TECHNICAL USER DOCUMENTATION

Privileged Frame Simulation

Complete End-User Manual

Installation and setup Configuration and commands

Orbital mechanics SR and static GR Full PF construction

Quantum channel QKD and ground links Reproducibility

Edition 1.17.0

20 September 2026

Package release:
`full-comparison-2026-09-20-r26-numerical-relativity-workflows`

Compiled from the accompanying LaTeX source. Includes worked examples, mathematical notation, configuration and command references, diagrams, source references and a topic index.

Contents

List of Figures

Chapter 1

How to use this manual

This manual documents the complete delivered Privileged Frame Simulation package: its scenario engine, real satellite ephemeris interface, matched-state numerical orbit comparison, static-GR photon model, supplied full PF adapter, quantum channel, optional proper-time storage channel, configurable QKD forecasts and ground links, saved sessions, sensitivity experiments, local browser dashboard and a separate hosted SaaS edition. It is written for a reader who wants to install the package, choose a physically meaningful experiment, run it, understand every category of output and preserve the evidence needed to reproduce the calculation.

The documented release is `full-comparison-2026-09-20-r26-numerical-relativity-workflows`, distributed as `PF_Simulation_Standalone_and_SaaS_r26.zip`. This is manual edition 1.17.0. Chapter 31 documents Einstein Toolkit execution, kuibit analysis, bounded numerical-metric propagation, and explicit cross-workflow research companions. Chapter 30 documents optional scientific backends, published-source checks, measured-event replay, hosted engineering and vendor-worker routing, and explicit evidence boundaries. Chapter 28 adds explicit full-state/channel import, population-specific rates, prespecified sampling and convergence, traceable network resources, streaming archives and frozen-model acquisition validation. Chapter 27 documents hosted access, projects, managed inputs and execution policies while preserving standalone operation. Chapter 26 explains study purpose, physical topology and quantum experiments, including explicit mission coupling and complete density-matrix handoff. Chapter 25 adds nine configurable quantum protocol/component profiles, exact and sampled evidence, numerical checks and a separate narrative benchmark report. The configuration and GUI chapters document canonical physical settings, conflict rejection before overrides, migration of legacy imports and preservation of derived QKD defaults. Chapter 24 adds a scoped research risk-audit workflow, uncertainty-aware decisions, covariance-based observation diagnostics and hash-checked evidence bundles. Chapter 23 adds normalized research exports, conditional Monte Carlo uncertainty and held-out observation comparison while preserving the scenario engine. Chapter 22 documents the complete graphical interface, including graphical environment setup, workflow selection, configuration editing, execution, result inspection and a shared-UTC overview of live comparison forecasts. The preceding scientific and command-line chapters remain available. Chapter 21 adds configuration-controlled published-data benchmarks, including numerical reference comparisons, laboratory count analysis and satellite telemetry validation. Chapter 20 retains the QKD operating and mathematical reference, three protocol branches, ground-link assumptions and the source-audited Micius benchmark. Examples and implementation descriptions were checked against this release's Python, JSON, JavaScript, batch launchers, Markdown guides and saved execution evidence. Later package versions can change defaults, schemas or commands. Record your own release identifier before comparing outputs with this manual.

Read the first-run instructions before editing configuration. The supplied examples already exercise SR, static GR, phase noise, sensitivity, tracking and replay. A successful reproducible example establishes that the selected execution path works in your installation. It does not establish experimental correctness of the proposed PF theory or the empirical accuracy of an orbit or optical apparatus.

1.1 Reading routes

Your objective	Read first	Then use
Install and obtain a first result	Installation and first-run workflows	Saved results and the dashboard guide
Understand differences between the three scenarios	Scientific model and event construction	Emission timing, channel assumptions and scenario fields
Compare numerical dynamics with public orbit predictions	Matched-state comparison workflow	Force-model configuration, residuals and sensitivity

Your objective	Read first	Then use
Run current-UTC tracking with SR or GR	Live workflow and tracking configuration	Data provenance, timing-table validity and troubleshooting
Change phase noise or detector assumptions	Quantum model and configuration reference	Tomography, rate interpretation and controls
Reproduce a saved experiment	Session files and replay workflow	Hashes, validation and the reporting checklist
Investigate an unexpected graph	Reading graphs and dashboard controls	Result schemas, status fields and troubleshooting
Find an exact option or configuration key	Command reference or configuration tables	Topic index and package source map

The PDF has a linked table of contents, navigation bookmarks and a topic index. Search for exact terms such as `gr_static`, `phase_correlation`, `replay_validation.json` or `gate_reference`. Code blocks use literal option names. Values with units in their names are stored in those units unless explicitly stated otherwise. A display may convert metres to kilometres or seconds to nanoseconds without changing the file's units.

1.2 What counts as evidence

1.2.1 A model must actually execute

A calculated result must be traceable to the implementation that produced it. A reference number, fitted parameter or precomputed illustration must not be presented as a newly executed physical simulation. Analytical calculations are legitimate when they implement the stated model and their inputs, assumptions and output scope are disclosed; a more complicated numerical method is not inherently a more accurate physical model.

The production polarization channel constructs a Qiskit circuit, applies its configured channels and obtains the two-photon density matrix from `AerSimulator`. Fidelity and purity are calculated from that matrix. Independent closed-form Bell-state expressions check the result; they do not replace a failed backend. Optional tomography draws measurement outcomes from the calculated state and declared collection model. These are simulated samples, not detector observations. QKD forecasts use the declared optical, count and finite-key models, which need not be represented as quantum circuits.

Scenario 3 deliberately reuses Scenario 2's physical history and quantum state, then executes the full PF construction for each available reception pair. Its metadata and progress log identify this reuse. It is not a second propagation or density-matrix experiment and does not claim a new emission-control law. A standalone Scenario-3 request first computes the required physical history. Unavailable propagation, storage or PF selection remains explicitly unavailable; a backend error must not produce a canned successful result.

The r18 correction evidence includes unmodified production call traces and separately labeled failure-injection regression tests. A call trace establishes which implementation ran. Mathematical checks establish consistency with the implemented model. Neither alone establishes empirical apparatus accuracy.

The package supports several distinct levels of evidence. A model assumption is a chosen input, such as a Gaussian phase covariance or an illustrative drag density table. A calculated prediction follows from those inputs and the implementation. A software validation checks a defined numerical or logical property. A measured physical result requires independent measurements and calibration. Keep these levels separate when interpreting plots or writing a paper.

A successful solver status means the relevant numerical problem completed within its implementation checks. A small PF simultaneity residual checks the construction on the computed event pair. A small numerical-versus-SGP4 residual means those predictions agree over the selected interval. None of these alone demonstrates an experimentally preferred inertial frame, real detector timing precision, or the physical superiority of one emission controller.

1.3 Package scope at a glance

Included capability	Operational meaning	Main boundary
Circular and numerical orbital models	Generate prescribed satellite worldlines	Approximate dynamics with explicit force choices
Public GP/SGP4 tracking	Propagate published satellite elements at requested epochs	Prediction, not direct telemetry
Flat SR and static GR	Select photon, clock and event-coordinate conventions	GR is an isolated static spherical exterior model
Scenario 1	Coincident detector-worldline experiment	Distinct event structure; not the scoring target for other scenarios
Scenario 2	Separate detectors with retained old frame selection	Corrected event handling around the retained selector
Scenario 3	Full PF analysis of Scenario 2's same physical events	Per-pair diagnostic; no new predictive emission law
Qiskit polarization channel	Conditional density matrix, fidelity, purity and counts	Apparatus covariance is supplied, not inferred from frame labels
Configurable QKD	Intersatellite and dual-ground BBM92; single-arm decoy BB84	Forecasts with explicit counts, security assumptions and abort reasons; no operational keys
Dashboard	Read saved sessions, inspect plots and tables	Read-only; refreshing does not launch a simulation
Replay and sensitivity	Recompute frozen inputs and perturb declared parameters	Reproducibility and sensitivity are not calibrated uncertainty

1.4 Publication and verification status

This manual is typeset documentation, not a claim that the package or PF theory has been peer reviewed. The code includes mathematical checks, regression tests and saved experiments, with their limits stated in the validation chapter. The release's Windows batch files were inspected, while the recorded automated Python executions used Linux hosts. The browser dashboard's HTTP and JavaScript behavior tests passed, but direct browser inspection of the revised rendering and print layout remained blocked in the authoring environment. A supplied user PDF documented rendering of the earlier dashboard and motivated display fixes; it does not certify the revised layout.

Figures in this manual are explanatory diagrams or plots regenerated from explicitly identified saved results and equations. They are not screenshots of an internally verified revised dashboard. The manual itself has been rendered and visually inspected as a PDF.

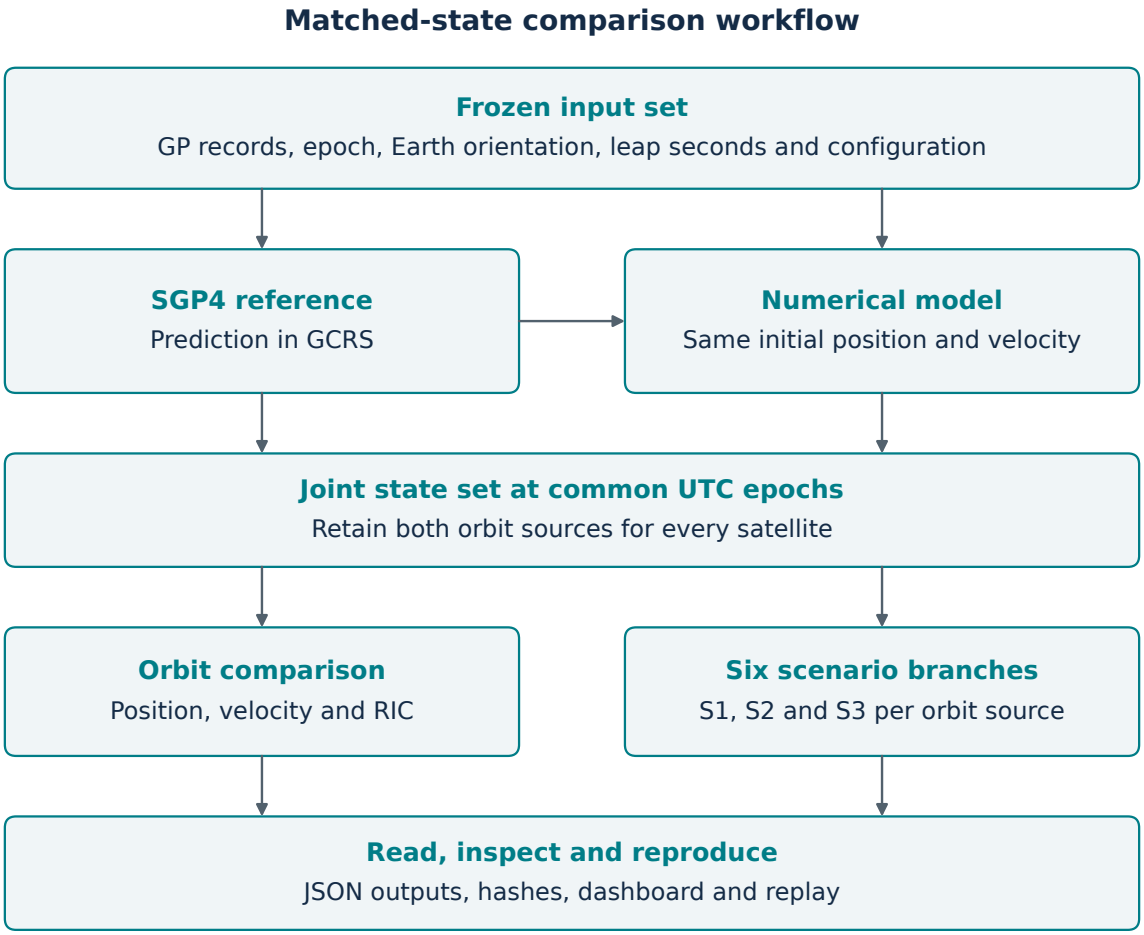

Figure 1: The matched-state workflow. Both orbit branches use a frozen reference-data context. Their common initial state makes later residuals interpretable as differences between propagation assumptions. The six scenario branches are three scenarios evaluated separately for each orbit source.

Chapter 2

Package orientation and experiment planning

2.1 Choose the appropriate entry point

Use `run_comparison_python38.bat` for the complete matched-state comparison, dashboard sessions, live forecast segments, frozen replay and sensitivity experiments. This is the main workflow for comparing independent numerical orbital dynamics with SGP4 predictions while keeping their initial states and reference epochs aligned.

Use `run_python38.bat` for the original scenario runner. It is useful for direct circular-orbit examples, individual scenarios, quantum controls and existing automation built around `scenario1.json`, `scenario2.json` and `scenario3.json`. It supports both SR and static GR, including tracking configurations. Its single-source output differs from the matched-state workflow. Both runners now share the same optional sampling and control defaults, configuration precedence and input-path rules.

Use `run_dashboard_python38.bat` only to inspect existing comparison sessions or sessions another process is producing. A dashboard does not generate orbital states, fetch a new satellite record, select a new force model or enable quantum noise. Those choices belong to the run command and its configuration.

The legacy `real_time_satellite_comparison.py` remains a compatibility tool. Its original circular-model comparison is explicitly unmatched. The full matched-state workflow replaces that limitation for new experiments. Do not combine residuals from the legacy and matched workflows as if they were the same experiment.

2.2 Plan one question per experiment

A useful experiment begins with a question that corresponds to a controlled change. For example: how does the numerical J_2 trajectory depart from SGP4 over a fixed 120-second interval? How does a one-metre initial radial perturbation change computed reception times? How do accepted counts change when the gate reference changes while the conditional polarization channel remains fixed? Each question identifies the independent change, the quantities that remain fixed and the output to inspect.

Do not change the orbit epoch, force model, detector gate, covariance and data source at once and then attribute the resulting difference to PF selection. Save a separate output directory for every variation. Use replay for reproducibility and sensitivity for explicitly declared parameter variations. Maintain a short experiment record containing the purpose, command, configuration filename and expected interpretation.

2.3 A practical first research sequence

1. Run the offline SR comparison preset and inspect its session status and branch validation files.
2. Confirm that the three satellites have identical numerical and reference initial positions and velocities at the first orbit sample.
3. Examine the position and velocity residual curves and the signed RIC components at later samples.
4. Inspect all six scenario rows at an epoch that actually has a photon ensemble.
5. Run the static-GR replay preset into a new directory, retaining the same reference dataset and a documented physical time interval.
6. Run the explicit covariance example and identify the changed apparatus inputs before interpreting fidelity or purity.

7. Perform a small sensitivity experiment and compare it with the nominal numerical branch.
8. Replay one saved session and read the exact recomputation result before relying on the workflow for larger studies.
9. Only then move to current-UTC acquisition, where data freshness and visibility are time dependent.

This sequence separates setup problems, model differences, data acquisition and experiment design. It also leaves a reproducible progression of saved outputs instead of a single result whose settings cannot be reconstructed.

2.4 Working directories and preservation

Extract the whole package into an ordinary writable directory. Launch commands from the package root, where the batch files and configuration presets reside. Avoid running individual files from inside the compressed archive. Keep saved sessions outside temporary folders you routinely clear. Retain the full session folder, not only `comparison.json` or an exported graph.

A complete comparison session contains captured inputs, provenance and integrity checks needed for recomputation. Moving the entire session to another folder is different from editing its JSON contents. Reading is safe; modifying a hashed file can invalidate replay. Perform exploratory transformations, CSV exports or manual annotations in a separate analysis directory. Both independent validator commands are read-only by default and can safely inspect a completed branch. If a new audit report is needed, use `--report` with a new filename outside the input run and every captured session. The original run-completion reports remain part of the preserved record.

2.5 Units and labels to check before analysis

Quantity	Native interpretation	Frequent mistake
Position and residual vectors	Metres in the stated chart	Mixing GCRS and GR areal coordinates
Velocity	Metres per second of the stated time coordinate	Rotating positions without consistently transforming velocity
Orbital angles	Radians in JSON orbit definitions	Entering degrees without conversion
Orbit elapsed time	TAI/TT-rate seconds in matched comparison rows	Joining by exact numeric equality with GR elapsed time
GR scenario elapsed time	TCG / Schwarzschild coordinate seconds	Treating coordinate time as every detector's proper time
Phase standard deviation	Radians of polarization phase	Treating it as an overall physically irrelevant phase
Pair rate	Expected accepted pairs per second	Calling it measured counts or conditional-state fidelity
PF residual	Numerical coordinate or tangent-space diagnostic	Interpreting it as a hardware clock accuracy specification

Use `epoch_utc` to align matching physical samples across the orbit and photon tables. Keep the difference sign convention with every export: orbit comparison quantities are numerical model minus the SGP4 reference prediction. Norms remove sign; RIC components retain directional information.

Chapter 3

Installation and working environment

3.1 Choose the workflow before installing

The package has two complete execution routes. Use the matched-state comparison workflow when you want a numerical orbit model and an SGP4 reference prediction to begin from the same satellite states, then compare their orbital and photon results. Use the original scenario runner when you want to study one configured orbit source directly, including synthetic circular orbits. Both routes support Scenarios 1, 2, and 3 and the `flat_sr` and `gr_static` photon models. Their command options and output layouts differ, so commands should not be mixed indiscriminately.

Your immediate goal	Recommended entry point
First complete example with saved orbital inputs	`run_comparison_python38.bat`
Compare numerical propagation with SGP4	`run_comparison_python38.bat`
View sessions already on disk	`run_dashboard_python38.bat`
Run synthetic or single-source photon scenarios	`run_python38.bat`
Recompute a frozen matched-state session	Comparison runner with `--replay`
Recreate a scenario image from result JSON	`plot_comparison.py`
Recheck independent result invariants	`validate_results.py`

The legacy comparison dashboard is a viewer for saved sessions. The newer workbench in Chapter 22 can configure and submit simulations; neither interface by itself collects measured photon data. Internet access is needed for package installation and fresh ephemeris/timing acquisition. The supplied offline examples use captured inputs and run without live downloads after dependencies are installed.

3.2 Extract the complete archive

Extract `PF_Simulation_Standalone_and_SaaS_r26.zip` into a writable folder, then open the extracted `privileged-frame-simulation` project folder. Confirm that it contains `README.md`, `run_comparison.py`, `config_comparison_replay.json`, the `dashboard` directory, `tracking_fixtures`, and `timing_data`. Keep the directory structure intact: an isolated Python script does not contain its configurations, browser assets, captured data, or vendored full PF implementation.

On Windows, open Command Prompt in that project directory. A path with spaces is acceptable when quoted. For example:

```
cd /d "C:\Users\Daniel\Downloads\privileged-frame-simulation"
```

Commands in this manual use Windows Command Prompt syntax. A caret at the end of a command line continues the command onto the next line. It must be the final character; do not add trailing spaces after it. A multi-line example may instead be entered as one line with the carets removed. In PowerShell, use one line or PowerShell's own continuation syntax. Prefix a batch filename with `.` and a backslash if PowerShell does not resolve it from the current directory.

3.3 Install without upgrading Python 3.8

The Windows setup explicitly selects the installed Python 3.8 interpreter and creates an isolated environment named `.venv-pf38`. It does not upgrade the system Python installation. Verify that the Windows Python launcher can find Python 3.8, then run:

```
py -3.8 --version
setup_python38.bat
setup_tracking_python38.bat
```

The first setup script installs `requirements.txt` and runs `pip check`. The second installs `requirements_tracking.txt` into the same environment and runs `pip check` again. A successful setup ends with a completion message. If either script reports failure, resolve the error before starting a simulation; an environment that exists but is only partly installed is not ready.

The tracking requirements are optional for purely synthetic scenario runs, but required for the matched-state comparison workflow, SGP4 tracking, and its offline replay examples. Offline replay avoids new downloads; it still uses the astronomy and SGP4 libraries to recompute the predictions.

The Python 3.8 branch pins Qiskit 1.2.4, Qiskit Aer 0.15.1, NumPy 1.24.4, SciPy 1.10.1, Matplotlib 3.7.5, typing-extensions 4.13.2, mpmath 1.3.0, and pytest 8.3.5. The tracking extension pins sgp4 2.24 and Astropy 5.2.2. `colorama` is pinned to 0.4.6. These are the package's declared compatibility choices, not instructions to upgrade every package independently.

The separate Python 3.12-or-newer requirement branch selects Qiskit 2.5.2, Qiskit Aer 0.17.2, Astropy 7.1.1, and the version ranges written in the requirements files. The Windows batch launchers are specifically named and written for Python 3.8. Use the appropriate interpreter directly for a deliberately separate modern-Python installation. The documented validation evidence identifies the actual tested runtime versions; the existence of a dependency marker is not a guarantee that every intermediate Python release was tested.

No IBM account, IBM cloud backend, Node.js installation, JavaScript package manager, map token, external web framework, or hosted dashboard account is required for standalone use. Hosted access uses the separate account flow in Chapter 27. Qiskit is used locally in the standalone edition. Unrelated IBM runtime or provider plugins in a global Python installation are one reason to use the isolated environment.

3.4 Verify the interpreter actually being used

The batch launchers call the environment's Python by an explicit path. Environment activation is therefore unnecessary. These commands should identify the expected interpreter and report consistent installed dependencies:

```
.venv-pf38\Scripts\python.exe --version
.venv-pf38\Scripts\python.exe -m pip check
.venv-pf38\Scripts\python.exe -m pip show qiskit qiskit-aer astropy sgp4
run_comparison_python38.bat --help
run_python38.bat --help
```

Avoid switching from a batch launcher to bare `py` midway through troubleshooting: bare `py` can select a different installation and its unrelated packages. For example, `py run_privileged_frame_simulation.py` may bypass `.venv-pf38`, whereas `run_python38.bat` selects it reliably.

All five batch scripts change their working directory to the folder containing the script. Relative output directories in their arguments therefore resolve under the package folder even if the launcher was called from elsewhere. Use a quoted absolute path when you deliberately want outputs in another location.

3.5 Other operating systems

The `.bat` files are Windows conveniences. On an installation with a suitable Python interpreter and compatible wheels, the equivalent operations are ordinary Python commands:

```
python3.8 -m venv .venv-pf38
.venv-pf38/bin/python -m pip install -r requirements.txt
.venv-pf38/bin/python -m pip install -r requirements_tracking.txt
.venv-pf38/bin/python -m pip check
.venv-pf38/bin/python run_comparison.py --help
```

Run these commands from the project directory. Use the actual interpreter name installed on your machine. This documents the direct entry points; it does not claim that every operating system, processor architecture, graphics stack, or browser was tested.

3.6 Updating an existing installation

Keep an existing working `.venv-pf38` environment and your experiment folders. Back up custom configuration files before copying a new release over the program files, because shipped presets may have the same names as locally edited presets. The updated package is complete, so a separate clean extraction is also a useful way to compare versions without overwriting custom work.

Preserve the `dashboard` directory as a unit when updating browser assets. After replacing JavaScript or CSS, reload the page; if old styling persists, perform a hard refresh. A browser tab can continue displaying cached assets after the Python files have changed. The accompanying `UPDATE_INSTRUCTIONS.md` describes the release's compatibility changes, and `MANIFEST.json` describes the shipped source rather than freshly generated experiment outputs.

Edition 1.17.0 documents release `r26-numerical-relativity-workflows`. The optional storage model is configured in JSON and defaults to disabled; Chapter 19 gives its complete reference. Standalone plots use recorded scenario identities when files are renamed; see Section 11.1. Both main runners leave finite-count tomography and additional channel controls off unless configured or explicitly enabled. To preserve the earlier standalone runner's sampled-output behavior, add `--sample-counts --controls`, or set both fields to `true` under the new shared `execution` object. This changes the default workload and the presence of sampled estimates; it does not change the exact conditional channel equations.

Relative tracking-cache and timing-file inputs now resolve against the configuration file's directory in every command-line configuration loader. Output paths still resolve against the process working directory. Check custom configurations kept outside the package root during migration. Independent validator commands now inspect inputs without rewriting their stored reports; use an explicit external `--report` destination when a new audit artifact is wanted.

Chapter 4

Guided operating workflows

4.1 First matched-state example

Start with the captured offline SR example. It contains fixed historical ephemeris and Earth-orientation inputs, so it separates installation and computation checks from current network availability:

```
run_comparison_python38.bat ^
  --config config_comparison_replay.json ^
  --output comparison_sessions\first_sr ^
  --dashboard --open-browser
```

This command creates a new comparison session, initializes a numerical J_2 trajectory from each reference GCRS state, evaluates both orbit sources, runs the selected photon scenarios, validates the scenario results, and keeps the local viewer available. The default preset requests a 600-second forecast with 31 orbit samples and 3 photon-ensemble epochs. These two grids serve different purposes: the dashboard can show an orbit sample for which no photon ensemble was calculated.

The console prints the dashboard URL, generally `http://127.0.0.1:8765/`. It may open before calculations finish. A running or incomplete session is not yet a validated final result. Wait for `Validated session:` and the final completion message before treating the session as complete. The server continues until Ctrl+C, allowing you to inspect the results. Closing the browser tab does not stop the Python server.

If you want a shorter initial computation, use a separate output folder:

```
run_comparison_python38.bat ^
  --config config_comparison_replay.json ^
  --duration 30 --samples 3 --scenario-steps 1 ^
  --output comparison_sessions\short_sr
```

This is a shorter forecast, not a looser numerical tolerance. A single photon step is evaluated at the start of its interval; it does not summarize every orbit sample. Reducing the number of photon epochs is a practical way to reduce the work performed by the full PF and GR calculations while keeping their configured numerical checks.

The comparison runner requires a new or empty session directory. Repeating the same command with a completed output folder causes an explicit error rather than overwriting that experiment. Choose `first_sr_02`, a timestamped name, or omit `--output` to obtain a new timestamped directory automatically.

4.2 Inspect a session without recomputation

Launch the viewer independently when you want to compare several saved sessions:

```
run_dashboard_python38.bat ^
  --sessions comparison_sessions ^
  --port 8765 --open-browser
```

The `--sessions` argument points to the directory tree containing complete session directories, not to a `scenario1.json` file or a PNG. The viewer discovers comparison sessions and excludes the nested `reference`,

`numerical`, and sensitivity result folders from its session picker. Refreshing this view reads updated files; it does not download new GP data or rerun an orbit.

To browse the verification sessions shipped with the package:

```
run_dashboard_python38.bat ^
  --sessions verification\full_comparison --open-browser
```

Those records document prior calculations. Their timestamps do not become current by opening them in a browser. For a live demonstration, use the live workflow below and point the viewer at its output root.

4.3 Compare SR and static GR

Create two separate sessions so their settings, captures, and results remain distinguishable:

```
run_comparison_python38.bat ^
  --config config_comparison_replay.json ^
  --spacetime flat_sr --output comparison_sessions\sr_case

run_comparison_python38.bat ^
  --config config_comparison_gr_replay.json ^
  --spacetime gr_static --output comparison_sessions\gr_case
```

The spacetime choice applies to all selected scenarios in both orbit branches. Orbit-comparison charts remain GCRS charts in both cases. GR changes the photon/timing/PF branch and its prescribed-worldline embedding; it does not convert the J_2 or drag numerical integrator into an exact Schwarzschild geodesic integrator. Compare UTC epochs, recorded settings, and propagation status before comparing numerical values.

The static-GR implementation requires Earth obstruction to remain enforced. A ray blocked by the opaque Earth sphere is outside its exterior-only propagation domain. Do not use `--unobstructed` as a workaround for a blocked GR link; that option belongs to explicitly virtual flat-SR experiments.

4.4 Enable an explicit phase-noise assumption

Phase noise is off in the ordinary replay and live presets. The named covariance presets enable it explicitly. A full noisy GR comparison with finite counts and configured/default sensitivity probes is:

```
run_comparison_python38.bat ^
  --config config_comparison_gr_noise.json ^
  --sample-counts --sensitivity ^
  --output comparison_sessions\gr_noise
```

For a command-line covariance override, specify both arm standard deviations and the correlation, including an explicit zero if independence is intended:

```
run_comparison_python38.bat ^
  --config config_comparison_replay.json ^
  --phase-sigma-a 0.8 --phase-sigma-b 0.5 ^
  --phase-correlation 0.35 ^
  --phase-source "Illustrative Gaussian covariance" ^
  --output comparison_sessions\sr_noise
```

The standard deviations are radians, and the correlation is dimensionless. The source label records your declaration; the software does not verify that a value came from a measured instrument. These example

numbers are illustrative. They are not inferred from path geometry, frame labels, or closeness to Scenario 1. With common covariance and the same source state, equal conditional fidelity and purity across scenarios are an expected channel result even if accepted counts differ.

To turn off phase noise in a custom command starting from a noisy configuration, set both arm sigmas to zero. For a fully self-describing run, also replace its phase source text with a zero-noise description. Source depolarization is separate and remains whatever the configuration specifies. Zero phase noise does not imply unit fidelity when the source is depolarized.

4.5 Run sensitivity experiments

Add `--sensitivity` to run one-at-a-time perturbations from the same baseline and frozen reference. If `comparison.sensitivity` is absent, the package uses four default probes: emitter radial position plus and minus 1 metre, and detector 1 along-track velocity plus and minus 0.001 metre per second. An explicitly empty sensitivity list means no probes.

```
run_comparison_python38.bat ^
  --config config_comparison_replay.json ^
  --duration 120 --samples 5 --scenario-steps 2 ^
  --sensitivity --output comparison_sessions\state_sensitivity
```

The drag example already enables two configured probes, so it does not require the extra switch:

```
run_comparison_python38.bat ^
  --config config_comparison_drag_illustrative.json ^
  --output comparison_sessions\drag_example
```

This preset supplies explicit spacecraft properties and a bounded illustrative atmosphere. It demonstrates the drag and sensitivity machinery; it is not a measured atmosphere or a calibrated model of the named satellites. Each probe creates additional numerical scenario work. More probes and more photon epochs increase execution time.

Read the perturbation metadata before interpreting a response. Deltas are additive in the named parameter's units. For example, a density-scale delta of 0.1 changes a baseline multiplier of 1.0 to 1.1; it is not a general instruction to increase any baseline by ten percent. Sensitivity output measures response to the chosen perturbation. It does not estimate a confidence interval or establish orbital accuracy.

4.6 Acquire current-UTC comparison sessions

Use an online preset for fresh tracking. The offline replay preset is intentionally rejected with `--live`:

```
run_comparison_python38.bat ^
  --config config_comparison_live_gr.json ^
  --live --live-iterations 10 --live-interval 30 ^
  --output comparison_sessions\live_gr ^
  --dashboard --open-browser
```

Each iteration starts a separately named forecast segment at the current UTC time. The segment freezes its captured inputs and matched initial states, then propagates forward over the configured horizon. A later acquisition never silently resets an earlier numerical trajectory. Live output directories contain children such as `segment_0001_...` and `segment_0002_...`.

The interval is the minimum start-to-start spacing. If computation and validation take longer than 30 seconds, the next iteration starts when the prior work finishes. It is not a guarantee of a 30-second refresh rate. GP

downloads follow the tracking cache policy, normally a two-hour refresh interval, while SGP4 positions are evaluated at every requested UTC epoch. A new live segment can therefore use the same GP payload as the previous segment and still have a new start state.

Both spacetime modes work in live comparison. This example explicitly enables illustrative phase noise in GR:

```
run_comparison_python38.bat ^
  --config config_comparison_live.json ^
  --live --live-iterations 10 --spacetime gr_static ^
  --phase-sigma-a 0.8 --phase-sigma-b 0.5 ^
  --phase-correlation 0.35 ^
  --phase-source "Illustrative Gaussian covariance" ^
  --output comparison_sessions\live_gr_noise
```

The bundled catalog IDs identify example orbital roles: emitter 25544, detector 1 42829, and detector 2 20580. They do not assert that the real spacecraft carry the simulated optical equipment. Actual object names are read from published responses. A blocked optical geometry or zero accepted rate is a valid modeled outcome, not automatically a tracking connection failure.

4.7 Exact offline replay

Replay a completed session into a new directory:

```
run_comparison_python38.bat ^
  --replay comparison_sessions\gr_noise ^
  --output comparison_sessions\gr_noise_recomputed
```

The command verifies captured files, uses the session-local orbital and timing data offline, recomputes the numerical and reference results, and compares the numeric sections with the original session. It writes `replay_validation.json`. The checked sections include orbit rows, scenario summaries, scenario effects, and sensitivity outputs. Seeded finite counts are part of reproducibility when they were enabled originally. An unambiguous historical physical alias can be canonicalized in the new replay copy; `replay_validation.json` records the original and effective choices under `physical_alias_migration`, together with `source_evidence_modified: false`. The original captured files remain unchanged. Conflicting historical aliases are rejected before a replay output directory is created. Resolve the scientific choice in a new configuration and run a new experiment rather than editing the sealed source session.

Replay deliberately uses the captured settings. Do not combine it with `--config`, a new epoch, spacetime, force model, duration, sample count, scenario selection, phase parameters, `--live`, `--sensitivity`, `--sample-counts`, `--no-sampling`, `--no-sample-counts`, `--controls`, or `--no-controls`. Such combinations are rejected. `--output`, viewer options, and `--quiet` remain available. To study changed assumptions, create a new ordinary session; that is a new experiment rather than exact replay.

Exact numeric identity is tested on the runtime that performs the replay. A change in source, numerical libraries, platform, or floating-point behavior can produce a reported mismatch. Preserve both sessions and inspect the failed section instead of treating the mismatch as a reason to rewrite the original results. Viewing a session, verifying its file hashes, and recomputing it are three distinct operations.

4.8 Use the original scenario runner

The original runner remains useful for synthetic visible-link studies without a matched-state comparison. These examples run all three scenarios and write their JSON and a static comparison image:

```
run_python38.bat --scenario all ^
  --config config_visible_links.json --spacetime flat_sr ^
  --output results_sr

run_python38.bat --scenario all ^
  --config config_gr_visible.json --spacetime gr_static ^
  --output results_gr
```

Its default `config.json` retains the original 30 snapshots over 20 days and the original synthetic profiles. Many of those geometries are Earth-blocked. The visible-link presets are explicitly different examples; they do not silently alter the original defaults.

The runner's live mode evaluates one current-UTC photon epoch per update, rather than a matched-state forecast horizon:

```
run_python38.bat --scenario all ^
  --config config_live_gr_covariance.json ^
  --live --live-iterations 10 --live-interval 5 ^
  --spacetime gr_static --output results_live_gr_noise
```

Completed live updates are stored in `live_0001`, `live_0002`, and subsequent folders. Each has its own current run manifest and validation report. Ctrl+C retains completed outputs and reports interruption. Use fresh output roots to keep experiments separate, even though this runner permits reuse and excludes older unselected scenario files from the current run manifest.

Chapter 5

Scientific models and interpretation

5.1 What the package computes

The package combines several models with different jobs. An orbital provider supplies satellite positions and velocities as functions of time. The photon solver determines release and reception events on those moving trajectories. A scheduler chooses nonnegative release holds according to an explicitly selected simultaneity target. The full PF adapter, when requested, constructs a frame for a specified anchored reception pair. A separate quantum channel predicts a conditional polarization density matrix. Collection and finite-shot sampling determine whether that state can supply simulated measurement counts.

Keep these layers separate when interpreting a result. A trajectory difference is an orbital-model comparison. A nearly zero transformed time difference is a timing or frame-selection residual. Fidelity compares a quantum state with a specified Bell state. Purity measures the mixedness of that density matrix. An accepted rate measures collection under the configured loss, gate, and visibility assumptions. None of these quantities can replace another.

Three levels of support apply throughout the manual. A mathematical model states equations and assumptions. Implementation validation checks that the software solves those equations consistently, including independent identities and limiting cases. Empirical validation requires relevant observations or independently calibrated physical inputs. The package includes substantial implementation validation and published physical frameworks, but its example configurations do not establish measured orbital accuracy, calibrated satellite polarization noise, or an experimental PF advantage.

The workflow is a simulation and analysis environment. It does not command satellites, distribute real entangled photons, synchronize physical oscillators, or ingest measured quantum tomography. The term “live” means that a workflow obtains recent orbital-element data and evaluates predictions at current UTC epochs. It does not turn those predictions into telemetry measurements.

5.2 Mathematical notation and units

All equations use SI units unless a field explicitly states otherwise. The speed of light is $c = 299\,792\,458\ \mathrm{m\,s^{-1}}$; the central body’s gravitational parameter is $\mu = GM$, in $\mathrm{m^3\,s^{-2}}$. Bold symbols denote vectors or matrices, as defined where introduced. A superscript T denotes transpose, $\dagger$ denotes Hermitian conjugation, $\|\cdot\|$ denotes the Euclidean norm of coordinate components, and $\mathbf{I}_n$ is the $n \times n$ identity matrix. Greek spacetime indices run from 0 to 3; repeated spacetime indices are summed. The metric signature is $(-,+,+,+)$, with coordinates $X^\alpha = (ct, x, y, z)$. Thus $\mathrm{d}s^2 = g_{\alpha\beta}\,\mathrm{d}X^\alpha \mathrm{d}X^\beta = -c^2\mathrm{d}\tau^2$ along a timelike worldline, where τ is proper time and t is the named chart’s coordinate time.

The symbol ρ denotes a quantum density matrix. To avoid ambiguity, the isotropic radial coordinate is written r_{iso} and atmospheric mass density is written ϱ_{atm}. The phase covariance matrix is Σ in photon order A, B; density-matrix basis vectors are ordered B, A, as required by the Qiskit representation. The fidelity F, purity P, probabilities, phase correlation κ, and depolarizing weight p are dimensionless. Angles and phases are quoted in radians, with numerical radian values used in exponentials and trigonometric functions. Ordinary-frequency bandwidth is quoted in hertz, not radians per second.

5.3 The three scenarios

Scenario	Physical event structure	PF or timing interpretation
1: coincident	Detector B uses detector A's worldline, producing coincident endpoints under the common release policy.	A controlled coincident-path benchmark with two distinguishable photon modes.
2: separated legacy frame	Two configured detector worldlines receive photons under the selected release policy.	The retained equal-snapshot-radius selector defines an arbitrary selected frame in SR, or a selected coordinate chart in static GR.
3: separated full PF	Uses the same release and reception events, geometry, and channel as Scenario 2.	Applies the supplied newer two-stage, anchored PF construction to that particular reception pair.

Scenario 1 is implemented by copying detector 1's synthetic profile or by sharing its tracked/numerical worldline callback. It is not a reference state inserted into another scenario's quantum calculation. The polarization subsystem still uses distinguishable modes A and B; it does not model the complete multiphoton optical apparatus needed to route and distinguish two photons following the same macroscopic path.

The output distinguishes acquired tracking inputs from effective role assignments. `effective_role_lineage` records the coincident control's receiver substitution, and `effective_tracking_provenance` identifies the trajectory actually used. The original `tracking_provenance` remains the acquisition record; it must not be interpreted as three distinct effective receiver paths in Scenario 1.

Scenario 2 retains the older selection objective with corrected Lorentz transformations. At the snapshot epoch, all three positions describe events at the same base-coordinate time. The selector seeks a minimum-speed subluminal boost that makes the two transformed, emitter-relative snapshot radii equal. It solves an eigenvalue problem derived from that radius condition. Equal initial radii select the zero boost. Some collinear unequal-radius inputs have no admissible solution and their selected-frame scheduling request is rejected. An ECI or no-delay request can still execute; an unavailable optional selected-frame diagnostic is recorded separately with its reason. Equal snapshot radii alone do not make the later moving-receiver detections simultaneous: scheduling is a separate calculation.

Scenario 3 is generated from a deep copy of Scenario 2's result, followed by an independent full-PF calculation at each epoch. The only supported `scenario3_release_policy` is `same_as_scenario2`. This prevents the frame construction from silently changing the physical experiment. When Scenarios 2 and 3 show identical quantum metrics or accepted counts, that is the expected consequence of sharing the same experiment and channel. It is not a demonstration that a PF controller recovered a degraded state.

For Scenario 3, `physical_history_source` identifies Scenario 2, `physical_history_reused` is true and `quantum_state_recomputed_for_scenario3` is false. The `physical_history=scenario2` progress annotation makes the same distinction visible during execution. The full-PF status describes the new per-pair construction, not an independent recalculation of the shared quantum metrics.

The full construction can be applied only when its required events and selection conditions are available. In flat SR, a mathematically valid photon reception behind Earth can still exist as a hypothetical vacuum continuation; the visibility layer separately sets Earth-respecting counts to zero. Static GR uses exterior rays only, so an obstructed link supplies no reception pair for PF selection. Read the propagation, visibility, and PF statuses together.

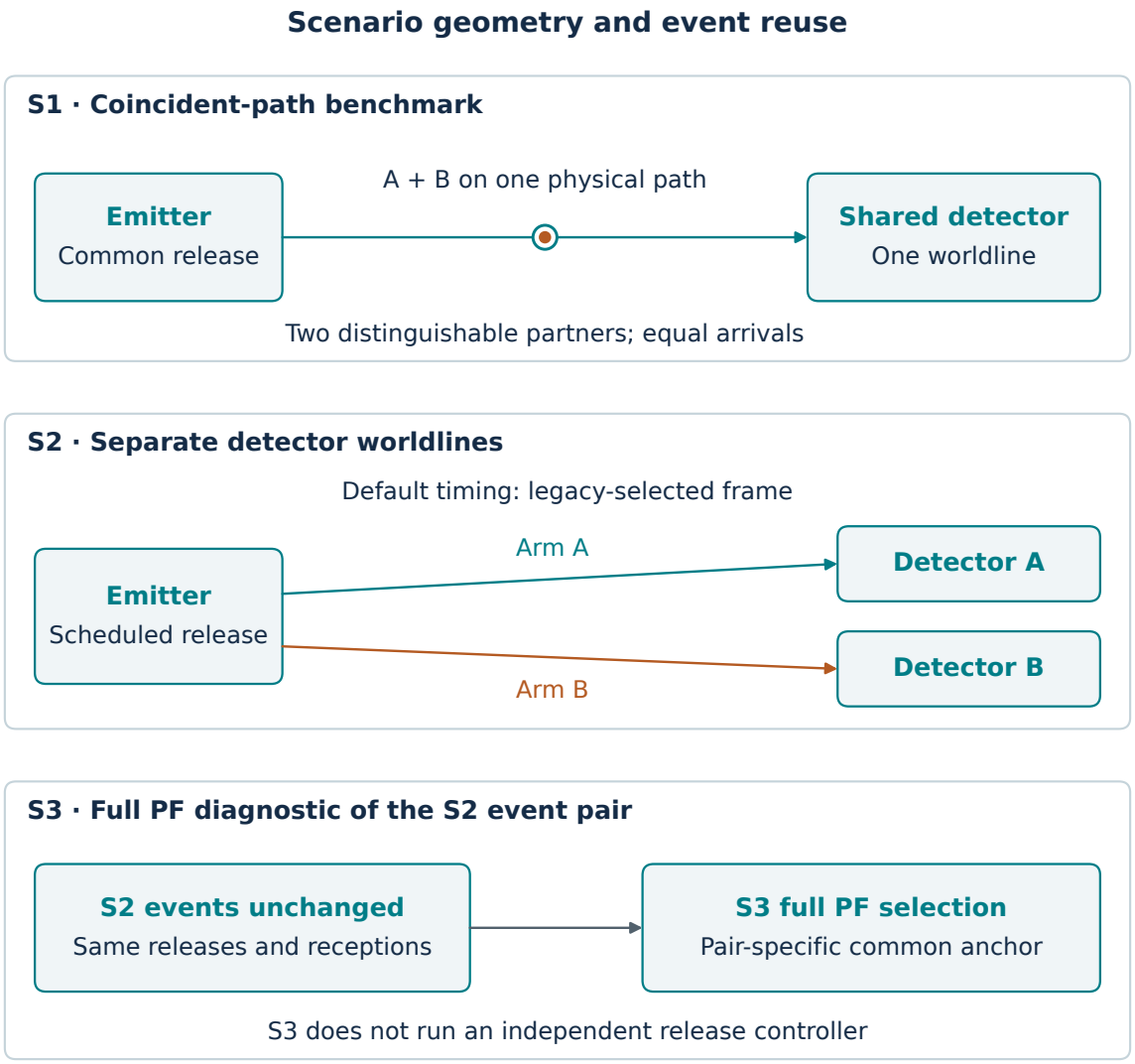

Figure 2: The three event structures. The single S1 connection represents two photon partners following the same physical path to the shared detector worldline. S2 shows the default legacy-selected-frame timing policy with separate detector worldlines. Scenario 3 consumes exactly the same physical release and reception events as Scenario 2; the new full construction is a separate diagnostic, not an independent emission controller.

5.4 Emission schedules and the meaning of simultaneity

The modeled pair is created once. A release hold represents a local optical delay or quantum memory for one partner before release from the moving emitter. It does not mean that an entangled pair is generated twice. By default the delay is ideal and adds no extra loss or decoherence. The optional `storage_channel` model instead integrates the emitter memory's proper elapsed time and applies independently configured polarization dephasing and retrieval loss; see Chapter 19. Its parameters require apparatus calibration for empirical predictions. Waiting before pair creation is not storage of an already-created state.

Each scheduler initially releases both partners at local epoch time zero and computes their receptions. It then identifies the partner that reaches the target simultaneity surface earlier and increases only that partner's release hold. One hold remains zero. In supported continuous timelike configurations, this defines the earliest target reception reachable with the chosen one-sided, nonnegative-hold policy. GR scheduling additionally has a finite hold-search bound and can return an unresolved result when the target cannot be bracketed.

Timing mode	Target	How to read the result
`selected_frame`	Equal reception times in the selected SR frame or GR linear chart.	The base-coordinate reception gap generally need not vanish.
`eci`	Equal reception times in the base coordinate system.	In static GR, the retained API name refers to Schwarzschild coordinate time, not UTC.
`none`	Both release holds are zero.	Moving receivers and unequal paths generally produce different arrival times.

For an SR boost velocity $\mathbf{V}$, define $\beta = \|\mathbf{V}\|/c < 1$ and $\gamma = (1-\beta^2)^{-1/2}$. The complete event transformation is

$$t' = \gamma\left(t - \frac{\mathbf{V}\cdot\mathbf{x}}{c^2}\right),$$
$$\mathbf{x}' = \mathbf{x} + \frac{\gamma-1}{\|\mathbf{V}\|^2}(\mathbf{V}\cdot\mathbf{x})\mathbf{V} - \gamma t\mathbf{V}.$$

The zero-velocity case is the identity, defined continuously rather than by division by zero. The code evaluates the spatial coefficient through the numerically stable identity $(\gamma-1)/\beta^2 = \gamma^2/(\gamma+1)$. For the reception differences $\Delta t = t_B - t_A$ and $\Delta\mathbf{x} = \mathbf{x}_B - \mathbf{x}_A$, the scheduling condition becomes

$$\Delta t' = \gamma\left(\Delta t - \frac{\mathbf{V}\cdot\Delta\mathbf{x}}{c^2}\right) = 0.$$

The selected-frame scheduler solves the transformed event-time condition directly. It independently solves the same physical target through that base-coordinate plane equation and reports the difference between the resulting release holds. Agreement is a coordinate-consistency check. By contrast, setting $\Delta t = 0$ defines a different simultaneity surface unless the extra term vanishes.

Reported gaps use detector B minus detector A. A negative gap therefore means B arrives earlier according to the named time coordinate. For each mode $j \in \{A, B\}$, the stored release hold h_j, flight time T_j, and arrival time t_j should satisfy $t_j = h_j + T_j$ in that same coordinate system. Full emission and detection events are retained, including selected-coordinate values and inverse-transform residuals. Do not subtract a PF-coordinate time from a base-coordinate time or combine a position at one epoch with a velocity at another.

The gate reference is another independent choice. `predicted_arrivals` centers a calibrated coincidence gate on the modeled arrival difference. `eci_zero` centers it at zero in the base coordinate-equivalent tags. `fixed` uses the configured center. A high acceptance rate under `predicted_arrivals` means that the assumed calibration follows the prediction; it does not demonstrate that raw asynchronous clocks were synchronized. Detector clock offsets are not estimated from data by this release.

5.5 The newer full PF construction

The authoritative implementation is the supplied, unmodified `vendor/privileged_frame_20260913.py`. The adapter checks the source SHA-256 before importing it. The adapter's public operation takes event A, event B, and an anchor, each with a time in seconds and a three-position in metres. In the simulation engine the events are the actual scheduled detections, and the anchor is the emitter's position at the start of that epoch, with local time zero.

In flat SR, the adapter subtracts the anchor event from both inputs exactly as a coordinate-origin change. The supplied Appendix-E selector first classifies the represented separation. Noncoincident timelike and null pairs have no subluminal PF under this construction and are excluded. Coincident inputs retain the supplied degenerate/coincident handling. A spacelike classification permits the construction to proceed, but does not guarantee that every subsequent constrained minimum is uniquely defined or numerically certifiable.

Stage 1 chooses a minimum-rapidity isotropic seed among boosts satisfying its equal anchor-relative Euclidean-radius constraint. Rapidity is monotonic in subluminal speed, so minimum rapidity relative to the initialization time axis can be found through a minimum-speed constrained problem. The supplied code reduces that problem to a quadratic system and uses analytic classification and a global-dual certificate. It is not a random search for any convenient frame.

Stage 2 works in the seed chart. Reception simultaneity fixes the correction boost's longitudinal component along the separation. The remaining transverse components must also satisfy the construction's anisotropic shell equality. The code reduces this to a quadratic zero set in a two-dimensional plane, which may be a circle, line, point, full plane, or empty set. The correction is the admissible minimum-rapidity solution when the selection is uniquely established. Ties, empty domains, and numerically uncertified cases remain explicit outcomes.

Let $\mathbf{x}''_A$ and $\mathbf{x}''_B$ be the anchor-relative spatial coordinates after applying the seed and correction boosts sequentially, and let $\beta_{\mathrm{c}} = \mathbf{V}_{\mathrm{c}}/c$ be the correction velocity in the seed chart. The independent second-stage shell check evaluates

$$Q_j = (\mathbf{x}''_j)^{\mathsf{T}} \left(\mathbf{I}_3 - \beta_{\mathrm{c}}\beta_{\mathrm{c}}^{\mathsf{T}}\right) \mathbf{x}''_j, \qquad Q_A = Q_B, \qquad t''_A = t''_B.$$

Here Q_j has units m^2. This is the correction chart's anisotropic quadratic form, not the Euclidean radius or a form constructed from the equivalent total boost. The seed and correction time axes are composed using four-velocity transformations. Two noncollinear boosts can include a Wigner rotation, so sequential and equivalent-total boosts need not have identical spatial axes. The adapter checks that their time coordinates agree, checks inverse transformations, and independently evaluates the correction-shell equality. A small final Euclidean-radius difference is not the defining second-stage shell test: the operational shell uses the correction chart's stated anisotropic form.

The full PF output is pair-specific and anchor-specific. Choosing a frame from a completed detection pair does not specify a unique predictive release-delay law, transfer that frame to arbitrary new pairs, calibrate satellite clocks, or introduce an interaction into the quantum channel. A future predictive controller would need a complete scheduling rule that chooses the anchor and target events before release and solves its coupled constraints. This package deliberately reports `delay_inference` as none.

5.5.1 Static GR and the common anchor tangent space

In `gr_static`, finite event coordinates are not subtracted and then treated as global Minkowski vectors. The supplied implementation numerically computes inverse exponential maps from the common anchor to the event points in the static spherical metric. These logarithm vectors are represented in one static orthonormal tetrad at that anchor. If O is the anchor, E_j is event j, and $e^{\hat{\alpha}}{}_{\mu}(O)$ is the orthonormal coframe at O, the construction is

$$\xi_j \in T_O M, \qquad \exp_O(\xi_j) = E_j, \qquad X_j^{\hat{\alpha}} = e^{\hat{\alpha}}{}_{\mu}(O)\, \xi_j^{\mu}.$$

The tetrad components use the tangent metric $\eta_{\hat{\alpha}\hat{\beta}} = \mathrm{diag}(-1, 1, 1, 1)$, with $X^{\hat{0}}_j = c\, t^{\tan}_j$. The same flat-space two-stage selector is then applied to those tangent-space components. The logarithm vectors are anchored geometric representations; their difference is not assumed to equal a global curved-spacetime displacement.

An exponential map follows a spacetime geodesic from a tangent vector at the anchor; a logarithm inverts that relation on a chosen local branch. This connecting geodesic serves the event-representation construction and must not automatically be identified with a physical emitted photon. The code follows a direct branch through mass continuation from the zero-mass geometry, with numerical refinement and sensitivity checks. It does not establish a globally unique logarithm, supply all winding/lensing branches, or construct a global simultaneity foliation of curved spacetime.

The adapter excludes anchors and events inside the spherical body. It conservatively rejects a direct anchor connector intersecting the body because no interior metric is supplied, and also rejects a resolved connector whose numerical path enters the body. Exponential-map reconstruction checks report how accurately the mapped tangent vectors reproduce the original event times and positions.

The output PF velocity in GR belongs to the anchor's orthonormal tangent chart. It is not a global Schwarzschild coordinate boost and is not a satellite's physical three-velocity at a remote event. If an output says `selected`, read that as a verified selection within the supplied construction, represented inputs, branch choice, and numerical checks. An extremely small PF time residual is an equation residual, not a claim of attosecond satellite-clock accuracy.

5.6 Spacetime, coordinates, and clocks

The `flat_sr` and `gr_static` choices apply to all requested scenarios. They select the photon/clock geometry and its interface to the orbital worldlines. They do not automatically switch the numerical comparison force model from Newtonian/J_2/drag into relativistic orbital dynamics.

Flat SR uses an Earth-centered inertial Cartesian approximation with the Minkowski metric. With tracked or numerical inputs, positions are GCRS-oriented and elapsed seconds follow the uniform TAI/TT rate. Satellite-like circular or numerical trajectories can accelerate in this flat benchmark; they are prescribed paths, not force-free straight-line SR geodesics.

Static GR uses an exterior Schwarzschild monopole with areal Cartesian positions and Schwarzschild coordinate time normalized to a stationary observer at infinity. Define $r = \|\mathbf{x}\|$, $\mathbf{n} = \mathbf{x}/r$, the Schwarzschild radius $r_{\mathrm{s}} = 2\mu/c^2$, and $f(r) = 1 - r_{\mathrm{s}}/r > 0$. The metric is

$$\mathbf{g}(\mathbf{x}) = \begin{pmatrix} -f & \mathbf{0}^{\mathsf{T}} \\ \mathbf{0} & \mathbf{I}_3 + (f^{-1} - 1)\mathbf{n}\mathbf{n}^{\mathsf{T}} \end{pmatrix}.$$

Equivalently, in spherical areal coordinates,

$$\mathrm{d}s^2 = -f\, c^2 \mathrm{d}t^2 + f^{-1}\mathrm{d}r^2 + r^2 \left(\mathrm{d}\vartheta^2 + \sin^2\vartheta\, \mathrm{d}\varphi^2\right).$$

Here ϑ is polar angle and φ is azimuth. For coordinate velocity $\mathbf{v} = \mathrm{d}\mathbf{x}/\mathrm{d}t$, define $v_r = \mathbf{n}\cdot\mathbf{v}$ and $\mathbf{v}_T = \mathbf{v} - v_r\mathbf{n}$. The implemented clock relation is

$$\begin{aligned} \frac{\mathrm{d}\tau}{\mathrm{d}t} &= \sqrt{f - \frac{v_r^2/f + \|\mathbf{v}_T\|^2}{c^2}}, \\ v^2_{\mathrm{local}} &= \frac{v_r^2}{f^2} + \frac{\|\mathbf{v}_T\|^2}{f}, \\ \frac{\mathrm{d}\tau}{\mathrm{d}t} &= \sqrt{f}\,\sqrt{1 - \frac{v^2_{\mathrm{local}}}{c^2}}. \end{aligned}$$

The final expression combines the static gravitational lapse with the local-static-observer kinematic factor. The speed in that factor must be the locally measured speed, not the uncorrected coordinate speed. Proper

elapsed time is obtained by integrating the rate along the specified worldline: $\Delta\tau = \int_{t_0}^{t_1} (\mathrm{d}\tau/\mathrm{d}t)\,\mathrm{d}t$. It is a physical interval under the metric model, but it does not define a clock's arbitrary synchronization offset.

Tracked GCRS positions are approximately identified with monopole isotropic coordinates before conversion to areal radius. Write the isotropic position as $\mathbf{q}$, its radius as $r_{\mathrm{iso}} = \|\mathbf{q}\|$, and $a = \mu/(2c^2)$. On the exterior branch $r_{\mathrm{iso}} > a$, define $s(r_{\mathrm{iso}}) = (1 + a/r_{\mathrm{iso}})^2$. Then

$$r = r_{\mathrm{iso}}\left(1 + \frac{a}{r_{\mathrm{iso}}}\right)^2, \quad \mathbf{x} = s\,\mathbf{q},$$
$$\mathbf{J} = s\,\mathbf{I}_3 + \frac{s'}{r_{\mathrm{iso}}}\mathbf{q}\mathbf{q}^{\mathsf{T}}, \qquad s' = -\frac{2a}{r_{\mathrm{iso}}^2}\left(1 + \frac{a}{r_{\mathrm{iso}}}\right).$$

The Jacobian $\mathbf{J} = \partial\mathbf{x}/\partial\mathbf{q}$ is dimensionless. Velocities use this full Jacobian, not merely the position scale factor. The separate time-rate conversion is

$$\frac{\mathrm{dTT}}{\mathrm{dTCG}} = 1 - L_G, \qquad L_G = 6.969290134 \times 10^{-10}, \qquad \mathbf{v}_{\mathrm{areal,TCG}} = \mathbf{J}\,(1 - L_G)\mathbf{v}_{\mathrm{GCRS,TT}}.$$

The tracked input velocities and uniform elapsed TT/TAI-rate seconds follow the package convention; this conversion is an explicit weak-field interface to the monopole chart. For Earth the leading areal-minus-isotropic radius difference is about 4.4 mm. That small algebraic difference does not imply millimetre ephemeris accuracy.

The numerical orbit-comparison tables and charts stay in GCRS even when photon propagation uses static GR. Their elapsed comparison time has the TT/TAI rate. Scenario photon offsets in GR use TCG-rate seconds and are joined to comparison states by their corresponding physical UTC epoch. Matching row indices or assuming those two elapsed numbers are exactly identical can associate different events. UTC, TAI, TT, TCG, proper time, and selected-chart time are distinct labels with explicit conversions.

The old Scenario-2 GR selected chart uses a constant Lorentz-shaped coordinate matrix $\mathbf{L}$. That is a coordinate change, not an isometry of the curved metric. With $X' = \mathbf{L}X$, its metric must transform as

$$\mathbf{g}'(X') = \mathbf{L}^{-\mathsf{T}}\,\mathbf{g}(\mathbf{L}^{-1}X')\,\mathbf{L}^{-1}.$$

For a local tangent $k' = \mathbf{L}k$, the scalar is preserved: $(k')^{\mathsf{T}}\mathbf{g}'k' = k^{\mathsf{T}}\mathbf{g}k$. The code checks nullness on transformed local photon tangents using this transformed metric. Applying the Minkowski interval formula to the finite endpoint displacement of a curved ray would be incorrect.

5.7 Orbital models and live ephemerides

5.7.1 Synthetic circular worldlines

Synthetic profiles use a spherical Earth radius of 6 371 000 m, gravitational parameter $3.986004418 \times 10^{14}\ \mathrm{m^3\,s^{-2}}$, altitude, plane inclination, and RAAN. Altitude is an increment above that sphere, not WGS84 geodetic height. RAAN rotates the orbital plane's node; RAAN equal to π does not itself make an orbit retrograde. Retrograde motion requires inclination above $\pi/2$.

The angular rate is $\Omega_{\mathrm{orb}} = \sqrt{\mu/r^3}$, in $\mathrm{s^{-1}}$. In the SR branch it parameterizes an accelerated circular benchmark. In the Schwarzschild branch it is also the exact coordinate angular rate of a circular timelike geodesic in the spherical exterior metric. The resulting proper clock rate for that GR circular family is $\sqrt{1 - 3\mu/(rc^2)}$. Numerical equality of some coordinate positions between branches is intentional; their metric and clock interpretation differ. The GR implementation requires the strictly stable circular-orbit domain $r > 6\mu/c^2$ for positive central mass.

Legacy fields such as `anomaly_rate`, spacecraft mass, cross section, and drag coefficient do not alter these synthetic circular trajectories. They are not the force inputs to the newer numerical orbit propagator. If testing drag, use the comparison workflow's explicit numerical model and its documented spacecraft/atmosphere configuration.

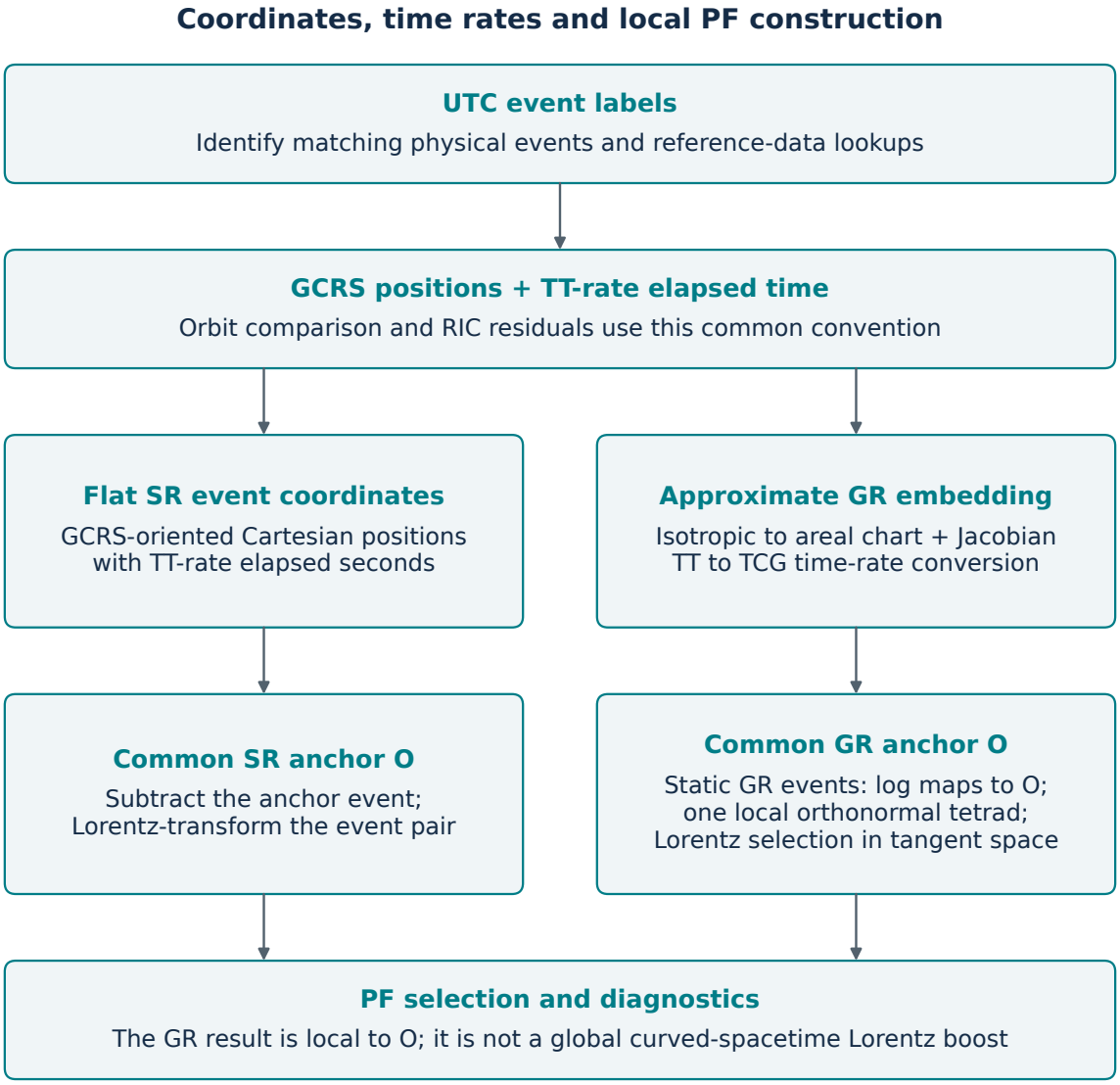

Figure 3: Frame and time conventions. Orbit residuals remain GCRS quantities in either mode. Entry into the static-GR model uses the documented approximate spatial-chart identification and TT-to-TCG time-rate conversion. The GR event construction then maps to an anchor tangent space before PF selection; this is not a global inertial Lorentz transformation of curved spacetime.

5.7.2 SGP4 predictions

The tracking provider obtains general-perturbation orbital elements from CelesTrak and propagates them with SGP4 using WGS72 conventions. It supports JSON GP data and validated TLE input. GP/TLE mean elements belong to their associated propagator; directly interpreting them as two-body osculating elements would change the model.

SGP4 yields TEME states. Astropy transforms both position and velocity, with the velocity attached as a Cartesian differential, into GCRS or ITRS as needed. Transforming velocity as though it were a position would omit frame-motion terms. The code uses split Julian dates when calling SGP4 to avoid unnecessary numerical time quantization.

Data retrieval UTC, element epoch, and propagation UTC have different meanings. The default permitted element age is seven days in either direction from the requested epoch. That is a configurable cutoff, not a guarantee of accuracy within seven days. The default refresh interval is two hours. An explicit stale-data override is recorded; network failures never silently create a synthetic satellite or select a replacement object.

UTC conversion additionally requires valid leap-second data. Earth-orientation transformations require IERS data and have their own coverage policy. The package's fallback leap-second file has a declared expiration, checked against the requested epoch; it does not authorize predictions beyond its coverage. `allow_degraded_iers` does not waive leap-second coverage. These checks preserve the coordinate contract rather than improve the underlying orbital-element accuracy.

5.7.3 Matched numerical orbit comparison

The full comparison initializes each numerical satellite with the reference SGP4 Cartesian position and velocity at exactly the same UTC epoch, in GCRS. It propagates that state independently afterwards. It does not repeatedly reset the trajectory to the reference within a segment. Live refreshes create new explicitly matched segments, so a discontinuity between segments must not be interpreted as a continuous accumulated residual history.

The `two_body` model integrates

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{v}, \qquad \frac{\mathrm{d}\mathbf{v}}{\mathrm{d}t} = -\frac{\mu}{r^3}\mathbf{x}, \qquad r = \|\mathbf{x}\|.$$

The `j2` model adds Earth's leading axisymmetric quadrupole acceleration. For fixed unit symmetry axis $\mathbf{k}$, $z = \mathbf{x} \cdot \mathbf{k}$, reference radius R, and dimensionless coefficient J_2, its extra acceleration is

$$\mathbf{a}_{J_2} = \frac{3J_2\mu R^2}{2r^5}\left[\left(\frac{5z^2}{r^2} - 1\right)\mathbf{x} - 2z\mathbf{k}\right].$$

The axis is normalized and held fixed over the segment; its default is GCRS $\mathbf{k} = (0, 0, 1)^\mathsf{T}$. This approximation does not include a full time-dependent Earth gravity field, precession, nutation, or polar motion in the force law. The coordinate-transformation use of IERS data does not silently add those dynamics to the numerical model.

The `j2_drag` model adds ballistic drag from a rigidly corotating atmosphere. With atmospheric angular velocity $\omega = \omega\mathbf{k}$, mass density ϱ_{atm}, spacecraft drag coefficient C_D, projected area A, and mass m,

$$\begin{aligned}\mathbf{v}_{\mathrm{rel}} &= \mathbf{v} - \omega \times \mathbf{x},\\ \mathbf{a}_{\mathrm{drag}} &= -\frac{1}{2}\varrho_{\mathrm{atm}} C_D \frac{A}{m}\|\mathbf{v}_{\mathrm{rel}}\|\mathbf{v}_{\mathrm{rel}}.\end{aligned}$$

Here ϱ_{atm} is in $\mathrm{kg\,m^{-3}}$, A is in $\mathrm{m^2}$, and m is in kilograms, yielding acceleration in $\mathrm{m\,s^{-2}}$.

Each spacecraft requires explicit mass, area, and drag coefficient. A positive density table is interpolated logarithmically between supplied spherical altitudes and multiplied by the configured scale. The code refuses

extrapolation outside that interval. Winds, attitude-dependent area, and a predictive space-weather atmosphere are not included. The density source label and classification record a declaration, not an independently verified calibration. SGP4's BSTAR is not converted into measured spacecraft drag properties.

The numerical model's default reference radius is 6,378,137 m, used for J_2 normalization, the numerical exclusion surface, and tabulated density altitude. The photon model's opaque sphere is 6,371,000 m. These different spherical conventions are explicit; neither represents ellipsoidal terrain or realistic near-limb atmospheric visibility. Near-grazing applications need a more complete Earth and atmosphere model.

DOP853 integrates forward through the requested horizon plus margin and backward through the negative margin, providing dense interpolation only within that domain. Photon release/intercept calls can use the margin. Calls outside it fail rather than extrapolate. Relative and separate position/velocity absolute tolerances control numerical integration, but are not physical orbit-error bounds. A force evaluation at/below the exclusion surface or outside the density table can reject a segment, including a trial solver stage near a boundary.

Selecting static GR for photons maps numerical worldlines into the Schwarzschild chart with the stated Jacobian and time-rate conversion. It does not make Newtonian/J_2/drag trajectories exact Schwarzschild geodesics, incorporate an oblate-Earth GR metric, or add frame dragging, solar radiation pressure, third-body gravity, maneuvers, or higher gravity harmonics.

5.7.4 Reading residuals and RIC components

The comparison residual is numerical prediction minus SGP4 prediction at the same epoch and in the same GCRS chart. Its norm is a model disagreement, not a measured tracking error. Zero initial residual follows from initialization and is not an independent validation success.

RIC components resolve the position residual in the reference orbit's instantaneous basis. Let $\mathbf{r}_{\mathrm{ref}}$ and $\mathbf{v}_{\mathrm{ref}}$ be the reference state and $\Delta\mathbf{x} = \mathbf{x}_{\mathrm{num}} - \mathbf{r}_{\mathrm{ref}}$. The basis and signed residuals are

$$\mathbf{e}_R = \frac{\mathbf{r}_{\mathrm{ref}}}{\|\mathbf{r}_{\mathrm{ref}}\|}, \qquad \mathbf{e}_C = \frac{\mathbf{r}_{\mathrm{ref}} \times \mathbf{v}_{\mathrm{ref}}}{\|\mathbf{r}_{\mathrm{ref}} \times \mathbf{v}_{\mathrm{ref}}\|},$$

$$\mathbf{e}_I = \mathbf{e}_C \times \mathbf{e}_R, \quad \begin{pmatrix} \Delta x_R \\ \Delta x_I \\ \Delta x_C \end{pmatrix} = \begin{pmatrix} \mathbf{e}_R^{\mathsf{T}} \\ \mathbf{e}_I^{\mathsf{T}} \\ \mathbf{e}_C^{\mathsf{T}} \end{pmatrix} \Delta\mathbf{x}.$$

The signed components identify radial, along-track, and cross-track differences. They are projections of the difference vector, not independent satellite coordinates. If the reference angular momentum cannot define that basis, the code reports an undefined RIC status.

Sensitivity probes change one declared input or model at a time. They show how outputs respond to that perturbation. A density-scale change, initial-position offset, or force-model switch does not produce a calibrated confidence interval. A confidence claim would require an input probability model and justified uncertainty propagation.

5.8 Photon propagation and physical visibility

For release time t_e and reception time $t_d \geq t_e$, the flat solver finds a future reception on a moving detector by solving

$$c(t_d - t_e) = \|\mathbf{x}_{\mathrm{det}}(t_d) - \mathbf{x}_{\mathrm{emit}}(t_e)\|\,.$$

It uses the emitter position at each actual release, so a held photon is not launched from the emitter's obsolete initial position. Null-distance residuals and full event inverse transformations support the numerical audit.

In static GR the ray solver integrates the exterior Schwarzschild null Hamiltonian with conserved energy normalization. In coordinates $X^\alpha = (ct, \mathbf{x})$, write the ray covector as p_α. Its normalization is arbitrary for a null ray; the solver fixes $p_0 = -1$. For affine parameter λ, the equations are

$$H = \frac{1}{2} g^{\alpha\beta} p_\alpha p_\beta = 0,$$
$$\frac{\mathrm{d}X^\alpha}{\mathrm{d}\lambda} = \frac{\partial H}{\partial p_\alpha}, \qquad \frac{\mathrm{d}p_\alpha}{\mathrm{d}\lambda} = -\frac{\partial H}{\partial X^\alpha}.$$

The code integrates a scaled version of these equations, preserving the null constraint and fixed conserved energy. A boundary-value solve finds a direct ray between endpoints; the moving-receiver iteration makes its endpoint coincide with the detector worldline. This is not a gravitational multiplier pasted onto a flat travel time. Local nullness, angular-momentum behavior, endpoint conditions, and interception residuals are reported. Returned ray samples are spaced in affine parameter; their associated coordinate times must be used when interpreting them.

The supported domain is a direct exterior ray around an opaque spherical body. Interior propagation, atmosphere, plasma, rotating-body effects, multiple lensing images, and general winding branches are absent. An occulted path provides no GR flight time or observations. An unresolved numerical solution is distinguished from physical occultation; its rates remain unknown rather than being reported as successful zero-noise detections.

GR link records also report emitter and detector clock rates, integrated proper intervals, and a frequency ratio computed from photon covector/observer four-velocity contractions. With normalized observer four-velocity $u^\alpha = (1, \mathbf{v}/c)^\alpha/(\mathrm{d}\tau/\mathrm{d}t)$, the measured frequency is proportional to $-p_\alpha u^\alpha$. Therefore

$$\frac{\nu_{\mathrm{det}}}{\nu_{\mathrm{emit}}} = \frac{(-p_\alpha u^\alpha)_{\mathrm{det}}}{(-p_\alpha u^\alpha)_{\mathrm{emit}}},$$

using the same ray-energy normalization at both endpoints. These quantities contain the modeled kinematic and gravitational effects. Their presence does not automatically retune an optical filter, change the channel covariance, or transform the configured detector bandwidth into a full spectral-transfer model. The collection model's bandwidth and jitter remain specified apparatus inputs.

5.9 The polarization channel and Qiskit metrics

The simulated state concerns polarization in calibrated horizontal/vertical (H/V) bases, with two modes labeled A and B and qubit values $0 \equiv H$, $1 \equiv V$. Qiskit's computational basis is ordered $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}_{BA}$. The default Bell target is the singlet $|\Psi^-\rangle$; `phi_plus`, corresponding to $|\Phi^+\rangle$, is also supported:

$$|\Psi^-\rangle = \frac{|01\rangle - |10\rangle}{\sqrt{2}}, \qquad |\Phi^+\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}}.$$

The target is the chosen ideal Bell state, not Scenario 1's computed state. Each scenario is evaluated against that independently specified target.

For the default Bell preparation, the source first receives the configured depolarizing mixture, followed by the configured polarization phases. The general full-state/local-channel input path in Chapter 28 retains the full matrix and uses its separate numerical reference; the Bell-family formulas below do not apply to an arbitrary imported source. For chosen Bell target $|\psi\rangle$ and depolarizing weight $p \in [0, 1]$, the source state is

$$\rho_{\mathrm{src}} = (1 - p)|\psi\rangle\langle\psi| + \frac{p}{4}\mathbf{I}_4.$$

The source depolarizing parameter defaults to $p = 0.005$. Turning phase noise off therefore does not make the source perfectly pure. With zero phase noise, default source quality and zero deterministic phase, the exact conditional metrics with storage disabled are $F = 0.996250000$ and $P = 0.992518750$.

The two accumulated H/V relative phases have a zero-mean Gaussian distribution in the order A, B:

$$\phi = \begin{pmatrix} \phi_A \\ \phi_B \end{pmatrix} \sim \mathcal{N}(\mathbf{0}, \Sigma), \qquad \Sigma = \begin{pmatrix} \sigma_A^2 & \kappa\sigma_A\sigma_B \\ \kappa\sigma_A\sigma_B & \sigma_B^2 \end{pmatrix}.$$

The standard deviations σ_A, σ_B are nonnegative radians and correlation $\kappa \in [-1, 1]$. Covariance entries have units rad^2. The matrix is positive semidefinite under those constraints, including singular perfectly correlated limits. If either variance is zero, its cross covariance is zero; a Pearson coefficient for the zero-variance variable has no independent meaning.

The current main configuration defaults to zero phase noise. Nonzero values require an explicit correlation and a parameter-source declaration. That declaration is retained in output, with `phase_parameter_source_verified` false. The program does not fit phase measurements or verify that a named source applies to the modeled satellite apparatus.

For each phase pair the operation is the local unitary $U(\phi) = R_z(\phi_B) \otimes R_z(\phi_A)$, with $R_z(\phi) = \exp(-\mathrm{i}\phi Z/2)$ and $Z = \mathrm{diag}(1, -1)$. Averaging those operations gives a completely positive, trace-preserving dephasing channel. Define basis index $j = 2b + a$, where $a, b \in \{0, 1\}$ label the A and B bits, and define

$$\mathbf{g}_j = \frac{1}{2}\begin{pmatrix} 1 - 2a \\ 1 - 2b \end{pmatrix}, \qquad M_{jk} = \exp\left[-\frac{1}{2}(\mathbf{g}_j - \mathbf{g}_k)^{\mathsf{T}}\Sigma(\mathbf{g}_j - \mathbf{g}_k)\right].$$

Then $\mathcal{E}(\rho)_{jk} = M_{jk}\rho_{jk}$. For a spectral factorization $\mathbf{M} = \sum_\ell \lambda_\ell \mathbf{w}_\ell \mathbf{w}_\ell^\dagger$, the diagonal Kraus operators are

$$K_\ell = \sqrt{\lambda_\ell}\ \mathrm{diag}(\mathbf{w}_\ell), \qquad \mathcal{E}(\rho) = \sum_\ell K_\ell \rho K_\ell^\dagger, \qquad \sum_\ell K_\ell^\dagger K_\ell = \mathbf{I}_4.$$

The multiplier is positive semidefinite and has $M_{jj} = 1$, which give complete positivity and trace preservation, respectively. The code handles numerically insignificant spectral roundoff and checks Kraus completeness explicitly. The implementation evaluates the exact Gaussian characteristic function for every density-matrix coherence, factors the resulting positive semidefinite multiplier into diagonal Kraus operators, and checks completeness. Qiskit Aer applies those Kraus operators in a density-matrix simulation; independent analytic formulas check its output.

For the singlet, the coherence contrast is governed by the phase difference; for $|\Phi^+\rangle$, it is governed by the phase sum:

$$C_{\Psi^-} = \exp\left[-\frac{1}{2}\left(\sigma_A^2 + \sigma_B^2 - 2\kappa\sigma_A\sigma_B\right)\right],$$
$$C_{\Phi^+} = \exp\left[-\frac{1}{2}\left(\sigma_A^2 + \sigma_B^2 + 2\kappa\sigma_A\sigma_B\right)\right].$$

With the corresponding contrast denoted by C, source depolarization p, and deterministic phase θ applied to mode A, the reference formulas for the base channel are

$$F = (1 - p)\frac{1 + C\cos\theta}{2} + \frac{p}{4},$$
$$P = (1 - p)^2\frac{1 + C^2}{2} + \frac{1 - (1 - p)^2}{4}.$$

Here $F = \langle\psi|\rho|\psi\rangle$ is squared fidelity to the pure Bell target and $P = \mathrm{Tr}(\rho^2)$ is purity. For a physical two-qubit density matrix, $0 \le F \le 1$ and $1/4 \le P \le 1$. A deterministic unitary phase can lower fidelity to that target while leaving purity unchanged. Random phase averaging can lower both. Equal-strength, perfectly correlated phase rotations protect the singlet's relevant coherence, whereas the corresponding $|\Phi^+\rangle$ coherence responds differently. This is a property of the selected state and coupling, not a privilege assigned to a path label.

For example, $\sigma_A = 0.8$ rad and $\sigma_B = 0.5$ rad with $\kappa = 0.35$, default source depolarization, and zero deterministic phase produce approximately $F \approx 0.865468879$ and $P \approx 0.766471722$. Supplying that same covariance to coincident and separated layouts gives the same exact state metrics when storage is disabled or the storage coherence factors also agree. With storage enabled, replace the base contrast by the combined contrast derived in Chapter 19.

The former geometry-derived Gaussian correlation rule is absent from the current channel. Path separation, proximity to the coincident arrangement, selected-frame velocity, PF selection status, and orbital residuals do

not set covariance. Obsolete correlation-length and correlation-time configuration keys are rejected. Coincident Scenario 1 is a useful physical-event control, but is not used as an ideal numerical benchmark from which Scenario 2 or 3 is penalized.

Fixed covariance is itself an experimental-control assumption. Real changes in propagation paths, optical components, or exposure could change noise. Modeling those effects requires an identified polarization coupling and calibrated statistics, potentially a field covariance integrated along each path. The optional storage model adds the specific local memory mechanism documented in Chapter 19; it does not infer propagation noise from path separation. The present Gaussian ensemble contains no temporal drift process across shots or epochs and no claimed quantum degradation from relativity of simultaneity alone.

5.10 Counts, timing gates, and tomography

The collection calculation keeps conditional state quality separate from the chance of observing a pair. The simplified pair survival probability is $\eta_A \eta_B$, where $\eta_A, \eta_B \in [0, 1]$ are the configured collection efficiencies. A Gaussian arrival-time gate supplies another factor, and Earth visibility supplies a mask where enforced. The source rate multiplied by the accepted probability yields the expected accepted pair rate. The equations immediately below describe storage disabled. When enabled, the memory survival product multiplies the collection probabilities as specified in Chapter 19.

Let σ_ν be the configured ordinary-frequency bandwidth in hertz, interpreted by the model as a Gaussian spectral intensity standard deviation. Its transform-limited temporal intensity width σ_t and the arrival-difference spread σ_Δ are

$$\sigma_t = \frac{1}{4\pi\sigma_\nu}, \qquad \sigma_\Delta = \sqrt{2\left(\sigma_t^2 + \sigma_{\text{jit}}^2\right)},$$

where σ_{jit} is the standard deviation of each detector's independent timing jitter. Equal individual pulse widths and independent timing contributions are assumptions of this convention. Write the mean gate residual as $\delta = \Delta t - t_{\text{center}} + \epsilon_{\text{gate}}$, where Δt is the modeled B-minus-A arrival gap, t_{center} is the chosen gate center, and ϵ_{gate} is the configured calibration error. For gate half-width $w \geq 0$ and standard normal cumulative distribution function Φ, the acceptance factor is

$$p_{\text{gate}} = \Phi\left(\frac{w - \delta}{\sigma_\Delta}\right) - \Phi\left(\frac{-w - \delta}{\sigma_\Delta}\right).$$

The implementation uses an algebraically equivalent, tail-stable evaluation. For visibility factor $V_{\text{vis}} \in \{0, 1\}$, selected according to the enforced visibility policy, source pair rate R_{src}, and emitted-pair budget N_{emit} per measurement setting,

$$\begin{aligned} p_{\text{acc}} &= \eta_A \eta_B p_{\text{gate}} V_{\text{vis}}, \\ R_{\text{acc}} &= R_{\text{src}} p_{\text{acc}}, \qquad N_{\text{acc},s} \sim \text{Binomial}(N_{\text{emit}}, p_{\text{acc}}). \end{aligned}$$

Here s labels a Pauli measurement setting. Rates are pairs per named source-run coordinate second; the package does not silently reinterpret them as pairs per source proper second. Unresolved propagation supplies an unknown acceptance rather than inventing a value for V_{vis}. Other spectral shapes, correlated detector jitter, dispersion, frequency-dependent loss, dark counts, accidental pairs, dead time, and saturation need additional models.

In flat SR the output provides both controlled-unobstructed and Earth-respecting probabilities; the selected enforcement setting decides which drives simulated counts. Static GR requires exterior visibility and rejects an unobstructed-through-Earth request. The phase ensemble and acceptance are assumed independent, so gating and loss do not reweight the conditional polarization state. If arrival errors and polarization phases are physically correlated, a joint model is needed.

For finite-shot tomography the code samples an accepted binomial count independently for each of the nine Pauli settings XX through ZZ, then measures that many polarization pairs in Aer. The first setting letter

measures A, while reported bit strings follow BA ordering. The emitted-pairs budget is per setting; it is not the total over all nine settings. Shared seeds provide common random numbers for controlled scenario comparisons, not shared physical photons across experiments.

The exact channel's F and P are distinct from `tomography.fidelity_estimate`, the biased plug-in purity, and the bias-corrected purity estimate. Counts-only linear inversion is not clipped or projected onto a positive density matrix. Finite sampling can therefore produce a negative reconstructed eigenvalue or an estimator outside the physical interval. Those outcomes concern the estimator, not a nonphysical exact channel. The bias-corrected purity requires enough counts for every needed observable.

When no accepted detections exist, the exact conditional channel can still be calculated, but there is no tomography estimate of an observed state. A nonzero F displayed next to a zero accepted rate must be read as a conditional model prediction. An unresolved propagation result similarly supplies no observations. Increasing an assumed source rate does not turn a blocked link into a measurable one.

5.11 What validation establishes and what remains open

The retained checks include independent flat event transforms and inverse transforms; moving-endpoint light-cone residuals; transformed curved-metric scalar consistency; Schwarzschild geodesic and clock limits; full-PF seed/correction, causal-domain, inverse-map, and selection-status checks; and Qiskit results against analytic channel metrics. Orbital tests include eccentric Kepler references, circular closure, energy and angular-momentum conservation, J_2 versus a potential gradient, drag signs and limits, and chart/time Jacobians. Tracking tests exercise published SGP4 reference cases and explicit time/data failures.

Reproducible saved sessions preserve relevant source responses, timing data, configuration, hashes, software context, and results. Hash verification establishes stored-data integrity. Recomputing a saved session checks numerical reproducibility under a declared runtime and tolerance policy. Neither is an independent physical measurement.

A scientifically defensible report should state the orbital provider and age of its data, force model, frame and timescale, spacetime choice, visibility policy, emission and gate policy, phase covariance and its calibration status, exact versus sampled metric type, and unresolved statuses. Report the relevant numerical residuals alongside their meaning rather than interpreting small residuals as measurement accuracy.

Claims that remain outside this release include an empirically confirmed PF scheduling advantage; a unique full-PF emission controller; calibrated quantum noise derived from orbital geometry; operational clock-offset estimation; precise quantum-link throughput; a complete rotating multipole Earth GR model; and measured trajectory accuracy. The package is useful for defining controlled calculations, finding inconsistencies, quantifying declared model differences, and preparing hypotheses for tests with independently justified inputs.

5.12 Implementation sources for this chapter

Subject	Authoritative package files
Scenarios and shared release policy	`simulation_engine.py`; `README.md`
Legacy selection and SR scheduling	`event_timing.py`; `relativity.py`
GR scheduling, rays, and clocks	`gr_event_timing.py`; `gr_photon.py`; `orbital_models.py`; `ORBITAL_PHYSICS.md`
Full PF and source provenance	`full_pf_adapter.py`; `vendor/privileged_frame_20260913.py`; `vendor/SOURCE_PROVENANCE.json`
Tracking, coordinates, and time data	`real_time_satellite_tracking.py`; `TRACKING_MODEL.md`
Numerical forces and comparison	`numerical_orbits.py`; `comparison_workflow.py`; `NUMERICAL_ORBITS.md`; `SCIENTIFIC_COMPARISON_REVIEW.md`
Quantum channel and acceptance	`quantum_channel.py`; `channel_geometry.py`; `MODEL.md`
Sensitivity and reproducibility	`comparison_sensitivity.py`; `SENSITIVITY_ANALYSIS.md`; `SESSION_FORMAT.md`

Chapter 6

Configuration reference

6.1 How configuration is applied

The package uses UTF-8 JSON configuration files. A file describes the simulated apparatus, orbital inputs, timing policy, and optional comparison experiment. Start with the shipped preset closest to your task, copy it to a new filename, and change only the settings needed for that experiment. Preserve the original preset so that a known example remains available for troubleshooting.

JSON uses double-quoted names and strings, lowercase `true`, `false`, and `null`, and numbers without unit suffixes. Comments and trailing commas are not valid JSON. Write a microsecond as `1e-6`, rather than `"1 microsecond"`. All physical inputs use SI unless their names explicitly say otherwise. Angular orbital elements and phase noise use radians, not degrees. The main loaders reject nonfinite JSON constants such as `NaN` and `Infinity`.

Canonical and legacy aliases in the submitted file are checked for agreement before any command-line or GUI override is applied. Conflicting aliases are configuration errors; an override cannot conceal them. Once that check succeeds, an explicit override takes precedence over the corresponding valid file value. Otherwise, an explicitly supplied setting takes precedence over the implementation default. A preset is a complete example and may deliberately use a different value from the implementation default. This distinction is particularly relevant to numerical integration: the underlying numerical model defaults to `two_body` and a 60-second maximum step; comparison presets select `j2` and a 30-second maximum step.

There are two related configuration consumers. The standalone scenario runner uses top-level `steps` and `time_interval`. The full comparison workflow uses `comparison.duration_s`, `comparison.samples`, and `comparison.scenario_steps`. It constructs separate SGP4 and numerical branches, overwriting each branch's top-level sampling interval, step count, and orbit source. Editing top-level `steps` in a comparison preset therefore does not increase its photon-ensemble count. The saved comparison configuration now makes that relationship explicit: top-level `steps` equals `comparison.scenario_steps`, and `time_interval` contains the actual branch coordinate-time endpoints. The normalized root `orbit_source:sgp4` identifies the frozen reference; the paired numerical branch is still constructed and recorded separately.

All command-line configuration loaders resolve relative `tracking.cache_dir`, `tracking.leap_second_file`, and `comparison.iers_file` against the configuration file's parent directory. This rule is shared by both main runners and the retained comparison command. An omitted tracking-cache path uses `tracking_cache` under that same configuration directory. Absolute paths remain absolute. Output paths retain their separate working-directory convention. If moving a configuration into a separate folder, move the required input tree with it, update its relative paths, or use absolute paths; copying a JSON file does not copy its tracking inputs.

The execution, quantum-channel, storage-channel and numerical-force schemas reject unknown names. Other areas, including the top-level object, circular orbit profiles, and comparison options, do not provide a universal unknown-key check. A misspelled optional setting can therefore be ignored. Confirm the effective configuration in the saved manifest and metadata after every edited run. A successful JSON parse means only that the file is syntactically valid; it does not show that every intended setting was applied.

6.2 Core scenario settings

Setting	Default or accepted values	Meaning and interaction
`time_interval`	`[0, 1728000]`	Two finite coordinate-time endpoints in seconds; end must be at least start. Used by the standalone runner. The default spans 20 days.
`steps`	`30`	Positive integer number of independent photon ensembles, including evenly spaced endpoints when more than one step is requested.
`spacetime`	`flat_sr` or `gr_static`	Selects the photon, clock, coordinate-interface, and full-PF geometry branch for every requested scenario.
`propagation_model`	Legacy compatibility alias	Accepted when unambiguous, normalized to `spacetime`, then removed. If both keys are present, they must resolve to the same mode.
`orbit_source`	`circular`	`circular` supplies synthetic orbital worldlines; `sgp4` supplies tracked predictions. `numerical` is installed internally by the comparison workflow.
`timing_mode`	`selected_frame`	`selected_frame`, `eci`, or `none`; determines the physical release schedule used by the scenario engine.
`frame_velocity_m_s`	Omitted	Optional finite three-vector with magnitude strictly below the speed of light. Overrides the legacy snapshot frame selector.
`epoch_utc`	Current UTC for tracking	Explicit timezone-aware ISO epoch, preferably ending in `Z`. Circular benchmark times do not acquire UTC merely by adding this field.
`scenario3_release_policy`	`same_as_scenario2`	The only supported policy. Scenario 3 reuses Scenario 2's physical releases and receptions before applying the newer full PF construction.
`velocities`	Built-in profiles	Historical name for the three circular orbital profiles; it is not a direct Cartesian velocity array.
`quantum_channel`	Default apparatus object	Supplies polarization, collection, and finite-count settings described below.
`execution`	Shared opt-in calculation settings	Controls `sample_counts` and `controls` consistently in both main runners.
`tracking`	Required for SGP4	Supplies catalog IDs and data policy.
`pf_geometry_options`	Omitted	Advanced numerical options for Scenario 3's GR anchor mapping, not a PF emission-delay law.
`satellite_to_ground`	Absent or `false`	Legacy scenario-engine switch; `true` remains rejected on that path. Configure the separate QKD ground-link workflow in Chapter 20.

The JSON value aliases `sr` and `schwarzschild` normalize to `flat_sr` and `gr_static`, respectively. Either accepted field name may contain an alias, but new files should use only the canonical `spacetime` field and canonical value. Two fields that normalize to the same mode are accepted and collapsed to that single field. Two fields selecting different modes are rejected, including when a subsequent `--spacetime` option would otherwise replace them. A missing choice defaults to `flat_sr`. Saved effective configurations retain only `spacetime`; they do not retain a second propagation-model selector. This naming correction changes no SR or GR equation.

`selected_frame` schedules equality of the selected chart's reception times. In flat SR that chart is inertial; in GR the retained legacy timing chart is a linear change of coordinates with the curved metric transformed consistently. It is not a global inertial frame. `eci` schedules equality of base-coordinate reception times; the historical name remains in GR, where the base time is Schwarzschild coordinate time. `none` releases both partners at the common initial epoch. JSON compatibility aliases `Privileged`, `PF`, and `privileged` map to `selected_frame`; `simultaneous` maps to `none`. The older `apply_QCS_corrections` key is accepted as a compatibility alias. If both timing keys are supplied, their normalized values must agree before any override. The effective configuration retains only `timing_mode`. Old `RTT` or `round-trip` modes are rejected because they were not implemented as a measured round-trip protocol.

Migrating an earlier configuration. A legacy-only file such as `{"propagation_model": "schwarzschild"}` normalizes to `{"spacetime": "gr_static"}`. If both old and new keys exist, first decide which mode the experiment intends, retain that canonical choice and remove the obsolete alias. A file selecting `gr_static` in one key and `flat_sr` in the other is rejected even with an explicit `--spacetime` option. Likewise, `timing_mode:eci` conflicts with `apply_QCS_corrections:Privileged`. Equivalent aliases are accepted after normalization; a different physical choice is not.

Save the corrected file under a new name and run into a fresh output directory. Inspect the saved canonical `spacetime` and `timing_mode` values before comparing results. Preserve historical sealed evidence unchanged. An ambiguous captured configuration cannot be made an exact replay by guessing which old field was intended; use an explicit new configuration and a new session. Section 22.3.1 explains the same checks and the base/override/result summary in the graphical editor.

The optional frame velocity changes the legacy schedule's target chart. It does not directly set Scenario 3's new PF velocity, which is calculated from that scenario's anchored detection pair. A zero vector makes the legacy selected chart coincide with the base chart. Comparison presets use this explicit zero vector; therefore their selected-frame scheduling and base-coordinate scheduling coincide by construction. Do not infer that every arbitrary boost would produce the same simultaneity relation.

An explicit `frame_velocity_m_s` is identified as a configured boost; automatic legacy selection is identified separately. For ECI and no-delay runs, failure of the optional legacy selector does not prevent the requested physical schedule. Its selected-frame diagnostic records unavailability and the reason, rather than a fabricated boost or zero offset. A requested selected-frame calculation still requires a valid selected or explicit chart.

One epoch is an independent apparatus ensemble evaluated at an orbital snapshot. The time interval is not a continuous integration of the two-photon density matrix over days. Increasing `steps` increases the number of evaluated orbital snapshots. It does not increase the photon budget per tomography setting or reduce statistical uncertainty within one epoch.

6.3 Shared optional calculations

The top-level `execution` object is the canonical place to choose optional calculations for either main runner. Its keys must be actual JSON booleans; strings such as `"false"`, integers such as `0`, and `null` are invalid.

Setting	Default	Meaning
`execution.sample_counts`	`false`	Enables finite-count tomography alongside exact channel predictions.
`execution.controls`	`false`	Enables the additional Qiskit density-matrix channel-control calculations.

For example, the following block enables both options independently of which runner loads it:

```
{
  "execution": {
    "sample_counts": true,
    "controls": true
  }
}
```

The corresponding `comparison.sample_counts` and `comparison.controls` fields are retained as compatibility aliases and are accepted by both main runners. When a canonical field and its alias are both present, their values must agree; a conflict is a configuration error, even if a later CLI choice would otherwise override the option. New configurations should prefer `execution` and avoid redundant aliases.

An explicit CLI choice takes precedence over valid file settings. `--sample-counts` and `--controls` enable the respective calculation. `--no-sampling` (alias `--no-sample-counts`) and `--no-controls` disable it, including when the configuration enables it. Leaving both choices absent preserves the file setting; if the setting is absent everywhere, it is false. Contradictory enable and disable flags for one calculation are rejected. New effective configurations store the resolved choices canonically under `execution`, removing the duplicated legacy fields after checking agreement. Existing captured sessions retain their historical configuration bytes for replay. The run header prints `sample_counts=on/off; controls=on/off`, and run metadata records the effective choices.

These optional calculations are distinct from the noise model. Disabling count sampling removes synthetic tomography; it does not turn off phase noise or change the exact density matrix. Disabling controls omits auxiliary calculations; it does not disable independent run validation.

6.4 Circular orbit profiles

The `velocities` object must contain exactly `emitter`, `detector1`, and `detector2` when supplied. Each describes a circular orbit. Scenario 1 always replaces detector 2's worldline with detector 1's worldline, regardless of their separate entries; this is how the coincident comparison case is constructed. Scenarios 2 and 3 retain the separate detector profiles. In an SGP4 run, catalog-derived worldlines are used instead of these synthetic profiles.

Profile field	Units and accepted range	Actual effect
`altitude`	Metres, strictly positive; required	Sets radius to 6,371,000 m plus altitude.
`inclination`	Radians in $[0, \pi]$; default `0`	Tilts the orbital plane. Values below $\pi/2$ are prograde, above it retrograde.
`raan`	Finite radians; default `0`	Rotates the plane around the reference z-axis; reduced modulo 2π.
`arg_perigee`	Finite radians; default `0`	An in-plane phase reference; a circular orbit has no unique perigee.
`mean_anomaly_at_epoch_rad`	Finite radians; default `0`	Initial phase at model time zero, added to `arg_perigee`.
`eccentricity`	Only `0`; default `0`	Noncircular profiles are rejected.
`latitude`	Only `0`; default `0`	Nonzero legacy latitude is rejected as an ambiguous orbital initial condition.
`anomaly_rate`	Retained legacy metadata	Ignored by the corrected circular dynamics.
`satellite_mass`, `cross_section`, `drag_coeff`	Retained legacy metadata	Do not apply atmospheric drag or modify circular motion.

Mean motion is calculated from the central gravitational parameter, rather than the arbitrary historical `anomaly_rate`. The model uses $n = \sqrt{\mu/r^3}$, where $\mu = GM = 3.986004418 \times 10^{14}\,\mathrm{m^3\,s^{-2}}$; the returned

Cartesian velocity is the derivative of the returned position. A right ascension of π does not itself make an orbit retrograde. Change inclination to select a retrograde plane.

In `flat_sr`, these circular trajectories are prescribed accelerated worldlines in Minkowski spacetime. In `gr_static`, they are stable circular test-particle geodesics of the spherical Schwarzschild model, using areal radius and coordinate time normalized at infinity. Neither profile includes J_2, drag, maneuvers, or third-body perturbations. For numerical J_2 or drag experiments, use the full comparison workflow's `numerical_model`, rather than adding more fields to a circular profile.

The visible-link presets deliberately place the satellites close in orbital phase: emitter phase 0, detector 1 phase 0.05 rad, and detector 2 phase 0.1 rad, with common inclination 0.5 rad and RAAN zero. They are controlled examples, not catalog-derived spacecraft trajectories. The default wide-separation profiles can be Earth-blocked for much of the 20-day snapshot sequence; a zero accepted rate in those snapshots is consistent with the geometry.

6.5 Quantum channel: polarization and phase covariance

All entries in this section belong under `quantum_channel`. These parameters are shared by the selected scenarios unless the user runs separate configurations. No current rule calculates polarization noise from detector separation, path closeness, orbital frame name, or distance from the coincident scenario. The former geometry-correlation keys are rejected explicitly.

Setting	Default and range	Meaning
`bell_state`	`singlet`; also `phi_plus`	Selects both the prepared Bell state and the fidelity target.
`source_depolarizing`	`0.005`, in `[0,1]`	Two-qubit source mixture weight toward the maximally mixed state.
`phase_sigma_a_rad`	`0`, nonnegative	Standard deviation of arm A's Gaussian H/V polarization phase.
`phase_sigma_b_rad`	`0`, nonnegative	Independent marginal standard deviation for arm B.
`phase_correlation`	`null` with zero noise	Explicit covariance parameter in `[-1, 1]`; required when either phase standard deviation is nonzero.
`phase_parameter_source`	Nonempty assumption label	User-supplied explanation of the origin of the parameters. The software does not verify a calibration claim.
`deterministic_relative_phase_rad`	`0`, finite radians	Fixed polarization phase on arm A; affects fidelity but is unitary and does not itself mix the state.
`seed`	20260915, nonnegative integer	Reproducible sampling seed, offset by `100 * epoch_index` within the run.

The permitted phase-standard-deviation ceiling of `1e150` radians is only a numerical guard against floating-point overflow. It is not a physically useful operating range. Both sigmas must be finite numbers, not `null`, booleans, or arbitrary strings. The cross-covariance is $\kappa\sigma_A\sigma_B$, giving a positive-semidefinite 2×2 Gaussian covariance matrix for $-1 \leq \kappa \leq 1$. Negative covariance and the singular endpoints are supported. If one marginal variance is zero, the covariance is zero for any allowed κ and a statistical Pearson correlation is undefined; κ remains a configured parameter.

With both sigmas zero, `phase_correlation: null` is accepted and normalized to zero because the covariance vanishes. With either sigma nonzero, an omitted or null correlation is rejected: the program will not silently assume independent phases. Set `phase_correlation: 0` only when independence is the intended apparatus assumption. The legacy equal-arm `phase_sigma_rad` spelling is supported only when neither per-arm sigma appears; mixing these alternatives is an error.

The nonzero covariance presets use $\sigma_A = 0.8\,\text{rad}$, $\sigma_B = 0.5\,\text{rad}$, and $\kappa = 0.35$. Their labels explicitly state that these are illustrative and uncalibrated. A source string such as "measured on apparatus X" is provenance supplied by the operator; it is not proof that the parameters were measured correctly. A calibrated application should retain the measurement procedure, operating bandwidth, acquisition interval, units, fitted covariance, and parameter uncertainty outside the simulation as supporting records.

For the singlet, Gaussian phase averaging suppresses coherence according to the variance of the phase difference. For `phi_plus`, the relevant combination is the phase sum. Equal common-mode phases therefore have different effects on these two states. The `phi_plus` control exists partly to reveal this state dependence. The phase parameter is an H/V polarization phase, not an unobservable overall phase of the entire two-photon state.

Turning phase noise off does not set the source depolarization to zero. With the default source mixture and no deterministic phase, the exact conditional prediction is fidelity $F = 0.996250$ and purity $P = 0.992518750$. These values need not vary between orbital snapshots when the channel parameters are unchanged. With the supplied unequal-arm noisy example, the conditional singlet values are approximately $F \approx 0.865468879$ and $P \approx 0.766471722$. These are calculations under the stated channel, not experimental satellite measurements.

To run a noise experiment, replace the covariance fields in a copied preset with the following valid JSON object entries. The object below is a self-contained `quantum_channel` value; omitted apparatus fields receive their implementation defaults.

```
{
  "bell_state": "singlet",
  "source_depolarizing": 0.005,
  "phase_sigma_a_rad": 0.8,
  "phase_sigma_b_rad": 0.5,
  "phase_correlation": 0.35,
  "phase_parameter_source": "Illustrative covariance; not calibrated"
}
```

6.6 Optical bandwidth, timing gates, collection, and counts

The following entries also belong under `quantum_channel`. They describe collection and finite-count sampling. In this package they do not automatically change the conditional polarization density matrix. A lost photon or rejected timing pair reduces available observations; it is not represented by inserting a false polarization outcome into the two-qubit state.

Setting	Default and range	Meaning
`bandwidth_hz`	`5e9`, strictly positive	Single-photon Gaussian intensity spectral RMS in Hz.
`detector_jitter_s`	`1e-10`, nonnegative seconds	Per-detector Gaussian timing standard deviation.
`gate_half_width_s`	`6e-10`, nonnegative seconds	Acceptance half-width about the chosen coincidence-gate center. Zero gives zero acceptance.
`gate_reference`	`predicted_arrivals`	`predicted_arrivals`, `eci_zero`, or `fixed`.
`gate_center_s`	`0`, finite seconds	Gate center used only for `fixed`.
`gate_timing_error_s`	`0`, finite signed seconds	Residual timing-calibration offset added to the calculated gate residual.
`efficiency_a`, `efficiency_b`	Each `0.4`, in `[0,1]`	Polarization-independent arm transmission and collection factors.
`enforce_earth_occlusion`	`true`, boolean	Applies the Earth visibility mask to the accepted rate. Required in GR.

Setting	Default and range	Meaning
`emitted_pairs_per_setting`	8192, nonnegative integer	Emitted-pair budget separately for each of the nine two-qubit Pauli settings.
`source_pair_rate_hz`	1e6, strictly positive	Pair rate per source coordinate-time second used for expected accepted rates.

The timing-width convention is explicit: the single-photon temporal standard deviation is $\sigma_t = 1/(4\pi\sigma_\nu)$, where σ_ν is the configured `bandwidth_hz`, and the differential timing spread combines two such photons and two independent detector jitters. The code calculates the probability that the resulting Gaussian timing difference lies inside the symmetric gate. Its stable tail calculation can return a probability effectively zero when a narrow gate is far from the arrival difference.

With `predicted_arrivals`, the gate is centered on the calculated arrival gap, so an accurately calibrated gate can accept temporally separated detections. With `eci_zero`, the gate center is zero in the base coordinate-equivalent time tags. With `fixed`, it is the configured `gate_center_s`. The residual is the actual arrival gap minus the center plus `gate_timing_error_s`. This is an electronic association/calibration model, distinct from the release policy. Changing the gate reference does not reschedule photon emission or alter the propagation trajectory.

For a genuinely fixed gate, include `gate_center_s` explicitly even though normalization supplies zero by default. All gate widths and offsets refer to calibrated coordinate-equivalent tags, not raw asynchronous satellite clock readings. In GR, use the stated Schwarzschild coordinate-time convention and the reported clock-rate information; do not subtract uncalibrated proper-time clock readings as though they were the same coordinate.

The pair survival factor is $\eta_A\eta_B$, where η_A and η_B are `efficiency_a` and `efficiency_b`. The controlled unobstructed acceptance additionally multiplies by the gate probability. Earth-respecting acceptance becomes zero when either link is blocked. Expected rate is the source rate times the applicable acceptance probability. The efficiencies are fixed input factors: there is no automatic inverse-square loss, telescope aperture budget, atmospheric scintillation, beam divergence, or polarization-selective attenuation model hidden behind them.

In flat SR, setting `enforce_earth_occlusion` to false permits a clearly labeled controlled unobstructed collection calculation; the results still report Earth-blocked geometry separately. In static exterior GR, false is rejected because no interior-Earth propagation metric is supplied. Neither a phase-noise option nor the PF selector removes a blocked optical link.

Finite tomography draws accepted counts from a binomial distribution for each measurement setting and samples the corresponding outcomes. `emitted_pairs_per_setting` is not the accepted count, not a count rate, and not the total budget across all settings. Zero accepted counts provide no measurable tomography estimate, even though the conditional channel prediction remains defined. A missing or unresolved GR propagation solution can instead leave collection probabilities unavailable; `null` must not be interpreted as zero.

6.7 SGP4 tracking, caches, UTC, and Earth orientation

Use `tracking` when the orbit source is SGP4 or when running a full comparison. These entries are passed directly to the CelesTrak provider. Correct boolean JSON types matter: use `false`, not the string `"false"`, because the provider casts several flags to booleans.

Setting	Default	Purpose and constraint
`catalog_ids`	Required	Object mapping `emitter`, `detector1`, and `detector2` to positive NORAD catalog IDs.
`cache_dir`	`tracking_cache`	Directory for exact GP response records and acquisition metadata.

Setting	Default	Purpose and constraint
`max_age_days`	7	Positive limit on the absolute difference between requested propagation epoch and element epoch.
`allow_stale`	`false`	Explicitly permits age-limit exceedance for a labeled analysis; does not improve orbital accuracy.
`offline`	`false`	Requires valid cached records and disables automatic timing downloads.
`refresh_hours`	2	Positive maximum age of a recently downloaded cache record before online refresh is attempted.
`iers_auto_download`	`true`	Allows timing-data update when online; automatically ineffective when offline.
`allow_degraded_iers`	`false`	Opt-in acceptance of degraded Earth-orientation coverage, with provenance indicating the condition.
`data_format`	`json`	`json` for CelesTrak GP data, or `tle` for two-line elements.
`timeout_s`	20	Positive HTTP timeout in seconds.
`endpoint`	CelesTrak GP HTTPS endpoint	Must begin with `https://`; the provider adds `CATNR` and `FORMAT` query parameters.
`leap_second_file`	Automatic selection	Optional path to a current IERS leap-second table.

The default endpoint is `https://celestrak.org/NORAD/elements/gp.php`. Shipped tracking examples assign ISS catalog 25544 to the emitter, TECHNOSAT 42829 to detector 1, and HST 20580 to detector 2. These assignments illustrate the software; they do not imply that those spacecraft carry the modeled photon source, detectors, optical memories, or a compatible experiment. A physical mission requires suitable hardware, pointing, visibility, and communications beyond the catalog orbit selection.

Two different ages must be distinguished. `refresh_hours` measures time since a record was downloaded, using acquisition timestamps. `max_age_days` measures time from the elements' epoch to the requested propagation epoch. A newly downloaded record can still contain old elements. Signed element ages can be negative when the requested epoch precedes the elements' epoch; the rejection test uses absolute age. The dashboard displays actual element ages, rather than interpreting the configured seven-day threshold as a measured age.

The cache stores the exact response text, source URL, acquisition UTC, format, catalog ID, and SHA-256 hash. Offline mode requires matching, valid records for every requested role. A corrupt cache fails validation; a download failure does not silently substitute circular trajectories or stale data. Cache files use names such as `25544_json.json`. Preserve the entire record, not merely a copied orbit line, when reproducing a run.

Leap seconds and Earth-orientation parameters are distinct inputs. Leap-second tables support UTC/TAI/TT conversions; IERS Earth-orientation data support the TEME/GCRS/ITRS transformations. Supplying a leap-second file does not by itself supply Earth-orientation coverage. An explicit `comparison.iers_file` pins the supplied IERS-A table during calculation in the command-line workflows; it is honored even when the standalone scenario runner is used. Automatic Earth-orientation downloads are disabled while that explicit table is selected, and the table must cover the requested use under the configured coverage policy. The bundled timing file is checked against the requested epoch; an old bundled table is not accepted indefinitely. A package that runs offline at its captured September 2026 epoch may legitimately require refreshed data at a future date.

For online operation, the shipped live presets use `offline: false` and `iers_auto_download: true`. The provider attempts supported timing-data sources and validates requested-epoch coverage. If the network or

all valid sources are unavailable, address the reported input problem; do not bypass it by treating old data as current. The error about a leap table expiring in 2023 is a timing-input coverage error, not evidence that GR propagation or quantum calculation failed.

For reproducible historical experiments, use a captured session or the replay presets, retain their declared epoch, and leave offline mode enabled. Changing an offline fixture to today's UTC without replacing its inputs is not a valid live run. `allow_stale` labels a deliberate exploratory exception to the orbital age policy, whereas `allow_degraded_iers` concerns Earth orientation. Neither option disables leap-table expiry checks or certifies predictions as accurate.

6.8 Full-comparison settings

All entries in this section belong under `comparison`. The workflow freezes the reference GP and timing inputs, obtains a matched GCRS Cartesian initial state, and integrates the numerical branch from that same state. It compares numerical predictions with SGP4 predictions at shared UTC epochs.

Setting	Implementation default	Effect
`duration_s`	`600`	Prediction horizon in TAI/TT-rate seconds; finite in `[0, 604800]`.
`samples`	`31`	Orbit comparison sample count, integer from 1 through 10,001.
`scenario_steps`	`3`	Photon-ensemble epochs per branch and scenario, integer from 1 through 1,001.
`scenarios`	`[1,2,3]`	Nonempty unique subset of scenario numbers, sorted internally.
`sample_counts`	Compatibility alias; effective default `false`	Alias of `execution.sample_counts`, accepted by both runners; prefer the canonical field.
`controls`	Compatibility alias; effective default `false`	Alias of `execution.controls`, accepted by both runners; prefer the canonical field.
`sensitivity_enabled`	`false`	Runs configured one-at-a-time probes, or the four defaults if no list is supplied.
`sensitivity`	Omitted	Optional list of explicit perturbation specifications; an empty list requests no variants.
`iers_file`	Omitted	Explicit IERS-A table; resolved relative to the configuration and honored by the comparison, standalone and retained-comparison loaders.

Orbit samples and photon ensembles use independent evenly spaced grids. With duration 600, samples 31, and scenario steps 3, orbit rows occur every 20 seconds, while photon ensembles occur at 0, 300, and 600 seconds. Selecting the 100-second orbit row in the dashboard therefore correctly shows no photon ensemble. Increase `scenario_steps` only if more ensemble epochs are needed; it can materially increase GR and full-PF compute time.

One sample or one scenario step selects the starting epoch. A zero horizon is permitted, although multiple requested samples then repeat the same time. GR photon branches convert the comparison horizon to TCG-rate seconds so their UTC endpoints agree with the TAI/TT-rate orbital sampling. A tiny numerical difference between `time_s` and `elapsed_seconds` is consequently not sufficient to declare a mismatched epoch; inspect the UTC labels and coordinate conventions. In a new saved effective configuration, the photon `time_interval` is `[0,duration_s]` for flat SR and $[0, \text{duration\_s}/(1 - L_G)]$ for static GR, using the package's TCG–TT rate

convention. These are the settings actually used by the photon branches, not an unrelated interval inherited from the starting preset. Top-level `steps` likewise records the resolved `scenario_steps` value.

`sample_counts`, `controls`, and `sensitivity_enabled` must be actual JSON booleans. Sampling and controls follow the shared execution rules and both have explicit CLI disable forms. Sensitivity has an enable-only CLI switch; set `sensitivity_enabled` to `false` in a copied configuration when it must be disabled. Exact replay uses the captured configuration and prohibits simulation overrides, because changing the assumptions would create a new experiment rather than a replay.

6.9 Numerical force and integrator settings

These entries belong under `numerical_model`. They affect the matched-state numerical orbit branch, not the SGP4 reference. Selecting `gr_static` does not convert these Newtonian force equations into an exact general-relativistic ephemeris. It changes the photon/clock interface and anchor mapping while preserving the documented numerical orbital approximation.

Setting	Default and domain	Meaning
`model`	`two_body`	`two_body`, `j2`, or `j2_drag`.
`mu_m3_s2`	`3.986004418e14`, positive	Central gravitational parameter in $m^3\,s^{-2}$.
`earth_radius_m`	`6378137`, positive	J_2 reference radius, numerical exclusion surface, and altitude reference for the drag table.
`j2`	`1.08262668e-3`, nonnegative	Dimensionless oblateness coefficient, active for J_2 models.
`axis_gcrs`	`[0,0,1]`	Nonzero finite vector normalized internally; fixed symmetry/rotation axis in GCRS.
`rotation_rate_rad_s`	`7.292115e-5`, nonnegative	Corotating atmospheric angular speed used by drag.
`rtol`	`1e-11`	Relative integrator tolerance in `[2.3e-14,1e-3]`.
`atol_position_m`	`1e-4`, positive	Absolute tolerance for each position component, metres.
`atol_velocity_m_s`	`1e-7`, positive	Absolute tolerance for each velocity component, m/s.
`max_step_s`	`60`, positive	Maximum internal integration step in seconds. Presets use 30.
`spacecraft`	Required for drag	Per-role physical coefficients, detailed below.
`atmosphere`	Required for drag	Explicit bounded density table and provenance.

The numerical provider uses SciPy's DOP853 adaptive integrator and dense output. `max_step_s` is an internal integration cap, not the interval between output samples. A 30-second cap and 20-second reporting grid are compatible. Tighter tolerances can test numerical convergence; they do not make uncertain initial ephemerides or an incomplete force model more accurate. Distinguish integration error from model disagreement and real orbit uncertainty.

The numerical radius 6,378,137 m differs from the 6,371,000 m spherical surface used by the circular benchmark and photon occultation model. The difference is explicitly part of the model conventions. Drag-table altitude is measured relative to the numerical force model's configured radius. Do not transfer an altitude table from a different radius convention without checking the resulting density coordinates.

`two_body` contains central Newtonian gravity. `j2` adds the fixed-axis quadrupole term. `j2_drag` additionally uses velocity relative to the specified corotating atmosphere and an explicit ballistic coefficient for each space-

craft. The fixed axis does not include a complete time-dependent Earth-orientation gravity field. Solar/lunar gravity, solar radiation pressure, high-order geopotential harmonics, maneuvers, and space-weather-driven thermospheric dynamics are not automatically added by choosing J_2.

Unknown numerical model keys are rejected. Keep a copy of the effective force configuration from the saved result whenever varying constants. Altering `mu_m3_s2` affects this orbital branch; it does not globally replace all Earth constants in photon propagation, SGP4, or the supplied PF source. Such a change is a controlled force-model sensitivity experiment, not an automatic simulation of a different gravitating body throughout the entire package.

6.10 Explicit drag and atmosphere inputs

For `j2_drag`, `spacecraft` must define every satellite role, with exactly three keys in each entry: positive `mass_kg`, positive `area_m2`, and positive `drag_coefficient`. These names differ from the historical circular-profile names. Editing `velocities.emitter.drag_coeff` does not populate `numerical_model.spacecraft.emitter.drag_coefficient`.

The `atmosphere` object accepts exactly `source`, `classification`, `density_table`, and optional `density_scale`. `source` must be a nonempty description. `classification` must be `illustrative`, `empirical_model`, or `measured`. `density_scale` defaults to 1 and must be nonnegative. A zero scale turns off the density contribution while retaining the declared drag model and its other validation requirements.

`density_table` needs at least two rows. Each row contains exactly `altitude_m` and `density_kg_m3`. Altitudes must be nonnegative and strictly increasing; densities must be positive finite values. The implementation interpolates logarithmic density between supplied altitudes and never extrapolates outside the table. Cover the complete numerical propagation path, including solver trial steps, rather than only the listed output epochs.

The illustrative drag preset supplies densities from 200 km to 1,000 km and labels both its atmosphere and spacecraft specifications as software-test assumptions. Its coefficients must not be described as measured values for ISS, TECHNOSAT, or HST. For an empirical profile, identify the atmospheric model, epoch, solar and geomagnetic inputs, altitude convention, and extraction method. A label alone is not a calibration. For an observed profile, retain measurement provenance and uncertainty separately.

An out-of-range density error is preferable to silent extrapolation. To resolve one, supply a justified wider table, shorten the forecast horizon, or choose a different supported force model. Reducing the maximum step can help avoid an excessively large trial excursion near a hard boundary, but does not justify inventing densities or extending a model beyond its physical validity.

6.11 Sensitivity experiment specifications

Each element of `comparison.sensitivity` describes one finite additive perturbation around the unchanged baseline. It requires `name`, `parameter`, and a finite nonzero `delta`. The name must be unique, 1-80 characters, start with a letter or digit, and otherwise contain only letters, digits, underscores, periods, or hyphens. At most 100 variants are accepted.

Parameter	Additional fields	Delta units and restrictions
`initial_position_m`	`role`, `axis`	Metres added along a Cartesian or initial RIC axis.
`initial_velocity_m_s`	`role`, `axis`	m/s added along the chosen axis.
`j2`	None	Dimensionless additive coefficient; active `j2` or `j2_drag` required; result nonnegative.
`drag_density_scale`	None	Dimensionless additive multiplier; active drag required; result nonnegative.

Parameter	Additional fields	Delta units and restrictions
drag_coefficient	role	Additive dimensionless value; active drag and explicit baseline required; result positive.
area_m2	role	Square metres added to explicit drag area; result positive.
mass_kg	role	Kilograms added to explicit mass; result positive.
phase_sigma_a_rad, phase_sigma_b_rad	None	Radians added to marginal phase sigma; result within supported nonnegative range.
phase_correlation	None	Additive change within [-1,1]; explicit baseline correlation required.
gate_timing_error_s	None	Signed seconds added to gate calibration residual.

role is emitter, detector1, or detector2. axis is x, y, z, radial, along_track, or cross_track. RIC directions are calculated from the baseline initial state and held fixed for the perturbation: radial points outward, cross-track follows orbital angular momentum, and along-track is cross-track crossed with radial. For an eccentric orbit, this transverse direction is not necessarily parallel to the complete velocity vector.

Every variant starts from the baseline, rather than accumulating earlier perturbations. Reference GP inputs and the epoch remain fixed. The quantum baseline for a new sensitivity run is the current normalized quantum_channel configuration. A retained comparison_quantum_input provenance copy is reused only when it normalizes to the same channel; after a channel edit it is refreshed. Thus an old provenance copy cannot silently replace a newly configured phase or detector setting. As elsewhere, the equal-arm phase alias is normalized before an explicit per-arm command override; mixing incompatible legacy and current phase fields remains a configuration error. Each variant's saved numerical_model records its actual force-model settings, including a perturbed J2 or drag input; its saved quantum inputs likewise describe that variant rather than the baseline. Initial-state probes deliberately break exact initial equality by their declared amount; the unperturbed baseline remains matched. Names do not determine physics: the delta is always additive. A density scale of 1 with delta 0.1 becomes 1.1; if the original scale were 2, the same delta would produce 2.1, not a ten-percent increase.

The omitted-list default runs four probes: emitter radial position plus and minus 1 m, and detector 1 along-track velocity plus and minus 0.001 m/s. The supplied drag preset instead requests a 0.1 density-scale addition and a 1 m emitter radial addition. These are illustrative input probes, not measured uncertainties, posterior distributions, Monte Carlo ensembles, or confidence intervals.

This object can replace the sensitivity list in a copied comparison preset:

```
[
  {
    "name": "emitter_radial_plus_1m",
    "parameter": "initial_position_m",
    "role": "emitter",
    "axis": "radial",
    "delta": 1.0
  },
  {
    "name": "gate_bias_plus_50ps",
    "parameter": "gate_timing_error_s",
    "delta": 5e-11
  }
]
```

Set `sensitivity_enabled` true or add `--sensitivity` to execute the list. A phase-sigma probe that introduces nonzero noise also requires an explicit baseline correlation; normalization of an originally null correlation is not treated as a user's independence assumption. Variant phase labels and density labels are marked illustrative, preserving the original source as provenance.

6.12 Advanced GR full-PF numerical controls

`pf_geometry_options` is an optional top-level object forwarded to the supplied extended inverse-exponential-map backend. It applies only to the newer full PF's GR anchor-tangent mapping. The adapter fixes the mapping method to `geodesic-log-extended`; do not supply a `method` key or copy the vendor's entire standalone static-background configuration into the package runner.

All active options must be finite and positive. Count budgets must be integers. These are scaled internal numerical controls, not directly metres, seconds, uncertainty estimates, or a calibration target. For normal end-user runs, retain the defaults and inspect failure diagnostics before changing them.

Active setting	Default	Function or additional bound
`rtol`	`1e-10`	Relative integration tolerance; `[1e-13,1e-5]`.
`correction_atol`	`1e-18`	Absolute tolerance on scaled correction variables; `[1e-23,1e-10]`.
`refinement_factor`	`0.01`	Tightens the refinement solve; `[0.001,0.5]`.
`max_log_step`	`0.35`	Maximum step in transformed integration parameter; at most 1.
`path_step_fraction`	`0.2`	Local path-step guard; at most 0.25.
`max_endpoint_relative`	`1e-9`	Relative endpoint acceptance limit; at most `1e-3`.
`max_refinement_relative`	`1e-8`	Relative refinement/branch agreement threshold; at most `1e-3`.
`max_invariant_relative`	`1e-8`	Relative invariant-drift limit; at most `1e-3`.
`max_jacobian_condition`	`1e6`	Endpoint Jacobian condition-number ceiling.
`min_jacobian_singular_value`	`1e-8`	Lower singular-value guard on the endpoint map.
`continuation_steps`	`4`	Initial source-mass continuation budget.
`max_continuation_steps`	`64`	Maximum continuation attempts; at most 1,024 and not below the initial budget.

Active setting	Default	Function or additional bound
`max_newton_iterations`	12	Endpoint-correction iteration budget.
`max_integration_steps`	20000	Per-integration step budget; at most 100,000.
`min_exterior_fraction`	1e-8	Guard against entering an unresolved near-horizon exterior margin.
`max_path_dynamic_range`	1e14	Maximum supported path-to-anchor scale ratio.

The extended validator also accepts inherited `atol` (default `1e-12`), `max_step` (0.125), and `max_nfev` (80), but the currently selected extended backend uses `correction_atol`, `max_log_step`, and its Newton/continuation budgets instead. Likewise, `max_spatial_fraction`, `max_time_fraction`, and `max_horizon_fraction` are compatibility names ignored by this backend. Changing these inherited names should not be presented as an effective tuning experiment.

The direct Schwarzschild photon solver has its own internal Python-function controls: the normal scheduling path uses ray relative tolerance `1e-12`, reception-time tolerance 2×10^{-14} s, 65 saved ray samples, and a one-second maximum release hold. These are not exposed as JSON or CLI controls in the package runner. `pf_geometry_options` does not change them. Adding invented top-level keys such as `ray_rtol` will not configure the scheduler. API-level experimentation requires a deliberate code change and renewed validation.

Tighter residual thresholds test internal numerical consistency within the chosen exterior metric. They do not establish an accurate satellite clock model, global curved-spacetime simultaneity, global uniqueness of the logarithm, or a newly discovered emission controller. A domain-excluded or uncertified full-PF status is an explicit result. Do not weaken a guard merely to turn that status into `selected`.

6.13 Catalog of shipped configuration presets

The following table lists the mission and comparison presets. Separate published-reference benchmark presets are documented in Chapter 21. The circular and tracking presets are intended for the standalone scenario runner; the six comparison presets add the full workflow's configuration blocks. Verification subdirectories also contain captured effective configurations used as evidence. They are historical run inputs, not additional current defaults.

Preset filename	Model and noise	Intended use
`config.json`	Circular SR; phase off	Original wide-separation geometry, 30 epochs over 20 days.
`config_illustrative_noise.json`	Circular SR; sigmas 1/1 rad, correlation 0	Same broad snapshot interval with illustrative independent noise.
`config_visible_links.json`	Circular SR; phase off	Visible-link geometry, 10 epochs over 60 s.
`config_visible_covariance.json`	Circular SR; 0.8/0.5 rad, correlation 0.35	Same visible-link family with explicit illustrative covariance.
`config_gr_visible.json`	Circular GR; phase off	Visible-link GR example, 5 epochs over 60 s.
`config_gr_covariance.json`	Circular GR; 0.8/0.5 rad, correlation 0.35	Visible-link noisy GR example, 5 epochs over 60 s.
`config_storage_sr.json`	Circular SR; base phase noise off; storage enabled	Illustrative memory channel, 5 epochs over 60 s; see Section 19.8.
`config_storage_gr.json`	Circular GR; base phase noise off; storage enabled	Same memory example with static-GR propagation and clocks; see Section 19.8.
`config_tracking_replay.json`	Offline SGP4 SR; phase off	Fixed capture at 2026-09-16 17:13:22.933650000 UTC.

Preset filename	Model and noise	Intended use
`config_live_tracking.json`	Online SGP4 SR; phase off	Current tracking example; one ensemble per live update.
`config_live_covariance.json`	Online SGP4 SR; illustrative covariance	Live tracking with noise already enabled in the file.
`config_live_gr.json`	Online SGP4 GR; phase off	Static-curved live propagation example.
`config_live_gr_covariance.json`	Online SGP4 GR; illustrative covariance	Static-curved live propagation and noise.
`config_comparison_replay.json`	Offline comparison SR/J_2; phase off	Default full comparison; 600 s, 31 orbit samples, 3 ensemble epochs.
`config_comparison_gr_replay.json`	Offline comparison GR/J_2; phase off	Same frozen reference and horizon with the GR photon/PF branch.
`config_comparison_gr_noise.json`	Offline comparison GR/J_2; illustrative covariance	Noisy frozen comparison, 600 s/31 samples/3 ensemble epochs.
`config_comparison_live.json`	Online comparison SR/J_2; phase off	Live forecast segments, 60 s/7 samples/1 ensemble epoch.
`config_comparison_live_gr.json`	Online comparison GR/J_2; phase off	Corresponding GR live forecast segments.
`config_comparison_drag_illustrative.json`	Offline comparison SR/J_2/drag; phase off	Explicit illustrative drag table and two enabled sensitivity probes.
`config_qkd_intersatellite.json`	BBM92 with two satellite receivers	Configurable entangled-pair QKD forecast; see Chapter 20.
`config_qkd_ground.json`	BBM92 with two ground stations	Dual downlinks with explicit optical and detector factors; SR or static-GR override.
`config_qkd_decoy.json`	Single-arm decoy BB84	Signal, weak-decoy and vacuum pulse forecast; all Z intensities retained.
`config_qkd_micius_type.json`	Illustrative Micius-type apparatus	Ground geometry and channel assumptions; not a reconstructed historical pass.
`config_qkd_satquma.json`	Decoy BB84 with selectable SatQuMA-compatible security	Mission forecast using the reference-matched finite-key model; see Section 21.6.

The actual filenames are authoritative; do not infer a nonexistent `config_comparison_live_noise.json` or automatically load another file based on a naming pattern. To add noisy live comparison, copy `config_comparison_live_gr.json` or use its phase CLI overrides with an explicit correlation and source label.

All full-comparison replay presets use epoch `2026-09-16T18:32:00Z`, the cached capture under `tracking_fixtures/capture_20260916`, the bundled leap-second file, and the captured IERS-A file under `verification/tracking/timing_inputs`. They are intentionally reproducible historical examples. The live comparison presets omit a fixed epoch and use online data with the separate `tracking_cache` directory. The tracking-only replay preset uses a different historical epoch and should not be expected to reproduce the full comparison's visibility or residual plot.

The 19 retained non-QKD presets use the singlet Bell target, source depolarization 0.005, predicted-arrival gates, efficiencies 0.4 in each arm, and Earth occlusion enabled. All except the explicitly noisy presets have zero phase sigmas. Finite-count sampling and channel controls default off in both runners; enable them in `execution` or with `--sample-counts --controls` when wanted. These switches concern workload and sampled estimates, not the definition of fidelity or purity. QKD has its own protocol count mode under `qkd.security.count_mode`; the QKD presets specify their own source, ground and optical assumptions in Chapter 20.

6.14 A reliable configuration-editing procedure

First choose whether the task is a synthetic scenario, an SGP4 scenario, or a full model-versus-reference comparison. Copy a corresponding preset, preserving its input path conventions. Change the physical assumptions in the appropriate nested object, rather than relying on a filename or an obsolete field. Keep covariance provenance and drag provenance explicit whenever their values are illustrative.

Next run a short example with a new output directory. For a scenario, use a small positive `steps`; for a comparison, shorten `duration_s` and reduce `samples` and `scenario_steps`. Check the reported spacetime, orbit source, timing mode, quantum parameters, epoch, element ages, and any propagation or PF status before extending the horizon. A short run at one epoch cannot prove visibility throughout an orbit, but it can expose a bad path, wrong type, ignored legacy setting, or incompatible noise assumption.

Finally inspect the saved effective inputs and validation report. Keep the original configuration with its results or preserve a frozen comparison session. When reporting a comparison, state both the reference-data epoch and the simulation epoch, the actual force model, quantum assumptions, and whether the result is conditional or count-supported. Reproducibility depends on these inputs together; reproducing a chart's appearance alone is insufficient.

Chapter 7

Reading results and preserving evidence

The package produces numerical predictions, optional synthetic observations, diagnostic checks, and provenance records. These serve different purposes. The density matrix predicts a conditional polarization state; sampled counts imitate finite data acquisition; propagation diagnostics test the calculation; manifests identify the inputs and saved outputs. A useful analysis keeps these layers distinct and records the configuration that connects them.

Begin every review with the session status and propagation status. Then inspect the physical assumptions, acquisition provenance, accepted count rate, and quantum quantities. A small residual can demonstrate a well-solved equation without establishing accurate satellite ephemerides. High predicted fidelity can coexist with zero detected photons. A successful full-PF selection reports a property of the supplied event pair; it does not establish a new emission controller or an observed improvement in quantum-state quality.

7.1 Find the right output family

There are two main output layouts. The established simulation launcher writes scenario files directly into its output directory. The matched-orbit comparison workflow adds an outer session containing two complete simulation branches: the frozen SGP4 reference and the independently propagated numerical model. It can also include sensitivity variants and exact replay results.

File or folder	What it contains	How to use it
`comparison.json`	Matched-orbit rows, concise scenario rows, differences, sensitivity summaries	Main entry point for analysis and dashboard display
`session_manifest.json`	Session lifecycle, captured-input hashes, runtime and artifact hashes	Establish which run, inputs, and outputs belong together
`inputs/config.json`	Captured computation configuration	Recover the actual experiment settings
`effective_config.json`	Full effective configuration in each scenario-output folder	Inspect canonical settings, applied overrides, and branch-specific inputs
`inputs/gp/`	Exact acquired GP records	Preserve the orbital reference used in this session
`inputs/earth_orientation.ecsv`	Effective Earth-orientation table with units	Preserve coordinate-conversion inputs
`inputs/Leap_Second.dat`	Selected leap-second list and original validity limit	Preserve UTC-related timing inputs
`inputs/effective_erfa_leaps.json`	Effective ERFA leap table	Preserve the actual process-level time conversion state
`reference/`	Scenario results using frozen SGP4 worldlines	Inspect the reference prediction branch
`numerical/`	Scenario results using the Cartesian numerical model	Inspect the independent force-model branch
`sensitivity/<case>/`	Full numerical results for a requested perturbation	Trace a sensitivity summary to its detailed calculation
`replay_validation.json`	Exact equality results from an offline recomputation	Available after a replay, rather than every ordinary run

Within each scenario-output folder, `scenario1.json`, `scenario2.json`, and `scenario3.json` contain the requested scenarios. `run_manifest.json` binds the selected result files to a run identifier. `validation.json` records their independent numerical audit. The original simulation launcher also creates `comparison.png` unless plots are disabled. The matched comparison workflow uses the dashboard and does not automatically create that static image. The plotting command is given in Section 11.1; Section 13.9 covers replotting and independent validation.

Normal simulation runs and comparison branches save their actual effective configuration as `effective_config.json`. Consult this file to recover the canonical settings and applied overrides used by that branch. The run manifest's `effective_config_sha256` binds the exact saved file bytes; validation rejects a missing or changed snapshot when this field is present. The existing `config_sha256` retains its canonical-JSON-content meaning and is not the file's byte hash. Older manifests without the new field remain readable, but do not provide this additional snapshot-integrity check. These hashes establish consistency with the recorded manifest, not independent authenticity or scientific validity.

A completed comparison session normally has outer status `completed`, branch run status `complete`, and validation status `passed`. These spellings deliberately differ. During execution, a branch can be `running` or `results_written`; its validation can be `not_completed`. A partially populated folder is useful for diagnosis but is not a completed result set.

Keep an original completed session unchanged. Put exported analyses and annotations in a separate directory. The dashboard and both independent validator commands are read-only by default. A validator prints its new audit to standard output; request an external `--report` filename to retain it. The destination guard prevents overwriting a stored report or inserting one into a captured session. Initial `validation.json` reports are still written during run completion and then included in the session record.

7.2 Understand the schemas before importing data

The current comparison and session-manifest formats have their own `schema_version` of 1. Current scenario results use `metadata.schema_version` equal to 5 with storage disabled, or 6 with storage enabled. Schema 6 includes proper-storage-time and channel audit records; see Chapter 19. Retained schema-4 records remain auditable. Do not assume that matching filenames imply matching schemas across historical package versions. In particular, an old geometry-derived phase correlation or an obsolete delay offset is not interchangeable with the current explicit covariance and event-based timing records.

JSON `null` means unavailable or not computed. It does not mean zero. In Python it becomes `None`; in the dashboard it generally appears as an em dash or an explicit unavailable message. Do not replace missing arrival times, unknown probabilities, or undefined PF velocities with zero during spreadsheet import. Doing so can turn a failed or inapplicable calculation into apparent perfect simultaneity.

Use the unit suffixes in field names. Positions ending in `_m` are metres, velocities ending in `_m_s` are metres per second, times ending in `_s` are seconds, and angular quantities ending in `_rad` are radians. Some arrays have an order declaration next to them. Preserve that declaration when converting them into tables.

Chapter 8

Orbit comparisons and physical interpretation

8.1 The main comparison document

`comparison.json` is the most compact record of a matched-state experiment. Its `epoch_utc` is the common initial physical epoch. `spacetime` selects the photon/timing/PF treatment. `model_config`, `model_provenance`, `reference_provenance`, and `quantum_assumptions` state what was actually calculated. The `interpretation` and `segment_policy` strings summarize the intended scope.

Field group	Contents	Interpretation
`initial_states`	Initial position and velocity for each role	Nominal numerical states match the reference at the common epoch
`rows`	Model and reference positions, velocities, residuals, UTCs, RIC components	Full sampled orbital comparison
`summary`	Maximum position and velocity difference per satellite	Maxima over stored orbit samples, not uncertainty bounds
`scenarios`	One concise row per orbit source, scenario, and photon epoch	Joins orbital predictions to photon and quantum outcomes
`scenario_effects`	Numerical-minus-reference changes at matching scenario UTCs	Changes caused by the orbit branch under otherwise retained assumptions
`sensitivity`	Independent input perturbations and their results	Controlled experiments around the numerical baseline

The reference fields are named `tracked_position_m` and `tracked_velocity_m_s` for historical compatibility. They are SGP4 ephemeris predictions derived from published GP data, not measured telemetry. The model fields are the numerical force-model predictions. The signed Cartesian residual is always model minus reference. Its Euclidean norm discards direction and is nonnegative.

Both orbit branches start at the same GCRS position and velocity. Consequently, the nominal residuals at the initial epoch are zero by construction. Their later separation tests differences between propagators, force assumptions, and time evolution. It does not show that either trajectory is the actual satellite path. A deliberately perturbed initial state in a sensitivity case is an exception to the initial zero.

8.2 Read radial, in-track, and cross-track residuals

`position_difference_ric_m` stores three signed components in the order radial, in-track, cross-track. The reference position defines the outward radial direction. The reference angular momentum defines the orbit-normal cross-track direction. Their cross product defines the transverse in-track direction. For an eccentric orbit, in-track is not necessarily exactly parallel to velocity.

A positive radial residual places the numerical satellite farther outward along that instantaneous reference basis. A negative in-track residual places it behind in the defined transverse direction. These components are coordinates of the same position difference, not three independently measured errors. Their squared sum agrees with the squared position-residual norm to floating-point precision when the basis is defined. If the reference has no usable angular momentum, `ric_status` explains the undefined basis and the components remain unavailable.

Do not infer a force error from one component alone. A changing in-track residual can follow from a small phase difference accumulated over time, while radial and cross-track components can reflect both force-model and orientation differences. A sensitivity experiment can establish how a specified input changes those patterns; it cannot by itself establish the real parameter error.

8.3 Match times by physical epoch

Orbit rows use TAI/TT-rate elapsed seconds. In `gr_static`, scenario elapsed times use the package's TCG/Schwarzschild convention. These numbers can differ slightly while describing the same UTC event. The dashboard joins scenario and orbit rows by `epoch_utc`, rather than demanding equality of their elapsed floating-point values.

The bundled `gr_noise_sensitivity` example makes this visible. Its final orbit row is at 120.0 seconds, while its final GR scenario ensemble has elapsed time approximately 120.00000008363148 seconds. Both have UTC `2026-09-16T18:34:00.000000000Z`. Treating the elapsed values as identical would be inaccurate; failing to join the records would also be wrong.

Orbit sample count and photon ensemble count are separate settings. The 120-second example has orbit samples at 0, 30, 60, 90, and 120 seconds, but photon ensembles only at the first and last physical epochs. At 30 seconds, an empty photon table means that no ensemble was computed there. It does not imply failed propagation, zero counts, or missing data from a completed calculation.

8.4 A concrete orbital reading exercise

Open `verification/full_comparison/smoke_sr/comparison.json`. This saved example spans 30 seconds and contains three orbital epochs for three roles. Its maximum position differences are approximately 0.601684 m for the emitter, 0.713600 m for detector 1, and 0.746773 m for detector 2. All initial states match exactly. Those modest differences are predictions of the two propagation models under the saved inputs; they are not an accuracy specification.

The longer saved `gr_noise_sensitivity` example spans 120 seconds. Its maxima are approximately 2.666875 m, 3.063653 m, and 3.201510 m respectively. The largest velocity difference is approximately 0.01402544 m/s, belonging to detector 1. Selecting that satellite in the dashboard changes the displayed maximum to the maximum for the selected satellite over the entire stored interval. Moving the epoch slider alone does not change the definition of that summary maximum.

Chapter 9

Scenario records, timing, and quantum results

9.1 Navigate a scenario JSON file

Each scenario file has `time_steps`, `epochs`, `fidelity_list`, `purity_list`, `PF_offsets_list`, `ECI_offsets_list`, and `metadata`. The parallel top-level lists provide convenient plotting data. The full `epochs` objects are the authoritative detailed records. Use their metadata and status fields whenever a compact list could conceal a missing result.

Epoch section	Contents	Reading guidance
`timing`	The release policy actually used and its computed links	Start here for emission holds and detection events
`selected_frame_timing`	Timing aimed at the retained selected-frame target	Compare with the actual mode; do not assume it is always selected
`eci_simultaneity_timing`	Alternative ECI-target timing calculation	A different simultaneity target can require a different schedule
`geometry`	Arrival gaps, gating, loss, visibility, and expected rates	Separates acceptance from conditional state quality
`quantum`	Density matrix, fidelity, purity, covariance, optional tomography	Separates exact channel predictions from synthetic estimates
`controls`	Optional alternate-channel calculations	Empty when controls were not requested
`initial_states_source`	Actual states supplied to this epoch	Prefer these to legacy profile labels when using tracked or numerical orbits
`initial_proper_clock_rates`	Initial clock rate relative to declared coordinate time	Diagnostic rates, not a complete detector clock-calibration model
`full_pf`	New full construction on the computed reception pair	Present in Scenario 3 only

The legacy `PF_offsets_list` name does not mean that Scenario 3 has acquired a new full-PF emission law. Scenario 3 preserves Scenario 2's release and reception events and adds the new construction diagnostically. Read `metadata.scenario3_release_policy` and `full_pf.delay_inference` before interpreting any offset list.

The roles are also important. In Scenario 1, detector A's worldline is used for both channels to create the coincident-detector case. The actual detector 2 orbit displayed in the comparison dashboard is still the detector 2 role orbit; it is not replaced in that general trajectory view. Scenarios 2 and 3 use the separate detector worldlines.

9.2 Read an emission schedule

`timing.emission_holds_s` contains nonnegative holds for channels `a` and `b`, measured from that snapshot's local time origin in the stated coordinate convention. A hold is not automatically a proper waiting time on an onboard clock. For each link, `emission_time_s`, `arrival_time_s`, and `flight_time_s` describe the physical schedule. The basic consistency relation is arrival time minus emission time equals flight time.

The emitter and detector are moving. `emission_position_m` belongs to the emission event; `detection_position_m` belongs to the solved interception event. A snapshot range divided by the speed of light is not a substitute for checking this moving-endpoint solution. Recorded transformed emission and detection events, inverse errors, and null residuals allow the coordinate calculation to be audited without confusing it with a new physical path.

Arrival gaps use detector B minus detector A. Read the label of each gap: a base-coordinate gap, selected-chart gap, and anchor-tangent PF gap are not interchangeable. In GR, the old selected-frame transformation is a linear chart transformation with its metric pulled back; it is not a global inertial Lorentz symmetry of curved spacetime. The new GR full PF acts on events mapped into the common anchor's tangent space.

A recorded residual around 10^{-17} s measures numerical satisfaction of a coordinate constraint. It is not a claim that real detectors or orbital data achieve that synchronization accuracy. Compare it with solver tolerances to assess computation, and with a separately calibrated instrument model to assess an experiment.

9.3 Distinguish a blocked ray from an unresolved calculation

For a resolved link, status `ok` means the implemented propagation problem was solved. It does not alone say that both links are collectable: inspect `earth_clear_pair` and the collection mode. Flat-SR unobstructed experiments can intentionally retain hypothetical vacuum propagation through an opaque Earth while also reporting Earth-respecting rates.

GR propagation uses an exterior Schwarzschild model. It does not provide a physical through-Earth replacement ray. An `occulted` result can therefore have unavailable emission holds and arrivals, zero accepted rate, and no reception pair on which to select the full PF. The reason field explains the direct-link restriction; the solver does not silently search a later orbital pass.

An `unresolved` result is different. It indicates that a numerical or supported-domain issue prevents a prediction. Its acceptance probability and expected rate remain unknown rather than being converted to zero. This distinction survives in JSON and should survive every analysis table you generate.

The saved `live_fresh` GR example illustrates obstruction. Its epoch is `2026-09-16T20:49:14.172157000Z`. Both orbit branches report `occulted` scenario propagation and zero accepted rate. Scenario 3 reports unavailable reception events. Its conditional fidelity remains about 0.99625 in that storage-disabled saved example, because the base channel can be specified even when the experiment yields no accepted photons. That is an expected model result, not an observation of high-fidelity entanglement through Earth.

9.4 Loss, bandwidth, gating, and accepted counts

`pair_survival_probability` describes the specified polarization-independent collection survival, multiplied by memory retrieval survival when storage is enabled. Schema 6 also retains the corresponding pre-storage quantities. `gate_probability` describes acceptance within the specified timing gate. `accepted_probability` combines the production collection assumptions, including Earth obstruction when enforced. `controlled_accepted_probability` and `earth_respecting_accepted_probability` make the alternative collection scope explicit where supported.

The expected accepted pair rate is source pair rate times accepted probability. The expected accepted pairs per setting is emitted pairs per setting times accepted probability. These are expectations, so they need not be integers. Sampled counts are integers and fluctuate around those expectations.

With a 1,000,000-pair/s source, efficiencies of 0.4 in each channel, and the saved default bandwidth, jitter, and gate, the visible example has pair survival 0.16 and gate probability approximately 0.9999720931. Its expected accepted rate is approximately 159,995.5349 pairs/s. At 8,192 emitted pairs per tomography setting, the expected accepted count is approximately 1,310.6834 per setting. The rate is expressed per the declared source-time coordinate; consult `rate_time_coordinate` before comparing different clock conventions.

Under the package's present assumptions, polarization noise and acceptance are independent. Changing the timing gate can change collected counts without changing the conditional polarization density matrix. Real

apparatus can couple polarization and acceptance, but that coupling is not inferred from a scenario name or a frame label here.

9.5 Exact fidelity and purity versus estimated quantities

`quantum.density_matrix_real` and `density_matrix_imag` combine to form the exact model density matrix. The basis and bit-order declarations specify how it is stored. The current output uses two polarization qubits with density-matrix bit order B A; do not casually swap detector labels when importing matrix entries.

Fidelity is evaluated against the declared Bell target. Purity is $P = \mathrm{Tr}(\rho^2)$. Neither is a score measuring proximity to Scenario 1. All scenarios with the same specified conditional channel can have identical fidelity and purity, even when their geometry and accepted counts differ.

With phase noise off, storage disabled, zero deterministic relative phase, and source depolarization 0.005, the saved predictions are $F = 0.99625$ and $P = 0.99251875$. Turning phase noise off does not make the source perfectly ideal. The optional ideal control also removes source depolarization and the deterministic relative phase.

For the singlet noise example, $\sigma_A = 0.8\,\mathrm{rad}$, $\sigma_B = 0.5\,\mathrm{rad}$, and $\kappa = 0.35$. Its covariance is

$$\Sigma = \begin{pmatrix} 0.64 & 0.14 \\ 0.14 & 0.25 \end{pmatrix} \mathrm{rad}^2.$$

The relative phase variance is $V_- = 0.61\,\mathrm{rad}^2$, giving coherence $C_{\Psi^-} \approx 0.7371233744$, fidelity $F \approx 0.8654688788$, and purity $P \approx 0.7664717221$. These are consequences of the explicitly supplied channel parameters. The source string describes them as illustrative, and `phase_parameter_source_verified` is false. The code does not turn an attribution string into empirical calibration.

Check `trace_real`, `minimum_eigenvalue`, `analytic_fidelity`, `analytic_purity`, and their reference errors when inspecting the exact channel. A physical density matrix has unit trace, is Hermitian, and has no materially negative eigenvalues. Small floating-point residuals should be considered against the validator's tolerances rather than read as experimental uncertainties.

9.6 Synthetic tomography

Tomography is optional. Without sampling, `tomography` is absent or null even if the expected rate is positive. With sampling, accepted counts are drawn independently for each measurement setting under the specified acceptance model; outcome counts are then drawn from the predicted polarization state. Nine combinations of the Pauli X, Y, and Z measurement settings are used.

`accepted_counts_per_setting` and `shots_per_setting` identify the actual accepted sample size. `counts` contains bit-string outcomes. `settings_order` says that the first setting letter measures A and the second measures B, while `count_bit_order` is B A. For a setting such as XX, the four outcome counts must sum to its accepted count, not to the emitted count.

The exact `fidelity` and `purity` fields remain the channel predictions. `fidelity_estimate`, `purity_plugin_biased`, and `purity_unbiased` are finite-count estimates. The plug-in purity estimate has positive finite-shot bias. The unbiased estimate removes the specified estimator bias; it does not guarantee a physical reconstructed matrix in every finite sample.

The first numerical epoch of the saved GR/noise example collected 11,824 pairs across nine settings. Its exact fidelity is approximately 0.865469, while its sampled estimate is approximately 0.875129. Exact purity is approximately 0.766472, plug-in purity is approximately 0.782473, and bias-corrected purity is approximately 0.780035. These differences are expected sampling fluctuations. The linear-inversion matrix has minimum eigenvalue approximately -0.02035; this estimator is not projected onto the positive-semidefinite set. That does not mean that the underlying simulated channel state was unphysical.

A zero accepted count cannot support a tomography estimate. Do not fill an unavailable estimate with the exact model value. Also, common random seeds across scenarios provide common random numbers for controlled

comparison, not independent real experiments. Identical count tables in otherwise identical channels are not independent confirmation of a theory.

9.7 Optional channel controls

When enabled, `controls` contains exact channel calculations named `phase_noise_off`, `ideal`, `phi_plus`, and `fixed_phase`. They are interpretive checks, not additional observed datasets. `phase_noise_off` removes both marginal phase noises and optional storage dephasing. `ideal` also removes source depolarization and fixed relative phase. `phi_plus` changes the Bell target and preparation, which changes how common phase fluctuations act. `fixed_phase` sets the deterministic relative phase to $\pi/2$ under the remaining channel assumptions. Both `phi_plus` and `fixed_phase` retain storage dephasing. These are conditional-state controls, not recalculated collection budgets.

The controls help separate random dephasing, coherent phase rotation, and source mixing. A deterministic unitary phase change can lower fidelity to a fixed target while retaining purity; therefore lower fidelity alone does not identify decoherence. An empty controls object means controls were disabled, not that a check failed.

Chapter 10

Using the comparison dashboard

10.1 Open and orient the viewer

Start the local server from the extracted project directory:

```
run_dashboard_python38.bat --sessions comparison_sessions --open-browser
```

The default address is `http://127.0.0.1:8765/`. Leave the terminal running while using the viewer; Ctrl+C stops it. To inspect the bundled evidence instead of your own sessions, supply `--sessions verification/full_comparison`. The browser assets are bundled; no account, external map service, JavaScript package manager, or web framework is required.

The viewer reads saved results. Refreshing does not run an orbit calculation, acquire a satellite observation, download new elements, or change a simulation. A separate live comparison command creates successive saved forecast segments that the viewer can discover. Every segment retains its own frozen input set and initial state.

Control	Action	Scope
Saved session	Select a discovered completed, running, or failed session	Changes the dataset being inspected
Projection	Choose XY, XZ, or YZ	Changes the two-dimensional trajectory projection only
Satellite	Show all roles or one role	Filters orbital curves, state rows, and orbital maxima
Inspect epoch slider	Select a stored orbit sample	Updates selected markers, state rows, and matching photon rows
Refresh every 5 s	Enable or pause automatic rereading	Preserves selection where possible; pauses when page is hidden
Refresh button	Request a fresh read immediately	Does not recompute the selected session

The session picker excludes nested numerical, reference, and sensitivity artifacts as independent sessions. If no sessions are found, verify the `--sessions` path and the presence of `comparison.json` or `session_manifest.json`. A bounded scan can omit deeply nested or numerous sessions; a warning instructs you to start the viewer on a more specific directory.

10.2 Read each panel

The connection badge distinguishes connecting, connected, paused, and refresh-failed states. The last-refresh time is a browser read time, not a new satellite observation epoch. If a refresh fails, existing plots can remain from the last successful read; the error message says so.

The summary shows lifecycle status, spacetime mode, reference UTC, and element-age range. Element age comes from each satellite's `signed_element_age_days`. The maximum allowed age in the configuration is an acceptance policy, not an observed age. A negative signed age means the element epoch lies later than the evaluation epoch; it is not automatically a corrupted timestamp. Stale reference flags remain explicit.

For the saved historical fixture, the actual ages range from approximately 0.580052 to 0.643720 days. The dashboard should therefore display a rounded range near 0.58 to 0.644 days, not 0.58 to 7 days. The latter would incorrectly mix the configured seven-day age limit with measured metadata.

Trajectories are GCRS coordinate projections in kilometres. Solid lines show the numerical model, dashed lines SGP4, and dots mark the selected epoch. The horizontal and vertical coordinate scales are matched within a projection. Short arcs and nearly overlapping lines are expected when both models start identically and are compared over a short interval. Switching projections can reveal a separation concealed in another view; it does not change the data or the three-dimensional residual.

Position difference and velocity difference plot Euclidean norms over elapsed orbit time. Their units are metres and metres per second. A vertical guide marks the selected epoch. The maximum summary cards refer to all stored times for displayed satellites, whereas the state table refers only to the selected epoch. Orbit charts remain GCRS in `gr_static`; that flag changes the photon branch's propagation and frame treatment, not the chart in which the comparison rows are stored.

The state-difference table provides norms and signed RIC components. It contains one row per currently displayed role at the selected time. Missing values stay unavailable.

The photon-and-quantum table shows both orbit sources and requested scenarios at the selected UTC. Its columns include propagation status, whether both links are clear, exact conditional fidelity and purity, expected accepted rate, full-PF status, selected PF time gap, emission holds, and arrivals. Full PF is not applied to Scenarios 1 and 2. A blank PF gap there is expected. The satellite dropdown does not turn this table into a different experiment; these are complete pair-based scenario summaries.

The no-ensemble message lists the available photon UTC epochs when none matches the selected orbit sample. Select one of those epochs using the slider. In the saved 120-second example, only the beginning and end have photon rows. An intermediate empty table is expected behavior.

The sensitivity panel lists requested cases, status, parameter, perturbation, and summary. It does not infer confidence intervals. If no probes were requested, its explicit empty-state message is correct. Open the full JSON or case folder for detailed rows and effects.

Four expandable sections expose original acquisition and frozen computation provenance, force-model assumptions, session lifecycle and replay metadata, and interpretation limits. Original acquisition identifies where and when inputs were obtained. Frozen computation identifies what was used in the run or replay. A saved historical session remains historical even if the dashboard is continuously refreshing it.

10.3 Printing and browser validation boundary

For a report, pause automatic refresh, select the session and epoch, choose a projection and satellite scope, and expand only the provenance sections you want printed. Use the browser's Print command and select Save as PDF. Dedicated print styles request landscape pages and rearrange chart cards into blocks; printing also suspends refresh and redraws canvases for the print layout.

Inspect the preview for chart clipping, table overlap, page breaks, and the completeness of expanded metadata. Browser print settings can override CSS page size or margins. Check the final PDF, particularly when long JSON provenance sections are expanded.

The revised viewer passed HTTP and JavaScript behavior checks, but actual revised browser rendering and corrected print pagination were not verified in the authoring environment because cloud browser access was blocked. Earlier user-supplied PDF output confirmed that the previous charts rendered and exposed a print-layout defect that was subsequently corrected in code. That earlier PDF is not evidence that the revised print layout has been visually validated. Treat local visual inspection as a remaining release check rather than a claimed completed test.

Chapter 11

Static plots and analysis recipes

11.1 Generate a four-panel quantum comparison

To plot the numerical branch of a completed session without rerunning physics:

```
.venv-pf38\Scripts\python.exe plot_comparison.py ^
  comparison_sessions\gr_noise\numerical\scenario1.json ^
  comparison_sessions\gr_noise\numerical\scenario2.json ^
  comparison_sessions\gr_noise\numerical\scenario3.json ^
  --output analysis\gr_noise_quantum.png
```

Choose an analysis folder outside the immutable session. To compare SGP4 quantum predictions, use the reference branch instead. The plotting command accepts one or more scenario files, rejects duplicate scenario numbers, and checks compatible array lengths. It reads results only; it neither samples new counts nor changes the state.

Labels come from `metadata.scenario` in each current result, so renaming files or changing their argument order does not change their scenario identity. A retained top-level `scenario` is accepted for older results. If both fields are present, they must agree. Recorded identities must be JSON integers 1, 2 or 3; Boolean values, strings and other numbers are rejected.

Only when neither identity field is present does the command use a standard `scenario1.json`, `scenario2.json` or `scenario3.json` filename, recognized without regard to letter case. A recorded identity takes precedence over the filename. A custom filename without a recorded identity is rejected instead of being numbered by argument order. Invalid, conflicting or duplicate identities stop the command before a plot is written.

For example, after copying Scenario 3 and Scenario 1 outputs into `analysis/full_pf.json` and `analysis/coincident.json`, respectively:

```
.venv-pf38\Scripts\python.exe plot_comparison.py ^
  analysis\full_pf.json analysis\coincident.json ^
  --output analysis\renamed_comparison.png
```

The plot labels these records S3 and S1 and orders the legend by scenario number. The stored fidelity, purity, rates and correlation values are preserved; this labeling correction does not change their predictions.

The four panels show conditional fidelity, conditional purity, expected accepted pair rate, and the supplied phase-correlation parameter. The horizontal axis uses seconds for short runs and days for runs reaching two days in magnitude. Connecting lines guide the eye between independent snapshot ensembles; they are not a continuous measurement of one evolving photon pair.

Flat-SR plots can show both controlled unobstructed and Earth-respecting rates. GR plots show exterior Earth-respecting rates, since there is no supported through-Earth GR continuation. Unresolved rates appear as gaps rather than invented zeroes. The correlation panel reports a supplied parameter; it does not estimate covariance from the satellite geometry. If either marginal sigma is zero, that correlation parameter does not determine an observable Pearson correlation.

Equal curves can overlap. Scenario-specific line styles and markers help identify coincident traces. Read the data or legend before concluding that a curve is missing. With a fixed channel, equal fidelity and purity across all three scenarios are expected and do not imply that propagation or scheduling were skipped.

11.2 Inspect a session with standard Python

The following examples use only the Python standard library unless stated otherwise. Save a snippet in a separate analysis file and run it using the package environment. Adjust the path to the session you want to inspect.

```
import json
from pathlib import Path

session = Path('verification/full_comparison/gr_noise_sensitivity')
data = json.loads((session / 'comparison.json').read_text())
print(data['status'], data['spacetime'], data['epoch_utc'])
for role, values in data['summary'].items():
    print(role, values['max_position_difference_m'],
          values['max_velocity_difference_m_s'])
```

To export orbit residuals for a spreadsheet, preserve both elapsed time and UTC:

```
import csv

fields = ['epoch_utc', 'elapsed_seconds', 'satellite',
          'position_difference_norm_m',
          'velocity_difference_norm_m_s']
with open('orbit_residuals.csv', 'w', newline='') as stream:
    writer = csv.DictWriter(stream, fieldnames=fields)
    writer.writeheader()
    for row in data['rows']:
        writer.writerow({key: row.get(key) for key in fields})
```

Do not label this export “satellite position errors.” “Numerical minus SGP4 position difference” preserves its actual meaning.

To inspect Scenario 3 at the final physical epoch and avoid the SR/GR elapsed-time mismatch:

```
last_utc = max(row['epoch_utc'] for row in data['rows'])
for row in data['scenarios']:
    if row['epoch_utc'] == last_utc and row['scenario'] == 3:
        print(row['source'], row['propagation_status'],
              row['full_pf_status'], row['selected_pf_time_gap_s'])
```

The saved GR/noise example should show selected PF results for both sources. A different live epoch can legitimately show unavailable PF selection when receptions are blocked or unresolved.

11.3 Reconstruct the exact density matrix

A compact matrix check can be made without NumPy:

```
result = json.loads(
    (session / 'numerical/scenario3.json').read_text())
q = result['epochs'][0]['quantum']
rho = [[complex(re, im) for re, im in zip(rr, ii)]
       for rr, ii in zip(q['density_matrix_real'],
                         q['density_matrix_imag'])]
trace = sum(rho[i][i] for i in range(4))
purity = sum(rho[i][j] * rho[j][i]
             for i in range(4) for j in range(4)).real
print(trace, purity, q['purity'])
```

This verifies the trace and purity arithmetic of the saved matrix. It does not replace the full independent validator, and a trace of one alone does not establish positive semidefiniteness. Use the installed NumPy eigensolver if you also want to check all eigenvalues.

To inspect sampled totals without confusing them with exact predictions:

```
tomography = q.get('tomography')
if tomography is None:
    print('No sampled tomography in this result')
else:
    for setting, outcomes in tomography['counts'].items():
        total = sum(outcomes.values())
        assert total == tomography['accepted_counts_per_setting'][setting]
    print(tomography['total_collected_pairs'])
    print('Exact F:', q['fidelity'])
    print('Estimated F:', tomography['fidelity_estimate'])
```

11.4 Follow a sensitivity result to its cause

Start with the perturbation metadata, not just the plotted maximum. It records the parameter, units, baseline value, new value, role, and axis where applicable. Every variant begins independently from the same baseline; perturbations do not accumulate in the order displayed.

The saved GR/noise sensitivity session has four default probes: emitter radial position plus and minus 1 m, and detector 1 transverse velocity plus and minus 0.001 m/s. In the positive emitter radial probe, the initial emitter position-residual norm is approximately 1 m by design. The nominal value is zero. The other two initial satellite states remain matched.

A perturbation can reduce the norm relative to SGP4 at a later sample. For example, the positive radial probe's emitter maximum is approximately 2.599937 m, compared with the nominal 2.666875 m. This does not establish that moving the real satellite estimate outward by 1 m is a valid correction. It shows a local response of one declared model comparison.

11.5 Publication-size vector figures

The supplied figure assets have two independently arranged sizes. Files in `figures/` have a native width of 160 mm and are placed at that width in this manual. Files with the same names in `figures/single_column/` have a native width of 85 mm. The narrow chart panels are stacked and the diagrams are rearranged to retain readable text. Select the appropriate file rather than shrinking a wide two-panel figure into a single column.

The assets retain vector lines and embedded TrueType fonts. Ordinary figure labels and legends are at least 8 points at the stated native width; mathematical subscripts can be smaller. Curve styles or markers supplement

colour. Legends are separated from data, and captions state the assumptions needed to interpret the examples. A different publication width requires another layout check; the supplied sizes do not certify compliance with an individual publisher's instructions.

```
\includegraphics[width=160mm]{figures/storage_loss.pdf}
\includegraphics[width=85mm]{figures/single_column/storage_loss.pdf}
```

To reproduce every asset, run `python generate_figures.py` in the LaTeX-source directory with NumPy and Matplotlib installed. The plotting and diagram generators are included. `figure_data/residuals.json` preserves the five original samples per satellite and records the originating saved comparison's checksum. The numerical simulation is not rerun to regenerate that figure. The analytic covariance and memory plots retain the documented formulas, parameter values and sample grids. Consult `FIGURE_GUIDE.md` for standalone captions, sizing and provenance.

Chapter 12

Worked visual analysis

12.1 Example A: separate orbital disagreement from optical state quality

Open the bundled saved comparison `verification/full_comparison/gr_noise_sensitivity`. Its nominal numerical model is J_2, its spacetime mode is `gr_static`, and its frozen reference epoch is 2026-09-16T18:32:00Z. Its five orbit sample times run from 0 to 120 TT-rate seconds, while its two photon ensembles correspond to the start and end UTC epochs. This is a deliberately short worked example; the values are not the output of every default preset or of a new current-UTC run.

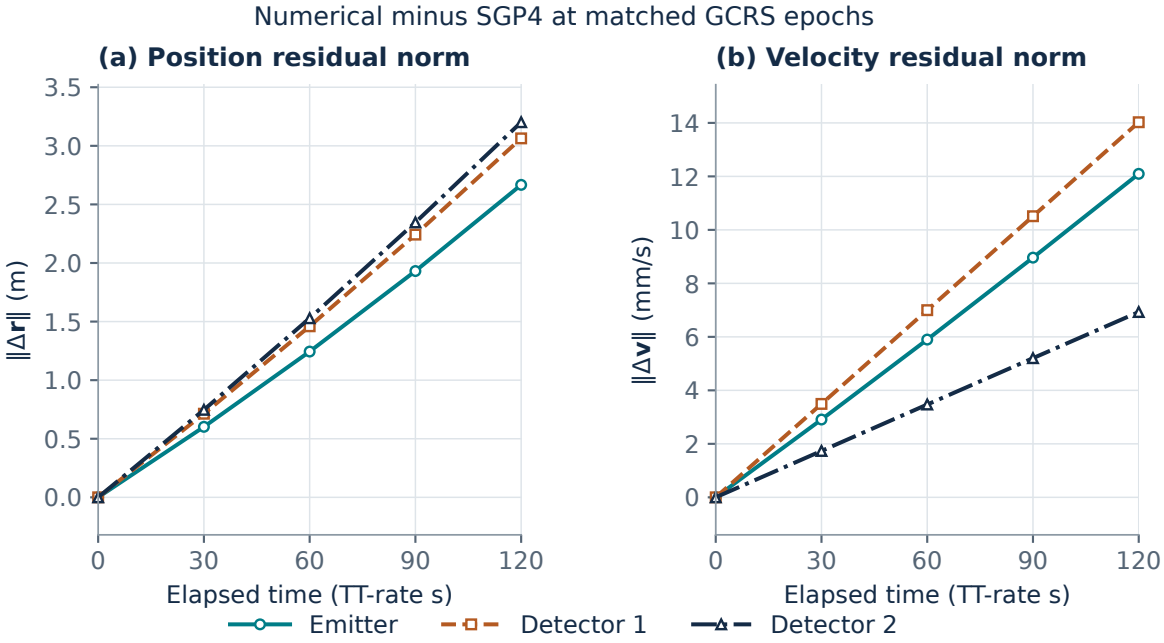

Figure 4: Numerical J_2 minus SGP4 residual norms from the saved `gr_noise_sensitivity/comparison.json`. Lines connect the five stored samples to aid reading; they are not additional sampled observations. Position differences are in metres and velocity differences in millimetres per second. The common zero at the initial epoch results from matched initialization. Distinct line styles and markers identify all three satellites in grayscale. These are differences between orbit predictions, not measured errors against the true orbit.

The final stored position norms are approximately 2.667 m for the emitter, 3.064 m for detector 1 and 3.202 m for detector 2. The maximum velocity differences over the interval are approximately 12.095, 14.025 and 6.928 mm/s respectively. This ordering shows why it is useful to inspect both position and velocity: the satellite with the largest position disagreement need not have the largest velocity disagreement.

The appropriate conclusion is that these two propagation models produce different predictions from a common initial state over this interval. A claim that the numerical satellite is 3.202 m away from its true position would require independent position measurements. A claim that SGP4 is more accurate merely because it is the reference would also be unsupported.

Now inspect the endpoint scenario summaries at `2026-09-16T18:34:00.000000000Z`. The saved GR photon elapsed value is about `120.00000008363148` seconds, not exactly 120. Join using the UTC label. All six branches have the same declared noisy channel, so their exact conditional metrics are approximately $F = 0.865468879$

and $P = 0.766471722$ even though their trajectories and event times differ. Both links are clear and the calibrated gate policy yields about 159,995.535 accepted pairs per modeled Schwarzschild coordinate second in this example.

The orbital and optical findings can coexist. A changed trajectory affects geometric timing. Fixed optical covariance leaves the conditional density matrix unchanged. Predicted-arrival gate calibration can preserve high acceptance. Inspect the arrival differences and release-hold differences in `scenario_effects` to quantify the timing effect instead of trying to infer it from an unchanged fidelity number.

12.2 Example B: use signed components to explain a norm

At the same endpoint, select detector 2 in the dashboard. Its displayed radial, along-track and cross-track position differences are approximately 0.231, -2.651 and -1.780 m. The Euclidean norm of those orthonormal components is about 3.202 m, matching the position residual norm. The negative along-track and cross-track components indicate directions in the reference orbit's instantaneous basis; they do not mean a negative distance.

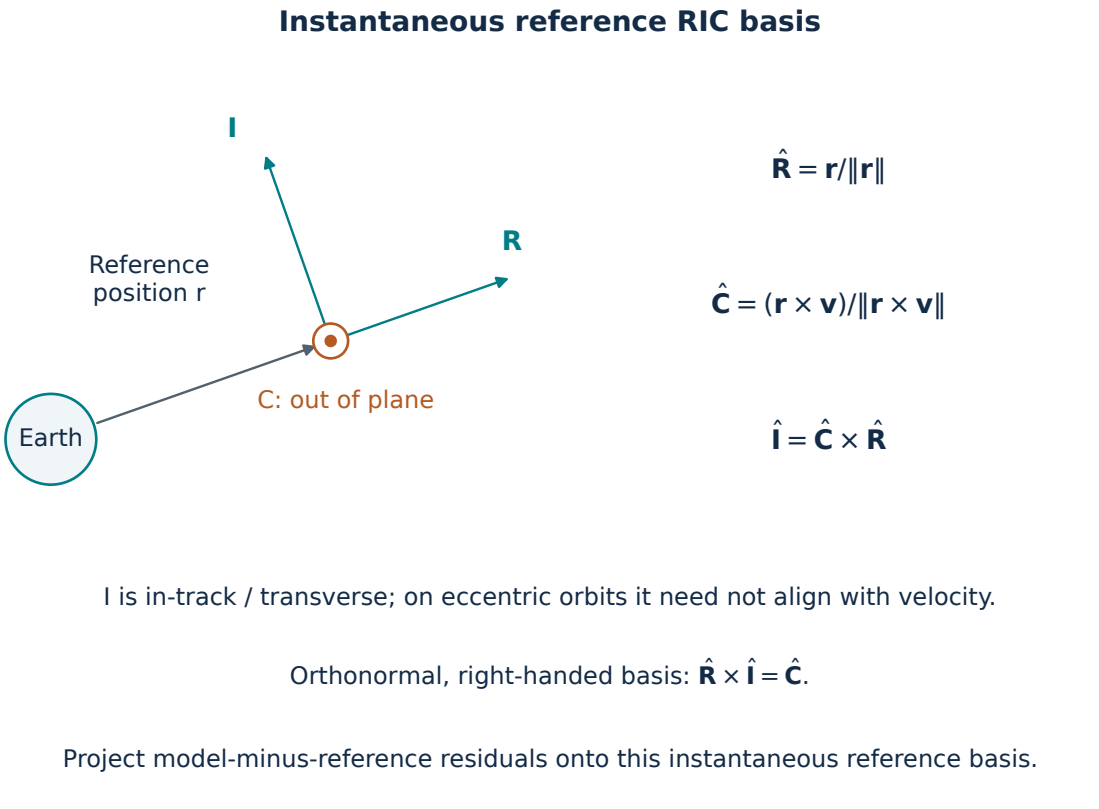

Figure 5: Definition of the instantaneous reference RIC basis. This diagram is schematic, not the actual saved satellite geometry. R is radial, C is normal to the plane defined by the reference position and velocity, and $\hat{\mathbf{I}} = \hat{\mathbf{C}} \times \hat{\mathbf{R}}$ completes the right-handed basis. The arrows denote unit-basis directions; the sketch is not to scale. For an eccentric orbit I is not generally parallel to the full velocity vector because that velocity can have a radial component.

A norm-only chart hides sign and can conceal a change in the dominant direction. For analysis, retain the three signed components together with the norm and UTC epoch. When comparing sensitivity variants, use the baseline-defined perturbation direction specified in the record; do not silently reinterpret it as a newly rotating direction at every later point.

12.3 Example C: distinguish a missing ensemble from a failed link

In the saved `drag_sensitivity_fixed` session, the orbit samples are at 0, 30 and 60 seconds. Photon ensembles exist only at 0 and 60 seconds. Selecting the 30-second sample should show the explicit no-ensemble message and list the available photon UTCs. It should not show six fabricated scenario rows or retain rows from the previous epoch.

A missing ensemble is a sampling decision. An `occulted` propagation result is a physical visibility classification within the model. An unresolved propagation result is a numerical or domain limitation. A zero accepted rate can arise from visibility or gate rejection. These are different situations and require different responses. Increasing `scenario_steps` can address sparse photon sampling; it cannot make an opaque-Earth ray physically visible or repair an invalid coordinate conversion.

For a concrete blocked-link example, inspect `verification/full_comparison/live_fresh` at its initial epoch. The saved session is a completed GR comparison, but its six scenario summaries report an occulted pair (at least one arm blocked) and zero accepted rate. Scenario 3 reports `reception_events_unavailable` rather than a selected PF. The ISS, TECHNOSAT and HST catalog assignments are illustrative orbital roles; no quantum hardware aboard these spacecraft is asserted. The phase-noise-off conditional F and P remain calculable because they describe a specified channel conditioned on collection, not an observed state from those blocked links.

12.4 Example D: interpret phase correlation independently of scenario labels

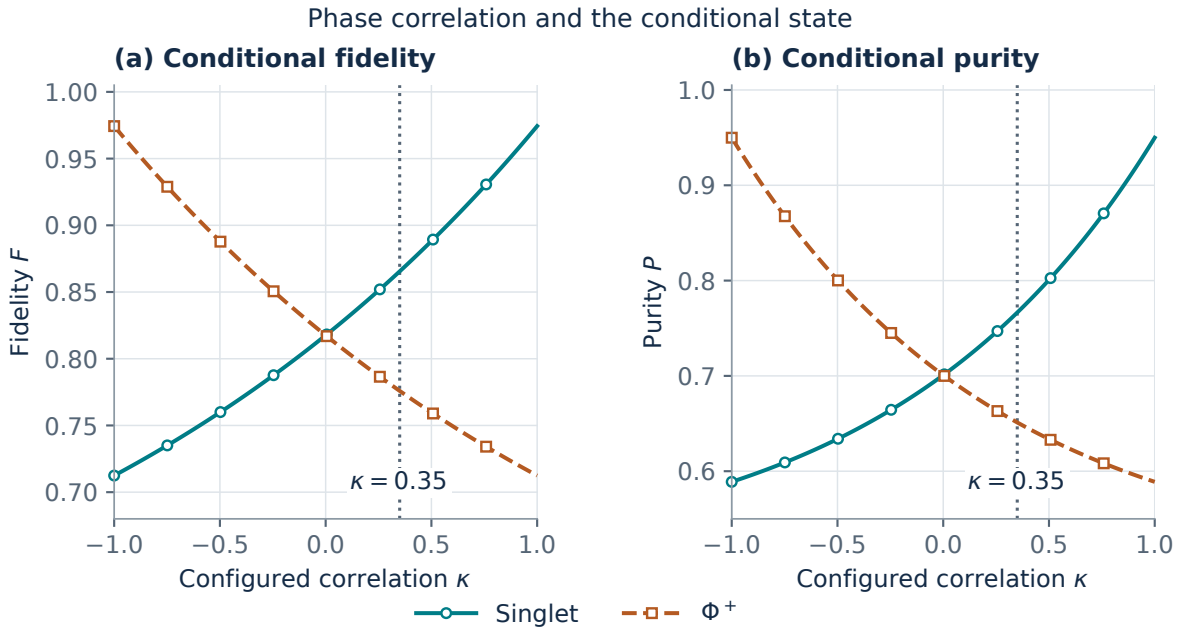

Figure 6: Analytic channel response for $\sigma_A = 0.8\,\text{rad}$, $\sigma_B = 0.5\,\text{rad}$, source depolarization $p = 0.005$ and zero deterministic phase. Solid and dashed curves distinguish the singlet and $|\Phi^+\rangle$. The labelled dotted line marks the configured correlation $\kappa = 0.35$. The curves come from the package's analytic Gaussian-channel equations; they are not measured satellite data, a fit to path separation, or an automatically generated package output.

For the singlet, stronger positive correlation reduces the variance of the relative polarization phase. For $|\Phi^+\rangle$, the relevant coherence depends on the phase sum, so the correlation dependence reverses. This difference follows from the state and noise coupling. It cannot be explained by a universal rule that coincident geometry always has higher fidelity than separated geometry.

The values plotted above would apply equally to any scenario supplied with those same channel settings. Changing only a frame label must not move a point along the correlation axis. A different covariance is a

different apparatus assumption and must be identified as such. To study a physical path dependence, first justify a model connecting the optical environment and polarization coupling to that covariance, then calibrate or independently constrain its parameters.

A deterministic phase experiment answers a different question. Change `deterministic_relative_phase_rad` while holding the random covariance fixed: fidelity to the chosen target can change while purity remains unchanged. This is a useful way to learn why fidelity and purity should be reported separately.

12.5 Example E: compare a nominal result with a sensitivity probe

The saved GR example includes positive and negative one-metre emitter radial perturbations and positive and negative 0.001 m/s detector-1 along-track velocity perturbations. Every perturbation is a distinct controlled numerical run around the nominal initial state, with the same frozen reference data. The sign and units belong in any plot legend or exported table.

Begin with the nominal `summary` and `scenario_effects`. Then choose one named sensitivity entry, read its `perturbation` record and inspect its stored summary and scenario changes. For a numerical output y and a perturbed input q, a forward finite response can be computed as

$$D_{+}y = \frac{y(q+\delta q) - y(q)}{\delta q}$$

only when the parameter supports that interpretation and the output exists in both cases. A central finite difference uses the paired positive and negative runs:

$$D_{0}y = \frac{y(q+\delta q) - y(q-\delta q)}{2\delta q}.$$

These finite differences approximate derivatives only when the response is sufficiently smooth and the perturbation scale is appropriate. A categorical force-model switch is a model comparison, not a derivative with metres or seconds in its denominator.

If a perturbation changes a link from clear to occulted, a local derivative of the accepted rate at that boundary may not be a useful stable description. Report the status transition directly. Similarly, a PF selector can change status or become sensitive near a selection degeneracy; an unavailable field must remain unavailable rather than being replaced by zero to make a graph continuous.

Where a finite secant response is recorded, it is the output difference divided by the supplied input delta. Its units follow that quotient. A 1-m position probe and a 0.001-m/s velocity probe cannot be compared as though their deltas were the same physical quantity. They are not probability distributions or confidence bounds. Gate probes can change acceptance alone under the current independence assumptions; phase-covariance probes can change the conditional density matrix because they explicitly change the apparatus model.

12.6 A reusable inspection checklist

1. Verify the session reports completion and inspect branch validation status.
2. Record the release, source fingerprints, epoch, orbit provider, force model and spacetime choice.
3. Check reference element age, leap coverage and Earth-orientation status.
4. Confirm the compared records share a physical UTC epoch and compatible chart and units.
5. Read propagation and visibility status before interpreting arrivals, PF diagnostics or rates.
6. Separate exact conditional metrics from finite-shot estimates and actual accepted counts.
7. Confirm Scenario 3 retains Scenario 2's release policy before attributing any difference to the full construction.
8. Identify every changed apparatus or force parameter when comparing sessions.
9. Preserve the original session and export analysis into a separate directory.
10. State the model assumptions and validation limits with every published figure.

Chapter 13

Complete command reference

13.1 Common conventions and precedence

Run any entry point with `--help` for its parser summary. Filenames following options should be quoted if they contain spaces. Numeric inputs use ordinary decimal or scientific notation, such as `1e-10`; unit symbols must not be appended to numeric values. UTC timestamps should include `Z` or an explicit offset, for example `2026-09-16T18:32:00Z`.

The selected JSON configuration supplies the base settings. First, the loader checks and canonicalizes physical and execution aliases. A conflict between `spacetime` and `propagation_model`, between `timing_mode` and `apply_QCS_corrections`, or between an execution field and its comparison alias is rejected before an override can mask it. Explicit command-line or GUI overrides then replace valid corresponding base settings; final normalization and validation check the effective configuration. Omitted overrides leave the configuration in force; normalization defaults apply only to missing keys. In particular, a parser option with no literal default does not necessarily imply that the resulting model option is unset.

All command-line configuration loaders, including both main runners and the retained comparison command, resolve relative `tracking.cache_dir`, `tracking.leap_second_file`, and `comparison.iers_file` paths against the configuration file's directory. Absolute input paths retain their meaning. The config filename itself, output directories, replay/session inputs, and explicit report destinations resolve from the invoking process's working directory. The Windows launchers first make that directory the project folder. Moving a configuration therefore requires moving its relative input tree or updating those input paths, while moving only the launch location does not change the meaning of the inputs in that file.

Both main runners resolve `execution.sample_counts` and `execution.controls` with the same precedence: an explicit enable/disable command-line choice, then the supplied configuration, then `false`. Omission of a CLI choice preserves the configuration. The retained `comparison.sample_counts` and `comparison.controls` names remain accepted aliases; conflicting values between a canonical field and its alias are rejected instead of silently choosing one.

13.2 Matched-state comparison: inputs and model controls

Invoke `run_comparison.py` directly with the environment's Python or use `run_comparison_python38.bat`.

Option	Accepted value / default	Operational meaning
`--config`	JSON path; package `config_comparison_replay.json` if omitted	Base configuration for an ordinary session.
`--output`	New directory; timestamped child of `comparison_sessions` if omitted	Session destination, or root of live segment directories.
`--replay`	Completed session directory	Offline recomputation using captured settings.
`--epoch-utc`	Explicit timestamp; configuration epoch or current UTC otherwise	Initial physical epoch; incompatible with `--live`.
`--spacetime`	`flat_sr` or `gr_static`; configuration value	Selects photon/timing/PF spacetime for all selected scenarios.
`--force-model`	`two_body`, `j2`, or `j2_drag`; configuration value	Numerical orbital force model; drag needs explicit inputs.
`--duration`	Seconds from 0 through 604800	Comparison forecast horizon in TAI/TT-rate elapsed seconds.

Option	Accepted value / default	Operational meaning
`--samples`	Integer 1 through 10001	Number of orbit-comparison samples across the horizon.
`--scenario-steps`	Integer 1 through 1001	Number of independent photon-ensemble epochs per branch.
`--scenario`	1, 2, 3, `both`, or `all`	Overrides configured scenario list; `both` means 1 and 2.

If missing from a custom configuration, normalized comparison defaults are 600 seconds, 31 orbit samples, 3 photon epochs, and Scenarios 1-3. The supplied offline comparison preset selects J_2; the supplied live preset selects J_2 with a 60-second horizon, 7 orbit samples, and 1 photon epoch. A custom configuration with no numerical-model section defaults to `two_body`. Inspect the actual preset instead of assuming every run uses the same defaults.

The upper duration limit is an input bound, not a claim of useful predictive accuracy for a seven-day extrapolation. The ephemeris-age policy and timing-data coverage still apply throughout a run, and force-model and dense-output limits can stop invalid predictions.

13.3 Matched-state comparison: channel and extra calculations

Option	Value / default	Operational meaning
`--sample-counts`	Explicit enable; otherwise shared configuration, then false	Adds seeded finite-count tomography.
`--no-sampling`, `--no-sample-counts`	Equivalent explicit disables	Overrides a configured sampling enable; keeps exact channel metrics.
`--controls`	Explicit enable; otherwise shared configuration, then false	Adds channel control calculations.
`--no-controls`	Explicit disable	Overrides a configured controls enable.
`--sensitivity`	Enable switch; false unless configured	Runs configured probes, or the four default probes if absent.
`--phase-sigma-a`	Nonnegative radians; configuration value	Arm A accumulated H/V phase standard deviation.
`--phase-sigma-b`	Nonnegative radians; configuration value	Arm B accumulated H/V phase standard deviation.
`--phase-correlation`	Number in [-1, 1]; configuration value	Explicit covariance correlation; required for nonzero noise.
`--phase-source`	Nonempty descriptive text	Records the declared parameter or calibration source.

Sampling and controls have explicit enable and disable forms in both runners. Supplying both forms for the same option is a usage error. Sensitivity remains a separate opt-in switch: omitting `--sensitivity` does not disable `comparison.sensitivity_enabled: true` in the drag example; set that field to `false` in a copied configuration to disable it. The comparison runner has no equal-arm `--phase-sigma` shorthand; specify its per-arm options.

13.4 Matched-state comparison: live execution and viewer

Option	Value / default	Operational meaning
`--live`	Enable switch	Starts current-UTC frozen forecast segments; requires online tracking.
`--live-iterations`	Positive integer; 10	Number of segments when live is enabled.
`--live-interval`	Seconds from 0 through 60; 30	Minimum start-to-start spacing.
`--dashboard`	Enable switch	Starts the local read-only viewer and keeps serving after computation.
`--port`	Integer 1 through 65535; 8765	Integrated dashboard TCP port.
`--open-browser`	Enable switch	Requests opening the viewer; requires `--dashboard`.
`--quiet`	Enable switch	Suppresses detailed scenario progress, not completion/error messages.
`-h, --help`	No value	Prints usage and exits without running a session.

Port and live-interval values are validated even when their feature is not requested. The integrated dashboard serves the chosen output directory; a separately launched viewer pointed at `comparison_sessions` is preferable when you want to browse many independent session roots.

13.5 Original scenario runner: selection and output

Invoke `run_privileged_frame_simulation.py` directly or use `run_python38.bat`. This command evaluates a single configured orbit source, not the two-branch matched-state workflow.

Option	Value / default	Operational meaning
`--scenario`	`1`, `2`, `3`, `both`, `all`; `all`	Selects scenarios; `both` retains the historical meaning 1 and 2.
`--steps`	Positive integer; configuration value	Overrides independent emission epochs; live forces one.
`--config`	JSON path; package `config.json`	Loads scenario and orbit settings.
`--output`	Directory; `results`	JSON, manifest, validation and optional image destination.
`--no-plots`	Disable switch	Skips `comparison.png`; keeps numerical output.
`--sample-counts`	Explicit enable; otherwise shared configuration, then false	Adds seeded finite-count tomography.
`--no-sampling, --no-sample-counts`	Equivalent explicit disables	Skips finite-count sampling even when the configuration enables it.
`--controls`	Explicit enable; otherwise shared configuration, then false	Adds extra channel control calculations.
`--no-controls`	Explicit disable	Skips controls even when the configuration enables them.
`--earth-occlusion`	Enable override	Enforces Earth blockage in accepted counts.
`--unobstructed`	Disable-occlusion override	Explicit virtual collection in flat SR only.
`--timing`	`selected_frame`, `eci`, `none`	Overrides reception timing target; `none` releases both at the epoch.

Sampling and controls share the comparison runner's opt-in defaults and `execution` configuration. Plotting remains enabled unless `--no-plots` is supplied. Enable and disable choices for the same calculation are mutually exclusive. `--earth-occlusion` and `--unobstructed` cannot be used together. The ordinary configuration already enforces obstruction, so the former is mainly an explicit override of a copied configuration.

13.6 Original scenario runner: physics, covariance and live options

Option	Value / default	Operational meaning
`--spacetime`	`flat_sr` or `gr_static`	Overrides configuration and applies to every selected scenario.
`--orbit-source`	`circular` or `sgp4`	Selects synthetic circular worldlines or tracked predictions.
`--epoch-utc`	Explicit UTC timestamp	Tracked run epoch; incompatible with `--live`.
`--simulation-only`	Require-synthetic switch	Requires `circular`; does not silently convert an SGP4 configuration.
`--phase-sigma`	Nonnegative radians	Sets both phase sigmas; incompatible with either per-arm option.
`--phase-sigma-a`	Nonnegative radians	Arm A override.
`--phase-sigma-b`	Nonnegative radians	Arm B override.
`--phase-correlation`	Number in [-1, 1]	Explicit correlation, required for nonzero phase noise.
`--phase-source`	Nonempty descriptive text	User-declared parameter source.
`--live`	Enable switch	Forces SGP4, one photon epoch, and interval [0, 0] per update.
`--live-iterations`	Positive integer; 10	Number of live updates.
`--live-interval`	Seconds from 0 through 60; 5	Minimum start-to-start live interval.
`-h, --help`	No value	Prints usage and exits.

In live mode, an explicit `--steps` value is superseded by the required one-epoch-per-update behavior. The live flag does not reset GR to SR. An equal-arm legacy `phase_sigma_rad` configuration is normalized before explicit sigma overrides; a configuration that mixes that legacy key with per-arm sigma keys is rejected. New configurations should use the two per-arm keys consistently.

13.7 Standalone dashboard

`comparison_dashboard.py` has `--sessions` (default `comparison_sessions`), `--port` (default 8765), `--open-browser`, and `-h`/`--help`. Its port accepts 0 through 65535. Unlike the integrated runner, port 0 is permitted and asks the operating system to choose a free port; read the actual URL printed in the console.

```
run_dashboard_python38.bat ^
  --sessions comparison_sessions --port 0 --open-browser
```

The server binds only to the local IPv4 loopback interface. It validates host headers and local requests, serves bundled static assets and read-only JSON endpoints, and rejects write methods. There is no public hosting, remote login, upload endpoint, or command-execution interface in the dashboard. Do not assume that opening a firewall port makes this an authenticated multi-user service. This paragraph concerns the legacy read-only results viewer; the separate hosted application is documented in Chapter 27.

The local endpoints are `/api/health`, `/api/sessions`, and `/api/session?id=...`. Session identifiers come from discovery; the server prevents path traversal and loading outside the allowed session tree. File sizes and session discovery are bounded. These implementation safeguards support local inspection; normal users can work entirely through the browser controls.

13.8 Retained comparison and tracking modules

`real_time_satellite_comparison.py` is an older, explicitly unmatched orbital comparison. It compares configured ideal circular trajectories with same-epoch SGP4 predictions without fitting their initial states. Large differences can therefore arise simply because the initial orbits describe different objects or positions. Use the new comparison runner for matched-state experiments. This retained entry point supports flat SR only and rejects a static-GR configuration. Use `run_comparison.py` for the full matched-state SR/static-GR workflow; passing a GR label to the retained tool must not produce an SR calculation under that label. The retained tool also rejects ground-receiver configurations. Use the full comparison workflow for its supported ground-link scenarios.

Option	Value / default	Meaning
`--config`	Required JSON path	Tracking catalog configuration and optional synthetic profiles.
`--epoch`	Timestamp; config epoch, then current UTC	Epoch option uses this spelling, not `--epoch-utc`.
`--steps`	Positive integer; 1	Number of prediction samples.
`--duration`	Finite nonnegative seconds; 0	Future prediction interval, with no wall-clock waiting.
`--output`	Path; `results/tracking_comparison.json`	Single output JSON file.
`-h, --help`	No value	Usage summary.

Example:

```
.venv-pf38\Scripts\python.exe real_time_satellite_comparison.py ^
  --config config_tracking_replay.json --steps 3 --duration 60 ^
  --output results\tracking_comparison.json
```

`real_time_satellite_tracking.py` is a provider module, not a separate interactive tracking command. Its public `CelesTrakProvider` and compatibility name `SatelliteTracker` are used by the runners. Starting that file directly does not launch a web application. `quantum_simulation.py` retains compatibility functions and delegates its executable entry point to the original scenario runner; the named batch launcher is the clearer user interface.

13.9 Replot and independently validate

`plot_comparison.py` accepts one or more positional scenario JSON paths and an optional `--output` path, defaulting to `comparison.png`. Scenario numbers must be unique. Labels follow the recorded scenario identity, using the legacy field or standard filename only under the fallback rules in Section 11.1. Unidentified renamed files and invalid or conflicting identities are rejected with exit status 2. It recreates the static photon-channel figure; it does not create the orbital dashboard or recompute physical events.

```
.venv-pf38\Scripts\python.exe plot_comparison.py ^
  results_gr\scenario1.json results_gr\scenario2.json ^
  results_gr\scenario3.json --output results_gr\comparison.png
```

`validate_results.py` and `validate_extended_results.py` each accept `--results`, defaulting to `results`, optional `--report PATH`, and the standard help flags. The general validator supports retained schema 4 and current schemas 5 and 6 and rejects a mixed-schema folder. The extended validator checks the current pipeline's schemas 5 and 6, including storage audit records when enabled. They operate on a scenario-run directory, not on an entire comparison root containing two branches.

```
.venv-pf38\Scripts\python.exe validate_results.py ^
  --results results_gr
```

Both validator commands print their recomputed JSON report to standard output and leave the input files unchanged by default. The stored `validation.json` written when the original run completed remains intact. This default is safe for the branches inside a hash-preserved comparison session.

To retain a new audit report, request a new file outside the input results and outside every captured run or session:

```
.venv-pf38\Scripts\python.exe validate_extended_results.py ^
  --results comparison_sessions\gr_case\numerical ^
  --report analysis_reports\gr_case_numerical_recheck.json
```

The report path is relative to the process working directory. Its parent directories are created only after destination checks succeed. Existing files, directories and symlinks are never overwritten. Destinations inside the input results, inside an enclosing run or session, or inside another captured run/session are rejected; resolving `..` or a symlink does not bypass this protection. Choose a new external filename for a second report. The external report records either a passed audit or a failed numerical/input audit. Exit status is 0 for passed validation, 1 for failed numerical/input validation, and 2 for a bad report destination or command usage. The Python `validate()` APIs remain read-only and raise on invalid data. Reading an existing report does not rerun validation.

For software regression testing, run the package's test suite from its root:

```
.venv-pf38\Scripts\python.exe -m pytest -q
```

Tests using tracking transformations require the optional tracking dependencies. The developer-only JavaScript behavior tests use Node.js, but Node.js is unnecessary to run the simulations or view the dashboard. A passing invariant test, a passing replay, and a correct visual rendering answer different questions; none alone establishes experimental validity of a physical hypothesis.

Chapter 14

Troubleshooting, recovery and reproducible work

14.1 Installation and dependency errors

If `py -3.8` is not recognized or cannot find Python 3.8, the launcher or that interpreter is unavailable on the machine. Confirm the installed interpreter rather than letting a bare `py` command choose another version. If the environment exists, use its full Python path to check its version. The setup script's printed error is the relevant evidence; a later simulation error may merely be a consequence of incomplete installation.

If pip says that `qiskit==2.5.2` cannot be installed on Python 3.8, you are using an obsolete or edited requirements file that lacks the release's environment markers, or explicitly requesting the incompatible version. The supplied `requirements.txt` selects Qiskit 1.2.4 for the Python 3.8 route. Recheck the file and run the setup in the intended extracted project.

Errors such as `Could not load 'ibm_backend'` or `cannot import name 'TypeAliasType'` are characteristic of conflicting packages in the interpreter being used. Use the isolated package environment and its pinned requirements. Do not add IBM cloud packages to solve a local simulation problem unless you are deliberately developing a separate integration. `pip check` helps identify dependency conflicts but does not replace importing and executing the actual application.

For a missing `sgp4` or `astropy` import, run `setup_tracking_python38.bat` after the base setup. Installing these packages into a different global interpreter will not repair `.venv-pf38`. If installation cannot download packages, retain the exact network or certificate error; changing simulation physics will not fix package retrieval.

14.2 Ephemeris, Earth-orientation and leap-second failures

Live data failures are reported explicitly. The provider does not substitute a synthetic satellite, silently change a catalog ID, or fabricate a valid ephemeris. Check the recorded source URL, catalog identifier, retrieval attempt, requested epoch, and cache policy. A stored GP record can still be too old for the requested propagation epoch even when its file exists.

The default element-age cutoff is seven days in either direction from the element epoch. This is an operational policy, not a promise of seven-day accuracy. Use the fixed historical capture at its documented historical epoch for offline examples. Do not advance that capture to the current date merely to make it appear live. A current download is also not guaranteed to be suitable for a much earlier historical epoch.

Three data concerns should be distinguished: orbital-element freshness, Earth-orientation coverage, and leap-second coverage. Refreshing one does not automatically repair the other two. In online mode, `tracking.offline` must be false and `tracking.iers_auto_download` may enable current timing-data acquisition. In offline mode, the required captures must already exist and cover the requested epoch.

The package checks leap-second coverage before using UTC conversions. It first uses the configured/automatic selection policy and can fall back to the bundled official `timing_data/Leap_Second.dat` when that list covers the epoch. This release's bundled list expires on 2027-06-28. Its date is not extended by the program. The console and provenance identify a fallback and its reason.

If no available list covers the requested epoch, obtain a current official list and supply `tracking.leap_second_file`, or repair online acquisition. An explicitly configured file is a strict override and is not silently replaced. `allow_degraded_iers` concerns Earth-orientation behavior; it does not waive leap-second validity. A covered

historical replay can remain legitimate after the list has expired relative to today's clock because the requested historical epoch is still covered.

Do not suppress a persistent leap-second or IERS exception just because a plot is desired. These time conversions are part of the calculation's coordinate contract. Conversely, a fallback message followed by completed, passing validation is not the same as an unresolved stale-table failure. Inspect the provenance to determine which source was actually used.

14.3 Physical unavailability versus numerical failure

`propagation=ok` describes the ray/interception calculation. `visibility=blocked` describes obstruction. In flat SR it is possible to calculate a mathematical vacuum ray through the Earth while masking that pair from accepted counts. In static GR, no interior-Earth metric is supplied, so a blocked connection is reported as occulted and reception events can be unavailable. These are distinct from network errors.

An unavailable full PF selection does not automatically invalidate a resolved photon pair. The new construction has its own event-separation, mapping, domain, and numerical certification conditions. Conversely, a tiny PF simultaneity residual does not turn a blocked pair into detected photons. Read the status and interpretation fields together, rather than sorting experiments by the smallest printed number.

An unresolved solver is not assigned a physical zero collection probability. Unknown acceptance is retained as unavailable, commonly `null` in JSON. A genuine blocked link can have zero acceptance. This distinction prevents a numerical failure from masquerading as a successful prediction of no photons.

If the numerical orbit model reports atmosphere-domain exit, invalid spacecraft parameters, a collision, or an out-of-range dense-output request, revise the requested experiment or supply a valid model domain. For the drag model, do not extend a low-altitude density formula blindly to orbital heights. The configured positive density table is used only inside its declared altitude interval.

14.4 Dashboard and plotting problems

If the browser cannot connect, keep the dashboard command running and use the exact loopback URL printed in its console. A server stopped by Ctrl+C no longer serves the page. If port 8765 is already in use, choose another port for both the command and URL, or let the standalone viewer select a free port with `--port 0`.

A blank session list usually indicates the wrong `--sessions` root or absence of comparison sessions, not missing quantum results. The viewer expects comparison-session structure. The original scenario runner's `results_gr` folder is normally inspected through its JSON, static PNG, and validator; it is not a substitute for a `run_comparison.py` session. Nested branch folders are deliberately excluded from the picker.

If an intermediate slider position says no photon ensemble exists, inspect the listed available photon UTC epochs. The orbit and photon grids are independent. With 31 orbit samples and 3 photon epochs across 600 seconds, most orbit samples have no photon row. The viewer joins these data by physical UTC, including GR, rather than treating TCG and TT elapsed numbers as interchangeable.

If fidelity and purity are flat across time, compare the channel inputs. With storage disabled, a time-independent covariance and source state produce time-independent exact conditional metrics. With storage enabled, also inspect the recorded proper storage times and coherence factors; changing holds can change the exact state even at fixed covariance. This can be correct even while orbit positions and counts change. If an accepted rate is zero, the conditional channel prediction can still be defined, but there is no collected ensemble from which to infer tomography.

Reload after updating dashboard assets. For printing, use the browser's Print or Save as PDF function and inspect every page for clipped tables, chart splitting, or overlap. The release includes dedicated landscape print styles and pauses refresh during printing. Corrected print pagination was not visually verified by the remote browser environment; automated DOM/server checks should not be represented as a substitute for checking the actual saved PDF.

14.5 Exit status and interrupted runs

The comparison and original scenario runners return 0 after normal non-viewer completion, 1 for caught runtime failures, 2 for invalid command usage, and 130 for interruption. With an integrated dashboard, successful computation is followed by continued serving; Ctrl+C stops that serving and reports interruption even though the completed session remains available. The standalone viewer handles Ctrl+C as an ordinary stop and returns 0.

In Command Prompt, inspect `%ERRORLEVEL%` immediately after the process returns. Runtime messages begin with `Comparison failed:` in the matched workflow or `Runtime error:` in the original runner. A usage banner accompanies argument errors, such as incompatible replay options. A terminal that still displays an active dashboard prompt is not necessarily a hung computation.

Completed sessions remain after interruption. A calculation interrupted before completion may retain partial outputs with `running` or `failed` status; use its manifest and validation report to determine what finished. Do not label partial files as a successful run merely because they contain several plausible epochs. Restart into a new output directory so the new attempt has a clear identity.

14.6 A repeatable experiment routine

For each study, copy the nearest preset into a clearly named custom configuration, record the scientific question and changed inputs, and choose a fresh output folder. Run the command with an explicit spacetime and scenario selection if those distinctions matter to the comparison. Retain the console log, configuration, complete session directory, and software version together. Keep expected physical outcomes separate from observed program results.

In Windows Command Prompt, capture both ordinary progress and error messages with `>` and `2>&1`. Keep this log outside the new session directory, because the comparison runner requires that directory to be empty when the calculation begins:

```
if not exist logs mkdir logs
run_comparison_python38.bat ^
  --config config_comparison_replay.json ^
  --output comparison_sessions\logged_sr ^
  > logs\logged_sr.txt 2>&1
set PF_RUN_EXIT=%ERRORLEVEL%
echo Exit code: %PF_RUN_EXIT%
type logs\logged_sr.txt
```

Output is redirected to the file during this run, so absence of terminal progress is expected. `>` replaces an existing log of the same name; use a new filename for a separate attempt. `>>` appends, which is useful only when you intentionally want several invocations in one log and label them clearly. If placing a launcher invocation inside another batch script, use `call run_comparison_python38.bat ...` so the outer script resumes after the launcher returns. Capture the exit code immediately; subsequent commands can change it.

Before interpreting the plots, check completion status, branch validation, data age, time conventions, visibility, and the declared channel assumptions. Compare matching UTC epochs and matching physical quantities. A chart of numerical-minus-SGP4 position is a comparison of model predictions; it is not a residual against telemetry. A conditional fidelity is referenced to the declared Bell target, not to the coincident scenario.

Preserve the complete session, including `inputs` and `session_manifest.json`, and replay into another directory. A screenshot cannot recover model settings or captured data. Manifest hashes detect changes; they are not publisher signatures.

For support, provide the exact command, complete error text, Python version, configuration and affected session. Include `comparison.json`, session and run manifests, `validation.json`, and affected scenario files. For browser issues, identify the selected session and epoch; attach a screenshot and the saved PDF for printing defects.

Chapter 15

Validation, replay, and troubleshooting evidence

15.1 Three different checks

File integrity, numerical consistency, and empirical accuracy are distinct. Hash verification establishes that the files match the manifest. The numerical validator independently checks declared equations and internal constraints. Empirical accuracy would require external measurements and calibrated apparatus assumptions; the package's passing tests do not provide that evidence.

The scenario validator checks state normalization and analytic quantum results, accepted-count consistency, emission schedules, transformations and inverse transformations, null constraints, GR endpoint and invariant diagnostics, proper clock rates, and full-PF construction constraints where defined. When both Scenarios 2 and 3 are present, it also checks that Scenario 3 retains the same physical events and quantum input.

Read the scope note in `validation.json` alongside `status`. Maxima such as `max_gr_endpoint_residual_m` or `max_full_pf_time_gap_s` are numerical residuals within that scope. The report also counts resolved epochs, clear epochs, unknown acceptance epochs, collected synthetic pairs, and PF statuses. These counts often explain a result faster than a single headline fidelity.

15.2 Verify a session without modifying it

Run this from the project directory using the configured Python environment:

```
from comparison_session import verify_session

manifest = verify_session(
    'verification/full_comparison/gr_noise_sensitivity')
print(manifest['status'], manifest['session_id'])
```

A missing file, altered size, hash mismatch, or unsafe captured path raises an error. Hashes detect changes relative to the manifest; they are not a publisher signature or proof that a data source was truthful.

For a read-only numerical recheck of a branch, use the ordinary validator command:

```
.venv-pf38\Scripts\python.exe validate_extended_results.py ^
  --results comparison_sessions\gr_case\numerical
```

Its JSON audit is printed to standard output; no saved result or report is rewritten. To keep the audit as a file, append `--report analysis_reports\gr_case_recheck.json`. The destination must be new and outside captured runs/sessions. Report-writing protection also resolves symbolic links and parent-directory segments before checking the destination. The general `validate_results.py` command follows the same policy.

The read-only Python API is also available for analysis scripts:

```
from pathlib import Path
from validate_extended_results import validate

report = validate(Path(
    'verification/full_comparison/gr_noise_sensitivity/numerical'))
print(report['status'])
```

Do not run Python with `-O` or `-OO`: the validator requires assertions and refuses optimized execution. A passed CLI audit exits with status 0; validation failure exits with status 1, and invalid report destinations or command usage exit with status 2. A preserved old session can still be checked using these commands. Source revision and optional effective-configuration metadata may differ across releases, so exact replay across a revision should be assessed through its explicit comparison report rather than assumed.

15.3 Recompute from frozen inputs

Replay is stronger than reopening the dashboard. It verifies the completed original session, uses its captured GP, EOP, leap data and configuration, disables network refresh in the replay provider, and computes a new session in a new output directory:

```
run_comparison_python38.bat ^
  --replay comparison_sessions\gr_noise ^
  --output comparison_sessions\gr_noise_recomputed
```

`replay_validation.json` reports equality of `rows`, `scenarios`, `scenario_effects`, and `sensitivity`. The saved `replay_gr_noise` evidence reports all four as true. Runtime paths, wall-clock creation timestamps, and host provenance are intentionally outside these numeric equality sections.

The session records Python and package versions and source hashes. Different libraries, platforms, or source revisions can change floating-point outputs. A mismatch is reported and should be investigated, not silently relabeled a pass. Preserve both sessions, compare environment provenance, and determine whether the change is an intended model revision, a numeric implementation difference, or a defect. A justified tolerance study is separate work; the current replay result is exact equality on the tested runtime.

Replay retains validity constraints. It does not extend an expired leap table, synthesize missing GP data, or silently permit stale elements. Frozen inputs make an old calculation reproducible at its original epoch; they do not make those inputs appropriate for a new current-UTC prediction.

15.4 Read errors in context

Observation	Likely meaning	Next inspection
Session `failed`	Acquisition, configuration, propagation, validation, or interruption failed	`error` and `error_type` in comparison/session/run records
Validation `not_completed`	Current run has not finished its audit	Run status and terminal log; do not reuse an earlier pass
PF `reception_events_unavailable`	No resolved pair on which to run the construction	Link propagation status and reason
Zero accepted rate with finite F/P	Conditional channel defined but no accepted observations	Visibility, gate, collection scope, sampled totals
Null accepted rate	Prediction unresolved, rather than demonstrated zero detections	Propagation diagnostics and reason

Observation	Likely meaning	Next inspection
No photon rows at selected time	Orbit sample has no matching ensemble	Available photon UTCs and configured ensemble count
No sensitivity cases	Sensitivity was disabled or no probes selected	Captured comparison configuration
Old plots after refresh failure	Browser retained last successfully loaded data	Connection banner, server terminal, selected session
Hash mismatch	Stored content changed or is incomplete relative to manifest	Preserve originals; compare paths, sizes, hashes
Replay mismatch	Recomputed numeric sections differ	Replay report and runtime/source provenance

For a support report, include the exact command, terminal output from the first error onward, package release identifier, Python version, relevant configuration, and complete session folder when feasible. Include both `run_manifest.json` and `validation.json`; a screenshot of a passing line alone cannot bind the claim to a particular scenario file. For display defects, include the selected session, epoch, projection, satellite filter, browser version, screenshot, and any PDF that demonstrates the issue.

A clear analysis conclusion names the declared model, frozen input epoch, collection mode, noise assumptions, prediction or estimator being reported, and its evidence. For example: “In the saved 120-second GR/noise example, the numerical and SGP4 branches predict the same conditional channel fidelity under the supplied covariance; both endpoint ensembles have exterior clear links. The numerical detector-2 trajectory differs from the reference by at most approximately 3.20151 m over the stored samples. These are model predictions, with independent numerical checks, rather than measured orbital or optical performance.” This preserves the result’s useful content without extending it beyond the calculation.

Chapter 16

Reporting, preservation and validation boundaries

16.1 What to retain with a result

Retain the entire completed session directory, the exact command and console log, the configuration used to launch it, the package release identifier and the Python/dependency versions. A graph alone is not a reproducibility record. A single `comparison.json` is useful for inspection but is not sufficient to recreate every frozen reference input or independently inspect each scenario's full event and quantum output.

For a published numerical comparison, give the UTC epoch and horizon, sample counts, reference provider, element ages, source and comparison frames, timescales, numerical force model, solver tolerances and integration bounds. State that the reference is SGP4 prediction when that is the case. For optical results additionally report emission policy, Bell target, source depolarization, phase sigmas, covariance correlation, deterministic phase, detector and bandwidth assumptions, gate policy, loss and Earth visibility enforcement. Identify which figures show exact conditional states and which show count-based estimates.

If presenting the full PF calculation, report the event pair, anchor definition, SR or GR representation, selection status and relevant numerical residuals. State whether a release policy was prescribed independently of the selection. This release uses the same physical schedule for Scenarios 2 and 3; it does not implement a new full-PF emission controller.

16.2 A concise experiment record

Create a text file outside the immutable session containing the following fields. Replace all example labels with your actual values.

```
Experiment title:
Question and controlled change:
Package release:
Python and dependency versions:
Command and configuration:
Session directory:
Reference UTC / duration / samples:
Reference-data provenance and age:
Orbit force model / chart / time coordinate:
Photon spacetime / visibility / emission policy:
Quantum target / covariance / parameter source:
Gate and count budget:
Validation and replay outcome:
Observed numerical differences:
Interpretation and remaining limitations:
```

Attach exported figures with explicit axis units, difference signs, sample counts and source session names. A connected line through samples is a plotting convention; it does not add observations between them. Avoid replacing unavailable values with zero, overlaying different live segments as one uninterrupted forecast, or calling an illustrative sensitivity span a confidence interval.

16.3 Historical execution evidence

Evidence	Recorded result	Scope of the conclusion
Revision 5 full Python 3.8.20 regression run	596 passed	Retained execution evidence; four expected historical timing warnings.
Revision 5 validator/reporting tests on Python 3.12.14	37 passed	Read-only audits, protected reports and compatibility on a second runtime.
Earlier comparison tests on Python 3.12.14	130 passed	Retained evidence for unchanged comparison modules.
Final focused dashboard HTTP tests	25 passed	Server/API behavior, access checks and result handling
Dashboard JavaScript behavior tests	9 passed	Actual script exercised using a mocked DOM/canvas environment
Revision 5 saved SR replay	Exact numeric sections matched	Retained frozen-input recomputation evidence.
Revision 5 saved GR/noise/counts/sensitivity replay	Exact numeric sections matched	Repeated with seeded counts and four sensitivity probes.
Actual current-UTC GP acquisition	Three successful downloads in the recorded empty-cache test	Online data acquisition worked at that recorded time
Revised dashboard browser rendering and printing	Not directly verified internally	Browser navigation and local-file preview blocked by the authoring environment

The table records earlier execution evidence retained with this package. Revision 6 adds targeted tests of recorded scenario labels, renamed and reordered files, legacy fallbacks and input rejection. Their results are recorded in `RELEASE_NOTES_R6.md` and `DOCUMENTATION_VALIDATION.md`. The simulation and quantum-channel implementations were unchanged by that plotting correction. Revision 7 adds the optional storage model and focused storage validation described in Chapter 19 and the delivered validation records. The focused browser-script tests do not substitute for real canvas rendering, browser interaction or print pagination. The supplied earlier user dashboard PDF showed functioning charts as well as print defects; the later display fixes have code-level checks but still need local browser inspection.

Numerical orbital tests include matched initial states, Kepler and circular limits, energy and angular momentum checks, J_2 force-gradient comparisons, drag sign and domain checks, and conversion Jacobians. Quantum tests compare the channel with independent analytic metrics and examine valid covariance limits. GR/PF tests check the specified numerical identities and event construction. The source records define what each test actually asserts.

No test count certifies operational satellite navigation accuracy, measured entanglement distribution, calibrated phase noise, or a universal physical interpretation of PF selection. Those claims require separate experimental evidence and independently justified inputs. This boundary is central to using the package scientifically.

16.4 Release identity and integrity

The archive documented here is `PF_Simulation_Standalone_and_SaaS_r26.zip`, release `full-comparison-2026-09-20-r26-numerical-relativity-workflows`; this is manual edition 1.17.0. The revised PDF and its editable LaTeX source are included with the release. Read the release identifier from `MANIFEST.json` and use the delivered `PACKAGE_CHECKSUMS.json` to verify the packaged content. An archive's own byte hash cannot be embedded recursively in a document inside that same archive; keep any separately distributed archive digest alongside the file it identifies.

Revision 17 groups same-directory image/PDF formats of one saved figure and separates figures from reports and documents in the GUI. Figure counts describe unique figures; format-file counts describe their available

representations. Published benchmark details remain in their saved JSON reports. This presentation change does not alter the underlying calculations or saved numerical evidence.

Revision 8 refines the nine vector figures and supplies independently arranged full-width and single-column versions. Legends, markers, labels, connectors and captions are updated; the simulation modules, configuration, formulas and saved numerical results are unchanged from revision 7. Section 11.5 documents native placement and reproduction.

Revision 7 adds configurable local-memory dephasing and retrieval loss, while preserving the disabled-storage behavior. Chapter 19 explains the new schema and model limits.

Revision 6 corrected standalone plot labels to use recorded scenario identities and validates identity conflicts before plotting. The shared execution defaults, configuration-relative input paths and read-only independent validators introduced in revision 5 remain in effect. The earlier differences are resolved behaviors documented in the current command and configuration references. They are not outstanding defects in this release. To recover the earlier standalone default workload, explicitly request `--sample-counts --controls`. Existing completed sessions remain immutable historical records; do not edit them merely to update their metadata to the new defaults.

`MANIFEST.json` covers the stated source/configuration/documentation scope. `PACKAGE_CHECKSUMS.json` records archive content checks. A session's own manifest records that session's capture and generated artifacts. These objects have different scopes. A hash confirms consistency with the stated bytes; it is not a signature from an independent trusted publisher and does not establish the truth of input data.

The full PF vendor source remains byte-for-byte the supplied `privileged_frame.py`, stored as `vendor/privileged_frame_20260913.py`. Its provenance record names `PF_Presentation_Corrections_2026-09-13(2).zip` as the supplied archive. Preserve that source and its notices when moving the package. This manual does not grant new redistribution rights or replace the licenses and notices of the package's components and dependencies.

Chapter 17

Glossary and source references

17.1 Glossary

Term	Meaning in this package
Accepted pair	A simulated emitted pair retained under efficiency, gate and visibility assumptions.
Anchor	The reference event used by the full PF construction; in GR it defines the common tangent space.
Areal radius	Schwarzschild radial coordinate for which a symmetry sphere has area $4\pi r^2$.
Bandwidth	Configured spectral scale used by the declared Gaussian temporal-envelope convention.
Bell target	The independently chosen ideal polarization state against which fidelity is evaluated.
Cartesian state	Three position components plus three velocity components in a specified frame and timescale.
Coincident	Events sharing a spacetime point; the scenario also shares the detector worldline for both modes.
Conditional state	Polarization density matrix given collection, distinct from collection probability.
Coordinate time	Time coordinate assigned in a specified chart; not automatically any detector's proper time.
Covariance	Matrix describing the assumed joint distribution of accumulated polarization phases.
DOP853	Numerical ODE integrator used for the comparison model's Cartesian worldlines.
Earth occultation	Blocking by the package's declared opaque spherical Earth model.
ECI	Earth-centered inertial terminology; some retained field names use it for a branch's base time chart.
ECEF / ITRS	Earth-fixed reference conventions; do not treat them as arbitrary global inertial frames.
Element epoch	Epoch to which a published orbital-element set refers.
Emission hold	Ideal delay of one photon before release from the moving emitter.
Ensemble	Quantum/channel and event calculation at a selected scenario epoch.
EOP	Earth orientation parameters needed for relevant celestial/terrestrial frame transformations.
Exponential map	Geodesic map from an anchor tangent vector to a spacetime point.
Fidelity	Squared state fidelity; with a pure Bell target, its expectation in the calculated density matrix.
Flat SR	Minkowski spacetime benchmark with prescribed satellite worldlines.
Full PF	The supplied two-stage, anchor-based construction integrated through the full PF adapter.

Term	Meaning in this package
Gate	Finite acceptance interval for the modeled difference in arrival tags.
GCRS	Geocentric Celestial Reference System used for the matched orbit comparison.
GP	General-perturbation orbital data published for propagation with an associated model.
IERS	International Earth Rotation and Reference Systems Service; timing and orientation inputs are distinct datasets.
J_2	Leading zonal quadrupole coefficient used by the declared approximate numerical gravity model.
Kraus operators	Operator representation of the completely positive, trace-preserving quantum channel.
Leap-second table	Data connecting UTC and atomic timescales with an explicit coverage policy.
Logarithm map	An inverse exponential map on a selected branch, used to represent GR events in a common tangent space.
Matched state	Identical initial position and velocity at one shared epoch before independent propagation.
Numerical residual	Difference used to diagnose a model or equation; its interpretation depends on its definition.
Occulted	A classified blocked propagation path, distinct from an unresolved solver failure.
PF	Privileged Frame terminology used for the supplied construction, not an experimentally established frame in this package.
Phase noise	Assumed randomness in H/V relative polarization phase, not a random global phase.
Proper time	Time accumulated along a worldline according to the selected spacetime metric.
Purity	$P = \mathrm{Tr}(\rho^2)$, a mixedness measure separate from fidelity.
RAAN	Right ascension of the ascending node; rotating the node does not itself change prograde motion to retrograde.
Replay	Recalculation from frozen captured inputs, accompanied by integrity and numeric comparison checks.
RIC	Radial, in-track and cross-track components in the reference orbit's instantaneous basis.
Scenario	A defined physical event structure and construction policy, not a quantum quality score.
Segment	One forecast initialized at one epoch with one frozen input set and matched initial state.
Sensitivity	Controlled response to a stated parameter change, not automatically statistical uncertainty.
SGP4	Propagator used with published GP/TLE mean elements to produce the reference prediction.
Static GR	Exterior static spherical Schwarzschild photon/clock model, not full rotating-Earth relativity.
TAI	International Atomic Time; its elapsed rate equals TT's in this implementation.
TCG	Geocentric Coordinate Time; related to TT through a defined constant rate factor.
TEME	SGP4's True Equator, Mean Equinox output frame.
TLE	Two-line orbital-element format designed for its associated propagation theory.

Term	Meaning in this package
Tomography	Estimation of state observables from the accepted simulated measurement counts.
TT	Terrestrial Time; differs from TAI by a constant offset and from TCG by a rate convention.
UTC	Civil atomic timescale with leap-second handling, used for shared physical epoch labels.
Visibility	Whether the declared link geometry permits propagation outside the opaque body.

17.2 Primary external references

These sources support the standard tools and conventions used by the package. The package source remains authoritative for its actual defaults, interfaces and approximations. Links to current documentation can describe newer library versions than the pins used here; use the versioned source where supplied. The external links were accessed and checked on 20 September 2026.

R1. Python Software Foundation, Python 3.8 virtual environments. The documented virtual-environment mechanism isolates a project's interpreter and installed packages. This supports the manual's installation approach; the exact dependency pins come from the package's requirements files. Python 3.8 venv documentation.

R2. SciPy, `solve_ivp`, version 1.10.1. The reference describes initial-value ODE solving, DOP853, dense output and local error controls. Those controls govern numerical error estimates and are not ephemeris accuracy guarantees. SciPy solve_ivp reference.

R3. Astropy, working with Earth satellites. The current stable satellite-coordinate guide provides background on the SGP4/TEME state interface and attaching velocity differentials for coordinate transformations. The package uses its pinned Astropy branch and tests its own frame/time adapter; the linked guide may describe a newer release. Astropy satellite guide.

R4. Astropy, IERS data access. Earth-orientation data have coverage and prediction policies. The linked current stable guide is background documentation; the package adds explicit configuration and failure handling around these facilities and treats leap-second coverage separately. Astropy IERS guide.

R5. CelesTrak, more frequently asked questions. This discussion explains the relationship between mean elements and the theory used to generate and propagate them. It supports initializing an independent numerical model from a propagated Cartesian state instead of treating TLE mean elements as osculating Kepler elements. CelesTrak element-conversion discussion.

R6. CelesTrak, GP data formats. This is the provider's documentation of its GP delivery formats. Acquisition format, propagation model and measurement status remain distinct; a public prediction feed is not quantum telemetry. CelesTrak GP format documentation.

R7. IERS Conventions (2010), Technical Note 36, chapter 10. This chapter defines the relevant relativistic reference-system framework and TT/TCG rate relation. The package's isolated-monopole GR interface is an approximation within that broader context, not a complete implementation of every IERS effect. IERS relativistic models and time coordinates.

R8. IBM Quantum, `quantum_info` API, version 1.2. The API specifies density matrices, purity and state-fidelity operations. The package's Gaussian phase covariance and collection model are its own explicit apparatus assumptions, validated against analytic formulas; the library does not supply empirical satellite-noise parameters. Qiskit quantum information reference.

R9. NOAA Space Weather Prediction Center, satellite drag. Atmospheric drag depends on environmental conditions. This provides context for treating the bundled density-table example as illustrative, rather than a calibrated forecast for the named spacecraft. NOAA satellite-drag explanation.

R10. Sean M. Carroll, Lecture Notes on General Relativity (1997). These author-written notes provide background on manifolds, curvature and black-hole geometry. They support the distinction between

local tangent-space Lorentz operations and global coordinate transformations; the package's actual geodesic algorithms and branch restrictions remain implementation-specific. Carroll GR lecture notes.

R11. John Preskill, Quantum Information, chapter 3 (2018 update). Sections on quantum channels, dephasing and Gaussian phase noise give a standard framework for the assumed random-phase channel. They do not calibrate the example covariance for any satellite or validate a geometry-dependent noise law. Preskill measurement and evolution notes.

17.3 Package source map

Area	Files to inspect
Release and user overview	`README.md`, `MANIFEST.json`, `PACKAGE_CHECKSUMS.json`, `UPDATE_INSTRUCTIONS.md`
Main comparison workflow	`run_comparison.py`, `comparison_workflow.py`, `FULL_COMPARISON_GUIDE.md`
Shared configuration and calculation policy	`execution_config.py`; canonical execution booleans, consistent input paths, explicit IERS context
Installation and launchers	`setup_python38.bat`, `setup_tracking_python38.bat`, the three `run_*python38.bat` launchers, `PYTHON38_COMPATIBILITY.md`
Scenario engine	`run_privileged_frame_simulation.py`, `simulation_engine.py`, `corrected_simulation.py`
Flat and curved event timing	`event_timing.py`, `gr_event_timing.py`, `relativity.py`, `gr_photon.py`
Full PF implementation	`full_pf_adapter.py`, `vendor/privileged_frame_20260913.py`, `vendor/SOURCE_PROVENANCE.json`
Circular and numerical orbital physics	`orbital_models.py`, `orbit_geometry.py`, `numerical_orbits.py`, `ORBITAL_PHYSICS.md`, `NUMERICAL_ORBITS.md`
Tracking and timing data	`real_time_satellite_tracking.py`, `TRACKING_MODEL.md`, `timing_data/PROVENANCE.json`
Polarization, collection and sampling	`quantum_channel.py`, `channel_geometry.py`, `quantum_simulation.py`, `MODEL.md`
Results and plotting	`result_reporting.py`, `plot_comparison.py`, `validate_results.py`, `validate_extended_results.py`
Frozen sessions and replay	`comparison_session.py`, `SESSION_FORMAT.md`
Sensitivity	`comparison_sensitivity.py`, `SENSITIVITY_ANALYSIS.md`
Dashboard	`comparison_dashboard.py`, `dashboard/index.html`, `dashboard/dashboard.js`, `dashboard/dashboard.css`
Validation records	`VALIDATION_REPORT.md`, `FULL_COMPARISON_VALIDATION.md`, `DASHBOARD_VALIDATION.md`, `DASHBOARD_CAPTURE_REVIEW.md`, `validation_logs/`
Worked evidence	`verification/full_comparison/`, `tracking_fixtures/capture_20260916/`

The test modules provide executable examples of accepted and rejected inputs, but their internal helper functions are not necessarily stable end-user interfaces. The command references and configuration tables in this manual describe the supported user-facing controls. Source changes, custom backends or new physical models require fresh validation and a new documented experiment identity.

Chapter 18

Mathematical notation and equation audit

This audit accompanies the revised scientific-model chapter. It checks that the printed equations express the implemented models, that dimensional conventions and indices agree, and that selected identities reproduce the package numerically. These checks establish implementation and documentation consistency; they are not observations validating a satellite instrument or a new physical theory.

18.1 Typesetting and notation conventions

All mathematical expressions in the scientific-model chapter are set as LaTeX mathematics. Code names, CLI options, JSON keys, and file paths remain monospaced identifiers. Vectors and matrices use bold symbols, with their types defined where introduced; transpose and Hermitian conjugation use T and $\dagger$. Products, inner products, norms, fractions, exponents, indices, sums, integrals, square roots, covariance matrices, and kets use their mathematical forms rather than programming syntax. Dimensions and units are declared where quantities are introduced.

The spacetime signature is $(-,+,+,+)$ and the temporal coordinate is ct. Proper time is τ; coordinate time is t in a named chart. The Schwarzschild metric is explicitly areal, while the conversion input radius is r_{iso}. The density matrix is ρ and atmospheric density is ϱ_{atm}. Photon covariance uses order A, B; Qiskit basis components use order B, A. Their different orderings are intentional and are reconciled explicitly by the indexed phase generator $\mathbf{g}_j$.

Fidelity is the squared pure-target fidelity $F = \langle\psi|\rho|\psi\rangle$; purity is $P = \mathrm{Tr}(\rho^2)$. The source depolarizing weight p is distinguished from gate and acceptance probabilities by subscripts. The normal cumulative distribution function Φ is distinguished from the Bell-state ket $|\Phi^+\rangle$. Source and detected pair rates use the model's coordinate-run seconds, while reported integrated clock intervals use proper seconds. Phase standard deviations use numerical radians and ordinary-frequency bandwidth σ_ν uses hertz.

18.2 Equation-to-implementation correspondence

Equation or invariant	Exact implementation checked	Scope
Complete event Lorentz boost, inverse, and interval	`relativity.boost_event`; `relativity.inverse_boost_event`; `relativity.spacetime_interval`	One common subluminal inertial boost in flat spacetime; a constant chart map in the legacy GR interface.
Reception simultaneity and one-sided holds	`event_timing.schedule_pair`; `event_timing._solve_holds`; `gr_event_timing.schedule_gr_pair`	The selected chart's event times, not equal snapshot radii alone.
Schwarzschild metric and proper-clock rate	`orbital_models.schwarzschild_metric`; `orbital_models.metric_clock_rate`; `gr_photon.coordinate_clock_rate`	Exterior areal coordinates, timelike observers, local-static speed distinguished from coordinate speed.
Proper-time integration	`orbital_models.proper_clock_elapsed`	Signed interval on a specified worldline; no inferred synchronization offset.

Equation or invariant	Exact implementation checked	Scope
Isotropic-to-areal map and full velocity Jacobian	`real_time_satellite_tracking.isotropic_to_areal_state`; `NumericalEphemeris.worldlines`	Exterior branch and separate TT-to-TCG rate factor; approximate GCRS identification explicitly retained.
Curved metric under a linear coordinate map	`gr_event_timing.chart_metric_diagnostic`	Local tangent scalar preserved by the transformed metric.
Circular angular and proper-time rates	`orbital_models.circular_worldline_for_model`	Stable Schwarzschild circular family or prescribed flat accelerated benchmark.
Newtonian, J_2, and drag accelerations	`numerical_orbits.OrbitalForceModel.acceleration`	Declared spherical radius, normalized fixed symmetry axis, supplied atmosphere and spacecraft parameters.
RIC projection	`comparison_workflow.compare_trajectories`	Numerical-minus-reference residual at one physical epoch; reference basis.
Moving-detector null interception	`event_timing.photon_intercept`; `gr_photon.gr_photon_intercept`	Actual release position and detector worldline.
Null Hamiltonian and conserved energy	`gr_photon.solve_null_geodesic`	Direct exterior Schwarzschild ray, fixed ray-covector normalization.
Observer frequency ratio	`gr_event_timing._complete_link`	Endpoint contractions using the same ray-energy normalization.
Full-PF seed, correction shell, and anchor tangent representation	`full_pf_adapter.select_full_pf`; `full_pf_adapter._selected_diagnostics`; vendored `select_pf_state_appendix_e`	Pair- and anchor-specific construction; correction velocity and correction-chart shell, not the equivalent total-boost shell.
Bell vectors and depolarized source	`quantum_channel.bell_target`; `quantum_channel.simulate_polarization_pair`	Singlet or $\|\Phi^+\rangle$; Qiskit BA ordering.
Gaussian covariance, characteristic function, and Kraus completeness	`quantum_channel.gaussian_phase_covariance`; `quantum_channel.gaussian_phase_kraus`	Supplied AB covariance, including singular correlated limits; no geometry-derived covariance.
Exact fidelity and purity	`quantum_channel.simulate_polarization_pair`	Independent analytic Bell-family formulas checked against Aer density matrices.
Temporal bandwidth, gate probability, and pair acceptance	`channel_geometry.gaussian_gate_probability`; `channel_geometry.channel_geometry`	Gaussian transform-limited convention, independent jitter, declared gate center, loss and visibility.

The GR adapter's logarithm maps were checked against the documented coordinate contract and existing exponential-map reconstruction tests. The chapter does not identify a tangent-space vector with a global Lorentz displacement or imply a unique logarithm branch. The full-PF correction-shell expression uses the correction boost in the seed chart, in accordance with the source adapter; a Euclidean radius or total-boost replacement would describe a different condition.

18.3 Independent numerical equation checks

The equation audit ran with Python 3.8.20 for revision 5. Its formulas and original scientific implementation were unchanged in revision 6. Revision 7 adds the separately documented storage channel in Chapter 19. These retained checks constructed reference expressions independently of the implementation paths that were compared. Floating-point residuals below are numerical discrepancies within each test, not physical uncertainty estimates.

Check	Observed result
Lorentz formula at zero boost, orbital speed, $0.2c$, and $0.8c$; interval invariance	Maximum normalized formula discrepancy 7.88×10^{-17}; interval relative discrepancy below 2×10^{-14}.
Clock-rate equivalence from radial/tangential speeds, local-static speed, and direct metric contraction	Agreement to the reported double precision in all nine sampled zero-mass, Earth-mass, and strengthened-field combinations.
Full spatial and time Jacobian	Maximum relative analytic discrepancy 3.35×10^{-16}; independent directional finite differences agreed within 2×10^{-10} relative.
Gaussian quantum channel and Bell metrics	36 independently formed matrix cases; largest absolute fidelity/purity discrepancy 1.67×10^{-15}.
Kraus completeness for six covariance configurations	Largest component discrepancy from $\mathbf{I}_4$: 1.55×10^{-15}.
Gaussian gate CDF, signed-gap symmetry, and zero-width limit	Maximum absolute discrepancy 1.94×10^{-16} in 20 reference combinations.
J_2 acceleration versus a finite-difference potential gradient with an oblique axis	Relative discrepancy 3.59×10^{-10}.
Full-PF correction-shell reconstruction for the satellite-scale SR fixture	Relative shell discrepancy 1.32×10^{-15}; selected-time gap approximately 6.94×10^{-18} s.

The independent quantum cases include zero phase noise, unequal variances, positive and negative correlation, exactly correlated and anticorrelated limits, a zero marginal variance, both supported Bell targets, zero and complete source depolarization, and nonzero deterministic phase. They check both the direct Gaussian coherence multiplier and the matrix produced from the Kraus operators.

The two worked quantum values were also rerun through Qiskit Aer with sampling disabled:

Configuration	Fidelity F	Purity P
Default source, phase noise off	0.996250000000000	0.992518750000000
$\sigma_A = 0.8$ rad, $\sigma_B = 0.5$ rad, $\kappa = 0.35$, default source	0.865468878759834	0.766471722077740

The displayed formulas were checked for the limiting values $\mu \to 0$, $\mathbf{V} \to \mathbf{0}$, $p \to 0$, $p \to 1$, zero phase variances, and singular correlation endpoints. The units of the Lorentz time term, metric clock rate, coordinate Jacobian, force terms, phase covariance, gate arguments, and pair-rate factors are consistent. Numerical integration, branch-selection, visibility, and calibrated-input limitations remain part of the documented model.

Chapter 19

Configurable quantum-memory storage channel

Revision 7 adds an optional physical link between the emission schedule and the conditional polarization state. A computed release hold can mean that an already-created qubit spends a specified proper time in a local memory on the emitter. The selected memory model applies polarization dephasing and retrieval loss during that interval. It is controlled by the top-level `storage_channel` object and is **disabled by default**.

This chapter is the complete operating and mathematical reference for that extension. It distinguishes physical holding history, conditional state quality and successful retrieval. It also explains why two frame labels describing the same physical schedule still produce the same state and counts.

19.1 What changes, and what remains a controlled assumption

The existing scheduler supplies a nonnegative coordinate-time hold for each partner. The extension evaluates the emitter's trajectory during each hold, converts that duration to elapsed proper time, and applies identical physical laws to every scenario. The Gaussian phase covariance, source mixture and deterministic phase remain separately configured base-channel inputs. No scenario receives a coherence penalty because its paths are separated or because its selected frame is non-PF.

The hardware interpretation is two distinguishable memory modes, A and B, co-moving with the emitter. The pair is created once at the epoch's local coordinate time zero. Each mode is released after its own computed hold. In the retained one-sided scheduler one hold is zero; the mathematical channel supports both nonnegative holding durations.

A hold is not a new entangled-pair generation event. If an experimental controller waits and only then creates the pair, that waiting time does not expose a nonexistent state to storage noise. Such a preparation policy requires a different event model. Likewise, a long optical fiber is not automatically equivalent to the emitter-local memory assumed here: its motion, attenuation, polarization transfer and distributed exposure would need their own model.

The extension does not create a new predictive PF scheduling controller. Scenario 3 continues to share Scenario 2's releases, receptions and storage history. Their exact quantum states and rates therefore remain equal under equal inputs, even when the storage channel is enabled. A future comparison of distinct physical control policies must first supply distinct, causally specified schedules.

19.2 Physical basis and scope of the model

Experiments establish that photonic-qubit memories have hardware-dependent coherence and retrieval efficiency. Körber and colleagues report a single-atom cavity memory whose coherence is protected through mapping between different internal bases. Vernaz-Gris and colleagues report a polarization-qubit memory and explicitly distinguish conditional fidelity from storage-and-retrieval efficiency. These results motivate modeling both quantities separately. They do not calibrate the satellite apparatus in this package.[1]

The implemented `independent_markovian_memory` model assumes constant, nonnegative dephasing and survival-decay rates in each memory's proper time, independent between the two modes and defined in the calibrated

[1] M. Körber et al., *Decoherence-protected memory for a single-photon qubit*, arXiv:1712.03668. P. Vernaz-Gris et al., *Highly-Efficient Quantum Memory for Polarization Qubits in a Spatially-Multiplexed Cold Atomic Ensemble*, arXiv:1707.09372, Nature Communications **9**, 363 (2018).

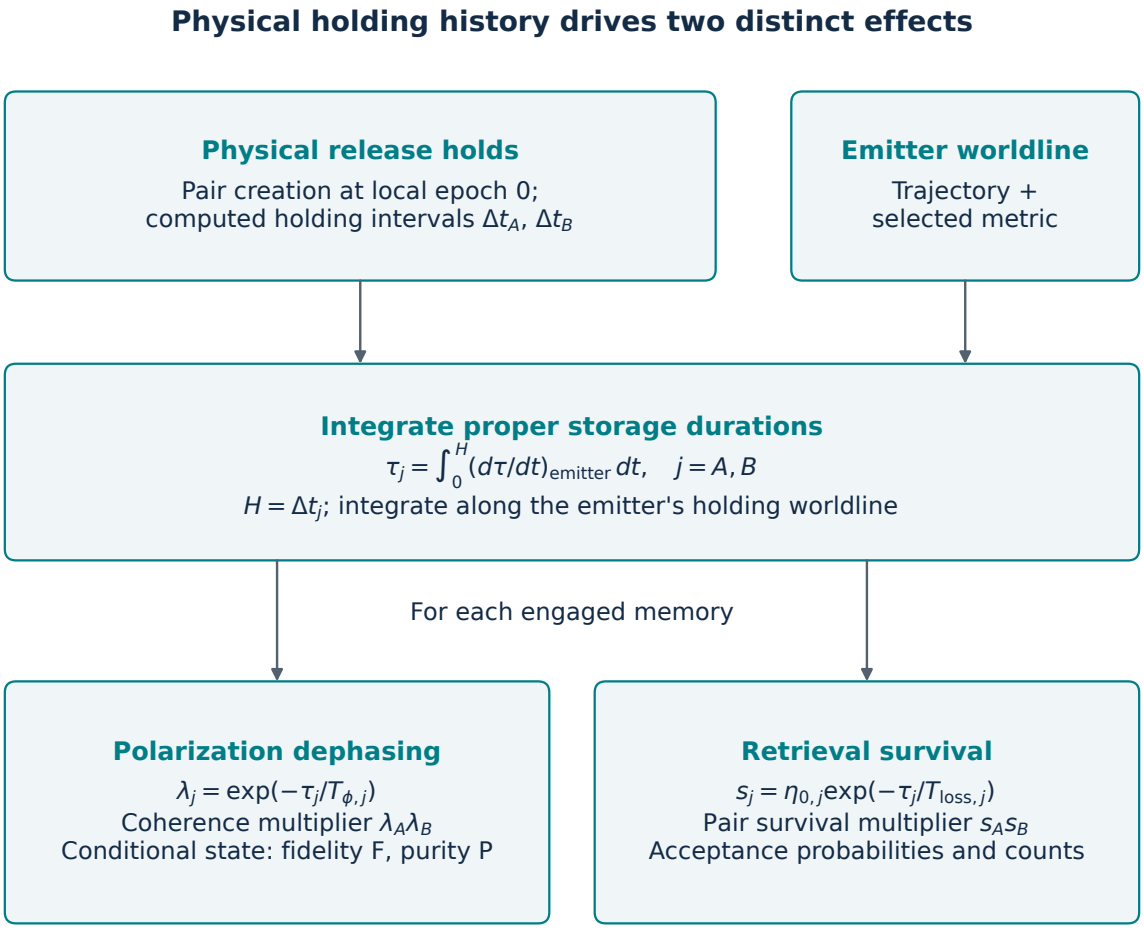

Figure 7: Storage is evaluated after the physical release schedule is known. Proper time drives separate dephasing and retrieval-survival calculations. Dephasing changes the conditional state; polarization-independent retrieval loss changes the collection budget. Frame labels do not enter either memory law.

local H/V bases. Polarization-basis transport along a photon path is not added by this model. This gives exponential laws and a completely positive polarization channel. It is an explicit phenomenological model, not a universal description of quantum memories. Echo sequences, correlated memory noise, nonexponential decay, polarization-dependent loss, background photons and time-varying control pulses are outside its scope.

The parameter-source string records the user's stated provenance. It is not checked against a measurement, instrument calibration or paper. The shipped millisecond examples are illustrative sensitivity settings. They are not fitted satellite-memory parameters, and their output is not an experimental prediction with certified uncertainty.

19.3 Proper storage time in SR and static GR

Let $h_j \geq 0$ be the computed coordinate-time hold for mode $j \in \{A, B\}$, measured from the epoch's pair-creation event. The storage duration is the proper elapsed time of the emitter memory:

$$\tau_j = \int_0^{h_j} \alpha_{\mathrm{e}}(t)\,\mathrm{d}t, \qquad \alpha_{\mathrm{e}}(t) = \frac{\mathrm{d}\tau_{\mathrm{e}}}{\mathrm{d}t} > 0.$$

The integration uses the same emitter state callback and spacetime mode as the event calculation. It does not substitute the receiver's clock, a selected-frame reception gap, a PF-coordinate time or the photon flight time for the memory's elapsed time.

For `flat_sr`, with emitter coordinate velocity $\mathbf{v}_{\mathrm{e}}(t)$,

$$\alpha_{\mathrm{e}}(t) = \sqrt{1 - \frac{\|\mathbf{v}_{\mathrm{e}}(t)\|^2}{c^2}}.$$

This expression applies along the prescribed timelike emitter trajectory; the emitter need not have constant velocity. For constant speed it gives $\tau_j = h_j/\gamma_{\mathrm{e}}$. The integral is of the material memory's worldline. It does not assign proper aging to a photon on a null path.

For `gr_static`, use the same areal Schwarzschild chart as the photon and clock module. With $f = 1 - r_{\mathrm{s}}/r$, radial velocity v_r, and tangential velocity $\mathbf{v}_T$,

$$\alpha_{\mathrm{e}}(t) = \sqrt{f(r(t)) - \frac{v_r(t)^2/f(r(t)) + \|\mathbf{v}_T(t)\|^2}{c^2}}.$$

Both the static gravitational lapse and kinematic contribution are included. Multiplying a gravitational factor by a Lorentz factor computed from an arbitrary coordinate speed would not generally reproduce this metric expression. At a stationary fixed radius, $\tau_j = h_j\sqrt{f}$. In the flat limit, the SR expression is recovered. The model requires the trajectory to remain in the supported exterior region and timelike throughout the integration.

The current implementation uses 32-point Gauss–Legendre quadrature and a 16-point comparison. For nodes z_k and weights w_k on $[-1, 1]$, the n-point estimate is

$$\tau_j^{(n)} = \frac{h_j}{2}\sum_{k=1}^{n} w_k\,\alpha_{\mathrm{e}}\!\left(\frac{h_j}{2}(z_k + 1)\right), \qquad n \in \{16, 32\}.$$

The 32-point node times, mapped integration weights, sampled positions and velocities, and clock rates are retained for audit. The 16-point comparison is stored as a duration scalar, together with the absolute difference; its full node set is not saved. The acceptance tolerance is $10^{-15}\,\mathrm{s} + 10^{-10}|\tau_j^{(32)}|$. Agreement of two quadrature orders is a numerical consistency check, not a rigorous error bound for an arbitrary discontinuous trajectory. The supplied smooth orbit providers and supported hold intervals define the intended use. Invalid or unavailable trajectory samples cannot be silently replaced by a unit clock rate.

19.4 Independent local dephasing

For each memory, the input `t_phi_s` is a proper-time pure-dephasing constant $T_{\phi,j} > 0$, measured in seconds. The surviving qubit's H/V coherence is multiplied by

$$\lambda_j = \exp(-\tau_j/T_{\phi,j}), \qquad 0 \leq \lambda_j \leq 1.$$

A JSON `null` dephasing time disables this term, giving $\lambda_j = 1$. It is not interpreted as a zero-second coherence time. Zero and negative numeric time constants are invalid.

The normalized, trace-preserving polarization operation is

$$\mathcal{D}_j(\rho) = \frac{1+\lambda_j}{2}\rho + \frac{1-\lambda_j}{2} Z\rho Z, \qquad Z = \mathrm{diag}(1,-1).$$

Its Kraus operators are

$$K_{j,0} = \sqrt{\frac{1+\lambda_j}{2}}\,\mathbf{I}_2, \qquad K_{j,1} = \sqrt{\frac{1-\lambda_j}{2}}\,Z, \qquad \sum_{m=0}^{1} K_{j,m}^{\dagger} K_{j,m} = \mathbf{I}_2.$$

The off-diagonal single-qubit element is multiplied by λ_j; the populations do not change. Nonnegative Kraus weights establish complete positivity, and the completeness relation establishes trace preservation. The convention is explicitly a coherence multiplier: there is no hidden factor of two in $T_{\phi,j}$.

In the package's B A basis order, the pair operation is $\mathcal{D}_B \otimes \mathcal{D}_A$. It composes with the configured Gaussian phase channel and fixed relative phase. These diagonal phase operations commute; the independent memory contributions multiply the relevant Bell coherence by $\lambda_A\lambda_B$. They do not modify the configured Gaussian covariance or replace its parameter provenance.

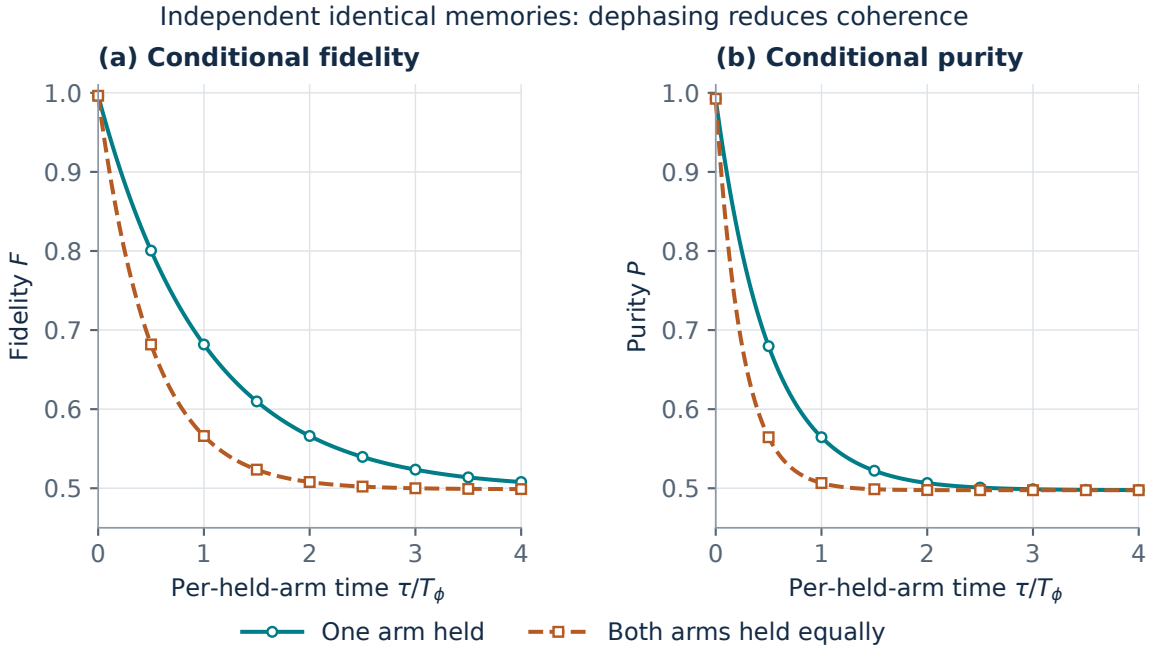

Figure 8: Analytic channel illustration, not a satellite run. Identical independent memories, zero base phase noise, zero deterministic phase, and source depolarization $p = 0.005$ give these conditional metrics. One held arm gives contrast $e^{-\tau/T_\phi}$; two equally held arms give $e^{-2\tau/T_\phi}$. The retained one-sided scheduler normally uses the first structure. No loss is included in this figure.

19.5 Retrieval survival and zero-hold bypass

The independent retrieval-survival model is

$$s_j = \eta_{0,j}\exp(-\tau_j/T_{\mathrm{loss},j}), \qquad 0 \le \eta_{0,j} \le 1.$$

Here $\eta_{0,j}$ is `retrieval_efficiency`, a fixed storage-and-retrieval survival factor, and $T_{\mathrm{loss},j}$ is `lifetime_s`. A null lifetime removes the duration-dependent decay, leaving $s_j = \eta_{0,j}$. This lifetime describes photon retrieval, not a polarization-qubit amplitude-damping time that drives one polarization toward the other.

The joint survival multiplier is $s_A s_B$. Conditional on survival, polarization-independent retrieval loss does not change the normalized polarization state. A larger erasure-space description could represent the lost-photon sector explicitly; this package instead keeps its probability in the collection budget. It does not renormalize a partially lost two-qubit matrix and call the resulting operation amplitude damping.

`bypass_zero_hold` determines the apparatus interpretation at exactly zero hold:

- With the default `true`, a zero-held arm bypasses its memory: $\lambda_j = s_j = 1$, even if its configured retrieval efficiency is less than one. Any strictly positive resolved hold engages that memory.
- With `false`, both arms undergo the storage/retrieval interface, including at zero duration. Then $\lambda_j = 1$ at zero time but $s_j = \eta_{0,j}$. Two zero-duration arms can therefore have pair survival $\eta_{0,A}\eta_{0,B} < 1$.

This discrete bypass policy can produce a jump in rate between zero and an arbitrarily small positive hold when $\eta_{0,j} < 1$. That is a specified switch in optical routing, not a discontinuity in the exponential decay law. Choose the flag to match the intended apparatus.

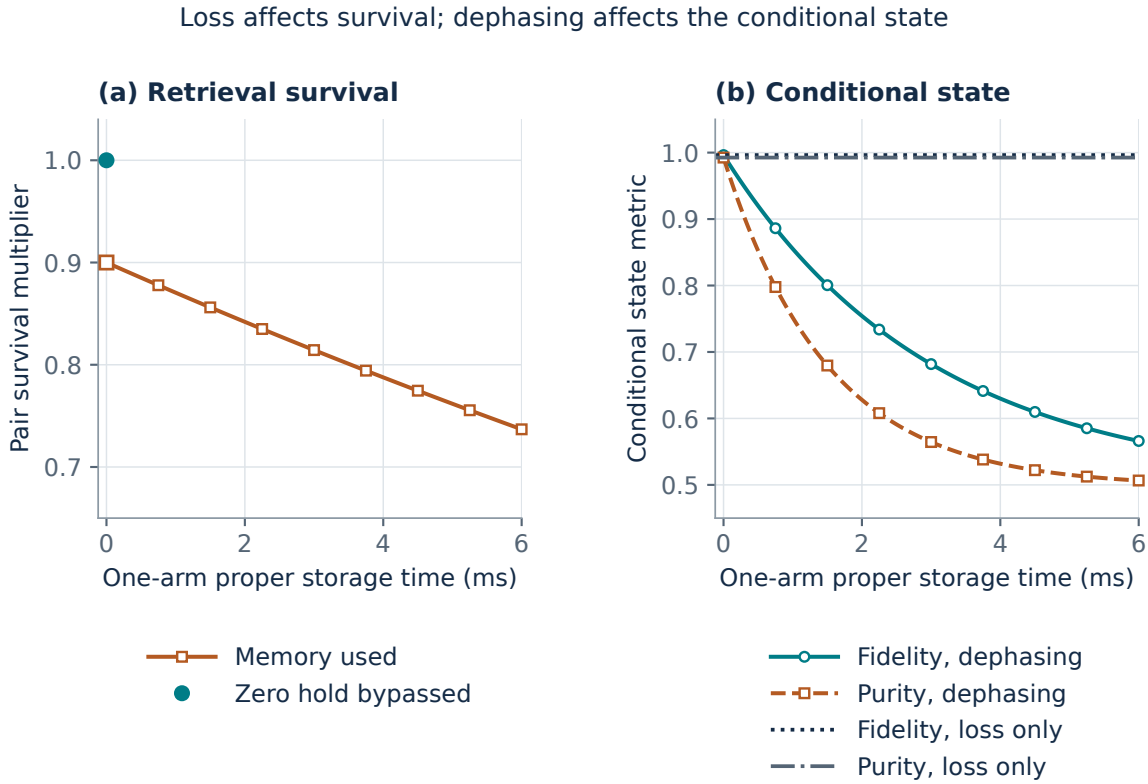

Figure 9: Illustrative one-arm memory with $T_\phi = 3\,\mathrm{ms}$, $T_{\mathrm{loss}} = 30\,\mathrm{ms}$, $\eta_0 = 0.9$, $p = 0.005$, zero base phase noise and zero fixed phase. Panel (a): retrieval changes the rate multiplier. The filled circle at zero hold indicates bypass, and the open square indicates engagement of the memory interface. Panel (b): loss alone preserves the conditional state, whereas dephasing lowers its metrics. These curves are analytic sensitivity illustrations, not fitted hardware data.

19.6 Combined Bell metrics and accepted counts

Let G denote the base Gaussian Bell-coherence factor defined in Chapter 5; it is C_{Ψ^-} for the singlet or C_{Φ^+} for the other supported target in that chapter's notation. The total contrast is

$$C = G\lambda_A\lambda_B.$$

For source depolarization p, deterministic relative phase θ, and the corresponding pure Bell target, the analytic checks become

$$\boxed{F = (1-p)\frac{1 + C\cos\theta}{2} + \frac{p}{4}}, \qquad \boxed{P = (1-p)^2\frac{1+C^2}{2} + \frac{1-(1-p)^2}{4}}.$$

These expressions apply to the supported Bell-family source and diagonal channels. They are not general formulas for an arbitrary two-qubit state. The code still calculates the density matrix and its metrics; the formulas provide an independent numerical reference.

At zero dephasing and zero deterministic phase, the default source gives $F = 0.99625$ and $P = 0.99251875$. With one arm at $\tau = T_\phi$, the memory contrast is e^{-1}, even though the coordinate hold can differ from that proper duration. At complete Bell-coherence loss, $C \to 0$, the limits are

$$F \to \frac{1}{2} - \frac{p}{4}, \qquad P \to \frac{1}{4} + \frac{(1-p)^2}{4}.$$

For $p = 0$ the limiting state is a mixed two-component Bell-diagonal state with $F = P = 1/2$, rather than the maximally mixed state with purity $1/4$. Dephasing removes coherence but does not equalize every population. A fixed unitary phase can change fidelity while leaving purity unchanged; it should not be interpreted as random decoherence.

Let $p_{\mathrm{acc}}^{\mathrm{pre}}$ be the applicable accepted probability before memory retrieval, including collection, gating and visibility. The revised probability and rate are

$$p_{\mathrm{acc}} = s_A s_B\, p_{\mathrm{acc}}^{\mathrm{pre}}, \qquad R_{\mathrm{acc}} = R_{\mathrm{src}}\, p_{\mathrm{acc}}.$$

The same multiplier applies to the controlled-unobstructed and Earth-respecting budgets where those budgets are defined. The stored pre-storage values make that multiplication auditable. Photon bandwidth, gate width and detector jitter are unchanged by this memory model. There is no assumed coupling between polarization dephasing and the acceptance selection within an epoch. Hardware with such a correlation requires a joint conditional model.

The rate remains per the declared source-run coordinate second, as in the existing count model. Applying a proper-time decay law does not silently convert the source repetition rate to pairs per proper second, or impose memory capacity, recycling time or duty-cycle constraints. The configured source rate assumes sufficient parallel modes or memory capacity to support that workload; it is not a prediction of achievable throughput for one physical memory.

19.7 Complete configuration reference

Omitting `storage_channel`, or setting its `enabled` member to `false`, preserves the existing storage-disabled calculation. The disabled mode keeps scenario-result schema 5. Explicitly enabling storage activates schema 6 and requires a nonempty `parameter_source` string. The named model is the only supported one in this release; arbitrary model names are rejected.

JSON member	Default / domain	Meaning
`enabled`	`false`; Boolean	Apply the storage extension.
`model`	`independent_markovian_memory`	Independent exponential memory channels.
`parameter_source`	Required nonempty string when enabled	Stated calibration or explicit illustrative assumption. Not independently verified.
`bypass_zero_hold`	`true`; Boolean	Skip memory use for exactly zero-held arms.
`arms.a`, `arms.b`	Separate objects	Parameters for the corresponding photon/memory mode.
`t_phi_s`	`null`, or finite positive seconds	Pure-dephasing coherence decay time. Null disables dephasing.

JSON member	Default / domain	Meaning
`lifetime_s`	`null`, or finite positive seconds	Retrieval-survival decay time. Null disables duration-dependent loss.
`retrieval_efficiency`	`1.0`; finite number in $[0, 1]$	Fixed survival for an engaged memory interface.

The next object belongs at the same level as `quantum_channel`, `spacetime` and `velocities`. It is not nested inside `quantum_channel`, which contains the existing polarization and collection settings. This illustrative block uses the same memory constants as the supplied SR and GR examples; the rest of each preset supplies its orbit and collection settings.

```
"storage_channel": {
  "enabled": true,
  "model": "independent_markovian_memory",
  "parameter_source": "Illustrative memory parameters; not calibrated",
  "bypass_zero_hold": true,
  "arms": {
    "a": {
      "t_phi_s": 0.002,
      "lifetime_s": 0.02,
      "retrieval_efficiency": 0.95
    },
    "b": {
      "t_phi_s": 0.003,
      "lifetime_s": 0.03,
      "retrieval_efficiency": 0.90
    }
  }
}
```

To isolate memory dephasing, set both lifetimes to null and both retrieval efficiencies to 1. To isolate retrieval loss, set both `t_phi_s` values to null. To obtain an enabled identity channel, set both time constants to null and both efficiencies to 1; this exercises the audit path while leaving physical predictions unchanged. Disabling the channel is a separate choice and produces the legacy-compatible schema-5 result.

Units are seconds, not milliseconds. A 2-ms time is `0.002`; a 2-second time is `2`. Do not use strings such as `"2 ms"` or nonfinite values such as `Infinity`. Use JSON null for the disabled term. Unknown storage keys, invalid parameter types and unsupported model names are configuration errors. A successful run should still be checked against its recorded effective configuration.

19.8 Run SR, GR and tracking cases

The storage parameters are configuration-driven. No new command-line switch silently changes their calibration. Start with the supplied presets and use the normal spacetime option for the selected metric:

```
run_python38.bat --scenario all --config config_storage_sr.json ^
  --spacetime flat_sr --output results_storage_sr

run_python38.bat --scenario all --config config_storage_gr.json ^
  --spacetime gr_static --output results_storage_gr
```

These commands use the standalone scenario runner and produce `scenario1.json`, `scenario2.json`, `scenario3.json` and the normal run artifacts. Use an output folder distinct from earlier runs. Changing a configuration file does not update an existing saved result or its graph; rerun the scenario calculation.

For live tracking, copy the desired live configuration and add the same top-level object. For example, if that edited file is named `config_live_storage.json`, run:

```
run_python38.bat --scenario all --config config_live_storage.json ^
  --live --live-iterations 10 --spacetime gr_static ^
  --output results_live_storage_gr
```

The edited live filename above is an instruction to create your own configuration, not another shipped preset. Normal TLE/GP acquisition, IERS coverage, ephemeris age and exterior-ray visibility requirements still apply. A successful storage configuration does not make an Earth-blocked GR reception available.

The full comparison workflow also passes the top-level storage object to its scenario branches. Copy the relevant comparison preset, add the object and use the existing `run_comparison.py` command. Each branch uses its own propagated emitter trajectory and computed holds. Compare the same physical epoch and inspect its rate-time convention; a difference between numerical and SGP4 storage predictions can reflect their underlying orbit histories.

To compare the existing selected-frame and base-coordinate timing policies, repeat an otherwise identical run with `--timing selected_frame` and `--timing eci`, each in a separate output folder. Inspect actual holds, not just frame names. These existing policies can change physical storage time. Neither option turns Scenario 3 into a predictive full-PF scheduler.

19.9 Read schema-6 results and unavailable outcomes

Each enabled epoch includes a `storage` record. Read its status before interpreting the quantum or count fields. Successful application records the per-arm holding history, proper durations, memory factors and numerical integration evidence. It also records the pair survival multiplier and the declared source of the parameters. Per-arm keys include `coordinate_hold_s`, `proper_hold_s`, `interaction_applied`, `coherence_factor`, `survival_probability`, and `proper_time_quadrature`. The aggregate `total_coherence_factor` is the memory-only product, and `pair_survival_probability` is the memory-only retrieval product. The configured provenance is explicitly marked `parameter_source_verified: false`.

Record / quantity	What to inspect
`metadata.schema_version`	6 for enabled storage; 5 when disabled.
`storage.status`	Applied or unavailable; never infer success from the configuration alone.
`storage.arms.a`, `storage.arms.b`	Coordinate hold, integrated proper time, coherence factor, survival and quadrature evidence for each mode.
`quantum.storage_coherence_factors`	Separate A/B coherence multipliers passed to the polarization calculation.
`quantum.analytic_coherence_factor`	Base Gaussian contrast; its established meaning is preserved.
`quantum.analytic_storage_coherence_factor`	Memory product $\lambda_A\lambda_B$.
`quantum.analytic_total_coherence_factor`	Combined contrast $G\lambda_A\lambda_B$.
Geometry's `pre_storage_*` quantities	Collection probabilities before the memory survival multiplier. Reconstruct a pre-storage rate using the source pair rate.
Exact and sampled state metrics	Exact values describe the conditional channel; tomography still requires accepted detections.

A useful inspection order is: effective storage configuration; release holds; proper durations; per-arm bypass status; coherence factors; survival factors; combined contrast; exact density matrix; expected rate; then sampled estimates. Verify the two products separately. A low rate with unchanged fidelity can be correct for a loss-only memory. Lower fidelity and purity at unchanged rate can be correct for a dephasing-only memory.

If the scheduler cannot resolve the physical release holds, the enabled storage channel is unavailable too. Exact storage-dependent fidelity, purity and state entries are null rather than fabricated from zero holding time. If

obstruction already establishes zero collection, the count budget remains zero; otherwise an unresolved required factor leaves it unknown. Null means unavailable, not zero fidelity or zero purity. The disabled base channel can still be specified without this additional physical history, so its legacy behavior is intentionally different. Invalid trajectory samples, non-timelike motion or failure of the quadrature convergence check are calculation errors; the runner stops rather than silently converting them into successful identity channels.

A modeled conditional state can exist while retrieval probability is zero, for example with a zero retrieval efficiency and a known holding history. It is then a formal conditional-channel prediction with no collected sample, not an experimentally accessible tomography estimate. This differs from an unknown holding history, where the state itself cannot be computed under the enabled model.

The four-panel comparison plot continues to display exact conditional fidelity, purity, rates and the specified Gaussian correlation. That correlation panel is not the total memory contrast and need not change when storage decoherence changes. Inspect the saved coherence factors alongside the plot. No general spatial or temporal closeness rule has been reintroduced. Comparison summary rows additionally expose storage status, proper holds, coherence and survival so that orbit-branch comparisons can be traced back to their physical memory histories.

19.10 Worked SR and GR output

The package includes five-epoch illustrative runs under `verification/r7/storage_sr` and `verification/r7/storage_gr`. The following table reads their first epoch with the supplied memory settings, default source $p = 0.005$, zero specified base phase noise and zero fixed phase. It displays computed predictions, not measurements.

Case	Hold A (ms)	Proper A (ms)	Fidelity	Purity
SR, Scenario 1	0	0	0.996250000	0.992518750
SR, Scenario 2/3	1.238303420	1.238303419	0.766604480	0.640998295
GR, Scenario 1	0	0	0.996250000	0.992518750
GR, Scenario 2/3	1.238303421	1.238303420	0.766604480	0.640998295

Mode B has zero hold in these examples. In the first SR Scenario 2 record, the unrounded proper hold is approximately 0.001238303419492834 s, and

$$\lambda_A \approx 0.5384009642, \qquad s_A \approx 0.8929644885, \qquad \lambda_B = s_B = 1.$$

The first-epoch expected accepted pair rate is approximately $142870.331\,\mathrm{s}^{-1}$ for Scenario 2 and Scenario 3, compared with $159995.535\,\mathrm{s}^{-1}$ for the bypassed Scenario 1. The GR values differ slightly below the displayed state-metric precision: both scheduling and the proper-clock integral contribute. These small SR/GR differences are not evidence that one spacetime implementation is incorrect or that a PF improves the channel.

The mechanism is directly traceable. The separated schedule holds A in the emitter memory; the declared dephasing law reduces its Bell coherence, and the retrieval law reduces collection probability. Scenario 1's zero-held modes bypass the memory under the explicit routing policy. Scenario 3 inherits Scenario 2's complete history, so it inherits the same storage consequences. Neither comparison uses Scenario 1's state as a numerical benchmark or introduces a penalty for spatial separation.

To reproduce an individual record analytically, use its full-precision `proper_hold_s`, not the rounded table. Compute the two exponentials, multiply the Gaussian and memory contrasts, evaluate the Bell formulas, and reconstruct the pre-storage accepted rate as `source_pair_rate_hz` times `pre_storage_accepted_probability`. Multiply that rate by the memory survival product. The supplied validators automate those consistency checks. Preserve the original JSON while making any independent audit.

19.11 Controls, comparisons and interpretation

The named `phase_noise_off` control turns off both the configured Gaussian dephasing and the optional storage dephasing. It keeps the source mixture and deterministic phase. The `ideal` control also removes those

remaining state imperfections. The `phi_plus` and `fixed_phase` controls retain the actual memory coherence factors. These controls calculate alternative conditional states; they do not rewind the physical schedule or produce new count budgets with loss removed.

For a physical storage on/off experiment, rerun with a copied configuration and change `storage_channel.enabled`. For a loss-only or dephasing-only study, use the null-time settings described above. Keep source quality, base phase covariance, gating, orbit inputs, sampling budgets and seeds fixed when testing one mechanism. Keep the source string honest about whether the parameters are measured or illustrative.

Scenario 1 may have zero holds and therefore bypass storage, whereas a separated layout may require one positive hold. A resulting difference then arises from different memory exposure under the specified control policy. It is not computed from geometric closeness to Scenario 1, and Scenario 1's result is never inserted as the target density matrix. The fidelity target remains the selected ideal Bell state.

Equal numerical values can also remain correct. Examples include storage disabled; null dephasing times; zero holds with bypass; equal proper exposure to equal memories; or Scenario 2 and Scenario 3 sharing the same complete physical history. Increasing precision on a graph cannot reveal a difference absent from the model inputs.

Coordinate changes alone preserve the proper storage duration of a fixed memory worldline and hence this channel's predictions. Different physical schedules can instead have different durations and outcomes. The relevant causal chain is schedule, memory exposure, channel and measurement. A coordinate-selected simultaneity label is not an extra noise source or a protection mechanism in the implemented equations.

19.12 Validation, limits and defensible conclusions

The revision-7 regression run on Python 3.12.14 passed 695 tests and 208 subtests, including 63 tests in the three new storage-focused test files. Two expected warnings came from historical time-data fixtures. Integration checks covered SR and GR, offline comparison branches and live-loop execution using a controlled tracking provider. Python 3.8 syntax compatibility was checked for the changed Python modules; this was not a new Python 3.8 runtime execution, live-network acquisition, hardware experiment or browser-rendering certification.

The validation separates parameter domains, mathematical channel identities and integrated workflow behavior. The delivered tests and validation reports document the actual execution evidence; their scope should not be expanded into an empirical validation claim.

Validation layer	Required relationship or limit
Configuration	Finite positive decay times or null; bounded efficiencies; Boolean enable/bypass flags; declared parameter source.
Proper time	Constant-speed SR and fixed-radius GR agree with closed-form clock rates; changing coordinates of the same worldline leaves elapsed proper time unchanged.
Quadrature audit	Stored 32-point nodes/rates reconstruct that integral; the 16-point duration is a recorded comparison scalar. This checks numerical consistency, not ephemeris truth.
Channel structure	Kraus completeness, Hermiticity, unit trace, positive semidefiniteness and agreement with the combined Bell formulas.
Ideal limits	Disabled storage reproduces prior behavior; enabled null times and unit efficiencies produce the identity memory; zero hold follows the explicit bypass policy.
Loss separation	Retrieval survival scales accepted probabilities and rates without altering the normalized conditional density matrix.
Scenario integration	Shared Scenario 2/3 history gives identical storage and quantum results; no scenario-index or baseline-distance penalty enters the memory law.
Unavailable inputs	Unknown holding history cannot be converted into a successful zero-decoherence result.

The reproducible vector figures in this chapter are generated by `generate_figures.py`, using only the stated analytic assumptions. They illustrate mechanisms rather than serving as evidence from a satellite run. Their PDFs are included in the source package; Python is not required to compile this manual.

For quantitative hardware use, measure conditional polarization transfer and retrieval probability against actual proper storage duration over the intended operating range. Check whether an exponential model and independent memories fit those data, report uncertainties, and avoid counting the same loss or dephasing twice in the base and storage parameters. The current model has no automated calibration fitting, memory saturation, dead time, cross-talk, dynamical decoupling or non-Markovian history.

This extension permits physically stated timing-dependent decoherence and loss studies. It does not demonstrate a PF advantage, force a separated-path degradation, or establish a new law of quantum dynamics. Those conclusions would require an explicit predictive scheduling comparison and independent physical evidence.

Chapter 20

Configurable QKD, ground links and evidence benchmarks

The protocol layer introduced in edition 1.4.0 turns explicitly modeled optical events and conditional quantum states into *key-length forecasts*. It supports entanglement-based BBM92 between satellite receivers, entanglement distribution from one satellite to two ground stations, and three-intensity decoy-state BB84 on a single satellite-to-ground arm. Each calculation retains the assumptions that connect geometry to counts and counts to a secret-key estimate. The output contains no operational cryptographic key, key-management service, implemented authenticated classical conversation, or proof that a laboratory device satisfies the security assumptions.

This distinction is necessary even when a reported key length is positive. A positive forecast means that the stated model and statistical bound leave a positive length after the specified deductions. It is not a record of exchanged secret bits. A completed calculation can return zero key because of low counts, large error, insufficient test data or finite-size penalties. Unavailable geometry instead leaves the count/key estimate unresolved; a null result is not zero key.

A completed key calculation with no usable net key reports an abort/no-key reason, including when authentication consumes an otherwise positive gross budget. Inspect both `secret_key_bits` and `abort_reason`; computational completion alone is not a usable-key result. The SatQuMA mission adapter follows the same common-schema convention.

Security probabilities are evaluated with stable logarithmic expressions where possible. Positive input probabilities whose required allocated tails or auxiliary probabilities cannot be represented by the supported floating-point calculation are rejected explicitly. They are not silently enlarged, clipped to a convenient security level or converted into a successful key forecast.

20.1 Scope, compatibility and the three scenarios

The original scenario, orbit-comparison and storage workflows remain available. The legacy top-level `satellite_to_ground` switch in the original runner remains unsupported there; it is not an alias for the new ground-QKD configuration. Use the new QKD workflow described below. A ground-link configuration and a satellite-link configuration must not be compared as though only their protocol name had changed. They have different receivers, geometry, optical losses and generally different accepted-event histories.

Scenario 1 retains its role as the coincident-worldline baseline. Scenario 2 retains the earlier separated-detector frame selector with the corrected physical event handling. Scenario 3 performs the full PF diagnostic construction on Scenario 2's physical history. For the same protocol, detector settings, density matrix, randomization policy and physical schedule, Scenario 2 and Scenario 3 must have the same QKD forecast. A different coordinate diagnostic cannot independently improve the secret-key rate. A genuinely different emission schedule would need to be specified as a new experiment before any difference could be interpreted causally.

The QKD optical-arm efficiencies replace the legacy scalar collection efficiencies in the separate QKD report; they do not multiply those legacy efficiencies a second time. The QKD layer uses the existing separation between conditional state quality and polarization-independent loss. Optional storage changes the conditional state through dephasing and the probability of detection through retrieval survival, as described in Chapter 19. For BBM92 these are applied once in the physical pipeline. The separate decoy pulse source does not support the entangled-pair memory model; that combination is rejected. The key estimator must not apply a second hidden storage penalty or a scenario-number penalty.

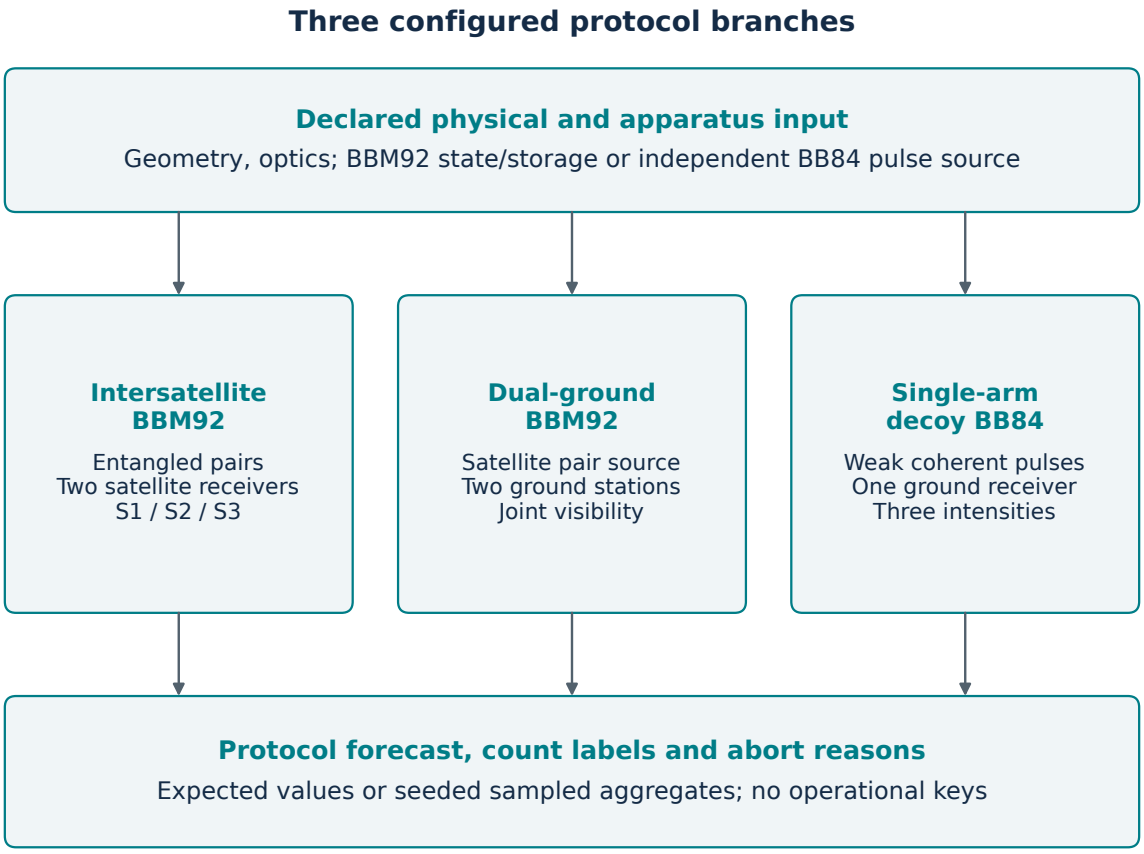

Figure 10: The QKD extension has three configurable branches. The shared physical inputs precede the protocol estimate. Choice of protocol does not create an extra PF emission law or replace the recorded physical history. This is an architecture diagram, not measured experimental data.

Branch	Physical experiment	Protocol interpretation
Intersatellite BBM92	An entangled pair is delivered to two satellite receiver modes.	Z-basis matches provide the modeled raw key; X-basis matches estimate phase error.
Dual-ground BBM92	One satellite pair source sends one photon to each of two rotating stations.	Both arms must be usable at compatible times; joint detections define the accepted pair sample.
Single-arm decoy BB84	Phase-randomized weak coherent pulses reach one selected receiver.	Signal, weak-decoy and vacuum intensities bound the single-photon contribution; this is not a two-photon coincidence calculation.

20.2 Inputs, configuration and runnable examples

Run these examples from the simulation-code directory, after completing the installation in Chapter 3. Each command uses a complete supplied configuration, rather than the fragments below. Use a fresh output directory for each study.

```
run_python38.bat --config config_qkd_intersatellite.json ^
  --scenario all --output results_qkd_intersatellite

run_python38.bat --config config_qkd_ground.json ^
  --scenario all --spacetime flat_sr --output results_qkd_ground_sr

run_python38.bat --config config_qkd_ground.json ^
  --scenario all --spacetime gr_static --output results_qkd_ground_gr

run_python38.bat --config config_qkd_decoy.json ^
  --scenario all --output results_qkd_decoy

run_python38.bat --config config_qkd_micius_type.json ^
  --scenario all --output results_qkd_micius_type

python qkd_micius_benchmark.py --output results_micius_benchmark.json

python qkd_micius_integration.py --output results_micius_integration.json
```

On Linux or macOS, invoke `run_privileged_frame_simulation.py` with `.venv-pf38/bin/python` and the same arguments. Use your shell's continuation syntax, or put the command on one line. The Windows caret is a shell continuation character, not a Python option. The `config_qkd_micius_type.json` preset is an illustrative geometry and apparatus study. The separate benchmark command reads the frozen literature evidence and performs the declared consistency calculation. Neither command reconstructs an unavailable Micius ephemeris.

20.2.1 Top-level QKD configuration

Key under `qkd`	Value/role	Meaning
`enabled`	Boolean	Opts into the QKD calculation; existing non-QKD workflows remain available.
`protocol`	`bbm92` or `decoy_bb84`	Selects entangled-pair or weak-coherent-pulse protocol.
`basis_z_probability`	Probability	Independent probability of selecting Z at each endpoint; X uses the complement. Both bases need usable populations for finite estimation.
`integration_mode`	`trapezoid` or `independent`	Integrates a time sequence or accumulates explicit independent snapshot exposures.

Key under qkd	Value/role	Meaning
exposure_s	Positive seconds	Exposure assigned to each independent snapshot; it is not the time between timeline points.
seed	Integer	Controls the simulated sampling stream; it is not a cryptographic randomness source.
security	Object	Conservative-model security settings and the shared mission count-mode selection.
security_model	conservative or satquma	Selects a security calculation; the SatQuMA-compatible option applies to decoy BB84.
satquma_raw_key_basis	x or z	Selects the physical key population for that optional model.
satquma_security	Object	Separate reference-matched bound, epsilon, leakage and authentication settings; Section 21.6.
optical_links.a, optical_links.b	Arm objects	Explicit collection/detector models for the corresponding modes.
decoy	Object	Single-arm pulse source, intensity/probability choices and source/detector assumptions.

trapezoid requires at least two ordered epochs and positive total duration. It uses the sampled rates to estimate one accumulated block. independent assigns exposure_s to every supplied snapshot; it is appropriate only when those exposures are the intended experiment. The key bound is evaluated once on the accumulated block, not once per numerical integration point. Changing the grid must not accidentally change the number of independent security penalties. Live one-epoch updates require independent-exposure semantics.

Rates and exposures use the model's coordinate seconds: ECI time in SR and Schwarzschild time normalized at infinity in the static-GR branch. They are not automatically emitter proper-time rates. If a measured source rate is specified per proper second, convert it consistently before comparing against this coordinate-time count ledger.

The original runner also accepts --qkd-protocol bbm92 or --qkd-protocol decoy_bb84, --qkd-count-mode expected or --qkd-count-mode sampled, --qkd-exposure SECONDS, and --no-qkd. Selecting a protocol enables the extension. An explicit exposure override selects independent snapshot exposures. These flags do not replace the need to configure the physical source and optical arms. The legacy --sample-counts switch controls the original tomography workload; it is separate from the QKD protocol count mode.

The following is an illustrative QKD fragment to merge into an existing complete configuration. It does not define the orbit profiles, source quantum state, pair brightness or run interval.

```
"qkd": {
  "enabled": true,
  "protocol": "bbm92",
  "basis_z_probability": 0.5,
  "integration_mode": "trapezoid",
  "seed": 20260917,
  "security": {
    "finite_key": true,
    "count_mode": "expected",
    "ec_efficiency": 1.16,
    "authentication_bits": 128
  },
  "optical_links": {
    "a": {"mode": "gaussian"},
    "b": {"mode": "gaussian"}
  }
}
```

20.2.2 Security keys and defaults

Key under `qkd.security`	Default	Validation and interpretation
`finite_key`	true	Boolean; false selects the asymptotic reference estimate.
`count_mode`	expected	`expected` or `sampled`; the label describes the counts used in the bound.
`epsilon_secrecy`	10^{-9}	Secrecy-budget cap; effective allocations are recorded.
`epsilon_correctness`	10^{-12}	Correctness cap entering the verification deduction.
`epsilon_parameter_estimation`	10^{-10}	Cap on the parameter-estimation allocation.
`epsilon_privacy_amplification`	10^{-10}	Cap on the privacy-amplification allocation.
`epsilon_authentication`	10^{-12}	External authenticator assumption; no authentication protocol is executed.
`ec_efficiency`	1.16	Finite number at least one; modeled leakage overhead over the binary-entropy limit.
`ec_leakage_bits`	null	Optional nonnegative integer overriding the modeled leakage.
`authentication_bits`	128	Nonnegative integer cost per analyzed block.

Every epsilon must be finite and strictly between zero and one half. Unknown security keys are rejected, so a misspelled cap does not silently fall back to a default. An explicit leakage budget should represent the whole intended transcript cost under the chosen security argument.

20.2.3 Ground receiver configuration

Ground receivers are configured at top-level `ground_stations`. The keys `detector1` and `detector2` match the scenario receiver roles. A single-arm decoy run selects the relevant optical arm but still uses the declared physical scenario geometry.

```
"ground_stations": {
  "detector1": {
    "latitude_deg": 0.0, "longitude_deg": -5.0,
    "height_m": 0.0, "earth_model": "sphere",
    "rotation_mode": "analytic", "earth_rotation_phase_rad": 0.0
  },
  "detector2": {
    "latitude_deg": 0.0, "longitude_deg": 5.0,
    "height_m": 0.0, "earth_model": "sphere",
    "rotation_mode": "analytic", "earth_rotation_phase_rad": 0.0
  }
}
```

These are illustrative station coordinates, not recovered experiment coordinates. The engine uses `earth_model="sphere"` to match its opaque Earth surface. Latitude lies in $[-90, 90]$ degrees, longitude in $[-180, 180]$ degrees, and height is nonnegative. The default height is zero. `rotation_mode="auto"` selects Astropy orientation when a reference UTC epoch exists and analytic rotation otherwise. `analytic` uses an explicitly stipulated inertial phase and a default rate of $7.2921150 \times 10^{-5}\,\mathrm{rad\,s^{-1}}$. `astropy` requires the UTC epoch and valid Earth-orientation data. `iers_auto_download` and `allow_degraded_iers` both default to false. Opting into degraded data changes the provenance and must be reported. The lower-level ground module can represent WGS84 in SR; that does not authorize mixing a WGS84 surface with the engine's spherical occultation or static-GR boundary.

20.2.4 Optical arm keys and conventions

Each `qkd.optical_links.a` or `.b` object has the following normalization defaults. Their physical meanings are developed in Section 20.4.

Arm key	Default	Meaning
`mode`	gaussian	`gaussian` or `measured_loss`.
`wavelength_m`	810×10^{-9}	Positive wavelength in metres.
`waist_m`	0.05	Positive intensity $1/e^2$ beam-waist radius.
`divergence_half_angle_rad`	Null derives $\lambda/(\pi w_0)$	Positive $1/e^2$ half angle when explicit; cannot be smaller than the configured diffraction limit. The GUI preserves null so later wavelength/waist edits update the derived value.
`receiver_aperture_radius_m`	0.5	Nonnegative circular aperture radius.
`pointing_jitter_rad`	0	RMS angular jitter per transverse axis.
`pointing_bias_rad`	0	Magnitude of mean angular pointing offset.
`aperture_method`	analytic	Noncentral-chi-square integral or independent `quadrature` calculation.
`optical_efficiency`	0.8	Additional terminal transmission in Gaussian mode.
`detector_efficiency`	0.6	Photon detection probability; counted once.
`min_elevation_deg`	10	Ground-path geometric elevation mask, between 0 and 90 degrees.
`zenith_optical_depth`	0.1	Nonnegative extinction depth for the simple downlink atmosphere.
`background_rate_hz`	0	Detected background rate before dead time.
`dark_count_rate_hz`	100	Detected dark rate before dead time, for one effective detector per arm.
`dead_time_s`	0	Nonnegative nonparalyzable detector dead time.
`polarization_rotation_rad`	0	Configured deterministic local analyzer rotation.
`measured_loss_db`	Required in scalar mode	Nonnegative aggregate attenuation; includes all optical/channel losses.
`loss_includes_detector_efficiency`	Required Boolean in scalar mode	True avoids multiplying detector efficiency again; false applies it once afterward.

Using `measured_loss` does not disable visibility checks. It replaces the Gaussian optical-factor product with the supplied aggregate scalar. The setting is useful for sensitivity or published-summary studies, but its name does not establish that the supplied number was actually measured for this run's geometry.

20.2.5 Single-arm decoy options

```
"decoy": {
  "arm": "a",
  "pulse_rate_hz": 100000000.0,
  "intensities": {"signal": 0.6, "decoy": 0.15, "vacuum": 0.0},
  "probabilities": {"signal": 0.8, "decoy": 0.15, "vacuum": 0.05},
  "misalignment_probability": 0.01,
  "detection_window_s": 1e-9,
  "timing_jitter_s": 0.0, "timing_offset_s": 0.0,
  "detector_mode": "gated"
}
```

This object belongs under qkd, with protocol="decoy_bb84". The intensity labels are literal names, their probabilities are positive and sum to one. The default vacuum intensity is zero; a nonzero weakest decoy is also allowed when $\mu_{\text{signal}} > \mu_{\text{decoy}} + \mu_{\text{vacuum}}$ and $\mu_{\text{decoy}} > \mu_{\text{vacuum}} \geq 0$. The selected arm determines the single received optical channel. Pulse rate is not the entangled-pair source rate. Misalignment is a configured bit-error probability, and the detection-window duration converts detected background rates into per-pulse click probabilities under the model. detector_mode is gated by default; free_running also includes background outside the detection gates in the dead-time occupancy. Detection windows must not overlap: $R_{\text{pulse}}\Delta t_{\text{gate}} \leq 1$. Both detector modes reject dead-time occupancy at or above 0.01.

The single-arm gate integrates a Gaussian arrival-time distribution with nonnegative standard deviation timing_jitter_s and signed mean offset timing_offset_s, both zero by default, over the full detection_window_s. Zero jitter uses deterministic acceptance. These pulse-source timing parameters are distinct from the entangled-pair coincidence gate. A published timing width with an unspecified convention must not be silently treated as this standard deviation.

20.2.6 Protocol result fields

The optional extension has extension_schema_version=1. Existing scenario schema 5 (without storage) and schema 6 (with storage) are retained. A scenario file stores individual samples at epochs[i].qkd, the complete accumulated block at qkd_summary, and the estimator fields below at qkd_summary.security. The block's duration_s and secret_key_rate_hz provide the explicit rate denominator and result. If an epoch is unresolved, the block remains unresolved and security can be null; a null value is not a zero-bit successful estimate.

In sampled mode, per-epoch qber_z/qber_x are empirical fractions of simulated errors and detections. expected_qber_z/expected_qber_x retain the Born-model expectations. Optical count budgets remain labeled expectations. sampled_total_coincidences includes simulated basis-mismatched arrivals as well as matched ones. At block level total_coincidences follows the selected count mode, while expected_total_coincidences retains the optical expectation. These distinctions prevent a sampled error fraction from being paired with an unlabeled expected denominator.

Field	Interpretation
protocol, finite_key, count_mode	Identifies the selected estimator and kind of input population.
n_z, n_x, m_z, m_x	Matched-basis detections and corrected errors; expected values may be fractional.
qber_z, qber_x	Error fractions within their named detected populations; null when undefined.
phase_error_upper, phase_sampling_delta	Phase-error bound and the sampling contribution; read with the proof assumptions.
ec_leakage_bits, verification_bits, privacy_amplification_penalty_bits	Separate deductions from the useful entropy budget.
gross_secret_key_bits	Nonnegative estimate before the configured authentication consumption.
authentication_bits, secret_key_bits	Authentication consumption and net nonnegative forecast.
key_fraction_sifted	Net key estimate divided by the selected raw-key population: n_z for the conservative model or raw_key_count for SatQuMA, whose raw_basis identifies X or Z; zero when that population is zero.
security_status, security_assumptions, epsilon_budget	Forecast/simulated-count label, assumptions and effective security allocation.
aborted, abort_reason	Explicit zero-key outcome and reason; preserve in scans and reports.
operational_key_generated	Always false; a forecast does not generate deployable secret bits.

20.3 Ground worldlines, local elevation and visibility

A ground receiver is a worldline, not a fixed point in the inertial orbital chart. The station specification provides geodetic latitude, longitude and height. The integrated engine uses a spherical Earth of radius 6371000 m, matching its opaque body and the static-GR areal surface. On this sphere geodetic and geocentric latitude coincide. Earth rotation then determines the station's time-dependent position and velocity in the model's inertial chart. Record the epoch and orientation convention along with the station coordinates; a longitude without a rotation epoch does not determine an inertial observing direction. The lower-level station module also provides WGS84 coordinates for coherent SR use, but the integrated scenario path does not mix that ellipsoid with the spherical ray boundary.

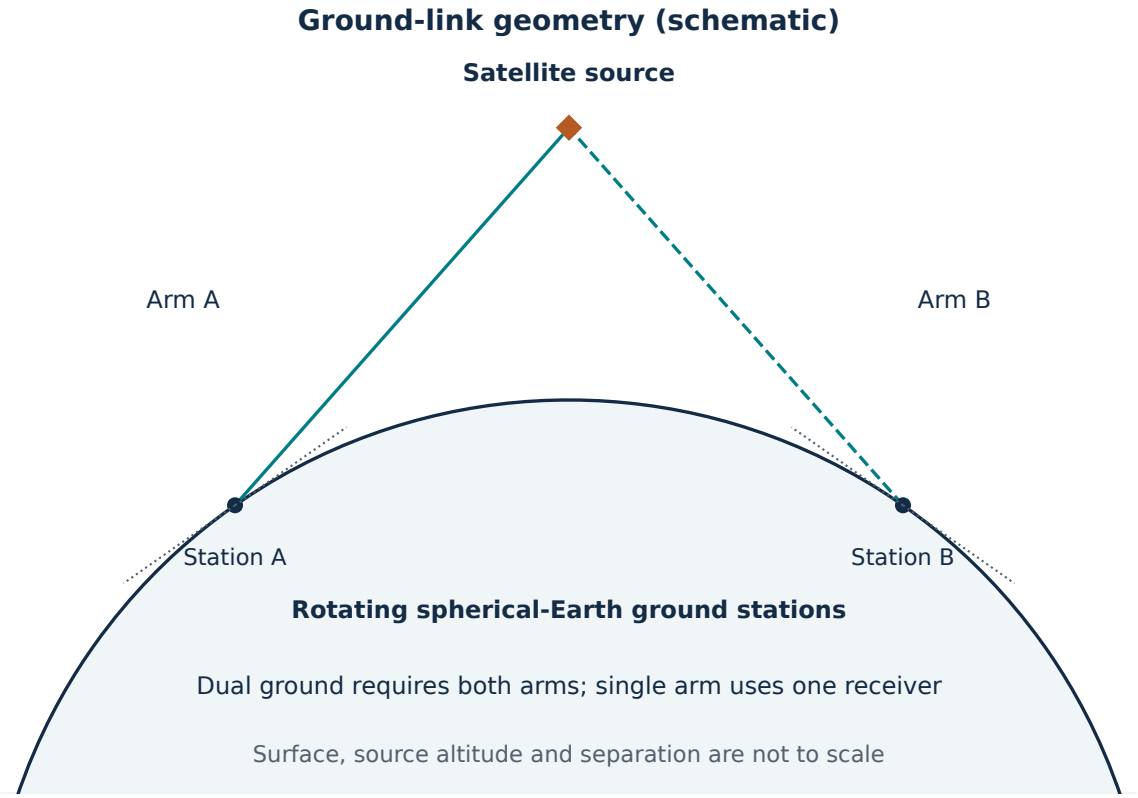

Figure 11: A dual-ground geometry has two distinct slant ranges and local elevations. Each link is checked against its own station horizon and elevation mask. The drawing is not to scale and does not reconstruct a Micius orbit. A single-arm BB84 run uses one receiver and one optical arm. Dotted lines indicate local tangents.

Let $x_g(t_d)$ be the station reception position, $x_s(t_e)$ the satellite emission position, and $\hat{u}_g(t)$ the station's outward geodetic normal. The endpoint-chord slant vector is $d = x_s(t_e) - x_g(t_d)$, its length is $L = \|d\|$, and the elevation is

$$e = \arcsin\left(\hat{u}_g(t_d) \cdot \frac{d}{L}\right). \tag{20.1}$$

The production link calculation respects these selected emission and reception events. Its elevation is a geometric endpoint-chord angle; it does not add atmospheric refraction or local velocity aberration. In GR, the existing ray status remains authoritative for propagation availability. A Euclidean endpoint separation, a geodesic path length, and a flight time are different quantities and retain their saved labels.

An elevation mask excludes low-elevation paths even when the line of sight is above the ideal surface. The mask is an operational assumption about usable observation geometry; it is not a model of local buildings, terrain, clouds, or telescope scheduling. Dual-ground operation requires the joint usable interval. Adding the individual visible durations can overstate the interval in which both photons from a pair are received. A zero jointly visible interval must not be converted into a successful zero-noise channel measurement.

20.3.1 SR and static-GR scope

The flat-SR branch uses moving station and satellite endpoints with the existing special-relativistic event and clock conventions. The `gr_static` branch uses the existing static spherical exterior model and its event-coordinate adapter. Station rotation is part of the receiver worldline; it does not turn the gravitational field into a rotating-Earth metric. Frame dragging, multipole gravity, atmospheric refraction, terrain, weather and a full geodetic time-transfer service are not introduced merely by selecting the GR mode.

Proper-time storage continues to integrate along the relevant emitter worldline under the selected spacetime convention. A static-GR run can therefore change timing and storage exposure without changing the declared apparatus noise model. It is essential to retain the model's time coordinate, proper duration and source-rate convention separately. The run record identifies the assumptions actually evaluated; an SR/GR difference is a model comparison, not a new calibration of station clocks.

20.4 Optical probability and detector assumptions

An optical model connects emitted trials to received events. Its declared factors include geometric collection, pointing or beam broadening, atmospheric transmission on ground paths, terminal transmission, detection efficiency and timing acceptance. Keep each factor dimensionless and bounded by one. A loss in decibels is a power/probability attenuation:

$$\eta = 10^{-\mathcal{L}/10}, \qquad \mathcal{L} = -10 \log_{10} \eta. \tag{20.2}$$

Two independent optical arm transmissions multiply. Their attenuation in decibels adds. This distinction matters for dual-ground BBM92, where a seemingly modest loss on each arm can make the pair coincidence probability small.

The Gaussian intensity radius is the $1/e^2$ radius. For range L, waist radius w_0, divergence half angle θ, and angular pointing jitter σ_θ per transverse axis,

$$w(L) = \sqrt{w_0^2 + (\theta L)^2}, \qquad s^2 = \frac{w(L)^2}{4} + (L\sigma_\theta)^2. \tag{20.3}$$

The default $\theta = \lambda/(\pi w_0)$ is the configured diffraction limit, and smaller values are rejected. For a centered circular receiver of radius a, the mean collected fraction is

$$\eta_{\mathrm{ap}} = 1 - \exp\left(-\frac{a^2}{2s^2}\right). \tag{20.4}$$

A nonzero mean pointing displacement is integrated with the noncentral-chi-square cumulative distribution; numerical radial quadrature is available as an independent calculation of the same aperture integral. This is a mean collection model for Gaussian jitter, not a stochastic turbulence or scintillation simulation.

The optical calculation is configurable rather than a claim of calibrated hardware throughput. Aperture, divergence, wavelength, pointing, extinction and efficiency must be documented with their conventions and provenance. A fitted net attenuation and explicit component losses must not be applied to the same path twice.

The downlink atmosphere uses $\eta_{\mathrm{atm}} = \exp[-\tau_z/\sin(e)]$ at positive geometric elevation, with zenith optical depth τ_z. Vacuum intersatellite arms do not receive this atmospheric factor. Uplinks are outside this extension's scope. Ground atmospheric transmission is only evaluated in its supported elevation domain. An airmass approximation can become unreliable near the horizon even before a numerical singularity occurs. An elevation mask and an explicit model domain provide a more interpretable configuration than extrapolating a simple law through the horizon.

Detector dark/background settings are event rates or per-gate probabilities as named in the configuration. A coincidence window in seconds cannot be treated as a dimensionless rate multiplier without the corresponding singles rates. Background coincidences are modeled as uncorrelated outcomes under the stated model; they can lower the observed correlation even when the normalized signal state is ideal. Timing jitter, spectral filtering, memory retrieval and gating describe different mechanisms. Their effects must remain identifiable in the count budget.

For continuous pair births the accidental-coincidence estimate is $R_{\mathrm{acc}} = R_{A,\mathrm{singles}} R_{B,\mathrm{singles}} \Delta t$, where Δt is the *full* coincidence window and the singles are the observed rates after the modeled nonparalyzable dead time. True coincidences use pair brightness, two arm efficiencies, timing acceptance and the two live fractions. Treating detector readiness as independent is an approximation for joint events; low dead-time and window occupancies are part of the stated regime. The integrated QKD workflow rejects a singles-window or dead-time occupancy at or above 0.01 rather than extending this approximation into a dense-click regime. This expression counts unrelated matches and does not implement a high-occupancy unique-click pairing algorithm or detector afterpulsing.

20.5 Normalized density matrices and basis errors

BBM92 uses the normalized conditional two-qubit density matrix already produced by the quantum channel. It does not obtain QBER by substituting $1 - F$ into a key-rate formula. Fidelity is one scalar overlap with a target state; two states with the same fidelity can distribute their errors differently between measurement bases.

For basis $b \in \{Z, X\}$, let $\Pi_a^{(b)}$ and $\Pi_c^{(b)}$ be the binary projectors for Alice and Bob. The implementation respects the Qiskit subsystem ordering $|BA\rangle$:

$$p_b(a, c) = \mathrm{Tr}\left[\rho \left(\Pi_c^{(b)} \otimes \Pi_a^{(b)}\right)\right]. \tag{20.5}$$

The probabilities must be real within numerical tolerance, nonnegative within the admissible floating-point tolerance, and sum to one. Hermiticity, unit trace and positive semidefiniteness are prerequisites for interpreting ρ as a conditional state.

Define $f_b \in \{0, 1\}$ as the documented Bob-bit flip for the target Bell correlation. The basis error is

$$q_b = \sum_{a,c:\ a \neq (c \oplus f_b)} p_b(a, c). \tag{20.6}$$

The singlet $|\psi^-\rangle$ is anticorrelated in both X and Z, so Bob is flipped in both bases. The $|\phi^+\rangle$ target is correlated in both, so neither basis is flipped. Other Bell targets require their own declared parity map. The parity convention belongs in the result, especially when comparing a published figure with an internally different basis-label convention.

Target	Expected Z / X parity	Bob flip Z / X
$\|\phi^+\rangle = (\|00\rangle + \|11\rangle)/\sqrt{2}$	Correlated / correlated	0 / 0
$\|\phi^-\rangle = (\|00\rangle - \|11\rangle)/\sqrt{2}$	Correlated / anticorrelated	0 / 1
$\|\psi^+\rangle = (\|01\rangle + \|10\rangle)/\sqrt{2}$	Anticorrelated / correlated	1 / 0
$\|\psi^-\rangle = (\|01\rangle - \|10\rangle)/\sqrt{2}$	Anticorrelated / anticorrelated	1 / 1

The public source channel offers `singlet` and `phi_plus`; the measurement helper also supports the other two Bell states for convention and algebra checks. For an ideal singlet, both corrected QBERs vanish even though the unflipped outcomes disagree. This is an effective test for a mistaken parity convention. A dephasing channel can preserve Z correlations while degrading X correlations; a single fidelity-derived error number would conceal that distinction. Source-state preparation, local polarization rotations and storage all act before the basis Born probabilities are calculated.

Loss does not change a normalized conditional density matrix merely by reducing the number of accepted trials. Background contamination can change the normalized *observed mixture* when independent random events enter the accepted sample. This is a separate, stated detector model, not an instruction to reduce the trace of ρ.

20.6 Every count has a sampling definition

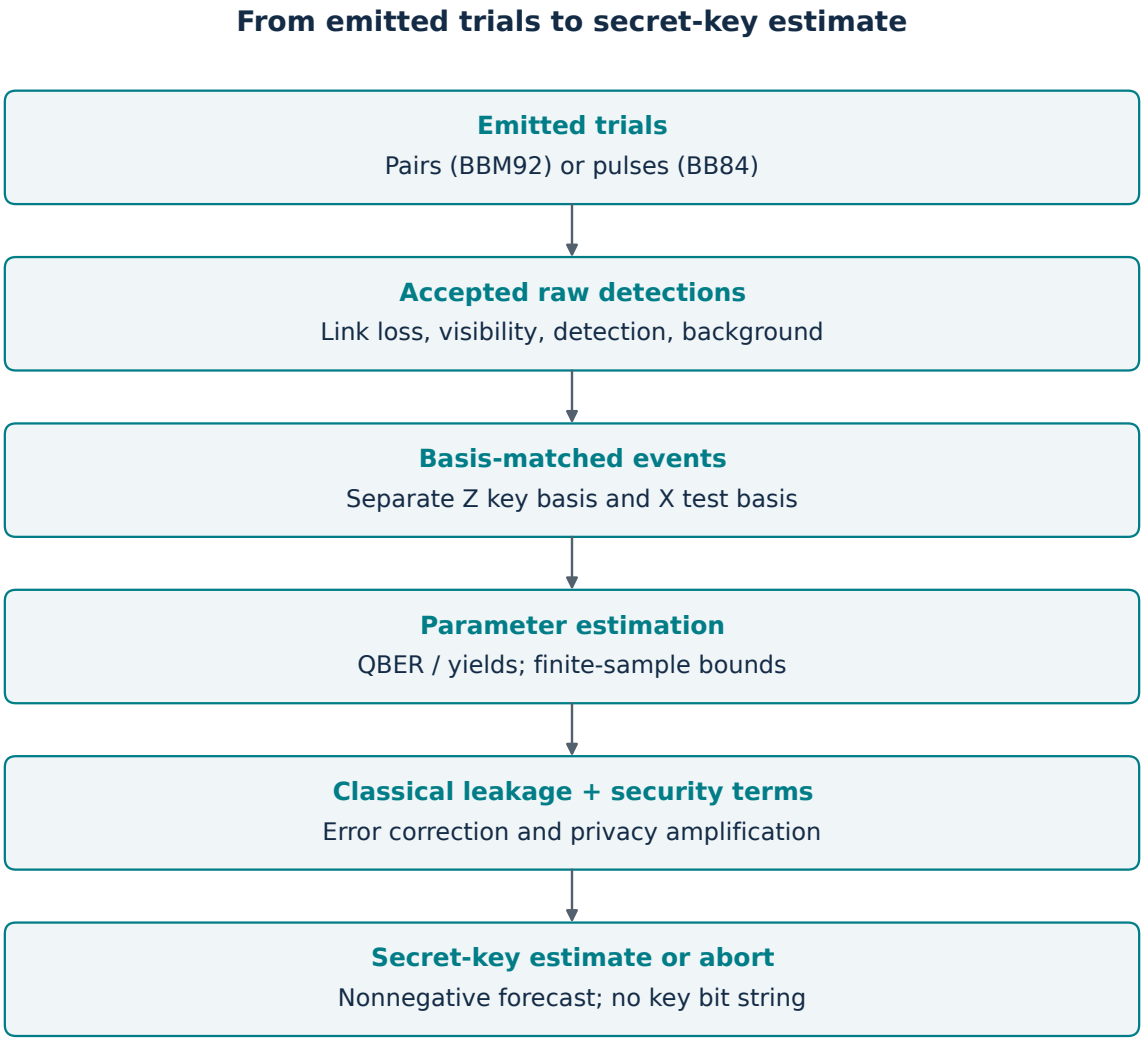

Figure 12: The count pipeline distinguishes emitted trials, accepted detections, basis-matched data, parameter-estimation bounds and the final secret-key estimate. BBM92 emitted trials are pairs; BB84 emitted trials are pulses. A rate or fraction is meaningful only with its denominator and integration duration. All displayed outputs remain forecasts.

An emitted pair rate, an accepted coincidence rate, a sifted-bit rate and a secret-key rate are not interchangeable. For a pair source with rate R_p, accepted signal probability η_{AB}, integration duration T, and independent Z-basis choices $p_{Z,A}, p_{Z,B}$, a simple stationary expectation is

$$\bar{n}_Z = T R_p \eta_{AB} p_{Z,A} p_{Z,B}. \tag{20.7}$$

Time-varying simulations accumulate the segment expectations instead. The X-basis test count uses the corresponding X-choice probabilities. An unbalanced basis choice can increase the nominal key-basis sample while making phase-error estimation weaker. In a small data block, the finite-size penalty can dominate any apparent sifting advantage.

Quantity	Definition	Common interpretation error
Emitted trials	Pairs for BBM92 or pulses for decoy BB84 over the stated interval.	Calling pulses, pairs and detected photons the same count.
Raw accepted events	Coincidences for BBM92; clicks for single-arm BB84 before basis selection.	Treating all raw events as key bits.

Quantity	Definition	Common interpretation error
Z matches n_Z	Accepted events in the chosen key basis at both endpoints.	Including mismatched bases or X test events in the key.
X matches n_X	Accepted events in the test basis at both endpoints.	Taking a small X sample as an exact phase-error measurement.
Error counts	Disagreements after the declared parity correction in each basis.	Reading physical anticorrelation as error.
Secret-key estimate	Nonnegative post-deduction length from the selected estimator.	Calling a forecast an exchanged or authenticated key.
Key rate	Estimated length divided by the declared acquisition duration.	Dividing by visible time when the result uses total elapsed time.
Sifted fraction	Estimated length divided by the explicitly named sifted population.	Confusing a fraction of sifted bits with a fraction of emitted trials.

20.6.1 Expected and seeded sampled modes

`count_mode="expected"` uses model expectations, which can be noninteger. Such outputs answer a forecasting question: what count budget does this apparatus model predict? Applying finite-size penalties to expected values provides a planning estimate; it does not turn expectations into an observed finite data record.

`count_mode="sampled"` uses a declared random seed to generate synthetic aggregate detections and errors according to the implemented sampling model. Counts are then simulated observations of that model, not measurements from a satellite. The seed controls repeatability, not security. Repeating a seed while changing source parameters does not guarantee that every underlying trial is physically paired between experiments unless the sampling implementation explicitly provides that coupling.

Record the count mode, seed, integration duration, effective apparatus configuration and code release in every comparison. Use several seeds when studying simulated sampling variability; do not present the best seed as typical performance. A deterministic expected run is useful for isolating parameter changes, while sampled runs illustrate fluctuations. Neither measures uncertainty in an uncalibrated optical parameter.

20.7 BBM92 asymptotic and finite-block estimates

The binary entropy is

$$h_2(q) = -q\log_2 q - (1-q)\log_2(1-q), \qquad h_2(0) = h_2(1) = 0. \tag{20.8}$$

For an upper bound on a phase error, the conservative entropy envelope is one when the bound reaches or exceeds one half. Extending the decreasing branch of h_2 above one half would incorrectly reward a worse upper bound with a longer key.

In the asymptotic BBM92 forecast, the X-basis error estimates the phase-error contribution in the Z key. With error-correction efficiency $f_{\mathrm{EC}} \geq 1$, the familiar bulk term is

$$n_Z[1 - h_2(q_X) - f_{\mathrm{EC}}h_2(q_Z)]. \tag{20.9}$$

The implementation still reports explicitly configured leakage and authentication deductions. The result is clipped to a nonnegative integer key length. The asymptotic mode is a diagnostic reference with sampling penalties removed; it should not be described as the secure length of a short physical run.

The finite BBM92 calculation follows the random-sampling structure of Tomamichel and coauthors. Set $n = n_Z$, $k = n_X$, and let ϵ_s be the effective smoothing allocation. The implemented deviation is

$$\delta = \sqrt{\frac{(n+k)(k+1)}{nk^2}\ln\frac{1}{\epsilon_s}}, \qquad \phi_Z^U = \min(1, q_X + \delta). \tag{20.10}$$

The key-length forecast has the structure

$$\ell = \max(0, \lfloor n[1 - h_2^U(\phi_Z^U)] - \lambda_{\mathrm{EC}} - \lambda_{\mathrm{cor}} - \lambda_{\mathrm{PA}} - \lambda_{\mathrm{auth}} \rfloor), \tag{20.11}$$

where h_2^U is the conservative entropy envelope. When leakage is not supplied explicitly, $\lambda_{\mathrm{EC}} = \lceil f_{\mathrm{EC}} n h_2(q_Z) \rceil$. The correctness and privacy-amplification deductions are

$$\lambda_{\mathrm{cor}} = \left\lceil \log_2 \frac{2}{\epsilon_{\mathrm{cor}}} \right\rceil, \qquad \lambda_{\mathrm{PA}} = \left\lceil 2 \log_2 \frac{1}{2\epsilon_{\mathrm{PA}}} \right\rceil. \tag{20.12}$$

The authentication deduction is a separately configured bit budget. It represents a resource assumption; the forecast does not execute a message authentication protocol.

The configured epsilon values are caps. The implementation derives a stricter allocation where needed and records the effective budget. For BBM92, the smoothing allocation is bounded by both the parameter-estimation cap and the secrecy cap: $\epsilon_s = \min(\sqrt{\epsilon_{\mathrm{PE}}}, \epsilon_{\mathrm{sec}}/4)$. The effective privacy-amplification value is at most $\min(\epsilon_{\mathrm{PA}}, \epsilon_{\mathrm{sec}}/2)$. Read the recorded budget rather than summing configuration names as though each name were an independent, fully spent failure probability. The epsilon metadata is not a finite-count guarantee in asymptotic mode. Failure budgets also compose over distinct used blocks; a per-block epsilon is not a promise for an unlimited sequence of passes.

No finite estimate is available without a usable key sample and test sample. A nonpositive post-deduction expression produces zero key and an explicit status. Smaller epsilon caps demand stronger assurances under the model and usually reduce the estimate. They cannot compensate for an incorrect detector model or an uncharacterized source.

20.8 Three-intensity decoy BB84

The single-arm branch assumes independently phase-randomized weak coherent pulses. An intensity μ denotes the mean photon number, so its emitted photon-number distribution is

$$P(N = j \mid \mu) = e^{-\mu} \frac{\mu^j}{j!}. \tag{20.13}$$

Signal, weak-decoy and vacuum settings have their own selection probabilities. The vacuum setting is a source condition, not an instruction to suppress detector background. Background clicks at a vacuum input help constrain the vacuum yield.

The decoy argument requires that, conditioned on photon number, the states sent at the different intensities are indistinguishable to the adversary in the degrees of freedom relevant to the proof. An arbitrary intensity-dependent spectrum, timing tag, polarization flaw or side channel invalidates that assumption. The code does not test a physical source for these properties.

For a simple threshold detector with transmittance η and background-click probability Y_0, the optical response has the familiar structure

$$Q_\mu = 1 - (1 - Y_0)e^{-\eta\mu}. \tag{20.14}$$

The implemented detector response and its errors are recorded in the output; this equation explains why a received pulse count is not a two-arm coincidence count. Detector efficiency belongs in η once. A misalignment error and a random background contribution must remain separately identifiable.

20.8.1 Bounds and retained populations

The estimator uses the analytic three-intensity bounds described by Lim and coauthors. The implemented variant retains *all Z-basis intensity classes* as the raw-key population and uses all X-basis classes for testing. Do not compare that population with a signal-intensity-only formula without explicitly reconciling the sifting convention.

Write $s_{Z,0}^L$ for the vacuum-event lower bound, $s_{Z,1}^L$ for the single-photon Z-event lower bound, $s_{X,1}^L$ for the single-photon X-event lower bound, and $v_{X,1}^U$ for the single-photon X-error upper bound. Intensity selection

probabilities and Poisson weights connect observed intensity-class detections to these bounds. The implementation uses analytic expressions, not a finite photon-number cutoff silently treated as exact.

In finite mode a common tail allocation is

$$\epsilon_t = \min(\epsilon_{\mathrm{sec}}/21, \epsilon_{\mathrm{PE}}/13, \epsilon_{\mathrm{PA}}). \tag{20.15}$$

The Hoeffding deviations used for detection and error aggregates are

$$\Delta_n = \sqrt{\frac{n_b}{2}\ln\frac{1}{\epsilon_t}}, \qquad \Delta_m = \sqrt{\frac{m_b}{2}\ln\frac{1}{\epsilon_t}}, \tag{20.16}$$

where n_b and m_b are the total detections and errors in the relevant basis. The resulting lower and upper bounds are clipped only in the conservative direction. A vanishing single-photon test lower bound cannot be replaced with an assumed perfect phase estimate.

For the explicit analytic elimination, write the intensities as $\mu > \nu+\omega$, $\nu > \omega \geq 0$, with preparation probabilities p_j, and define

$$\tau_r = \sum_j p_j e^{-\mu_j}\frac{\mu_j^r}{r!}, \quad n_{b,j}^{\pm} = \frac{e^{\mu_j}}{p_j}(n_{b,j} \pm \Delta_n), \quad m_{X,j}^{\pm} = \frac{e^{\mu_j}}{p_j}(m_{X,j} \pm \Delta_m). \tag{20.17}$$

Negative lower observations are replaced with zero before the exponential/probability factor. The vacuum and single-photon lower bounds are

$$s_{b,0}^L = \max\left(0, \tau_0\frac{\nu n_{b,\omega}^- - \omega n_{b,\nu}^+}{\nu-\omega}\right), \tag{20.18}$$

$$s_{b,1}^L = \max\left(0, \frac{\tau_1\mu}{\mu(\nu-\omega)-\nu^2+\omega^2}\left[n_{b,\nu}^- - n_{b,\omega}^+ - \frac{\nu^2-\omega^2}{\mu^2}\left(n_{b,\mu}^+ - \frac{s_{b,0}^L}{\tau_0}\right)\right]\right). \tag{20.19}$$

The single-photon X-error bound is

$$v_{X,1}^U = \min\left(m_X, \max\left(0, \tau_1\frac{m_{X,\nu}^+ - m_{X,\omega}^-}{\nu-\omega}\right)\right). \tag{20.20}$$

Finite lower event bounds are rounded downward and error upper bounds upward. Incompatible aggregate bounds abort. Asymptotic mode sets the observation deviations and phase-sampling correction to zero and retains real-valued bounds until final key rounding. The supplied vacuum-plus-weak-decoy example takes $\omega = 0$; the formulas also permit a nonzero weakest decoy under the stated ordering constraints.

A conservative Serfling replacement for Lim's approximate phase-sampling expression uses the retained single-photon lower bounds:

$$\delta_1 = \sqrt{\left(\frac{1}{s_{Z,1}^L} + \frac{1}{s_{X,1}^L}\right)\left(1+\frac{1}{s_{X,1}^L}\right)\ln\frac{1}{\epsilon_t}}, \qquad \phi_Z^U = \min\left(1, \frac{v_{X,1}^U}{s_{X,1}^L} + \delta_1\right). \tag{20.21}$$

The implementation aborts when a required single-photon lower count is zero. The count-estimation tails contribute at most $12\epsilon_t$ and phase sampling adds at most ϵ_t^2, whose sum respects the parameter-estimation cap. The retained effective secrecy budget is $21\epsilon_t$. The finite key expression is

$$\ell = \max\Bigg(0, \Bigg\lfloor s_{Z,0}^L + s_{Z,1}^L[1-h_2^U(\phi_Z^U)] - \lambda_{\mathrm{EC}} - \left\lceil 6\log_2\frac{1}{\epsilon_t}\right\rceil - \left\lceil \log_2\frac{2}{\epsilon_{\mathrm{cor}}}\right\rceil - \lambda_{\mathrm{auth}}\Bigg\rfloor\Bigg). \tag{20.22}$$

The output reports the bound populations, phase bound, deductions, effective budget and abort reason. A finite result can be zero where the asymptotic forecast is positive because the decoy and X-test data do not tightly constrain the useful single-photon population.

20.9 Security assumptions, status and abort handling

The protocol formulas assume an authenticated classical channel, trusted and characterized endpoint devices, valid random basis and intensity choices, the source assumptions used by the selected proof, and a detector/sifting model compatible with that proof. BBM92 does not become device independent merely because a Bell-state source or a CHSH number appears elsewhere in the package. Decoy BB84 does not remove the need for phase randomization or characterization of source leakage.

The forecast does not implement error-correction messages, verification hashes, universal-hash privacy amplification, authentication, secret storage or an adversarial hardware audit. It estimates the costs of these operations under declared assumptions. No sampled bit string should be exported as a usable cryptographic key.

Outcome	Correct handling
No visible/usable path	Report the geometry/propagation status, zero signal and any modeled background counts; do not call the exact conditional state an observation.
No Z matches	No key population exists; return zero key with its reason.
No usable X/decoy test data	The finite phase or single-photon bound is unavailable or uninformative; abort the key estimate.
Phase bound or key-basis QBER at least one half	Return zero key with an explicit abort. The entropy envelope is one above one half; a possible positive vacuum term does not override this conservative abort.
Leakage exceeds useful entropy	Return zero after deductions and retain the leakage budget in the report.
Invalid configuration or state	Reject malformed, nonfinite, out-of-domain or physically invalid inputs; do not silently substitute a favorable value.
Positive forecast	Report the count mode, proof assumptions, effective epsilon allocation and complete deductions alongside the estimate.

A fully specified blocked SR link can retain modeled local background counts while its signal vanishes. An occulted GR schedule with missing single-arm streams instead leaves the QKD counts unresolved and null; `true_pair_rate_upper_hz=0` records only a signal upper bound. It does not authorize filling every missing count or rate with zero.

An abort is a scientific result of the specified forecast. It should be preserved in parameter scans. Dropping zero-key or unavailable samples and averaging only successful points biases the reported performance. The correct aggregation depends on the planned operational policy and must be stated before interpreting a mean rate.

20.10 The Micius attachment: what it can validate

The supplied `1707.01339v1` paper concerns satellite distribution of entanglement over a separation exceeding 1200 km. It is not a complete QKD execution record. Its reported fidelity, Bell-test statistic, coincidence rate and loss are evidence about the published entanglement experiment. They do not supply an observed secret-key length, a decoy-state intensity record, or a full composable-security transcript.

The release therefore separates three questions: can the numerical formulas reproduce an explicitly stated simplified model; are that model's summary predictions compatible with selected reported statistics; and can the original time-dependent satellite experiment be independently reproduced? The first can be checked directly. The second has a conditional answer. The third remains unresolved with the available attachment.

20.10.1 Source-audited quantities and unresolved labels

Quantity	Attachment locator	Role in the benchmark
Source fidelity 0.907 ± 0.007	PDF page 4	Imported input to the Werner-state surrogate, not a held-out channel measurement.
Signal-to-noise ratio about $8:1$	PDF page 6	Imported mixing assumption; using it prevents an independent validation of the noise model.
Two-photon attenuation 64–82 dB	PDF page 6	Reported operating range; no complete per-event loss trace is provided.
Coincidence rate about 1.1 Hz	PDF page 6	Aggregate reported rate; it does not determine two separate arm efficiencies.
Ground separation 1203 km	PDF page 3	Experiment geometry context, insufficient to reconstruct an orbit.
Combined link distance 1600–2400 km	PDF page 6	Reported range, not an ephemeris.
Jitter 0.77 ns; full window 2.5 ns	PDF page 5	Timing assumptions; retain the full-window convention.
Fidelity at least 0.87 ± 0.09	PDF page 7	Reported entanglement witness statistic for conditional comparison.
CHSH $S = 2.37 \pm 0.09$	PDF page 7	Reported Bell statistic; raw angle/time records are absent.
1167 Bell-test coincidences in 1059 s	PDF page 7	Aggregate count/time cross-check; not a full Bell-test data file.
Figure 5 bar counts	PDF page 16	Visually transcribed internal figure labels; their sum conflicts with the body count.

PDF page numbers here refer to the supplied v1 attachment. The released evidence JSON preserves the source locators, imported-input roles, uncertainty convention, internal inconsistencies and the benchmark status. When using another paper version, recheck the locators and labels rather than assuming that page or figure numbering stayed fixed.

Figure 5 contains Z-basis bars $(3, 35, 29, 2)$ and X-basis bars $(2, 30, 33, 2)$, in the figure's ordering. They total 136 events, whereas the body reports 134 for the fidelity sample. Its visible X-basis anticorrelation also conflicts with the nearby stated ψ^+/positive-X interpretation. The software retains these as unresolved source inconsistencies. It does not silently relabel the bars, adjust two counts, or treat a choice of Bell parity as a recovered raw-data fact.

Conditional on interpreting the bars as singlet anticorrelations, the two-basis lower-bound calculation is

$$F_{\mathrm{lower}} = \frac{64}{69} + \frac{63}{67} - 1 \simeq 0.86784. \tag{20.23}$$

This reproduces a value near the printed lower bound. A simple independent-binomial propagation from those displayed counts gives a different uncertainty from the paper's quoted 0.09; it is a conditional calculation on a transcription, not a reconstruction of the authors' uncertainty analysis. The count discrepancy and parity ambiguity remain visible in the final status.

20.10.2 A frozen-parameter Werner-state consistency calculation

A Werner surrogate about a declared target Bell state is

$$\rho_W = v|\Psi\rangle\langle\Psi| + (1-v)\frac{I_4}{4}, \qquad F = \frac{1+3v}{4}. \tag{20.24}$$

The source fidelity therefore sets $v_s = (4F_s - 1)/3 = 0.876$. Interpreting the reported signal-to-noise ratio as uncorrelated white admixture gives a signal fraction $w = 8/9$. Without fitting the reported final fidelity or CHSH value, the surrogate predicts

$$F_{\mathrm{pred}} = \frac{1}{4} + w\left(F_s - \frac{1}{4}\right) = 0.834, \tag{20.25}$$

$$S_{\mathrm{pred}} = 2\sqrt{2}\, w\, v_s \simeq 2.2024. \tag{20.26}$$

The CHSH expression is the optimal Werner-state value under the assumed measurement convention, not a reconstruction of the experiment's angle-by-angle, time-tagged measurement history.

Propagating only the quoted source-fidelity uncertainty while fixing the approximate SNR gives

$$\sigma_{F,\mathrm{pred}} = w\sigma_{F_s} \simeq 0.00622, \qquad \sigma_{S,\mathrm{pred}} = 2\sqrt{2}\, w\frac{4}{3}\sigma_{F_s} \simeq 0.02347. \tag{20.27}$$

These are conditional standard uncertainties. The paper does not supply a full uncertainty distribution for the imported SNR; assigning it zero uncertainty here is a computational convention, not a claim that it was measured exactly. Source-model error, drift, background structure and correlations between reported quantities are not included in these narrow prediction intervals.

The predictions are compatible with the reported fidelity and CHSH summary values within the declared two-standard-deviation comparison. This supports a limited consistency statement about the chosen surrogate. Because source fidelity and SNR were imported from the same experiment, it does not independently validate the satellite channel, demonstrate exact reproduction, or establish a QKD key rate. The overall replication status remains `UNRESOLVED`.

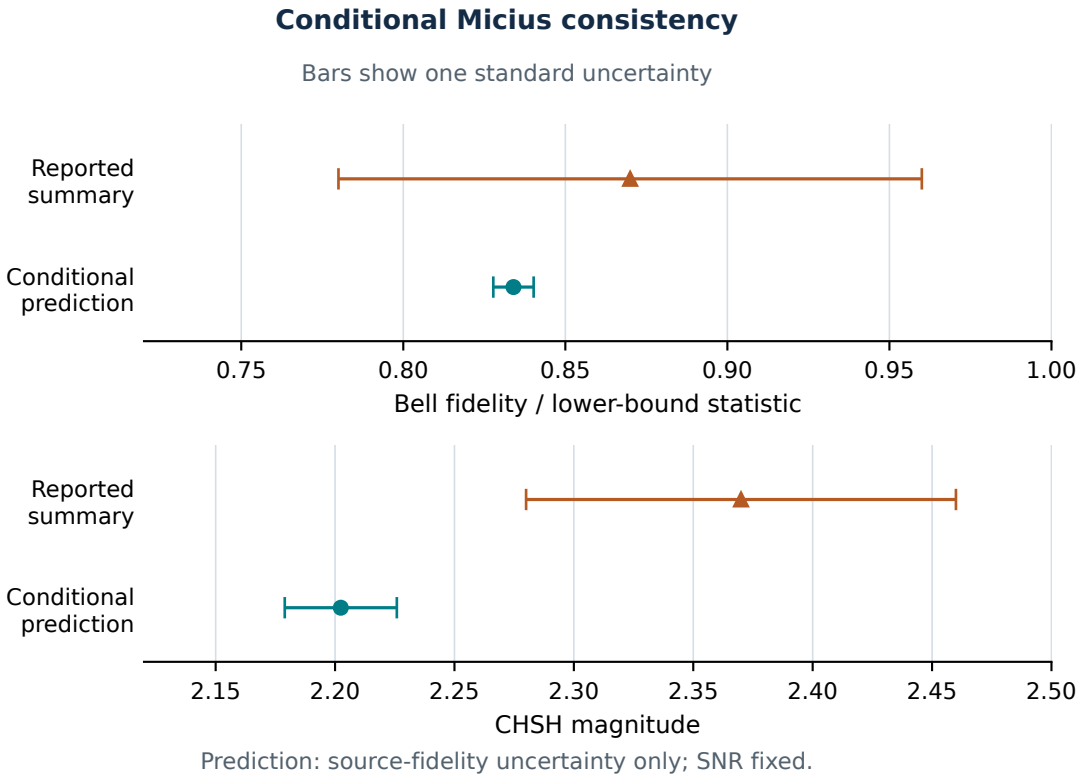

Figure 13: Conditional Werner-model predictions compared with the attachment's reported summaries. Bars show one standard uncertainty; the benchmark's stated diagnostic uses two combined standard deviations, with a one-sided comparison for the fidelity lower bound. Prediction uncertainty propagates source fidelity only while fixing the approximate imported SNR. These plotted intervals omit model and SNR uncertainty. The CHSH discrepancy is about 1.802 combined standard deviations. This is not an independent channel validation.

20.10.3 An end-to-end engine consistency test

The separate `qkd_micius_integration.py` benchmark invokes the actual scenario engine, Qiskit state propagation, optical/count pipeline and BBM92 key estimator. Its report retains all inputs and three complete engine runs. The controlled central fixture uses 250 coordinate seconds with two endpoint quadrature weights and 71 dB combined registered-photon loss, split equally between the arms. It does not modify the 21-epoch illustrative configuration used in Section 20.12.

The source fidelity and brightness are imported. The fixture interprets the reported 0.77 ns timing width as a Gaussian standard deviation and assumes a 5 GHz bandwidth. Those are declared fixture conventions; the paper does not establish that jitter-width interpretation. Storage, phase noise and detector dead time are absent in this diagnostic.

For the central loss only, the imported SNR fixes an effective arm-total background. With equal per-arm efficiency η, pair rate R, timing acceptance g, full window W and target signal-to-accidental ratio $r = 8$, the calibration is

$$B = \sqrt{\frac{R\eta^2 g}{rW}} - R\eta. \tag{20.28}$$

It gives about 2917.578 detected background events per second per arm. This is an explicitly calibrated input, not a prediction of sky background or an inferred count of physical detectors. No measured received fidelity or CHSH value enters that calibration.

The actual detected density matrix gives $F = 0.834$ and corrected $q_Z = q_X = 0.1106667$. An independent Pauli-correlation-tensor calculation of $S = 2\sqrt{\lambda_1 + \lambda_2}$, using the two largest eigenvalues of $T^{\mathsf{T}}T$, gives $S = 2.2024019$. Holding the calibrated background fixed at the 64/82 dB loss endpoints gives a modeled count-rate envelope of approximately $[0.0619969, 2.2132419]\,\mathrm{s}^{-1}$, containing the reported 1.1/s summary. This is an input-driven envelope, not a reconstructed pass average.

This zero-key result is not solely a finite-count effect. The assumed Werner-plus-white-accidental channel has symmetric error $q = 0.1106667$, for which even the ideal one-way asymptotic fraction

$$1 - 2h_2(q) \simeq -0.0038471 \tag{20.29}$$

is negative. Increasing the observation duration alone therefore does not create a positive key under this channel and this one-way bound. This conclusion does not exclude other protocols or a physically better channel.

The accumulated basis populations are approximately $n_Z = n_X = 29.5035$ and $m_Z = m_X = 3.26506$. The implemented finite estimator and an independent arithmetic check both return zero net forecast bits: the phase bound is one, with 18 EC bits, 41 verification bits, 65 privacy-amplification penalty bits and 128 authentication bits. The 20 declared software/budget checks pass for this fixture; deliberate source degradation and injected metric errors produce failed checks. The overall evidence status remains `UNRESOLVED`. Running the full software chain strengthens the implementation check while leaving the missing experimental evidence and imported-input limitations intact.

20.10.4 Later QKD experiments and security reanalysis

The 2017 entanglement-distribution attachment must not be conflated with the 2017 decoy-state QKD experiment or the 2020 entanglement-based QKD experiment. These use different observations and security analyses. Later work by Lim and coauthors revisited the finite-key analysis of the 2020 data and identified omitted parameter-estimation costs; this is a revised security interpretation of existing data, not a replacement empirical data set. The production conservative bound returning zero at the original security level is compatible with that conclusion but does not reproduce every threshold of the refined proof. The package does not claim an exact Micius key-rate replay. See Lim et al., 2021 and the separate benchmark scope in Chapter 21.

20.10.5 Evidence required for a stronger replication

A time-resolved replication would require the source and receiver ephemerides at the relevant epochs, station orientation and timing conventions, actual pass and acquisition masks, optical and detector calibration, source

brightness and multipair characterization, background records, time tags or bin-level counts, measurement settings, and the analysis choices used to combine them. The attachment does not provide that complete set. Missing data should remain missing rather than being replaced by an illustrative orbit and described as recovered.

For a QKD replication, further evidence would be needed: basis/intensity choices where applicable, error-correction leakage, parameter-estimation records, authentication assumptions, selected security proof and failure budget. An entanglement-distribution paper cannot supply those results through a change of terminology.

20.11 Reading outputs and designing controlled comparisons

20.11.1 Scenario files, comparison records and plots

Individual scenario files separate `epochs[i].quantum`, the legacy conditional channel result, from `epochs[i].qkd`, the optical/detector/protocol sample. For BBM92 the latter can contain a background-mixed detected density matrix, detected fidelity/purity and basis errors. These should not overwrite the original channel's conditional fidelity. The corresponding complete-block security result is `qkd_summary.security`.

The comparison document includes `scenarios[].qkd` sample data, `qkd_blocks[]` complete blocks tagged by source/scenario, `qkd_assumptions`, and `scenario_effects[].qkd_effects`. Ground stations are common prescribed receiver worldlines in both orbit branches; they are excluded from matched orbital initial conditions and satellite residuals. A ground receiver cannot be perturbed as an orbital detector state in a sensitivity experiment.

When QKD is enabled, the standalone runner adds `qkd_comparison.png` alongside its legacy comparison image. The separate plot shows Z/X errors, detected event rates and raw-versus- estimated secret bits for the complete block. The dashboard's optional QKD panel distinguishes selected-epoch true pairs, accidentals, single-arm pulse detections, detector fidelity/purity and basis errors. Its block table gives status, count mode, duration, coincidences, key estimate/rate and security-analysis QBER. A dash is an unavailable value; a zero-bit abort is a valid numerical result. The static manual figures do not constitute a screenshot certification of that browser layout.

20.11.2 A disciplined comparison sequence

Read the effective configuration before the headline key number. Check the topology and protocol, spacetime, receiver geometry, conditional state, optics and detector factors. Next check count mode and seed, emitted and accepted populations, Z/X sample sizes and basis errors. Finally inspect the finite/asymptotic selector, bound populations, phase-error upper bound, leakage, epsilon budget and final status.

Compare expected and sampled results only after accounting for their different sampling definitions. Compare asymptotic and finite estimates using the same underlying count budget; otherwise a changed apparatus can be mistaken for a statistical penalty. Compare S2 and S3 under the same physical history to verify invariance. Compare SR and static GR with the same declared source/receiver input and count policy, then trace any difference through timing, visibility, optical acceptance, storage and sample size before interpreting the final key change.

A useful parameter study varies one physically motivated quantity at a time, such as detector background, divergence, integration duration, basis bias or storage dephasing. Retain unavailable and zero-key points, and report whether a parameter is measured, assumed or fitted. A profile of key length against an assumed parameter is a sensitivity curve; it is not an experimental confidence interval.

For a ground pass, use sufficiently resolved time samples and inspect the visibility boundary. A coarse temporal grid can miss a short joint window or overestimate usable time. A convergence comparison of the same modeled pass helps bound discretization effects, but does not bound uncertainty in the real ephemeris or weather.

20.12 Worked results from the supplied configurations

The following values are extracted from the saved `validation_qkd` runs. They are expected-count finite-size forecasts, with the supplied apparatus assumptions, rather than measurements or generated secret keys. The first five rows use nine epochs over 120 coordinate seconds. Values are rounded for display; the scenario JSON preserves full precision and complete deductions. S2 and S3 are combined in a row only after verifying that their saved QKD results agree exactly.

Example	Scenario	Z raw events	Z QBER	Net bits	Bits/s
Intersatellite BBM92	S1	14343.17	0.003098	6752	56.267
Intersatellite BBM92	S2 = S3	3588.54	0.003100	618	5.150
Dual-ground BBM92	S1	239820.06	0.003100	187612	1563.433
Dual-ground BBM92	S2 = S3	186992.32	0.003101	143174	1193.117
Single-arm decoy BB84	S1 = S2 = S3	227805045.07	0.010004	82365033	686375.275
Micius-type illustrative	S2	27.33	0.079754	0	0

The single-arm example uses a separate 100 MHz pulse source, while the first BBM92 examples use a 1 MHz pair source and two-arm coincidence acceptance. The very different rates therefore do not establish a controlled protocol ranking. In the supplied decoy geometry the chosen arm A is unchanged across the three scenarios, explaining their equal result. S1 is a different physical event construction in the BBM92 comparisons; its larger number is not evidence of a PF advantage.

The Micius-type illustrative run uses 21 epochs over 250 seconds and a declared 71 dB combined scalar attenuation. It predicts about 109.33 total coincidences, or 0.4373 per coordinate second. After sifting it has only about 27.33 expected events in each test/key basis. The finite phase-error bound reaches its abort region, producing zero key with `phase_error_at_or_above_one_half`. This model configuration is separate from the literature consistency benchmark in Section 20.10; neither its rate nor its abort is a reproduced historical QKD outcome.

A separate static-GR ground integration check uses three epochs over 120 seconds. It gives 178787 net forecast bits for S1 and 133439 for each of S2/S3. Its time grid differs from the nine-epoch SR run, so their numerical difference must not be attributed to gravity. A matched-grid convergence study with otherwise identical inputs is required before isolating an SR/GR effect. Positive output alone does not establish that trapezoid integration has converged.

The editable source includes `figure_data/qkd_worked_results.json`, containing the extracted values, exact scenario-file paths and SHA-256 identities. It preserves the distinction between expected raw populations, net integer estimates and the key-rate denominator.

20.13 Reproducibility and evidence to retain

Retain the complete configuration, release identifier, effective normalized options, worldline/ephemeris provenance, station definitions, spacetime and timing conventions, integration domain, density-matrix ordering and Bell parity, optical parameter sources, count mode and seed, protocol output and every abort/status field. For a literature comparison, retain the source attachment identity, source page/figure locators, copied numerical facts, uncertainty assumptions and the distinction between imported inputs and comparison targets.

The code's tests check algebraic identities, parameter domains, count/protocol behavior and integration invariants. The benchmark checks published facts and an explicit surrogate calculation. These are complementary checks with different evidential scope. Passing a unit test cannot settle a missing experimental data issue, and a literature-compatible summary statistic cannot certify an implementation that uses the wrong Bell-state parity or double-counts loss.

The four new figures are editable vector artwork. Their Python source is `generate_qkd_figures.py`; the supplied full-width PDFs are 160 mm wide and the independently composed single-column alternatives are 85 mm wide. They illustrate the architecture, geometry and count definitions without inserting synthetic experimental measurements.

20.14 Primary sources for the QKD extension

The formulas and protocol assumptions should be read with their source models, not as interchangeable recipes. The software documents its own conservative allocations and forecast-only use explicitly.

Q1. Bennett, Brassard and Mermin (1992). The entanglement-based BBM92 protocol and the distinction between entanglement correlations and Bell-inequality security arguments. Quantum cryptography without Bell's theorem.

Q2. Tomamichel, Lim, Gisin and Renner (2012). Finite-key uncertainty and random-sampling structure used for the BBM92-style forecast, including its supplementary bounds. Tight finite-key analysis for quantum cryptography.

Q3. Lim, Curty, Walenta, Xu and Zbinden (2014). Practical finite-key decoy-state analysis and analytic three-intensity bounds underlying the decoy forecast. Concise security bounds for practical decoy-state quantum key distribution.

Q4. Yin and coauthors, supplied arXiv v1 attachment (2017). Primary numerical evidence for the conditional Micius consistency benchmark. The v1 labels and count inconsistencies are retained in the evidence record rather than silently corrected. Satellite-based entanglement distribution over 1200 kilometers.

Chapter 21

Published-data QKD validation profiles

Edition 1.5.0 adds three separately configurable reference workflows. They exercise different parts of the QKD calculation against evidence that is available for inspection: published SatQuMA numerical results, laboratory BBM92 observations, and Jinan-1 satellite telemetry. A single comparison cannot substitute for all three. Agreement with a reference security calculation tests numerical implementation; agreement with laboratory counts tests a stated detector/source model; agreement with satellite telemetry tests a stated optical and error model over the sampled conditions.

These profiles supplement the existing scenario engine. They do not replace the SR/GR controls, change the old or new PF construction, or add a new physical scheduling law. The benchmark names identify an external evidence source, not an additional scenario or inertial frame. The QKD scenarios described in Chapter 20 remain the place to simulate a configured emitter and its receivers. This chapter explains the additional evidence and the exact limits of the claims that a benchmark result can support.

21.1 Choose a reference by the question being tested

Profile	Evidence and intended use	What it does not establish
`satquma`	Published decoy-BB84 simulation outputs and a pinned author implementation. Check finite-key arithmetic, conventions and intermediate bounds.	An empirical satellite measurement, new optimizer equivalence, or a proof that a laboratory device meets the security assumptions.
`bbm92`	Laboratory entangled-photon timing/count records and published analysis outputs. Check coincidence-window dependence, accidentals, QBER and an asymptotic key-rate model.	A satellite pass, a complete composable finite-key execution or an independent prediction when fitted source/detector parameters are reused.
`jinan`	Public downlink range, count-rate, QBER and cumulative-key telemetry. Test a configured optical/error model on declared held-out observations.	Raw basis/intensity replay, exact historical key extraction, or entangled-pair fidelity and purity.

The three evidence classes must remain separate in a presentation or procurement comparison. SatQuMA is a scientific numerical reference, not observed detector data. The BBM92 data concern an entangled-photon laboratory setup, not a space-to-ground prepare-and-measure source. Jinan-1 is an empirical satellite BB84 reference, not the Micius entanglement experiment. Their useful overlap is the software's ability to declare apparatus/protocol conventions and expose residuals; their measurements are not interchangeable.

21.2 Evidence, calibration and held-out observations

Every comparison has three logically different ingredients. The *reference* is a published value or data record. The *model input* is a setting supplied to the calculation, such as an emission rate, range or detector response. The *prediction* is the value computed from those inputs. A reference quantity used to choose an input parameter is calibration evidence. It cannot also be counted as an independent confirmation of the same fitted effect.

For example, fitting an optical throughput to measured counts can be appropriate when the aim is to estimate that throughput. A subsequent count comparison on exactly those samples measures fit residuals. A compar-

ison on observations excluded from the fitting process measures held-out performance within that experiment and model class. It still does not establish predictive accuracy under different hardware, weather, wavelength or orbital conditions.

Declare the split and loss function before interpreting the comparison. Keep the same split when comparing candidate model revisions. Changing it repeatedly until a residual looks small turns the nominal test set into another calibration set. Adjacent samples from a satellite pass may also share disturbances, so a held-out score is not automatically an independent-sample confidence interval. Preserve the timestamps, sample mask and effective settings in the run output.

A numerical-reference comparison has a related restriction. Supplying published vacuum/single-photon lower bounds and phase-error upper bounds to a final key-length formula tests the final arithmetic. It does not test the process that produced those bounds. A stronger reference-code comparison starts from the same pulse, loss and protocol inputs and compares intermediate quantities as well as the final key length. Report these layers separately.

21.3 Run the benchmark suite

Use the same Python environment in which the package is installed. On Windows, the explicit environment interpreter avoids selecting a separate global Python installation. In the examples below, replace `.venv\Scripts\python.exe` if your environment has another name. The benchmark runner is a separate entry point from the scenario runner.

```
.venv\Scripts\python.exe run_qkd_benchmarks.py --profile all ^
  --output results_published_benchmarks

.venv\Scripts\python.exe run_qkd_benchmarks.py --profile satquma ^
  --output results_satquma

.venv\Scripts\python.exe run_qkd_benchmarks.py --profile bbm92 ^
  --output results_bbm92

.venv\Scripts\python.exe run_qkd_benchmarks.py --profile jinan ^
  --output results_jinan
```

On Linux or macOS, use the selected environment's `python` command and a backslash for shell line continuation, or enter each command on one line. Windows command prompt uses the caret shown above. Do not paste the Windows continuation character into another shell.

Option	Meaning
`--profileall`	Execute all three reference profiles and preserve their separate results.
`--profilesatquma`	Execute only the published numerical-reference workflow.
`--profilebbm92`	Execute only the laboratory BBM92 workflow.
`--profilejinan`	Execute only the satellite telemetry workflow.
`--configFILE`	Read a JSON benchmark configuration. It is not a scenario-engine configuration.
`--outputDIRECTORY`	Place new benchmark outputs in the selected directory. Use a fresh directory when changing a model or split.
`--no-plots`	Retain numerical outputs while skipping figure generation.
`--write-default-configFILE`	Write an editable suite configuration and exit without running.
`--require-empirical-pass`	Require all empirical profiles to pass their declared criteria; numerical-only references are exempt.

The configuration file is the authoritative description of a customized benchmark. Store it beside the results and examine the normalized configuration in the output. A plot alone does not preserve data selection, calibration controls or acceptance tolerances.

21.3.1 Export, edit and replay the effective configuration

Export a complete starting configuration into the package directory:

```
.venv\Scripts\python.exe run_qkd_benchmarks.py --profile all ^
  --write-default-config benchmark_custom.json

.venv\Scripts\python.exe run_qkd_benchmarks.py --profile all ^
  --config benchmark_custom.json --output results_custom_benchmarks
```

A suite file has `schema_version: 1` and a `profiles` object with `satquma`, `bbm92` and/or `jinan` settings. A single-profile object is accepted only when the command selects one profile explicitly. An empty or omitted profile object uses that profile's defaults. The parser rejects duplicate JSON keys, nonfinite values and unrecognized suite keys. Each profile additionally validates its own settings rather than silently ignoring misspellings.

Bundled default data paths resolve against the package directory. Explicitly overridden relative data paths resolve against the configuration file's directory. Exported defaults use resolved absolute paths for local editing. Move the associated data tree with a configuration or use deliberate absolute paths. The output directory resolves from the invoking shell. A copied configuration does not automatically copy its input files. The runner performs no network downloads. The released local data and its provenance are sufficient for the configured bundled workflows.

21.3.2 Outputs, exit status and automation

Detailed reports are JSON: `benchmark_suite.json` and the selected `satquma.json`, `bbm92.json` and `jinan.json`. The suite also saves `resolved_config.json` and a short `README.md`. Plot-enabled runs add PNG/PDF formats of each figure, not a narrative PDF report. Profile reports contain `numerical_validation`, `empirical_validation`, `summary`, `series`, `provenance`, `limitations` and the normalized `resolved_configuration`. Keep these fields together when exporting an excerpt for review.

Exit code 0 means the selected calculations completed without a failed numerical check. It does not mean that every empirical model agreed. Exit code 1 signals an input/configuration or execution error, and exit code 2 indicates a failed numerical validation. With `--require-empirical-pass`, exit code 3 additionally signals that at least one empirical profile did not explicitly pass its criteria. The status `within_configured_tolerances` counts as passing this engineering gate; the report's separate evidence limitations still apply. This opt-in gate is useful in automation precisely because a completed calculation with an honest empirical mismatch is a meaningful result.

21.4 Complete benchmark configuration reference

The shipped `config_benchmark_satquma.json`, `config_benchmark_bbm92.json` and `config_benchmark_jinan.json` are single-profile configurations. Select the corresponding profile explicitly when using one of them. The CLI profile `bbm92` maps to the internal JSON identifier `bbm92_laboratory`. The following tables describe every setting group; the supplied JSON and exported defaults preserve full numerical precision, complete arrays and source hashes.

21.4.1 SatQuMA reference and forecast controls

Key or group	Default and interpretation
`profile`	`satquma`.
`loss_file`	Bundled `FS_loss_XI0.csv`; columns are time in seconds, elevation in radians and total channel efficiency. Values must be finite, times unique and efficiency in $[0, 1]$.

Key or group	Default and interpretation
`published_csv_dir`	Bundled Figure 5a numerical tables used for final arithmetic checks.
`reference_file`	Frozen cases evaluated with the pinned author implementation. These references are distinct from the configurable forecast.
`relative_tolerance`, `absolute_tolerance`	10^{-9} and 10^{-7}, respectively, for numerical comparisons. These are not empirical uncertainty intervals.
`basis_x_probability`	0.7611; independent X selection at both ends. X is raw key and Z is the test basis. Both probabilities must be nonzero.
`source.pulse_rate_hz`	10^8 pulses/s.
`source.intensities`	$[0.7921, 0.1707, 0]$ mean photons per pulse, ordered signal, weak decoy and weakest decoy.
`source.probabilities`	$[0.7501, 0.1749, 0.075]$, aligned to those three intensities; each positive and their sum one.
`channel.excess_loss_db`	0 dB added to the supplied loss curve; nonnegative.
`channel.extraneous_probability_per_detector`	10^{-8} per detection opportunity per detector, not an arm-total count rate in hertz.
`channel.intrinsic_error`	0.001; dimensionless signal error probability.
`channel.afterpulse_probability`	0.001; the explicit pinned-reference linear afterpulse convention. The code rejects resulting click/error probabilities outside their physical domain.
`window.half_width_s`	210 s, including both endpoints around zero in the reference curve.
`window.sample_duration_s`	1 s exposure per selected row; spacings must agree. The default inclusive window has 421 exposure rows, not a 420-s trapezoidal integral.
`window.passes`	1 positive integer; repeated identical exposure blocks are pooled for the reference convention. It does not provide a new orbit for every pass.
`security`	The full SatQuMA security object in Section 21.6. Reference channel forecasts require `count_mode:expected`.
`sweep_excess_loss_db`	$[0, 3, 6, 9, 12]$ dB; independently evaluates the configurable forecast at these added losses.

The reference channel and the normal mission channel are distinct models. In particular, the reference reproduces its per-detector extraneous-click and afterpulse conventions and allocates each time bin's aggregate test error in proportion to its intensity detection counts. The mission pipeline retains its physical per-intensity click and error calculation. Selecting a common security bound does not silently replace either channel model.

21.4.2 BBM92 laboratory apparatus, calibration and acceptance

Key or group	Default and interpretation
`profile`	`bbm92_laboratory`; selected with CLI `--profilebbm92`.
`reference.path`, `reference.sha256`	Bundled derived record `mar20_run2.json` and its expected hash. A custom source must use the documented schema and an appropriate hash; null deliberately disables that hash assertion.
`model.pair_rate_hz`	1.5×10^6 generated pairs/s.
`model.loss_a_db`, `model.loss_b_db`	12 dB per arm; nonnegative.
`model.loss_includes_detector_efficiency`	False: the separately configured detector efficiencies multiply the losses. True avoids counting those efficiencies twice.
`model.detector_efficiency_a`, `model.detector_efficiency_b`	0.6 each, bounded in $[0, 1]$.

Key or group	Default and interpretation
`model.background_a_hz`, `model.background_b_hz`	500 and 1800 detected background counts/s per arm before the selected dead-time convention.
`model.dead_time_a_s`, `model.dead_time_b_s`	45×10^{-9} s each.
`model.detectors_a`, `model.detectors_b`	Four detector channels per arm; positive integers. Occupancy is distributed over this number in the model.
`model.background_deadtime_convention`	`all_counts` applies the live fraction to both signal and background; `published_signal_only` retains the reference alternative for explicitly labeled comparisons.
`model.accidental_convention`	`poisson_window` by default; `sparse_product` uses $S_A S_B W$. Both are approximations with domain checks.
`model.timing_width_s`	1.6×10^{-9} s under the selected width convention.
`model.timing_width_convention`	`pair_fwhm`, `pair_sigma` or `equal_detector_sigma`. FWHM divides by $2\sqrt{2\ln 2}$; equal independent detector widths multiply by $\sqrt{2}$ to give pair sigma.
`model.timing_offset_s`	0 s; mean pair-delay offset relative to the centered full coincidence window.
`model.source_fidelity`	0.955 for the declared Werner source about $\|\psi^+\rangle$; supported range $[0.25, 1]$. It is an assumed/calibrated source model, not inferred tomography.
`model.phase_sigma_a_rad`, `model.phase_sigma_b_rad`	0 rad each; optional Gaussian differential-phase noise.
`model.phase_correlation`	0; dimensionless correlation in $[-1, 1]$.
`model.relative_phase_rad`	0 rad deterministic phase.
`model.basis_z_probability_a`, `model.basis_z_probability_b`	0.5 at each receiver; independent choices.
`model.key_basis`	`both` retains both matched bases; `z` retains Z key with X testing in the asymptotic projection.
`model.ec_efficiency`	1.1, at least one; asymptotic reconciliation multiplier.
`calibration.mode`	`none` by default; `selected_windows` enables an explicit fit. Published apparatus estimates may already depend on this acquisition even with no additional fitting.
`calibration.indices`	$[25, 40, 55, 70, 85]$, zero-based distinct window-row indices, used only when fitting is enabled.
`calibration.parameters`	Pair rate, timing width and source fidelity by default. Optional fitted names also include per-arm loss and background.
`calibration.bounds`	One increasing interval per fitted parameter. Defaults: pair rate $[5 \times 10^5, 3 \times 10^6]$ Hz; width $[0.5, 3]$ ns; source fidelity $[0.8, 1]$. Initial values must lie inside their bounds.
`calibration.relative_rate_scale`, `calibration.qber_scale`	0.05 and 0.01; scales in the combined least-squares objective, not experimentally established uncertainties.
`calibration.max_nfev`	300 fit-function evaluations. Convergence status is retained.
`validation.holdout_indices`	Null selects rows not used for the optional fit; an explicit list can select another disjoint set. These are correlated window re-binnings of one acquisition.
`validation.minimum_window_s`, `validation.maximum_window_s`	10^{-10} to 10^{-7} s; eligible full-window range.
`validation.max_singles_window_occupancy`	0.01: requires $\max(S_A, S_B)W$ below this sparse-window threshold.
`validation.minimum_singles_to_true_ratio`	10: minimum single-arm to true-coincidence rate ratio for the selected approximation.
`validation.raw_rate_relative_tolerance`	0.05: per-row relative raw-rate residual tolerance.
`validation.qber_absolute_tolerance`	0.01 absolute probability, equal to one percentage point.

Key or group	Default and interpretation
`validation.minimum_fraction_within_tolerance`	0.95 of eligible rows must meet each criterion separately for the declared engineering gate.
`validation.numerical_atol`	10^{-9}, numerical consistency tolerance.

The model's live fraction in arm j, with n_j detector channels, is

$$L_j = \left[1 + \frac{(R_p\eta_j + B_j)\tau_j}{n_j}\right]^{-1}. \tag{21.1}$$

Under the default all-count convention, $S_j = (R_p\eta_j + B_j)L_j$. The true-pair rate is $R_T = R_p\eta_A\eta_B L_A L_B g(W)$. The alternative Poisson window approximation for accidentals is

$$R_A = \frac{[1-\exp(-S_A W)][1-\exp(-S_B W)]}{W}, \tag{21.2}$$

which approaches Equation 21.10 at low occupancy. The input rates, timing distribution and sparse-domain checks remain essential to its interpretation.

21.4.3 Jinan-1 alignment, optics, fitting and diagnostics

Key or group	Default and interpretation
`schema_version`, `profile`	1 and `jinan`.
`mode`	`calibrated` fits declared training data; `fixed` preserves supplied parameters.
`data.path`, `data.expected_sha256`	Derived `figure4.json` and its source hash. Null hash explicitly allows a custom file without that integrity assertion.
`data.duplicate_policy`	`last` keeps the last duplicate timestamp; `error` rejects duplicates. Resolution counts remain in provenance.
`data.max_range_interpolation_gap_s`	2 s; linearly interpolate only between range samples separated by at most this gap. No extrapolation beyond the range trace.
`data.zero_qber_is_missing`	True treats displayed zero QBER as unavailable telemetry, not a zero-error measurement.
`data.range_time_offset_s`	0 s; explicit timestamp offset applied to the range series before alignment.
`split.training_fraction`	0.6 of the overall elapsed interval when explicit intervals are absent.
`split.training_intervals_s`	Null by default; otherwise ordered, nonoverlapping elapsed-time intervals in seconds.
`split.exclusion_gap_s`	10 s around training intervals, excluded from held-out scoring.
`split.metric_block_duration_s`	20 s count blocks for fitting and scoring; not a claim of independent fluctuations. QBER uses changed telemetry plateaus separately.
`source.pulse_rate_hz`	625×10^6 pulses/s.
`source.intensities`, `source.probabilities`	$[0.8, 0.1, 0.001]$ photons/pulse and $[0.5, 0.25, 0.25]$. The third intensity is retained as nonzero.
`link.model`	`gaussian_aperture` or `effective_range`; both keep explicitly configured atmosphere and visibility assumptions.
`link.earth_radius_m`, `link.satellite_altitude_m`	6,371,000 m and 500,000 m; spherical geometry used to infer elevation from measured range. Altitude is an apparatus-model assumption, not a recovered ephemeris.
`link.reference_range_m`, `link.effective_reference_efficiency`, `link.range_power`	10^6 m, 10^{-4}, and 2 for the optional effective-range model. That efficiency is composite, rather than a separately known optical element.

Key or group	Default and interpretation
`link.wavelength_m`, `link.waist_m`	850 nm and 0.09 m.
`link.divergence_half_angle_rad`	5×10^{-6} rad; cannot be below $\lambda/(\pi w_0)$ in the Gaussian model.
`link.receiver_aperture_radius_m`	0.14 m radius, not diameter.
`link.pointing_jitter_rad`, `link.pointing_bias_rad`	10^{-6} rad jitter and zero mean bias; receiver-plane offsets scale with range.
`link.optical_efficiency`, `link.detector_efficiency`	0.15 initial optical throughput and 0.6 detector efficiency; each bounded in $[0, 1]$. The default fit adjusts optical throughput.
`link.zenith_optical_depth`	0.1 initial dimensionless optical depth; default fit parameter.
`link.min_elevation_deg`	0 degrees; reject signal below the selected elevation cutoff.
`detector.background_rate_hz`, `detector.dead_time_s`	100 counts/s and 0 s. A configured nonzero dead time remains subject to the sparse occupancy restriction.
`detector.gate_width_s`, `detector.timing_jitter_s`, `detector.timing_offset_s`	800 ps full gate, 312.13 ps Gaussian sigma, zero offset. Gate width may not exceed the pulse period.
`detector.count_observable`	`pre_gate` by default; `accepted` compares the modeled gated population. Choose according to the reported observation, not the better fit.
`detector.misalignment_probability`	0.008 initial signal error probability, bounded in $[0, 0.5]$; default fit parameter.
`detector.qber_intensity_selection`	`all` mixes configured intensity classes; `signal` evaluates the highest-intensity class alone.
`calibration.parameters`	Optical efficiency, zenith optical depth and misalignment by default. Detected background rate is another supported fitted parameter.
`calibration.bounds`	Optical efficiency $[10^{-6}, 1]$, depth $[0, 5]$, background $[0, 10^5]$ Hz, misalignment $[0, 0.5]$. Only selected parameters enter the fit.
`calibration.count_loss`	`log_rate` by default; `relative_rate` selects a relative residual. The loss is a fitting choice, not a measured uncertainty model.
`calibration.qber_residual_scale`	0.01 probability scale for the QBER component of the fit objective.
`calibration.max_nfev`	500 evaluations. Failed convergence raises an error; active bounds and local Jacobian rank are reported.
`thresholds.count_relative_mae_max`, `thresholds.qber_absolute_mae_max`	Null by default: report residuals without inventing a pass threshold. Supply finite positive criteria only when justified for the engineering question.

All length inputs use metres, time inputs seconds and phase/pointing angles radians unless the field explicitly says degrees. A 1-ns gate is `1e-9`, not `1`. QBER is a probability in the configuration and machine-readable output; a figure may convert it to percent for readability. An absolute QBER difference of 0.001 is 0.1 percentage point, not 0.001 percent.

21.5 SatQuMA: matching a published numerical calculation

The SatQuMA profile uses the numerical data associated with the 2023 finite-key satellite-QKD study and the author's separately pinned reference code. The public dataset version and the reference-code revision are distinct provenance fields. A dataset's version label does not prove which source-code commit generated every row. The comparison therefore preserves both identities and reports each tested layer.

SatQuMA uses X for the raw-key basis and Z for the test basis. The production generic profiles elsewhere in this package may label the key basis Z. Basis names alone are not physical disagreement, but swapping only their labels without swapping all populations and error conventions changes the calculation. Keep each benchmark's convention through the complete security analysis.

21.5.1 Final key-length arithmetic

For the supplied numerical-reference rows the gross key length is reconstructed from the published intermediate quantities as

$$\ell_{\mathrm{gross}} = \left\lfloor \max\left\{0, s^L_{X,0} + s^L_{X,1}[1 - h_2(\phi^U_X)] - \lambda_{\mathrm{EC}} - 6\log_2\frac{21}{\epsilon_s} - \log_2\frac{2}{\epsilon_c}\right\}\right\rfloor. \tag{21.3}$$

Here $s^L_{X,0}$ and $s^L_{X,1}$ are lower bounds on the vacuum and single-photon key-basis contributions, ϕ^U_X is an upper bound on the phase-error rate, and λ_{EC} is the reconciliation leakage. The binary entropy is

$$h_2(p) = -p\log_2 p - (1-p)\log_2(1-p), \qquad h_2(0) = h_2(1) = 0. \tag{21.4}$$

Counts and key lengths are dimensionless numbers of events or bits; ϵ_s and ϵ_c are dimensionless failure budgets. Do not mix percentages with probabilities in the entropy argument. The integer floor is applied to the completed expression.

21.5.2 Count intervals and the nonvacuum endpoint correction

For a count n_j, let $a = \ln(21/\epsilon_s)$. The Chernoff compatibility endpoints apply the shifts

$$\Delta^-_j = \frac{a}{2} + \sqrt{2n_j a + \frac{a^2}{4}}, \tag{21.5}$$

$$\Delta^+_j = a + \sqrt{2n_j a + a^2}, \tag{21.6}$$

$$n^-_j = \frac{e^{\mu_j}}{p_j}(n_j - \Delta^-_j), \qquad n^+_j = \frac{e^{\mu_j}}{p_j}(n_j + \Delta^+_j). \tag{21.7}$$

The Hoeffding option uses a common shift $\sqrt{\frac{1}{2}(\sum_j n_j)\ln(21/\epsilon_s)}$; the asymptotic option uses no shift. Signed interval endpoints are retained through the decoy combinations before the photon-population bounds are limited. Prematurely replacing a negative interval endpoint with zero can alter the reference finite-statistics calculation.

With $\tau_0 = \sum_j p_j e^{-\mu_j}$, the vacuum bound uses

$$s^L_{X,0} = \frac{\tau_0}{\mu_2 - \mu_3}\left(\mu_2 n^-_{X,\mu_3} - \mu_3 n^+_{X,\mu_2}\right), \tag{21.8}$$

subject to the implementation's explicitly recorded lower-bound floor and consistency gates. The subtracted term uses its *upper* endpoint to remain conservative. For $\mu_3 = 0$ it vanishes, which is why the vacuum reference fixtures can agree with the pinned code while retaining the corrected nonvacuum expression.

The final budget is aborted when required raw/test populations are absent, photon-population bounds are inconsistent, single-photon information is unavailable, the raw QBER reaches one half, or the phase bound meets the configured threshold. An apparently positive algebraic vacuum term does not bypass those gates. Observed-mode inputs must be integer populations; expected populations are forecasts.

An authentication debit is a separate operational accounting choice. A published gross secret-key length can match exactly while the package's net deployable-key forecast is smaller after a configured authentication debit. Show both quantities and the debit explicitly; do not hide the difference by adjusting the security budget or the reference value.

21.5.3 Reference calculation versus parameter optimization

The pinned reference equivalence is validated for a vacuum weakest decoy. When the weakest intensity is nonzero, the implementation uses conservative vacuum-bound interval endpoints rather than preserving an incorrect endpoint choice merely for numerical agreement. Such a run can differ from the pinned SatQuMA v2 implementation and is labeled accordingly. This boundary matters when transferring a security model to a source with a small but nonzero weakest intensity.

A fixed-parameter replay evaluates the same source, channel and protocol settings as a chosen reference calculation. It can reveal a background convention mismatch, wrong basis population, sample-window boundary or finite-statistics discrepancy without confounding those with optimizer behavior. Agreement at fixed parameters does not prove that two optimizers find the same optimum.

Numerical studies may use a continuous unfloored key expression as an optimization objective. The physical output is a nonnegative integer key-length budget after the prescribed floor and any separately reported deductions. When comparing values near zero or an integer boundary, retain full precision for the continuous quantity and compare the rounded quantity under the same rule.

21.6 Select the security model in a normal mission run

The shipped `config_qkd_satquma.json` demonstrates the selectable model on the ordinary mission pipeline. For example:

```
run_python38.bat --scenario all --config config_qkd_satquma.json ^
  --spacetime flat_sr --output results_satquma_mission_sr

run_python38.bat --scenario all --config config_qkd_satquma.json ^
  --spacetime gr_static --output results_satquma_mission_gr
```

These are mission forecasts, not replays of a SatQuMA reference loss curve. SR and static GR apply to all three selected scenarios. Scenario 3 continues to evaluate the full PF construction on Scenario 2's physical history. A successful QKD calculation does not upgrade a PF diagnostic status such as `correction_numerically_uncertified` into a certified construction.

The SatQuMA-compatible security equations are also selectable in the normal decoy-BB84 mission workflow. They are not restricted to replaying an external reference table. Set `qkd.security_model` to `satquma`; the retained `conservative` value is the default. The two models implement different statistical and accounting choices, so identical final key lengths are not generally expected.

`qkd.satquma_raw_key_basis` chooses `x` or `z`. The independent SatQuMA reference uses X for key and Z for testing; the mission adapter maps the selected physical populations into that convention. The QBER and count records retain the selected physical basis. This is a complete population mapping, not a change to the meaning of an observed error.

SatQuMA setting	Default	Meaning
`bound`	`chernoff`	Statistical interval model: `chernoff`, `hoeffding` or `asymptotic`.
`epsilon_secrecy`	10^{-9}	Secrecy budget for this bound.
`epsilon_correctness`	10^{-15}	Correctness budget for this bound.
`ec_method`	`logm`	Reference leakage estimate `logm`, or block-entropy estimate `block`.
`ec_efficiency`	1.16	Reconciliation efficiency multiplier, at least one.
`authentication_bits`	Inherits general mission setting when absent	Nonnegative integer debit per pooled block, deducted after gross-key flooring. A supplied SatQuMA value overrides inheritance; the standalone lower-level bound defaults to 0.
`phase_error_tolerance`	0.5	Predeclared abort threshold in $0, 0.5]$; a bound meeting or exceeding it returns zero key.
`count_mode`	`expected`	Lower-level bound label: `expected` or `observed`. Mission sampling controls determine the effective label.

These keys belong under `qkd.satquma_security`. When the SatQuMA model is selected, this object supplies the security budgets, statistical bound, leakage convention and authentication debit. The general `qkd.securi`

ty object does not replace explicit SatQuMA choices. When `qkd.satquma_security.authentication_bits` is absent, the mission adapter inherits `qkd.security.authentication_bits`, whose general default is 128. The GUI preserves that absence rather than freezing a copied value; adding an explicit SatQuMA debit intentionally pins it. The general `count_mode` also remains the mission pipeline's sampling control: `expected` uses expected populations and `sampled` uses simulated integer populations. An explicit conflicting SatQuMA count-mode label is rejected; `sampled` maps to the lower-level integer-count label `observed`. A sampled result is synthetic data, not an observed hardware transcript. The effective output label documents this distinction.

The SatQuMA `bound:asymptotic` setting controls its asymptotic limit; changing only the conservative model's `finite_key` flag does not select that limit. Read the chosen model and normalized security object in each output before interpreting a zero or positive key length. The selectable model applies to decoy BB84, not to the BBM92 estimator, whose assumptions and formula remain as documented in Chapter 20.

At the lower-level API, `satquma_security.decoy_key_bound` accepts three raw-basis populations, three test-basis populations, three test-error populations, a raw-error total, ordered intensities and intensity probabilities. The three classes are signal, weak decoy and weakest decoy. Required intensities satisfy $\mu_1 > \mu_2 + \mu_3$ and $\mu_2 > \mu_3 \geq 0$, with positive class probabilities summing to one. The source remains phase-randomized weak coherent pulses. This profile does not reinterpret an entangled pair source as a decoy source.

For the logarithmic leakage option, the implementation follows the pinned reference's `logm` convention, including its stated combination of binary entropy and natural-logarithm terms. Its numerical compatibility must not be generalized into a universal optimal reconciliation theorem. Authentication remains separate from the reference gross key length.

21.7 BBM92: timing windows, counts and error rates

The laboratory benchmark addresses a different part of the system: whether the source/detector count model and coincidence analysis follow the public BBM92 records under their declared timing conventions. The reference repository provides measured records and analysis outputs; it also reports source brightness, effective losses, timing response and noise parameters estimated from experimental data. Reusing these estimates is a declared calibration choice, not an independent measurement of the apparatus.

The bundled 150 window rows are re-binnings of one laboratory acquisition. They are not 150 independent experiments. The default profile imports published apparatus estimates and performs no new fit; optional selected-window fitting still shares the underlying record with its other window rows. An unused window is consequently not an independent held-out acquisition.

For a centered Gaussian pair-delay distribution with standard deviation σ_Δ, the accepted true-pair fraction inside a full coincidence window W is

$$g(W) = \operatorname{erf}\left(\frac{W}{2\sqrt{2}\sigma_\Delta}\right). \tag{21.9}$$

This formula uses a *full* width. Replacing W by a half-width without changing the factor of two changes the acceptance. A per-arm timing width is not automatically the width of the pair difference; independent Gaussian detector widths add in quadrature. A published histogram's fitted width must be interpreted using that source's exact convention before it is entered into this expression.

In a sparse, stationary approximation, unrelated detections contribute accidental coincidences proportional to the product of the single-arm rates and the window duration:

$$R_{\mathrm{acc}} \simeq S_A S_B W. \tag{21.10}$$

Here S_A, S_B are counts per second and W is in seconds, so R_{acc} has units s^{-1}. The precise accepted basis/parity factors and the handling of multiple candidate matches belong to the configured analysis convention. Do not insert another factor of two merely because the apparatus uses two arms.

A wider window accepts more of the true-pair timing distribution but also more unrelated detections. Counts can increase while QBER worsens and an asymptotic key-rate estimate falls. The optimum of a particular rate curve therefore depends on both the signal response and noise counts; it is not a universal timing constant or a preferred inertial frame. The benchmark exposes this trade-off rather than inserting a penalty based on distance from Scenario 1.

21.7.1 QBER and a rate are not a quantum-state reconstruction

For an accepted basis population, QBER is the error count divided by the accepted count under the specified Bell-parity convention. A singlet anticorrelation convention and a correlated target require different bit mappings. Scalar QBER values do not determine an arbitrary two-qubit density matrix. In particular, laboratory agreement in QBER or a coincidence histogram does not independently establish fidelity or purity of the simulated state.

The reference's asymptotic key-rate analysis must remain distinct from a finite-key bound. In a symmetric ideal one-way form the fraction is $1 - 2h_2(q)$; practical reconciliation introduces an efficiency penalty and two-basis formulations need the separate bit and phase estimates. A rate obtained with the selected asymptotic convention is not a count of cryptographic bits extracted from the public timestamp record. Finite-size estimation, reconciliation, verification, authentication and privacy amplification require additional choices and records.

21.8 Jinan-1: telemetry and a satellite model

The Jinan-1 reference comes from the 2025 Nature microsatellite-QKD publication and its public source-data archive. It provides actual satellite observations, including range, photon-count-rate, QBER, tracking and cumulative secret-key telemetry. These are stronger empirical constraints than a single headline rate, but they are not a complete event-level protocol transcript.

The archive explicitly warns of repeated or missing seconds in its telemetry. Count-rate and range columns have their own timestamps and need not contain the same number of samples. An implementation must align them according to a declared rule, retain gaps, and avoid pairing rows merely because they have the same row number. Similarly, summing a column of reported rates is not automatically a photon-count total; integration requires actual interval durations and a stated sampling model.

For a rate R_i measured at time t_i, a trapezoidal approximation over valid adjacent samples is

$$N \simeq \sum_i \frac{R_i + R_{i+1}}{2}(t_{i+1} - t_i). \tag{21.11}$$

This is an approximation with the time unit explicitly in seconds. It must not bridge an unobserved gap as though the rate were known there, and it must not be described as an exact detector-event count when the inputs are sampled telemetry.

21.8.1 Counts, range and background

A physically motivated range model predicts a received signal through an effective channel transmissivity. In a far-field approximation, geometric spreading produces an inverse-square range dependence before additional pointing, atmosphere, aperture and detector terms are included. This scaling is an approximation over its stated regime; a fitted normalization can absorb several unresolved physical factors. It is not a separate measurement of each one.

Random background detections can increase both the reported count rate and the error fraction. Their effect is different from deterministic polarization rotation, atmospheric transmission loss or receiver clock offset. The model and its fit constraints must keep these mechanisms identifiable where the available observations permit it. If only a composite parameter is inferred, label it as composite.

A fitted model that misses a held-out trend is useful evidence. It may indicate omitted pass-dependent attenuation, acquisition effects, pointing excursions or a mismatch between the modeled and reported count populations. Do not automatically call it a software defect, and do not suppress the failing samples. The result is a limitation of the tested model and dataset comparison until a concrete implementation error or additional measured input resolves it.

21.8.2 Cumulative key telemetry

The published cumulative key trace records the experiment's reported post-processing output. A step in that trace does not determine the raw-key counts, per-intensity yields or actual reconciliation leakage that produced the step. The archive's cumulative key values can be imported as reference observations without claiming to reproduce them. The absence of a new step is not by itself evidence of zero optical counts in that interval.

The supplied telemetry does not expose the complete per-block, per-basis and per-intensity detection/error record required for an exact finite-key replay. The benchmark therefore does not manufacture those records from a QBER curve or force a chosen security bound to match the published final total. A new exact protocol replay would require those underlying records and an explicitly matched proof and accounting convention.

21.9 Recorded release results and figure-reading guide

The following results come from the released default benchmark configurations and saved numerical reports. They demonstrate the reported comparisons for this release, not a universal accuracy or security specification. Re-running with another source file, fit split, loss model, count convention or threshold creates a different test.

21.9.1 Numerical reference: two levels of successful checks

All 158 selected published Figure 5a rows reproduce the reported integer key length from their supplied intermediate values. Separately, all 29 pinned-author-code cases pass the independent recomputation comparison, covering 348 real-valued quantities. This second layer recomputes the channel counts, decoy population bounds, phase bound, reconciliation leakage and final key expression; it does not reuse those intermediate values as predictions.

The default configurable reference forecast yields 4,652,997 gross bits in 421 s of inclusive one-second exposure bins. Its raw-basis QBER is approximately 0.00151904 and its phase-error upper bound approximately 0.00460830. Zero authentication debit is selected for this reference comparison, so gross and net values coincide. Those numbers are synthetic reference predictions, not satellite-observed key outputs.

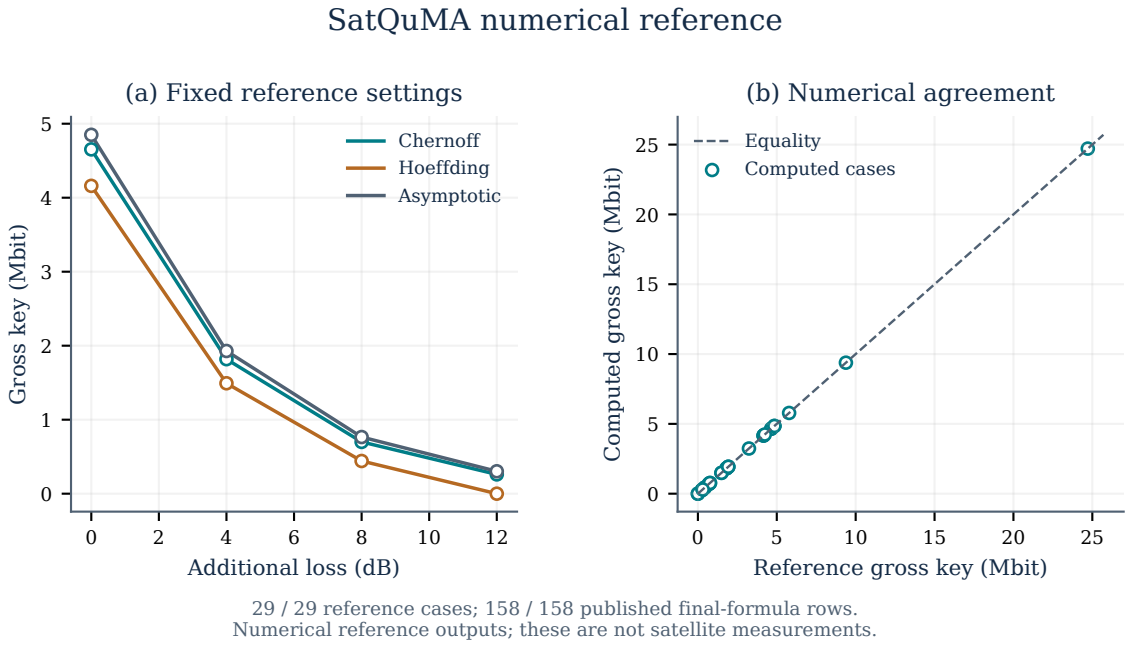

Figure 14: SatQuMA numerical-reference checks. Panel (a) shows the fixed reference settings under three statistical-bound conventions at selected additional losses. Panel (b) compares all tested gross key lengths with the separately executed pinned-author-code results. The equality line is a numerical target, not an experimental fit. The 158 final-formula rows reuse published intermediate bounds; the 29 reference-code cases recompute them. Neither comparison represents satellite measurements.

21.9.2 Laboratory BBM92: bounded agreement within one acquisition

The default laboratory profile performs no additional fit. Of the 150 available window rows, 75 satisfy the declared comparison range and model-domain gates. All eligible rows meet the configured 5% relative raw-rate and 0.01 absolute-QBER criteria. The relative raw-rate RMSE is 2.50755%; the absolute-QBER RMSE is 0.00351214, or 0.351214 percentage point. The largest eligible absolute relative raw-rate residual is 3.96277%, and the largest QBER residual is 0.755944 percentage point.

The criterion applies separately to each metric's eligible-row fraction. It is an engineering tolerance, not a simultaneous confidence band or a statistical significance test. The source and detector parameters already reuse information from this acquisition. The 75 eligible rows are therefore neither 75 independent experiments nor 75 fully independent held-out observations. Forty-four of the complete 150 rows exceed the configured sparse-window occupancy limit; additional rows are outside other selection criteria. Their exclusion is recorded rather than hidden.

Separately, 149 available reference asymptotic-rate values agree with their declared formula to about $1.3 \times 10^{-13}\,\mathrm{s}^{-1}$. That arithmetic check does not convert the projected rates into measured finite secret-key outputs. The release's overall laboratory evidence status remains partial despite the configured residual gate being satisfied.

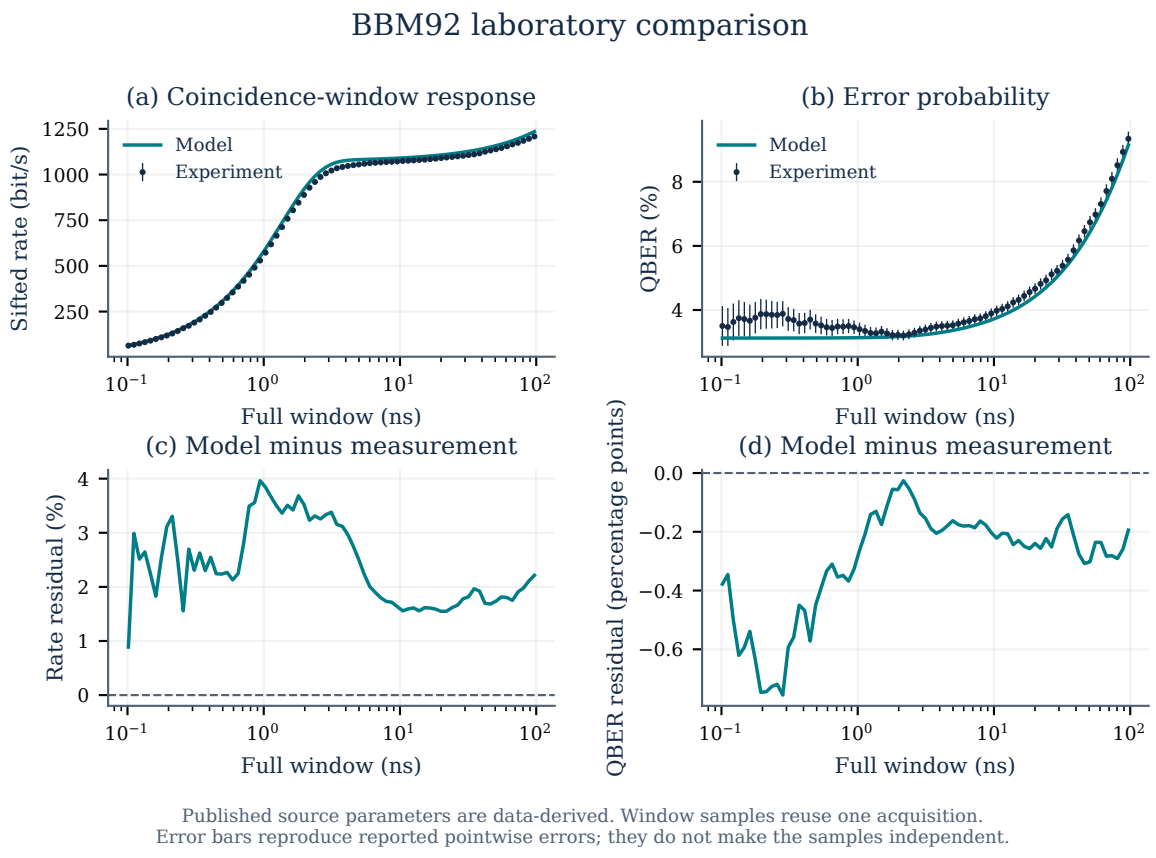

Figure 15: Laboratory BBM92 comparison over the displayed full-window range. Measured sifted rates and QBER are compared with the configured stationary source/detector model; lower panels show model-minus-data residuals. Error bars are the reference pointwise errors. All windows reuse one acquisition and the apparatus estimates are data-derived. A small residual or a successful engineering tolerance check does not establish independent experimental replication or finite-key security.

21.9.3 Jinan-1: an explicit residual rather than a forced match

The default temporal split uses the first 60% of the recorded interval for training and excludes the following 10-s boundary gap. The fit uses 14 populated 20-s count blocks and 13 changed QBER telemetry plateaus. It adjusts only optical throughput, zenith optical depth and signal misalignment. Their fitted values are approximately 0.316103, 0.429554 and 0.00826803, respectively. These are model-dependent fitted parameters; their local fit rank does not resolve physical confounding between source brightness, pointing, throughput and atmosphere.

Nine held-out count blocks give relative MAE 18.0119%, with count MAE approximately $2330.12\,\mathrm{s}^{-1}$. Six held-out QBER plateau transitions give absolute MAE 0.00129940, or 0.129940 percentage point. This QBER statistic compares the instantaneous model at packet report changes; the precise packet-integration supports are not available. It is a diagnostic of available telemetry, not a complete reconstruction of the experiment's packet errors.

The displayed key trace ends at 406,784 observed bits. The predicted final-key field is null: the benchmark does not infer an exact finite key from that trace. Its empirical status remains partial and formal statistical compatibility is not established. No measurement uncertainty model or default empirical acceptance threshold is invented. The observed mismatch is part of the result.

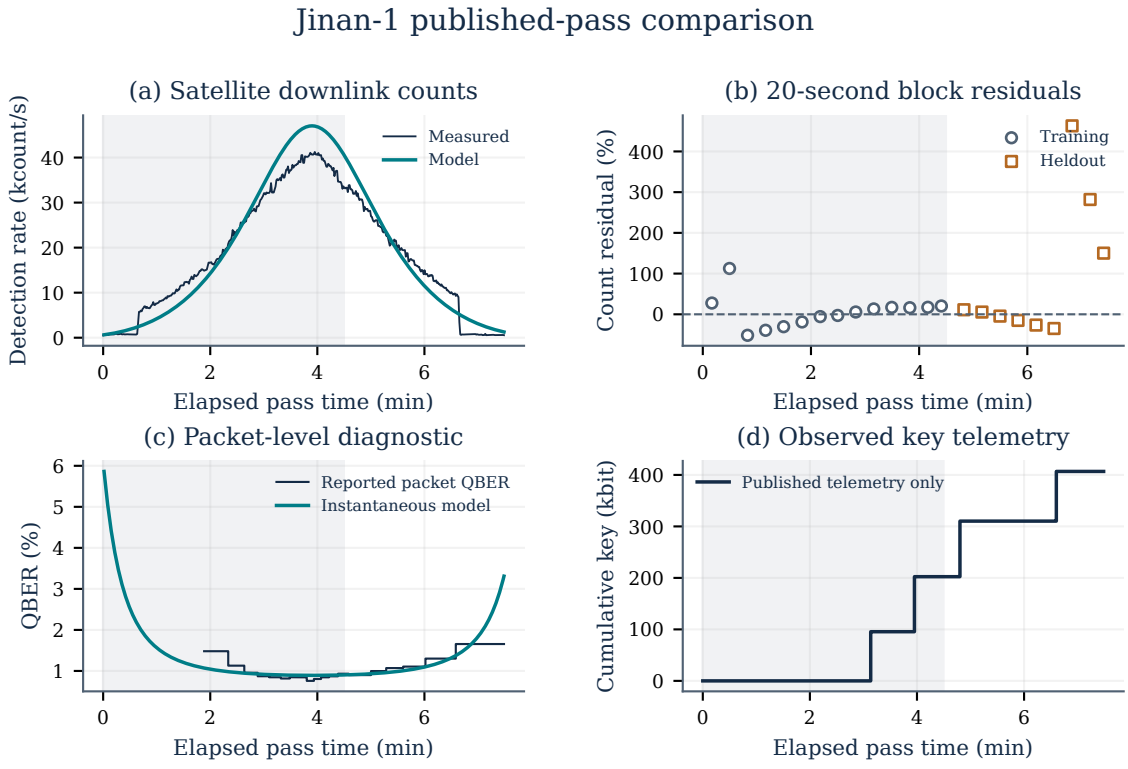

Figure 16: Jinan-1 pass telemetry and the configured count/QBER model. Shading identifies the training interval. Held-out count blocks follow the excluded boundary gap. The residual panel retains the large late-pass relative discrepancies; the count aggregate is normalized by mean observed rate. The QBER curve is an instantaneous prediction compared with packet-level reported values. The cumulative-key panel contains published telemetry only, with no predicted key curve. This is a partial, single-pass empirical comparison.

For clarity, the Jinan normalized count metric is

$$\text{relative MAE} = \frac{\frac{1}{N}\sum_i |\widehat{R}_i - R_i|}{\frac{1}{N}\sum_i R_i}. \tag{21.12}$$

It is not the average of individual relative errors. A low-count block can therefore have a large percentage residual while contributing modestly to this aggregate. The BBM92 relative RMSE instead evaluates $\sqrt{N^{-1}\sum_i[(\widehat{R}_i - R_i)/R_i]^2}$ on its eligible window rows. The two percentages are different diagnostics and should not be ranked as though they used identical normalization and data.

The values in the figures are regenerated from the saved suite report. Their solid curves are model predictions or numerical forecasts; reference observations and pinned-code values are identified separately in the legends. A key-sensitivity curve does not represent an empirical uncertainty band, and a finite-key bound is not a measured QBER. Read the figure caption and its parent report together.

21.10 Interpret numerical and empirical status separately

Read the profile's evidence type and effective settings before its status. A successful numerical check means the implemented expression and declared convention match the selected reference values within the specified tolerance. A successful empirical diagnostic means the chosen model passes the declared criterion on that selected data set. Neither word alone establishes broad physical accuracy.

A tight arithmetic tolerance should be appropriate to floating-point roundoff or an integer rounding convention. A broad empirical tolerance is a user-selected performance requirement unless it is derived from an explicit likelihood, covariance and uncertainty budget. Report a residual exceeding the tolerance as a failure of that declared check. Changing the tolerance changes the question being asked; it does not change the recorded observations or improve a model's prediction.

Keep zero-key, unavailable and failed outcomes in a sweep. Removing unfavorable windows, bins or passes after observing their results biases a summary. A missing value is not a zero error rate. A rejected configuration is not a physical zero-key prediction. A successful source-file integrity check confirms the file identity, not the truth of the measurements or adequacy of their interpretation.

21.11 Use the profiles in an engineering decision

A defensible evaluation begins with the intended operating regime and the quantity that matters: accepted count rate, error rate, finite key per pass or sensitivity to timing and optical loss. Choose the reference layer capable of constraining that quantity. First establish that the numerical implementation follows its equations; then use measured calibration inputs; finally evaluate predictions against observations excluded from calibration where such observations exist.

For a new receiver, record detector count conventions, timing-response widths, filter bandwidth, dead time, basis mapping and background measurement method. For a new satellite pass, preserve acquisition masks, time/range data and atmospheric/pointing inputs. For a security comparison, preserve the raw-key/test populations, decoy settings, leakage, finite-size proof and effective security budget. Similar headline key rates with different conventions are not equivalent results.

The new profiles can support regression control, model comparison, apparatus parameter studies and reproducible communication with a partner laboratory. They do not turn a simulation into an operational QKD system, certify a cryptographic implementation, or establish a PF advantage. Scenario 2 and Scenario 3 retain identical physical histories unless a separate predictive scheduling law changes the actual releases and interactions.

21.12 Primary sources and reproducibility boundary

B1. Sidhu and coauthors (2023). Finite key performance of satellite quantum key distribution under practical constraints. The associated numerical data are archived at Zenodo record 8101679. The author implementation is SatQuMA. Record the separately pinned code revision rather than inferring it from a dataset version label.

B2. Laboratory BBM92 reference. Realistic quantum network simulation for experimental BBM92 key distribution, with the public analysis and data repository. The package's comparison does not require the proprietary AQNSim simulator and does not claim to reproduce an unexposed internal implementation.

B3. Li and coauthors (2025). Microsatellite-based real-time quantum key distribution, Nature 640, 47–54. Source data: Zenodo record 14732295. The published article and an earlier preprint contain different headline maxima; compare the selected archived pass with its own published data rather than mixing maxima from different versions.

B4. Lim and coauthors (2021). Security analysis of quantum key distribution with small block length and its application to quantum space communications. This later security analysis motivates keeping the original Micius observations distinct from a revised key-length calculation. The benchmarks in this chapter do not claim to resolve an exact historical Micius finite-key replay.

Chapter 22

The complete graphical interface

This chapter describes standalone operation. The shared workbench is also available through the separate hosted application; Chapter 27 explains account access, project assets and hosted execution.

The graphical interface is the operating front end for the existing simulation, comparison and benchmark engines. It provides configuration editing, execution, run history, logs, result inspection and downloads in a local web browser. The launcher handles the local Python environment and browser server. Routine end-user operation does not require entering commands in a terminal. The command references in the preceding chapters remain available for automation and for reproducing the same calculation outside the interface.

The GUI does not introduce a different physical model. A selected workflow is translated into the same configuration and runner arguments used by the package's Python entry points. Consequently, the scientific limits, units, noise assumptions, PF interpretation and security-model scope explained earlier in this manual also apply to GUI runs. A preset is an editable starting point; its title is not a validation result. This edition includes explicit study-purpose, physical-topology and quantum-experiment controls, canonical configuration normalization and the resolved-spacetime display. See Chapter 26 for mission apparatus and complete-state coupling.

22.1 Start, install and close the application

Extract the complete package to a writable folder before opening the launcher. Do not launch files while viewing the ZIP archive. Keep the configuration files, benchmark data, documentation and Python modules together. A local folder is preferable to a network-synchronised folder for active calculations, because a running job writes several related files.

On Windows, double-click `PF_Simulation_GUI.pyw`. The supplied `Launch_PF_GUI.vbs` is an alternative launcher when the Python-file association is unavailable. An installed Python version supported by the package is still required. The package retains Python 3.8 compatibility; the GUI does not upgrade the user's Python installation. Python must include Tkinter for the launcher window. The browser itself needs no add-on and loads its application assets from the local package.

The launcher offers the following operations:

1. **Check environment**: inspect the selected interpreter and the installed simulation dependencies. Read the resulting status before starting a large run.
2. **Install dependencies**: create or use the package's local virtual environment and install the supplied requirements. This operation needs network access when the required wheels are not already available. It changes the local environment rather than the system Python version.
3. **Open GUI**: start the local application server and open the browser. The launcher displays its address and progress. An already running server can be opened again after the browser tab has been closed.
4. **Stop GUI**: end the GUI server after work is finished. First cancel active calculations or allow them to complete; closing a browser tab alone is not a request to cancel a calculation.

The launcher also permits choosing an installed Python interpreter with its graphical file picker. Stop the server before changing the interpreter or installing dependencies. Closing the launcher stops its server and active runs; use cancellation deliberately when work should remain incomplete.

The launcher prefers an existing `.venv-pf38` environment when available; otherwise its installation action uses `.venv-pf-gui`. The current installed Python creates that environment. If Python or Tkinter is absent,

install a supported Python distribution using its normal graphical installer, then reopen the launcher. On managed computers, dependency installation may be restricted by organisation policy; the environment check and installation log distinguish that problem from a simulation failure.

The GUI binds to the local loopback interface. It is a desktop application using a browser for display, not a hosted multi-user service. Do not expose its server through a public port-forwarding or reverse-proxy arrangement. Jobs execute with the filesystem permissions of the user who opened the launcher. Downloaded configuration files and result archives should therefore be shared deliberately.

Before replacing the application files, finish or cancel active work and close the launcher. Keep a backup of the existing `gui_runs` directory when its history is needed. When installing the update in a new folder, copy completed job directories into that installation's `gui_runs` directory, retaining each directory's identifier, request, configuration, logs and outputs together. Do not overwrite a different job with the same identifier. Reopening saved results in the updated interface changes their presentation; it does not rerun the simulation or rewrite the saved scientific evidence. The all-segments UTC overview also reads existing saved comparison segments. It requires no new simulation or recomputation of those runs.

22.2 Choose the workflow before editing fields

The top **Study purpose** selector determines which engine, defaults and result types are relevant. Mission workflows separately expose **Physical topology** and **Quantum experiments**; references show their applicability and standalone experiments show their apparatus scope. Select the workflow and then an applicable preset or benchmark profile. Select the link topology where the workflow permits it. Inspect the resulting fields before making detailed edits: changing the workflow or preset can replace its working configuration.

Workflow family	Main purpose	Correct interpretation
Scenario simulation	Run Scenario 1, 2 or 3, or the supported combinations, with SR/static GR and configurable channel, storage and QKD settings.	Mission predictions under the chosen geometry and apparatus assumptions.
Orbit comparison	Compare matched-state numerical dynamics with SGP4, optionally with sensitivity, live updates or a saved-session replay.	Agreement between predictors and internal consistency, not independent ephemeris truth.
Published QKD references	Run SatQuMA numerical arithmetic, BBM92 laboratory-window or Jinan-1 pass comparisons.	Numerical and empirical evidence remain separate; re-binned windows are not independent acquisitions.
Research precision	Propagate declared input distributions, export full research products and check configured study designs.	Conditional uncertainty and numerical stability; not calibrated prediction accuracy.
Research risk audit	Evaluate intended-use requirements, uncertainty coverage, observations and evidence records.	Missing evidence can block readiness despite successful execution.
Research acquisition validation	Calibrate and freeze the downlink count/QBER model, then apply it to declared independent acquisitions.	Scoped imported-observation agreement only; synthetic fixtures block empirical claims.
Micius reference benchmark	Evaluate the documented source-audited reference and conditional channel comparison.	Compatibility with selected reported quantities is distinct from exact experimental reproduction.
Quantum protocol benchmarks	Run Bell, teleportation, swapping, purification, memory, repeater, interference, QKD-sweep and channel-characterization profiles.	Numerical protocol/model checks with explicit assumptions; see Chapter 25.
Micius engine integration	Exercise the scenario/Qiskit calculation and its independent consistency checks.	Implementation evidence for the configured case; not an additional empirical measurement.

A QKD mission preset and a published QKD benchmark serve different purposes. For example, a Micius-type mission configuration supplies geometry, source and link parameters for a forecast. The Micius reference benchmark compares specified quantities with reported evidence. The Micius integration workflow intentionally keeps its controlled engine fixture fixed: its exposed run options vary the supported source fidelity, signal/background ratio, uncertainty multiplier and loss/calibration parameters. Its configuration object is empty; arbitrary mission geometry belongs in the separate Micius-type simulation preset. Neither should be substituted for the other when reporting validation. The same distinction applies to SatQuMA-compatible security calculations in a mission run versus the dedicated SatQuMA numerical benchmark.

22.2.1 Topology and applicability

An **inter-satellite** simulation uses satellite receiver worldlines. A **space-to-ground** simulation uses supported ground-station receivers and introduces their site coordinates, visibility/elevation and downlink assumptions. The application constrains available topologies for workflows whose data fix the experimental setting. A laboratory benchmark cannot become a satellite experiment merely by relabelling its topology. A ground-pass benchmark cannot provide an inter-satellite empirical reference without new appropriate data.

Topology selects the applicable configuration structure; it does not calibrate its parameters. A ground link still needs valid station coordinates, atmospheric and collection assumptions, and a useful visibility interval. An inter-satellite link still needs unobstructed geometry, source and detector specifications, and appropriate tracking inputs. Preserve the declared source of each parameter, especially when an illustrative preset is adapted for a real instrument.

The two-detector coincident construction also retains its original purpose. Scenario 1 uses the coincident detector profile; Scenario 2 uses the separated schedule selected by the older routine; Scenario 3 applies the full PF diagnostic to Scenario 2's physical event history. The GUI does not silently turn Scenario 3 into a new independently optimised emission controller.

22.2.2 The four workspace views

Configure contains the selectors labelled *Workflow*, *Link configuration* and *Starting configuration*. Its *Run options* panel stores execution choices separately from the engine configuration. Expand its advanced overrides to inspect the less frequently used options; a collapsed group is still part of the saved request. The configuration editor has *Fields* and *JSON* views, *Import* and *Export* controls, a field search and optional section templates. Use *Validate configuration* to check request structure and applicability, then *Run simulation* to submit it.

Runs lists the saved job history. Select a run to follow its log. *View results* opens its result workspace; *Use configuration* loads its settings as the basis for another run; *Cancel run* stops queued or active work. The *Follow log* control keeps the latest output in view.

Results presents available status cards, metrics, *Diagnostic guidance*, *Figures and visual outputs*, *Reports and documents* and *Saved outputs*. Use *Download run ZIP* to keep the job's files together. A figure card groups its available formats; selecting its preview opens a larger view.

Documentation links to the package documents and displays the local environment. The header's *Environment* control provides a quick route to that information. Dependency installation remains an explicit operation in the desktop launcher, not an automatic side effect of selecting a workflow.

The header's *Orbit viewer* control opens the model-versus-SGP4 trajectory display for saved comparison sessions. The *Configure live orbit comparison* shortcut in *Configure* prepares the comparison workflow for a current-UTC run. This is a separate result view from the scenario fidelity, purity and collection charts; the launch procedure is given below.

22.3 Edit a complete configuration

The form editor exposes the selected configuration's nested fields rather than limiting the user to a few headline controls. Expand the appropriate object or section to inspect its contents. Use the search facility to find a key or topic in a large configuration. Numeric, Boolean, text, null, object and array values must retain their JSON

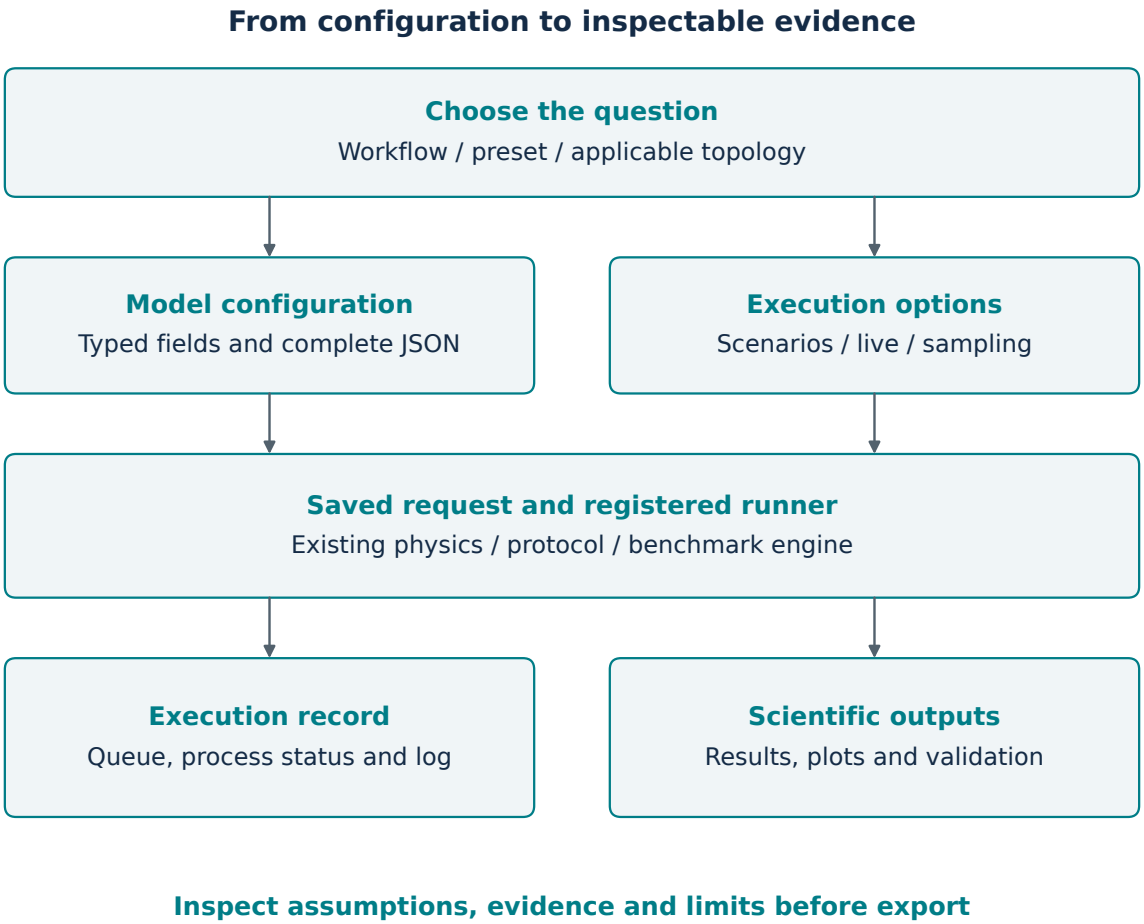

Figure 17: The GUI preserves the engine boundary. Workflow, topology and configuration produce a saved run request; the existing runner produces the scientific outputs. Execution status, validation evidence and empirical interpretation are inspected separately. The diagram describes software flow, not new physical evidence.

types. The advanced JSON editor and import/export controls provide access to the complete configuration, including supported optional keys that are not present in the starting preset.

A workflow also has **execution options**. These control matters such as scenario selection, live iteration count, sampling, controls and whether plots are generated. They are distinct from the model configuration. When an option overrides a configured value, the submitted job request and resolved output are the record of what was actually run. Save both when comparing experiments.

22.3.1 Canonical settings, overrides and migration

The GUI uses the same physical and execution alias checks as the command-line loaders. A configuration has one canonical spacetime selector, `spacetime`, and one canonical timing selector, `timing_mode`. An imported legacy `propagation_model` or `apply_QCS_corrections` field is migrated when its meaning is unambiguous and removed from the normalized configuration. If a canonical field and its legacy alias select different values, import is rejected and the current editor remains unchanged. The user must resolve the original conflict; selecting a later run override does not conceal it.

The resolved-spacetime summary distinguishes the base configuration, any explicit run override and the resulting mode. For example, a valid `flat_sr` base with an explicit `gr_static` run override selects static GR. An absent override preserves the base mode. A file containing both conflicting spacetime keys is invalid before this precedence rule is applied. Inspect the saved effective configuration to confirm the canonical value that reached the engine. The same agreement-before-override rule applies to timing and the sampling/control execution aliases.

For old files, retain a copy of the original and choose the intended canonical value explicitly. Remove the redundant alias, validate the new configuration and start a new run. A historical sealed session with ambiguous aliases is rejected on replay; do not edit its captured files or reseal it as though it were the original experiment. Prepare an explicit new configuration and new session instead. The correction concerns configuration identity and precedence, not a change to the SR, GR or quantum equations.

The editor also preserves intentionally derived or inherited QKD values. `qkd.optical_links.a.divergence_half_angle_rad` and the corresponding arm-B field remain null when divergence is to be derived from the current wavelength and waist at execution. An explicit number deliberately pins it. Likewise, omitting `qkd.satquma_security.authentication_bits` preserves inheritance from `qkd.security.authentication_bits`; adding a separate SatQuMA value overrides that inheritance. Completing the form does not freeze these derived defaults and silently disconnect them from later edits.

22.3.2 Configuration groups in a mission simulation

The tables in the earlier configuration and QKD chapters remain the authoritative field-by-field reference. The GUI groups help the user locate those fields; they do not change their meanings.

Configuration area	Typical choices	Check before running
Scenario and sampling	Scenario set, time interval, epoch count, count sampling, controls and seeds.	Counts, seconds and absolute UTC epochs are different quantities. Small pilot runs are useful before long sweeps.
Spacetime and timing	`flat_sr` or `gr_static`; selected-frame, ECI or unadjusted timing where supported.	A coordinate convention alone is not a channel-noise mechanism. Static GR remains the documented approximation.
Orbits and tracking	Circular profiles or SGP4 inputs; numerical force model and comparison settings; epoch and data files.	Use appropriate units, current timing data for live work and the intended offline/online policy.
Receiver geometry	Satellite receivers or ground stations, station coordinates and elevation limits.	A valid numerical trajectory can still yield no visible optical link.

Configuration area	Typical choices	Check before running
Source and quantum channel	Bell target, source quality, phase variances/covariance, deterministic phase and provenance.	Enter measured or explicitly illustrative apparatus parameters; do not add a scenario-index penalty.
Storage channel	Enable flag, proper-time dephasing/loss constants, retrieval efficiencies and bypass policy.	A null time constant and a zero time constant are not interchangeable. Avoid counting the same imperfection twice.
Collection and detectors	Optical geometry, efficiency, loss, detector timing, gating, background, dead time and rate assumptions.	Conditional state quality and accepted count rate are different outputs.
QKD	Protocol, bases/intensities, exposure/integration, count mode, leakage, security assumptions and finite-key model.	A zero secure-key budget can be the correct calculated outcome. No operational key is created.
Execution and provenance	Sampling/controls, output selection and source identifiers.	Preserve the configuration together with seed, version, inputs and produced results.

22.3.3 Numbers, units and validation

Enter values in the units stated by the key or field help. Metres and kilometres, seconds and nanoseconds, radians and degrees must not be mixed. JSON uses decimal points and does not accept `NaN` or infinity. Do not surround a numeric value with quotation marks. An empty string is not a valid substitute for a missing numeric value. Null is allowed only where the underlying model documents that meaning, such as a disabled decay mechanism.

Validation takes place in stages. The browser can reject malformed entries; the workflow adapter checks applicable options and topology; the simulation engine performs its domain checks and physics-specific validation. A form that can be submitted is not proof that the experiment is physically possible. Read engine warnings and result statuses even after a clean submission.

For correlated phase noise, retain the covariance constraints described in the quantum chapter. In particular, a configured correlation coefficient must be in its allowed interval, and variances must be nonnegative. The interface does not infer a measured correlation from the selected scenario label. For a fixed physical history, an alternative coordinate description must not create an additional decoherence contribution.

22.3.4 Import, export and paths

Export the working configuration before switching to a different preset when you want to keep an experiment. Importing a JSON configuration loads editable values; it does not execute the imported file as Python. Review all path fields and parameter-source strings after import.

Paths need particular care because the browser does not reliably expose the original directory of a selected upload. The GUI resolves its packaged relative inputs against the package directory, while explicit absolute paths identify local inputs unambiguously. If an imported configuration previously relied on paths relative to another folder, replace those paths with their intended absolute locations or copy the inputs into the documented package layout. Do not assume that a JSON filename alone identifies the directory containing its supporting data. The saved resolved configuration should be inspected before using a run as reproducibility evidence.

Exported configurations are plain editable JSON. They do not embed external GP records, timing tables or published reference data merely because they contain paths to those files. A complete run archive and, where applicable, the frozen comparison session are the appropriate evidence bundle for replay.

22.4 Run, monitor and cancel jobs

After reviewing configuration and execution options, submit the run from the GUI. The application creates a new job directory under `gui_runs`. The request, working configuration, process log and produced results are retained there. New jobs do not overwrite the scientific files of an earlier job. A persistent history allows completed work to be selected again after the browser has been reopened.

The queue separates *queued*, *running* and terminal job states. Open a job to follow its log and inspect results as they become available. Long live runs or benchmark suites can produce intermediate files before their final summary appears. A partially written result is not a completed benchmark; use the terminal job status together with the final reports.

Cancel an unwanted queued or running job using its cancellation control. A cancelled calculation may leave useful diagnostics and partial outputs, but these should be labelled incomplete. Cancellation is not a numerical failure, and it does not make a partial finite-key accumulation a validated complete exposure. If the server or computer stops unexpectedly, inspect the recovered job state and logs before relying on its output.

The application launches registered runners with explicit argument lists. The GUI has no field for arbitrary shell commands. This keeps the interface focused on supported package workflows and avoids changing the engines' scientific behavior to accommodate presentation needs.

22.4.1 Live tracking and comparison

For live scenario tracking, choose *Simulation · quantum states, timing and QKD* in the workflow selector. Select a *Current UTC tracking* starting configuration: *flat SR*, *static GR*, *phase noise*, or *static GR and phase noise*. Confirm *Repeat at current UTC* in *Run options*. Inspect *Live updates* and the update interval in the advanced run options, then select *Run simulation*. A job created this way already runs live SGP4 predictions; no second tracking connection must be launched from its results page.

Set a finite iteration count and a practical update interval. The runner uses current UTC for each update and requires the applicable online tracking/time-data policy. Do not simultaneously supply a fixed historical epoch and expect a current-UTC run. The observed interval includes calculation and file-writing time; the recorded UTC epochs, rather than an assumed fixed cadence, identify when the predictions apply.

Live tracking downloads or reuses public orbit-prediction inputs as configured; it is not direct telemetry from a spacecraft. A network outage, expired leap-second table, unavailable Earth-orientation input or stale orbit element is a data availability problem. Read its specific message rather than treating it as a PF, GR or quantum-channel failure. The GUI preserves the same flat-SR and static-GR choices for supported live runs.

A matched-state comparison runs both orbit-source branches with the documented initialisation. Inspect residuals, RIC components and model settings, not merely the final completion indicator. A replay uses the saved session's frozen inputs; changing the physical configuration while requesting an exact replay would change the experiment and is not allowed by that workflow.

Open the satellite trajectory viewer

The scenario workflow produces quantum, timing and collection outputs. The orbit comparison workflow also produces the numerical-model and SGP4 trajectories needed by the orbit viewer. To launch that view without a command line:

1. Double-click `PF_Simulation_GUI.pyw`, or `Launch_PF_GUI.vbs` on Windows, and choose *Open GUI* in the launcher.
2. Select *Orbit viewer* in the browser header to open the saved comparison display. Keep the GUI server running while using this view.
3. To obtain a new live comparison, return to the *Configure* workspace and select *Configure live orbit comparison*. The shortcut loads *Orbital mechanics · matched SGP4 comparison*, retaining the current flat-SR/static-GR choice.

4. Inspect the resulting *Current UTC orbit comparison · SR* or *Current UTC orbit comparison · GR* preset. Review catalogue identifiers, tracking policy, numerical force model, iteration count and interval before selecting *Run simulation*.
5. Return to *Orbit viewer*. In *Saved session*, select an individual comparison segment or *All segments — UTC overview* for its live run. Inspect trajectory and position/velocity residual plots together. The overview shows successive forecast windows on one UTC timeline as new segments are saved.

The viewer reads comparison-workflow outputs. A scenario-only live job does not automatically create the model-versus-SGP4 comparison files, and an empty session list before the first comparison is expected. Its trajectory projections show calculated ephemerides; they are not a ground-control display or a direct telemetry feed. A shorter residual indicates closer agreement between the two configured predictors, not independently measured orbit accuracy.

Review all forecast segments on a shared UTC timeline

Choose *All segments — UTC overview* in *Saved session* to inspect the saved `segment_0001…`, `segment_0002…` and later forecasts belonging to the same live comparison run. Each live run has its own overview choice; unrelated runs are not combined. The individual segment entries remain available with their existing timeline and detailed review controls.

Use the overview controls in this order:

1. Select the *Coordinate / spacetime subset*. Different coordinate charts or spacetime modes must be inspected in their own subsets before comparing their values.
2. Set *Forecast segment* to *All displayed segments*, or select one forecast to reduce visual overlap. Select the satellite and projection as needed.
3. Read the trajectory projection together with the position/velocity residual plots and the *Forecast starts and coverage* table. The table gives each segment's initialization UTC, first and last available sample UTC, point count and status. Initialization time and the first available sample need not coincide when output is incomplete.
4. Use *Open segment* in the table to open that forecast's individual saved-session view. Its original artifacts and configuration remain the evidence for that calculation.

The residual plots share the axis *UTC offset · s*, with its UTC origin printed beside the view. The origin is the earliest valid initialization or sample UTC across the whole group; selecting a subset does not reset it. Points use their recorded `epoch_utc` timestamps. Later forecasts therefore occupy their later UTC positions instead of restarting at a common display zero. The coordinate is a civil UTC offset for display; it is not a replacement for physical propagation duration or the saved `elapsed_seconds`, which may follow the engine's TT/TAI convention. Missing, invalid or unsupported UTC values, including unsupported leap-second timestamps, are reported rather than reconstructed from a segment number or assumed cadence. Inspect the coverage and status information before relying on an apparently missing point.

Each segment starts a new numerical forecast from its own SGP4-matched initial state. It does not continue the previous segment's integrator state. The supplied live comparison presets use a 60-second forecast window, while the runner's default minimum start-to-start interval is 30 seconds. These windows can overlap; calculation and file-writing time can move the actual starts farther apart. The overview uses recorded times instead of assuming the requested cadence.

For example, two hypothetical 60-second forecasts starting 30 seconds apart have the following coverage. These times illustrate the display and are not saved run results:

Forecast	Start UTC	End UTC
First	12:00:00	12:01:00
Second	12:00:30	12:01:30

Between 12:00:30 and 12:01:00, both forecasts can contain predictions for the same UTC instant. Both are retained as distinct traces. Lines are not drawn from the end of one segment to the beginning of the next. Vertical ticks on residual plots mark forecast initialization; circles on projected trajectories identify their first available samples. A residual returning near zero at a new start reflects the matched initial state. It is not new evidence of measured orbit accuracy.

This combined display is a sequence of separately initialized forecasts, not one continuous numerical integration or an independently measured spacecraft trajectory. Its time alignment helps compare the forecasts without changing the scientific interpretation of any segment.

The viewer checks for saved updates every five seconds. An active overview stays selected and includes newly saved segments when refreshed. *Follow newest session* is disabled while the overview is selected because the whole overview updates; the control remains available for individual-session review. A display refresh reads available results and does not request a new forecast, rerun the engine or rewrite saved evidence.

22.5 Read the results workspace

Select a job in the history to open its result workspace. The workspace combines execution status, available validation summaries, logs, plots and downloadable artifacts. Plot and report availability depends on the workflow and on how far it progressed. A disabled plot option intentionally produces no plot for that run. The result files remain authoritative when a condensed card shows only a subset of their fields.

Output family	What to inspect	Common interpretation error
Scenario state plots	Conditional fidelity, purity, phase/storage inputs and availability flags at each epoch.	Equal Scenario 2/3 values are not automatically a bug: those scenarios share physical history.
Collection and count plots	Visibility, accepted rate/probability, exposure, sampled versus expected counts and gate acceptance.	High conditional fidelity does not imply a usable count rate.
Timing/PF diagnostics	Propagation statuses, emission holds, simultaneity residual and construction validity.	A selected-frame residual is not direct evidence of a scheduling advantage.
Orbit comparison	Position/velocity residuals, RIC decomposition, force models, epoch and frozen inputs.	Predictor agreement is not a measurement of the true orbit.
QKD summaries	Protocol, sifted counts, QBER, leakage, finite-key terms, security assumptions and abort reason.	A positive forecast is not an operational cryptographic key or a security certification.
Published benchmarks	Numerical and empirical statuses, reference identity, calibration/holdout split, residuals and limitations.	A completed script is not necessarily an empirical pass.
Logs and metadata	Runner errors, warnings, request, resolved configuration and input provenance.	A missing graph alone does not identify the underlying cause.

22.5.1 Unique figures, format files and reports

The figure gallery groups image and PDF files with the same filename stem in the same directory into one figure card. For example, `figures/benchmark_jinan.png` and `figures/benchmark_jinan.pdf` are two format files for one figure. A matching stem in another directory remains a separate figure, so successive live-update snapshots are not merged. The displayed figure count is the number of unique figures; the format-file count is the number of available files. Open the image preview to enlarge it, or use the card's format links to open the saved files. A plot PDF is a vector version of the figure, not an additional narrative report.

Reports and documents lists available report/document artifacts separately; the heading is *Documents* when none is identified as a report. Known figure PDFs use *Open vector PDF*; known PDF reports use *Open report (PDF)*. An unclassified PDF uses the neutral *Open PDF* action; its extension alone does not establish whether

it contains a plot or narrative. Published-reference benchmarks save detailed JSON reports: the combined `benchmark_suite.json` and the selected profiles' `satquma.json`, `bbm92.json` and `jinan.json`. Read their numerical and empirical statuses, parameters, residuals, provenance and limitations together. These workflows do not generate an additional detailed narrative PDF report; their plot PDFs are figure formats. The short `README.md` summarizes status, while `resolved_config.json` records the suite settings. A benchmark's *Open result record (JSON)* link and the saved-output inspector open the recorded evidence without rerunning the benchmark. Plot-disabled runs can still contain these reports.

Interactive data charts provide a legend control for showing or hiding each series and a *Show data* control for its plotted values. Hiding a series changes only the display; it does not remove that series from the saved result. Saved figure formats remain individually downloadable from their grouped cards. Their presentation can differ from the interactive view while using the same results.

22.5.2 Live timelines, snapshot markers and coincident curves

Each live scenario update has its own run epoch and can contain only one sampled instant. Its saved `comparison.png` therefore shows one marker per available scenario, with no time-series segment to draw. A single-point snapshot is not an empty calculation. The GUI's aggregate live chart instead groups updates by scenario and places them at elapsed seconds relative to the first recorded UTC epoch. A run with ten updates and three scenarios can therefore show three ten-point series, rather than thirty one-point series all placed at local time zero. The plotted time comes from the recorded epochs, not from multiplying an update index by the requested interval.

When conditional-channel inputs remain unchanged, fidelity and purity may be constant across updates. They can also be exactly equal between scenarios. Coincident curves occupy the same coordinates; the interface does not offset their numerical values merely to separate their colours. Use the legend to hide other series or *Show data* to inspect each value. Saved single-update figures and their update labels remain useful for checking a particular instant.

A zero accepted pair rate belongs on the count graph. If all optical links are Earth-blocked, that zero is a physically meaningful collection outcome. The conditional fidelity and purity may still be calculated from the configured apparatus channel, but no accepted sample exists for tomography. The visibility and availability statuses must be read alongside state quality. Missing reception events leave arrival-time and full-PF reception diagnostics unavailable; they do not establish perfect synchronization. A zero maximum error in a validation report with zero resolved propagation epochs is not a successful timing test.

Open plots at their intended size before interpreting a small difference. Use the saved full-precision JSON values when checking a formula or comparing near-equal results. A screen thumbnail can hide axis exponents, confidence intervals or conditional-state qualifiers. Figures from different workflows can also use different horizontal coordinates: orbit snapshot time, pass time, block index and laboratory window duration are not interchangeable.

In *Saved outputs*, the *Inspect* control opens supported text files inside a scrollable dialog. JSON is pretty-printed for reading; this does not rewrite the saved result. Use *Open original* to open or save its original bytes. The in-app text inspector has a 15 MB limit; larger files remain available as artifacts for an appropriate external analysis tool.

Download individual artifacts for detailed analysis or download the job's archive to retain its complete evidence. Preserve configuration and logs with the graphs. Exported figures alone do not describe the source fidelity, phase covariance, storage model, tracking epoch or QKD security settings that produced them. A downloaded archive is a copy of a run's files, not an automatic rebuild of the entire application distribution. Share completed job evidence with *Download run ZIP*. Application repackaging excludes untracked files in the local `gui_runs` workspace by default, including job history and server connection state; include completed run evidence explicitly when it belongs in a distribution.

22.6 PASS, FAIL and recommendations

The GUI separates three questions:

1. Did the requested process execute and produce its outputs?
2. Did its declared numerical or implementation checks pass?
3. Does the available independent empirical evidence support the stated comparison, at the stated tolerance and under the stated assumptions?

These questions can have different answers for the same job. A benchmark can complete normally, pass arithmetic checks and still report partial or unresolved empirical agreement. The interface retains those distinctions rather than converting every completed job into a scientific PASS.

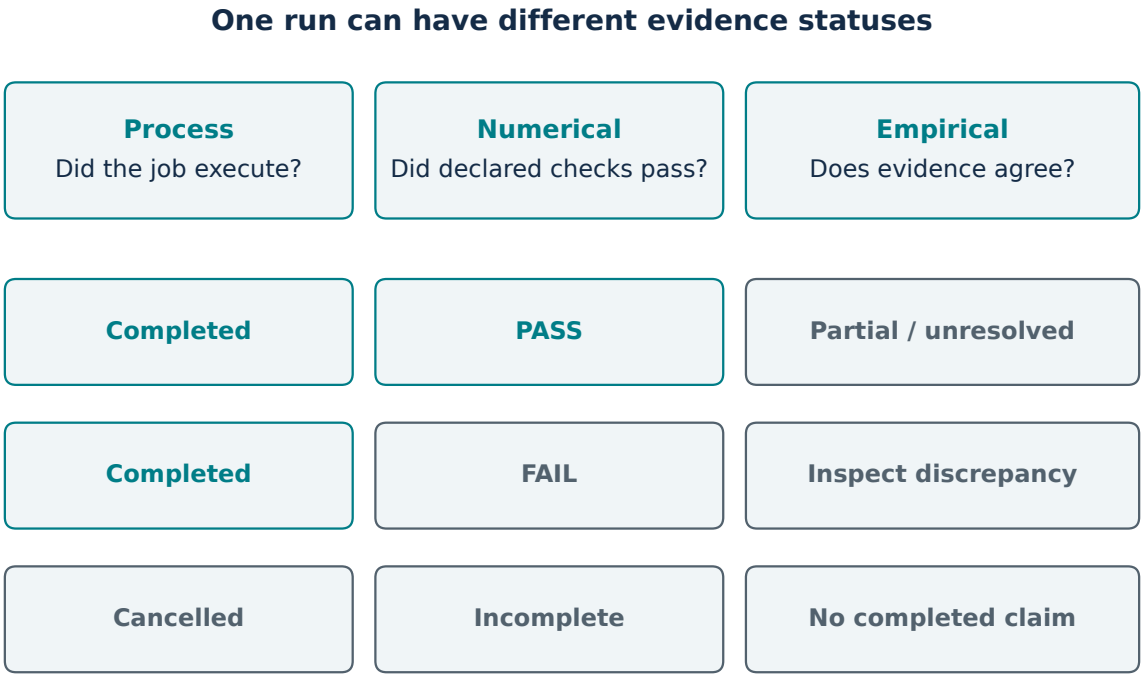

Figure 18: Read status by evidence layer. Process completion, numerical validation and empirical agreement answer different questions. The illustrative combinations show why a green execution indicator cannot replace a benchmark's reported empirical status. No experiment-specific result is asserted by this diagram.

Observation	Appropriate next step	What would not resolve it
Invalid configuration	Correct the named type, range, units, path or incompatible option, then rerun.	Ignoring the error or interpreting absent output as zero noise.
Tracking/data error	Check online/offline policy, network availability and the specified valid timing/orbit files.	Relaxing a scientific acceptance threshold.
No visible/accepted pairs	Examine line of sight, elevation, timing gate, optical loss and exposure for a physically useful interval.	Claiming high fidelity for a state with no measurable accepted sample.
Zero finite-key budget	Inspect count statistics, QBER, leakage and finite-size terms under the chosen security model.	Promising that a longer exposure always produces a key; an adverse asymptotic fraction may remain nonpositive.
Numerical check failed	Inspect the diagnostic residual, units, valid domain, reference identity and solver behavior; preserve the failure.	Increasing a tolerance solely to obtain a PASS label.

Observation	Appropriate next step	What would not resolve it
Partial empirical agreement	Review data provenance, fit/holdout separation, model limitations and uncertainty.	Treating fitted inputs or unavailable measurements as independent confirmation.
Cancelled/interrupted job	Keep its logs as incomplete evidence and resubmit a deliberate run if needed.	Reporting a partially accumulated result as a completed benchmark.

Recommendations are diagnostic suggestions, not automatic changes to the submitted configuration and not guarantees of a PASS. They identify settings or inputs worth checking when the result supports that advice. A change in detector background, loss or memory lifetime must correspond to a stated apparatus assumption or measurement; it is not a harmless display adjustment. A change in security level, sample size, reference data or acceptance criterion changes the claim being tested and must be recorded explicitly.

For a useful controlled study, duplicate the working configuration, alter one justified factor, and preserve both runs. Compare the underlying numerical values and assumptions rather than counting green indicators. In particular, do not select a favorable benchmark tolerance after examining the same residuals and present the resulting pass as a predeclared independent validation.

22.7 Worked graphical workflows

22.7.1 First scenario run without network inputs

Choose the scenario-simulation workflow and a supplied visible-link circular-orbit preset. Select all three scenarios and `flat_sr`. Start with a small number of epochs and leave live tracking disabled. Review source/channel provenance and whether phase noise, storage and QKD are enabled. Submit the job, follow its log, then inspect the scenario plots and JSON records.

Expected interpretation is more important than an expected headline number. With identical conditional-channel inputs and no schedule-dependent imperfection, the scenarios may have the same conditional fidelity and purity. Scenario 2 and Scenario 3 should share quantum outcomes when they share their full physical history and protocol. Rates can still differ from Scenario 1 because collection and timing are separate calculations. Record all of these observations rather than labelling equality as failure.

Export the configuration and job archive. For an SR/static-GR comparison, make a second run with the spacetime option changed and keep the remaining assumptions fixed. Read the stated GR approximation and use sufficiently precise numerical outputs; the presence of a GR mode does not imply a large observable difference.

22.7.2 A configured QKD mission study

Choose a QKD-capable scenario preset and the supported topology. For dual receiver entanglement QKD, inspect both links and the BBM92 assumptions. For a supported decoy-state downlink, inspect its active arm, three intensities, basis probabilities and selected security model. Set an exposure or integration policy that matches the modeled acquisition rather than summing unrelated independent snapshots without justification.

Run and inspect visibility, accepted counts and QBER before reading the final key-length forecast. A zero budget has a documented cause; inspect its terms. If considering a lower-loss optic or reduced detector background, save that change as a new design assumption and compare the two saved runs. Do not describe the model forecast as generation or delivery of a usable encryption key.

22.7.3 A published-data validation workflow

Select the dedicated benchmark rather than a similarly named mission preset. Retain its original reference data for the initial run. Review the profile's calibration choices and declared numerical/empirical criteria, then execute it. Inspect the numerical-validation and empirical-validation blocks separately. The BBM92 laboratory

acquisition, the Jinan-1 ground pass and the SatQuMA numerical reference do not provide interchangeable evidence.

For a custom sensitivity experiment, export a copy of the benchmark configuration, change the intended assumption, and retain a clear identifier for that variant. A different window selection, fit split or criterion is a new analysis. Keep the unaltered reference run alongside it. When testing Micius, retain the distinction between the source-audited reference calculation, the engine-integration checks and unresolved exact reproduction of the historical experiment.

22.7.4 A live run followed by a saved comparison

After the environment check succeeds, choose an online tracking/comparison preset. Review data-download permissions and timing inputs, set a small finite live iteration count, then run. Each update should show its actual UTC epoch and propagation status. Inspect the log if acquisition or time conversion stops. A valid current-UTC update is an orbit prediction at that epoch, not proof of a live spacecraft connection.

For reproducibility, retain the frozen session produced by the matched-state comparison. Use the replay workflow with that saved session to check numerical reproduction of its inputs. Perform changed-model comparisons as new sessions; replay and new experiments have different purposes.

22.8 Troubleshooting and operational limits

Symptom	Action
Double-click does not open the launcher	Verify extraction, Python/Tkinter installation and file association; use the supplied Windows alternative launcher.
Dependency check fails	Read the named missing/import-failing package and use the Install dependencies action. Retain the installation log if network or package availability prevents completion.
Browser page cannot connect	Check that the launcher server is running, reopen the displayed local address and inspect the launcher log. A browser tab can outlive a stopped server.
A configuration import changes path behavior	Inspect relative paths; use the intended absolute local paths or package-relative inputs. Export the corrected configuration as a new copy.
A job fails before creating scientific results	Read the process log and configuration validation message; distinguish dependency, input and model-domain errors.
Graphs appear identical	Check full-precision results and the actual channel/history settings. Shared input states or Scenario 2/3 histories can correctly yield equality.
Only one point appears in a saved live figure	A child snapshot can contain one epoch. Use the aggregate live chart for the recorded UTC timeline; use legend toggles and the data table when scenarios overlap.
The count plot stays at zero	Inspect visibility, gate acceptance and loss. An Earth-blocked link cannot be restored by changing phase noise. Conditional state quality is not a measured result without accepted counts.
The orbit viewer contains no session	Run the matched-SGP4 comparison workflow or select an existing comparison session. A scenario-only run does not produce the required trajectory comparison.
The orbit viewer repeatedly shows a forecast starting near zero residual	Select *All segments — UTC overview* to inspect successive initializations on their shared UTC axis. Forecasts restart from matched SGP4 states and may overlap; use the coverage table and *Open segment* to inspect a particular forecast.
A live run stops at a time-data check	Inspect the supplied UTC epoch, leap-second/Earth-orientation coverage and configured online/offline policy. Do not bypass validity checks without documenting their effect.
A completed benchmark is not an empirical PASS	Open its separate evidence blocks and limitations. Completion and numerical agreement do not supply missing measurements.

Symptom	Action
The history contains a cancelled/interrupted run	Preserve it as incomplete; make a new job for a complete calculation.

The interface does not add automatic instrument calibration, a global search for an optimal protocol, hardware control, public-server user accounts, an operational QKD service or experimental confirmation of the PF model. Its complete configuration access means that users can configure supported model features; it does not mean that every mathematically possible channel, orbit or quantum protocol is present. The earlier scientific chapters define those implementation boundaries.

22.9 Software organisation and reproducibility

The `pf_gui` package separates workflow/configuration definitions, job execution and persistence, result interpretation, HTTP serving and browser assets. The graphical launcher is a separate entry point. The existing simulation and benchmark engines remain the calculation layer. This division makes it possible to extend a workflow's form or result view without embedding physical equations inside user-interface code.

For every result used in a report, retain the selected workflow and topology, configuration, execution options, package release, input provenance, process log, final scientific files and relevant validation status. When sharing a saved job, state whether it used expected or sampled counts, live or frozen tracking inputs, SR or static GR, a configured or measured noise model, and a numerical benchmark or independent empirical reference.

The package's GUI tests and browser-validation report describe the checks actually performed for the release. A successfully rendered screen confirms that the interface can display its controls and results in the tested browser. It does not replace numerical regression tests or independent scientific validation. The same principle applies to the publication checks for this chapter: embedded fonts and correct mathematical notation certify document construction, not the truth of a modeled physical hypothesis.

Chapter 23

Research exports, uncertainty and observation validation

This chapter documents the research precision extension supplied with manual edition 1.8.0. It adds structured numerical exports, matrix diagnostics, conditional Monte Carlo summaries and comparison with user-supplied observations. It calls the same scenario engine described in the preceding chapters. The extension does not replace the propagation equations, introduce a new PF release policy, infer apparatus parameters from a frame label or establish an empirical accuracy specification.

The word *precision* here describes inspectable numerical data and an explicit uncertainty workflow. A file containing many decimal digits is not evidence that a satellite position, clock reading, optical efficiency or quantum state is known to that accuracy. Use this extension to expose assumptions, preserve reproducible tables and test a declared observation comparison. Treat its forecasts as conditional on the supplied physical model and input data.

23.1 What the extension preserves

Scenario 1 remains the coincident-detector baseline. Scenario 2 retains its configured physical release schedule. Scenario 3 applies the full-PF construction to those same Scenario-2 receptions. An export or uncertainty draw must preserve that relationship. Coordinate selection alone cannot create a different quantum channel, a higher count rate or additional secret bits for Scenario 3.

The state conditional on collecting both photons, the state reconstructed from finite counts and the BBM92 detected mixture answer different questions. The extension retains their source records; its principal polarization table diagnoses the conditional channel state. An unresolved event remains unavailable. An occulted link is not transformed into a solved reception, and missing observations are not exported as zeros.

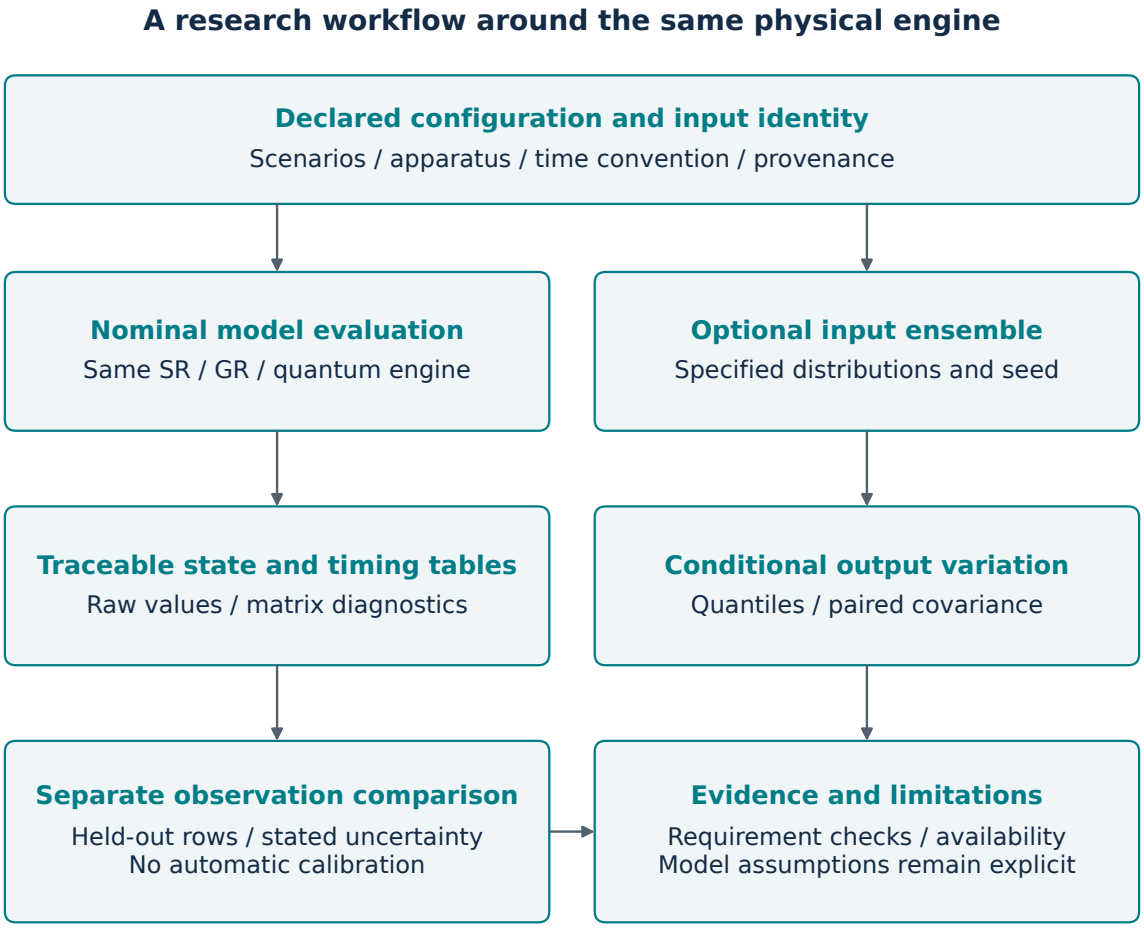

Figure 19: Research evidence built around the existing engine. Input-distribution sampling and observation comparison are separate branches. The saved numerical outputs remain tied to their configuration and model scope; none of the report branches changes the physical equations or certifies empirical accuracy.

23.2 Run the research workflow

23.2.1 Graphical operation

Open the launcher and choose the workflow *Research precision: uncertainty and evidence*. Its internal workflow identifier is `precision`. Choose a preset before editing the physical fields. The four starting configurations are `config_precision_research.json` for SR, `config_precision_gr.json` for static GR, `config_precision_qkd.json` for BBM92, and `config_precision_decoy.json` for decoy BB84. Their GUI preset identifiers are `research_sr`, `research_gr`, `research_bbm92` and `research_decoy`. They are illustrative starting points, not qualified mission designs.

The physical settings remain in the ordinary engine configuration. The new research settings are grouped under `precision`. Expand the typed editor or use the complete JSON editor for parameter arrays and correlation matrices. Select the scenarios there rather than looking for the simulation workflow's runtime scenario override. The only separate research runtime option is `no_plots`; it omits saved plot generation without changing the calculation. Inspect the effective configuration before submitting a run.

The result display keeps *Research calculation*, *Declared input uncertainty*, *Configured engineering requirements* and *Independent measurement evidence* as separate statuses. Inspect the nominal metric table, Monte Carlo interval table, requirement rows and any observation residuals. The joint output covariance display shows at most its first twenty axes; use the complete JSON artifact for the full matrix and axis definitions. Download the exported tables and metadata with the saved configuration. A completed job can still have unresolved measurement evidence or a failed configured requirement.

23.2.2 Command-line and Python operation

From the extracted package, the corresponding command is:

```
python run_precision_analysis.py --config config_precision_research.json --output results_precision
```

Add `--no-plots` when only the data products are required. Select another preset by changing the configuration filename. Configuration-relative measurement paths must remain anchored to the original configuration directory; moving an isolated JSON file can otherwise change what a relative path denotes. The output directory must be empty. Choose a new run folder rather than mixing old products with a new calculation. The research workflow supports 1–512 epochs, at most 512 ensemble draws and at most 20,000 scenario-epochs including the nominal run. Circular benchmarks are offline; SGP4 research runs require an explicit `epoch_utc` and `tracking.offline:true` with the required local ephemeris/time inputs. Numerical-provider injection and live online acquisition are outside this wrapper's supported scope.

The Python entry point is

```
from pf_precision.runner import run_precision_analysis
report = run_precision_analysis(
    config, output_dir, plots=True, progress=False, config_path=None)
```

Supply `config_path` when relative input paths need the same origin as the CLI. The wrapper resolves the engine's execution settings explicitly: the normal runner defaults for finite-count sampling and additional controls are false. Do not infer those choices from the lower-level scenario function's different Python defaults. This research runner additionally requires `execution.sample_counts:false` and, when QKD is enabled, `qkd.security.count_mode:expected`. It rejects sampled-count configurations so that the declared input ensemble does not silently mix in tomography or key-count sampling noise.

23.2.3 Research configuration reference

The following fields are additional configuration, not replacements for the ordinary `quantum_channel`, `qkd`, `storage_channel`, orbit or tracking sections.

Field below `precision`	Accepted values and behavior
`scenarios`	Nonempty list of unique integers 1, 2 and/or 3; default `[1,2,3]`. Canonical ordering preserves the shared S2/S3 realization.
`ensemble.enabled`	Boolean, default false. Requires a nonempty parameter list when true.
`ensemble.samples`	Integer from 2 to 512; default 32. This is an input-draw count, not the number of photons or scenario epochs.
`ensemble.seed`	Integer in $[0, 2^{32})$, default 20260918; controls the input draws.
`ensemble.parameters`	At most 32 distinct supported explicit parameter paths. Each entry supplies distribution, unit, source and uncertainty type.
`ensemble.correlation`	Optional symmetric positive-semidefinite matrix with unit diagonal, ordered exactly like `parameters`; supported only when all input distributions are normal.
`ensemble.invalid_draw_policy`	Only `fail`. No clipping or replacement sampling is performed.
`requirements`	Up to 256 nominal metric checks. Each entry declares scenario, metric, operator, threshold, unit and nominal basis.
`validation`	Optional observation-comparison settings. Requires file, source and explicit synthetic flag. It does not fit parameters.
`timing_budget`	Optional first-order frozen-endpoint SR uncertainty diagnostic; its covariance, axes and scope are defined in Section 23.6.

Each `ensemble.parameters` entry contains `path`, `distribution`, `unit`, `source` and `uncertainty_type`. The last field is either `epistemic` or `aleatory`; this is a declared role and does not establish provenance. A normal distribution requires positive `std`, has its mean at the explicit nominal configuration value and does not accept `low`/`high`. A uniform distribution requires increasing `low`/`high` containing the nominal value and does not accept `std`. Normal draws are unbounded; a draw outside the physical domain fails rather than being silently truncated. Use scientifically justified bounded uniform inputs where that model is appropriate.

Every sampled path must already be explicitly present in the configuration. Defaults hidden inside the physical engine are not silently turned into uncertain inputs. Supported path families and exact units are:

Prefix	Supported leaf names and units
`quantum_channel`	`phase_sigma_a_rad`, `phase_sigma_b_rad`, `deterministic_relative_phase_rad`: rad; `phase_correlation`, `source_depolarizing`, `efficiency_a`, `efficiency_b`: 1.
`quantum_channel`	`bandwidth_hz`, `source_pair_rate_hz`: Hz; `detector_jitter_s`, `gate_half_width_s`, `gate_center_s`, `gate_timing_error_s`: s.
`storage_channel.arms.a` or `b`	`t_phi_s`, `lifetime_s`: s; `retrieval_efficiency`: 1. The storage channel must be enabled and the sampled value must be finite and explicit.
`velocities.emitter`, `detector1` or `detector2`	`altitude`: m; `inclination`, `raan`, `arg_perigee`, `mean_anomaly_at_epoch_rad`: rad. These profile perturbations apply only to circular benchmark orbits.
`qkd.optical_links.a` or `b`	`pointing_jitter_rad`: rad; `optical_efficiency`, `detector_efficiency`, `zenith_optical_depth`: 1; `dark_count_rate_hz`, `background_rate_hz`: Hz; `receiver_aperture_radius_m`, `waist_m`: m; `measured_loss_db`: dB. QKD and the selected parameter's physical mode must be active.

Despite the legacy section name `velocities`, those circular-profile fields are orbital descriptors, not components of an arbitrary Cartesian state covariance. Orbit covariance and security epsilons are not interchangeable with the supported paths. An unsupported or duplicated path is rejected. Measured-loss mode does not use the Gaussian beam/pointing/throughput fields; `measured_loss_db` requires that mode. Detector efficiency cannot be counted separately when the supplied measured loss already includes it. Atmospheric optical-depth uncertainty requires a ground endpoint, and a decoy study may perturb only its selected optical arm.

For example, the following is a research-section fragment to add to a physical configuration that explicitly defines the two nominal phase sigmas and their physical phase correlation:

```
"precision": {
  "scenarios": [1, 2, 3],
  "ensemble": {
    "enabled": true, "samples": 64, "seed": 20260918,
    "invalid_draw_policy": "fail",
    "parameters": [{
      "path": "quantum_channel.phase_sigma_a_rad",
      "distribution": "uniform", "low": 0.12, "high": 0.18,
      "unit": "rad", "source": "Illustrative assumed interval",
      "uncertainty_type": "epistemic"
    }]
  },
  "requirements": [], "validation": {}
}
```

The interval above is an example assumption, not a measured specification. Keep the source description explicit when replacing it with a defensible apparatus estimate.

23.3 Exported products and aggregation rules

`precision_products.json` contains the structured tables and preserved source records. `precision_metadata.json` records table columns, units, encodings and conventions. The CSV files use UTF-8, comma-separated fields, JSON strings for nested matrices or source records, and the literal `\N` for unavailable scalar cells. A blank spreadsheet cell, zero and `\N` should not be treated as equivalent. Parse JSON-valued cells before using their arrays; do not split them by internal commas.

File	Row identity	Content and aggregation warning
`timing.csv`	Scenario, epoch index, optical arm	Release/reception events, coordinate conventions, clock rates/available proper intervals, gate fields, visibility, PF status and equation residuals. Pair-level fields repeat on both arm rows.
`polarization.csv`	Scenario and epoch index	Conditional density matrix, physical phase covariance, matrix diagnostics, Pauli state correlations and preserved quantum/storage source records. One row is not a count sample.
`key_blocks.csv`	Scenario aggregate block	One protocol-security budget over the configured exposures, including deductions and abort reason. Scenarios are alternatives; their keys are not additive.
`key_epochs.csv`	Scenario, epoch index, optical arm	Per-arm optical information and epoch statistics linked to the aggregate block. Repeated pair counts must not be summed across arm rows.

The structured source records retain information not flattened into a scalar column. For example, finite-count tomography remains under the quantum source record and detected BBM92 matrices remain with their QKD epoch record. The principal Pauli diagnostics apply to the conditional channel matrix. Physicality uses a declared 10^{-9} tolerance for trace/Hermiticity/eigenvalue checks; the Hermiticity error is a Frobenius norm. Invalid source matrices are retained and identified rather than repaired by the exporter.

An epoch index is zero-based. `snapshot_time_s` refers to the scenario snapshot, while each link's emission and reception times are local offsets from that snapshot. Preserve both pieces when building a timeline. In static GR, legacy source names containing `eci` refer to the base Schwarzschild chart where documented; the normalized table labels the base arrival gap accordingly.

23.3.1 Nominal metrics and requirements

Research metrics summarize the selected scenario; they are not arbitrary per-epoch observation fields. The exact metric names and units are:

Metric	Unit	Scope
`fidelity_mean`, `purity_mean`	1	Arithmetic epoch mean of the conditional state quantity.
`accepted_pair_rate_mean_hz`	Hz	Arithmetic epoch mean, not an exposure-weighted pass average.
`arrival_gap_abs_max_s`, `gate_residual_abs_max_s`	s	Maximum absolute available gap/residual.
`max_null_residual_m`	m	SR numerical null-distance residual.
`max_interception_residual_s`	s	GR numerical interception residual.
`max_ray_null_relative_residual`	1	GR normalized ray-null residual.
`secret_key_bits`	bit	One whole-run security bound.
`secret_key_rate_hz`	bit/s	Bound divided by configured exposure duration.
`qber_z_mean`, `qber_x_mean`	1	Arithmetic epoch mean of the reported expected-mode QKD error fraction.
`detected_pair_rate_mean_hz`	Hz	Arithmetic epoch mean of the BBM92 true-plus-accidental pair rate; unavailable for decoy BB84.
`unresolved_epoch_count`	count	Epochs without a resolved reception schedule or available arrival gap.

An unavailable branch-specific metric remains unresolved. A metric with some missing epochs is marked partial and carries the available/total epoch counts. Requirements compare only fully available nominal metrics. Their operators `min` and `max` respectively mean value at least, or at most, the declared threshold. The only supported requirement basis is `nominal`; the current checker does not apply a percentile criterion.

```
{"scenario": 2, "metric": "fidelity_mean", "operator": "min",
 "threshold": 0.9, "unit": "1", "basis": "nominal"}
```

This example checks an illustrative engineering target. Passing it does not establish observation agreement. Report a failed or unresolved check rather than editing the threshold after inspecting the result.

23.4 Four meanings that must remain separate

Quantity	What it describes	What it does not establish
Numerical residual	Agreement with a solved equation, independent expression, matrix property or coordinate inverse.	Accuracy of an ephemeris, clock or measured apparatus parameter.
Physical phase covariance	The assumed Gaussian distribution of polarization phases acting on the photons.	Uncertainty in estimates of the covariance parameters themselves.
Input-propagated variation	Spread of outputs after drawing declared model inputs from specified distributions.	Coverage of future observations when those distributions or the model discrepancy are unvalidated.
Observation residual	Difference between a compatible measured quantity and its matched prediction.	Identification of every apparatus parameter, an independent test after tuning to the same rows, or an operational security certification.

An example already present in the package has a Decimal inverse-time residual of order 10^{-51} s, while the float-based photon null-distance residual is of order 10^{-8} m. These test different computations. Neither states that a satellite clock or the physical photon path is known to that uncertainty. The scenario serializer stores ordinary floating-point numbers, including matrix entries; it does not preserve a 50-digit physical state simply because the Lorentz algebra uses a wider Decimal context.

The proper-clock integral describes elapsed proper time on a supplied worldline from a stipulated coordinate epoch. It does not infer clock offset, oscillator noise, drift calibration or synchronization with UTC. The current clock helper returns the integral and does not export its quadrature error. Do not substitute the photon root tolerance or an inverse-transform residual for that missing quantity.

23.5 Density-matrix and Pauli conventions

23.5.1 Preserve the state and its interpretation

The two-photon basis is $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$ in B, A bit order. Thus the first tensor factor is photon B, even though a measurement-setting label such as `XZ` means X on A and Z on B. Every matrix export must carry that convention. Its real and imaginary elements belong to the same complex matrix,

$$\rho_{ij} = R_{ij} + \mathrm{i} I_{ij}, \qquad i, j = 0, 1, 2, 3. \tag{23.1}$$

Rows and columns are indices in the stated basis; they are not arm labels. The extension changes the table representation, not the matrix normalization. No trace renormalization, eigenvalue clipping or positive-semidefinite projection is performed merely to make a diagnostic pass.

The conditional channel matrix describes the modeled polarization state when both photons are collected. Scalar loss and timing acceptance are handled outside that state. The detected BBM92 matrix additionally includes the modeled analyzer rotations and accidental-coincidence pedestal. A linear-inversion tomography matrix is an estimator from the simulated measurement counts; its minimum eigenvalue may be negative. Its unphysical fluctuation is evidence about the estimator and count budget, not a reason to silently alter the saved state. Decoy BB84 uses its own phase-randomized pulsed source; it does not turn the entangled-pair matrix into the BB84 source state.

23.5.2 Matrix diagnostics

For an available matrix, inspect its Hermiticity residual, trace, spectrum, purity and target fidelity together. The standard relations are

$$\rho = \rho^{\dagger}, \qquad \operatorname{Tr}\rho = 1, \qquad \rho \succeq 0, \tag{23.2}$$

$$P = \operatorname{Tr}(\rho^2), \qquad F_{\psi} = \langle\psi|\rho|\psi\rangle. \tag{23.3}$$

Numerical tolerances in these tests classify representational consistency. They are not confidence levels. A target-state fidelity and a state purity measure different properties: a deterministic relative phase can change the first without reducing the second.

Let $\sigma_0 = I$, and let $\sigma_x, \sigma_y, \sigma_z$ be the Pauli matrices. In the declared B, A order define

$$P_{\mu\nu} = \sigma_\mu^{(B)} \otimes \sigma_\nu^{(A)}, \qquad r_{\mu\nu} = \mathrm{Tr}(\rho P_{\mu\nu}). \tag{23.4}$$

The exported real/imaginary correlation matrix R uses row order I,X,Y,Z for B and column order I,X,Y,Z for A; its entry $R_{\mu\nu}$ is $r_{\mu\nu}$ above. It is a state correlation matrix, not a Pauli transfer matrix, quantum process matrix or Choi matrix. For a normalized Hermitian two-qubit state,

$$\rho = \frac{1}{4} \sum_{\mu,\nu \in \{0,x,y,z\}} r_{\mu\nu} P_{\mu\nu}, \qquad r_{00} = 1. \tag{23.5}$$

The coefficients are dimensionless expectation values. The identity coefficient is fixed by normalization and is distinct from the fifteen nonidentity observables.

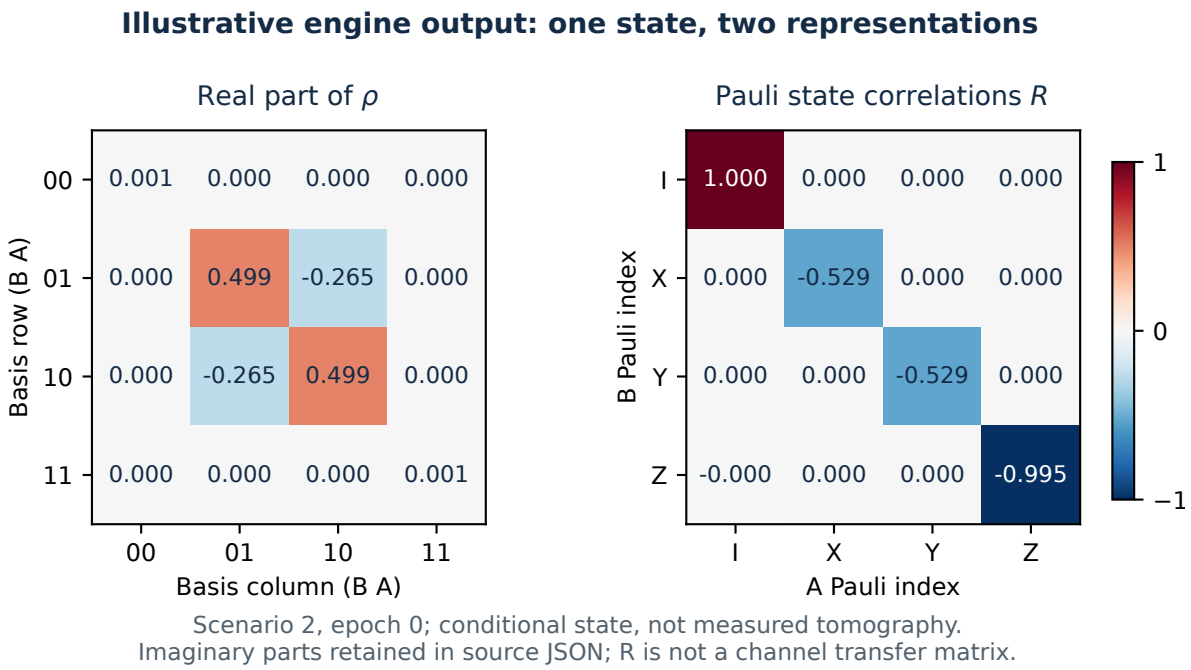

Figure 20: The real part of an actual illustrative engine density matrix and its Pauli state-correlation representation. Both refer to the same conditional polarization state; imaginary parts are retained in the frozen figure data. The model inputs are assumptions and the figure is not measured tomography or a process characterization.

A quantum observable covariance may be defined from the symmetrized product,

$$\Gamma_{ij} = \frac{1}{2} \mathrm{Tr}\left[\rho(P_i P_j + P_j P_i)\right] - r_i r_j. \tag{23.6}$$

For a physical state and Hermitian Pauli observables this covariance is real and positive semidefinite up to numerical roundoff. It describes quantum fluctuations. Incompatible observables cannot all be measured jointly on a single pair. Therefore Γ is not automatically the covariance of an experiment's tomography estimator, a fitted calibration vector or the Monte Carlo draws. Those quantities require their own sampling and measurement definitions. Equation 23.6 explains this distinction; the polarization exporter supplies R, not this observable covariance Γ.

23.5.3 Why matrix ordering matters

When importing a table into another numerical package, reconstruct the complete complex matrix before calculating a spectrum or partial trace. Preserve index origin, basis order and the real/imaginary pairing. Do

not infer that a CSV sorted alphabetically by a label is already in tensor-product order. A swap between A, B and B, A must be applied consistently to the matrix, Pauli labels, measurement conventions and target vector.

23.6 A declared endpoint and clock uncertainty budget

The optional `precision.timing_budget` computes a narrowly defined first-order uncertainty diagnostic for resolved flat-SR links. It holds the actual emission and detection endpoints fixed when differentiating the observable. It is not the Jacobian of the full moving-receiver scheduling solution and is not an orbital-state covariance propagator.

Let the clock tag convention be $t_{\text{tag}} = t_{\text{source}} + b$, with clock offset b in seconds. For emitter and receiver endpoints $\mathbf{r}_E, \mathbf{r}_R$ in the same source inertial axes, define

$$z = \frac{\|\mathbf{r}_R - \mathbf{r}_E\|}{c} + b_R - b_E, \tag{23.7}$$

$$\mathbf{n} = \frac{\mathbf{r}_R - \mathbf{r}_E}{\|\mathbf{r}_R - \mathbf{r}_E\|}, \tag{23.8}$$

$$H = \begin{bmatrix} -\mathbf{n}^{\mathsf{T}}/c & \mathbf{n}^{\mathsf{T}}/c & -1 & +1 \end{bmatrix}, \tag{23.9}$$

$$u_z^2 = H \Sigma H^{\mathsf{T}} + u_J^2. \tag{23.10}$$

Covariance entries must be finite numeric values. Boolean values and numeric strings are rejected before unit, symmetry and positive-semidefinite checks.

The input covariance Σ has eight axes in this exact order:

$$(r_{E,x}, r_{E,y}, r_{E,z}, r_{R,x}, r_{R,y}, r_{R,z}, b_E, b_R). \tag{23.11}$$

The first six axes have unit m and the last two unit s. Each covariance entry has the product of its row and column units: position–clock entries therefore have unit m s. The budget may contain signed covariance contributions. Do not replace a negative cross contribution by zero; only the total propagated variance must be nonnegative within numerical tolerance.

The settings are:

Field below `timing_budget`	Contract
`enabled`	Boolean, default false.
`model`	When enabled, exactly `frozen_endpoint_sr`.
`parameter_source`	Required nonempty provenance/assumption description; the workflow does not verify a calibration claim in this text.
`input_covariance`	Required finite symmetric positive-semidefinite 8×8 matrix in the axes above. A zero-variance axis requires its entire covariance row and column to be zero.
`independent_jitter_s`	Nonnegative standard uncertainty of the complete observable, default zero. Its square is added exactly once.
`assume_independent_jitter`	Boolean, default false. Must be true when a nonzero independent jitter term is supplied.

The matrix validity check uses normalized correlations so that metre-scale entries cannot conceal an invalid clock-offset covariance at much smaller numerical scales. The output retains the covariance, axis units, link Jacobian, coordinate convention, signed endpoint-position, clock-offset, position–clock cross and independent-jitter variance contributions, total variance and `standard_uncertainty_s`. Read `timing_budget.json` and the timing-budget section of the research report.

Static-GR links, missing or failed links and coincident endpoints are explicitly unavailable for this diagnostic. An unavailable uncertainty is not zero. The same supplied covariance is applied to each eligible link; this does not specify cross-link or cross-epoch covariance. The budget does not infer clock-offset means, propagate orbital dynamics, model stochastic oscillator behavior or validate the stated uncertainty against measurements. Avoid

double-counting detector jitter already included in the covariance or in another uncertainty term. Numerical solver residuals are not components of Σ unless a separate, justified numerical-error model has been supplied outside this diagnostic.

23.7 Conditional Monte Carlo uncertainty

The ensemble workflow repeatedly evaluates the existing scenario engine after drawing selected input parameters from declared distributions. If $\theta^{(k)}$ is draw k, the result is

$$y^{(k)} = f\left(\theta^{(k)}; \mathcal{M}, \mathcal{E}\right), \qquad k = 1, \dots, N, \tag{23.12}$$

where $\mathcal{M}$ is the fixed model and $\mathcal{E}$ denotes the retained ephemeris and execution conventions. This construction propagates only the uncertainty represented in the supplied draws. It does not discover omitted physics or make a parameter distribution empirical by assigning it a seed.

For complete finite vector outputs, the sample covariance uses the usual finite-sample denominator,

$$\overline{y} = \frac{1}{N}\sum_{k=1}^{N} y^{(k)}, \qquad S = \frac{1}{N-1}\sum_{k=1}^{N}(y^{(k)} - \overline{y})(y^{(k)} - \overline{y})^{\mathsf{T}}. \tag{23.13}$$

At least two available draws are required for a sample variance. The covariance of simulated outputs is distinct from Equation 23.6. It also differs from a fitted-parameter covariance and from the phase covariance already supplied to the polarization channel.

The implemented covariance includes only metric columns that are fully available in every completed paired draw. Other columns are listed under `dropped_axes`; they are not filled with zero and no pairwise deletion is used. This retains a consistent sample set for the whole matrix. Entry (i, j) has unit $\mathrm{unit}_i \mathrm{unit}_j$, so raw entries across different metric types cannot be compared just by their magnitude.

An empirical interval bounded by the $\alpha/2$ and $1-\alpha/2$ sample quantiles is a central interval of that simulated ensemble. It is not a confidence interval for an unknown mean, nor a guaranteed prediction interval for future hardware. Quantiles themselves have Monte Carlo error. Small sample counts are useful for testing the workflow and are inadequate for reliable tail probabilities. For strongly nonlinear or bounded outputs, a mean and standard deviation alone can conceal asymmetry, clipping at a physical boundary or a mixture of aborted and non-aborted protocol blocks.

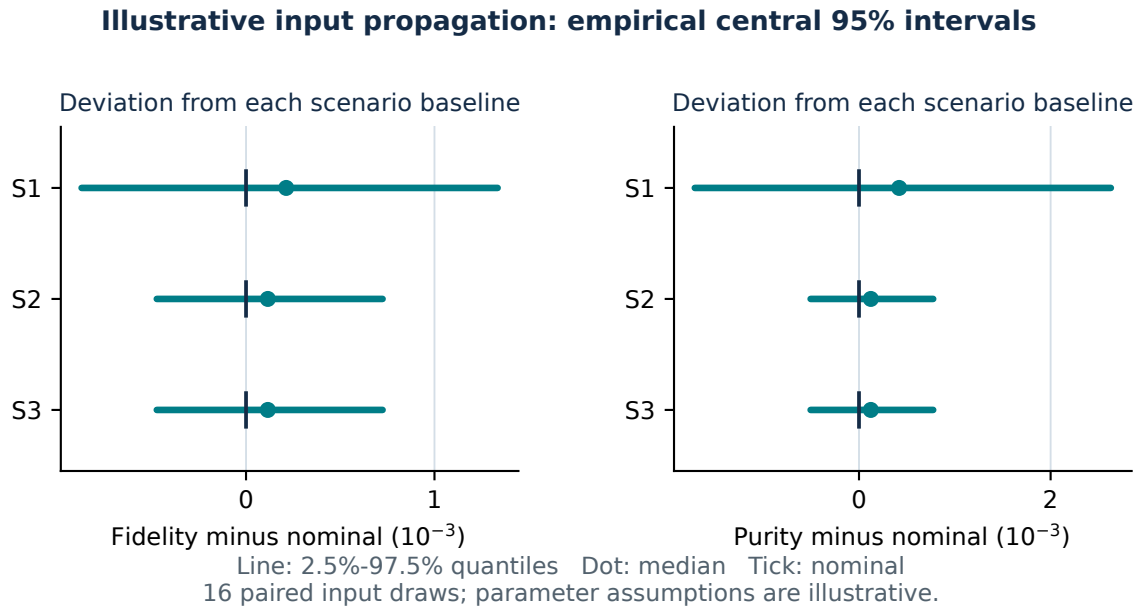

Figure 21: Empirical 2.5th, 50th and 97.5th percentiles from an executed illustrative input ensemble, plotted as deviations from each scenario's nominal value in units of 10^{-3}. The marker at zero is the nominal configuration. Equal S2/S3 results preserve their shared physical channel. These intervals propagate the declared assumptions only; their small demonstration sample count does not establish tail convergence or empirical prediction coverage.

23.7.1 Unavailable outputs and QKD aborts

A valid zero-key bound and an unavailable key forecast are different outcomes. Report successful and available sample counts beside every summary. This implementation suppresses a metric's mean, standard deviation and quantiles when that metric is not fully available in every requested draw. Partial counts and status remain visible. A failed or incomplete ensemble suppresses all success-only aggregate intervals and covariance. The failed draw remains in the run record, and execution fails rather than clipping, replacing or resampling it. The nominal outputs already written remain inspectable but do not constitute a completed ensemble report.

QKD key lengths are discrete and depend on a block's full count and security budget. Propagate the configured model to the block output. Do not sum independent per-epoch key bounds when the engine's defined calculation aggregates statistics first. Keep the authentication cost, leakage convention, phase-error bound and security epsilon budget explicit. An ensemble interval on a simulated key rate does not replace any protocol-security bound.

23.7.2 Separate parameter uncertainty from measurement sampling

The existing engine uses shared seeds across scenarios to support paired comparisons. Sampling new physical parameters and sampling new detection counts are different operations. This research wrapper disables finite-count sampling and requires expected-mode QKD. Its ensemble describes variation of declared input parameters. The ordinary simulation workflow remains available for finite-count experiments, with its own shot budgets and estimator conventions. A combined parameter-and-measurement study would need a separate declared sampling policy. Changing the seed cannot make an uncalibrated apparatus parameter measured.

Do not estimate uncertainty by pooling different scenario epochs indiscriminately. Their changing geometry can dominate the spread even when every input is known exactly within the model. Compare corresponding epochs or define a scientifically meaningful run-level statistic before calculating an ensemble distribution.

23.8 Observation comparison and calibration discipline

An observation file should identify the quantity measured, its units, the measurement and time convention, the corresponding scenario or configuration, its split role and any stated uncertainty. A residual is meaningful only after those semantics agree. In particular, conditional polarization fidelity is not the same observable as fidelity inferred from detected coincidences, coordinate time is not a clock offset, and an expected optical count rate is not a realized integer count.

For a prediction $\hat{y}_i$ and observation y_i, let

$$d_i = \hat{y}_i - y_i, \qquad \text{bias} = \frac{1}{n}\sum_i d_i, \qquad \text{RMSE} = \sqrt{\frac{1}{n}\sum_i d_i^2}. \tag{23.14}$$

The mean absolute error is $n^{-1}\sum_i |d_i|$. These descriptive statistics do not require a Gaussian model. If a supplied standard uncertainty u_i is appropriate to the comparison, a standardized residual d_i/u_i provides a scale comparison. It is not automatically a normal-test statistic: systematic uncertainty, model uncertainty, shared calibration errors and correlated rows may all be absent from that denominator.

23.8.1 Exact CSV and comparison configuration

The current CSV interface compares *whole-scenario aggregate metrics* from the table in Section 23.3; it does not join individual timestamped observations to epochs. Supply exactly these six header names:

```
scenario,metric,value,standard_uncertainty,unit,split
2,fidelity_mean,0.94,0.02,1,validation
2,purity_mean,0.90,0.03,1,training
```

These rows are synthetic examples, not observations of the supplied apparatus. The first row is eligible for a comparison; the second is retained but excluded from held-out validation. The accepted split values are `validation`, `calibration` and `training`. The validator does not fit the latter two kinds of rows. The selected scenario must be present in the run, and metric names and units must exactly match the supported metric schema.

Use a corresponding configuration fragment such as:

```
"validation": {
  "measurements_file": "observations/example.csv",
  "source": "Synthetic demonstration; not measured data",
  "synthetic": true,
  "sigma_limit": 3.0,
  "split": "validation"
}
```

This fragment belongs under `precision`. The `synthetic` flag must be explicitly true or false. The configured comparison split is always `validation`. An omitted validation section, an empty object or `measurements_file:null` leaves measurement comparison unconfigured. The file is limited to 8 MiB and 10,000 data rows; malformed rows, nonfinite numbers, unknown columns, wrong units and negative uncertainty are rejected. Empty or zero standard uncertainty leaves a row unresolved; the workflow does not invent an error bar. A nonfinite or partial prediction also cannot produce a consistent comparison.

For an eligible row with positive supplied uncertainty, the comparison criterion is $|d_i/u_i| \le k$, where k is `sigma_limit`. That parameter defaults to 3 and must be positive. It is a predeclared comparison threshold, not a parameter to tune after seeing residuals. The current implementation does not add the Monte Carlo predictive variance to u_i^2, model residual covariance, adjust for multiple comparisons or calculate a scientific qualification probability.

The report retains each row and its exclusion/availability reason, the resolved file path and SHA-256 identity, source label and threshold. With synthetic data, the overall evidence status is `synthetic_only`, even when individual residuals satisfy the criterion. A source label and `synthetic:false` do not independently verify provenance: `source_verified` and `empirical_validation_established` remain false. Aggregate agreement is a useful diagnostic, not a completed empirical validation claim.

23.8.2 Split before inspecting the held-out residuals

Declare training or calibration rows separately from held-out validation rows. Fit only on the permitted calibration data, freeze the resulting rule, then evaluate the held-out data once. Repeated tuning after reading held-out residuals turns that set into development data. Reserve another independent set before making a new validation claim.

Use acquisition groups, passes or time blocks when adjacent samples share noise, overlapping count windows or a calibration. Randomly splitting individual rows does not remove that dependence. The existing BBM92 and Jinan benchmark chapters explain their own source-specific splits and limitations. A generic CSV comparison does not silently inherit those benchmarks' empirical status.

The research workflow is not an unrestricted optimizer for all channel and orbital parameters. A single Bell-state fidelity generally cannot identify both phase marginal variances, phase correlation, source depolarization and a deterministic phase. A rate alone cannot identify source brightness separately from both optical efficiencies. Such degeneracies require additional independent observables or justified constraints. Record them as unresolved rather than reporting an arbitrary fitted vector as a unique apparatus calibration.

23.9 Numerical checks and qualification beyond this release

The retained saved-result validators test numerical consistency under the declared SR/static-GR, orbit and channel assumptions. Their PASS status is valuable implementation evidence. It does not establish flight accuracy, traceability to a timing standard, empirical input distributions or certification of a cryptographic service. A failed validation should lead to inspection of the equation, units, availability state and input identity; relaxing a tolerance to make a report pass changes the test.

A defensible accuracy statement requires the following additional evidence:

1. An operational definition of each measurand, its coordinate/time/basis convention, intended use and accepted domain.
2. Independent, timestamped observations with instrument provenance, calibration history, stated uncertainty and acquisition grouping.
3. Input uncertainties and dependencies justified from those observations, including orbit state, clock behavior, pointing, background and apparatus parameters that materially affect the requested output.
4. A model-discrepancy assessment for omitted effects. Static spherical GR, approximate ephemerides, idealized memory loading and sparse-count assumptions must remain within an evidenced domain.
5. Numerical convergence studies appropriate to the quantity of interest, separate from physical uncertainty, and tests against independent formulations.
6. Prespecified held-out validation and interval-coverage analysis, including failure and availability rates, repeated across the required operating regimes.

Correlated orbital-state propagation, stochastic oscillator models, full apparatus parameter inference and independent operational qualification are future extensions requiring additional data and validation. They are not enabled merely by exporting a covariance-shaped table. The present workflow makes such work easier to inspect and reproduce while retaining the existing physics and its explicit limits.

23.10 Figure regeneration and source guidance

The three native-width vector figures in this chapter are regenerated with

```
python documentation/latex/generate_precision_figures.py
```

The workflow diagram is conceptual. The matrix and interval figures read the frozen actual-output extract `documentation/latex/figure_data/precision_worked_results.json`. Its source-file hashes bind the numbers to an executed nominal SR study with three epochs, all three scenarios, configured memory dephasing and sixteen declared input draws. The phase-input distributions and memory parameters are illustrative. No hardware observations are used in these figures. Rounding matrix labels to three decimals is a plotting choice; the JSON retains the underlying numerical values.

To capture a newly executed study deliberately, use

```
python documentation/latex/generate_precision_figures.py --capture validation_precision/nominal_sr
```

This replaces the frozen figure extract and its provenance with that run's report/products. Review the resulting scope, values and captions before rebuilding the manual. Ordinary figure regeneration does not execute a new simulation, retrieve an ephemeris or invent observational data. The supplied PDF figures are sufficient to compile the manual without running Python.

The following primary references guide future uncertainty and qualification work. Their inclusion documents relevant methods and distinctions; it does not claim that this package conforms to a standard or has passed a certification.

- JCGM 100:2008, Guide to the expression of uncertainty in measurement: measurement models, input covariance and uncertainty contributions.
- JCGM 102:2011, Extension to any number of output quantities: joint output uncertainty, beyond isolated scalar error bars.
- NASA-STD-7009B, Standard for Models and Simulations (2024): declared model use, acceptance criteria, assumptions, uncertainty and evidence of credibility. Project applicability must be assessed separately.
- ETSI GS QKD 011 V1.1.1, Component characterization (2016): apparatus measurement procedures and uncertainty/calibration considerations.
- ETSI GS QKD 016 V2.1.1, Protection profile for a pair of prepare-and-measure QKD modules (2024): the distinction between a security proof's assumptions, their implementation and a certified system. A performance forecast is not that certification.

Chapter 24

Research risk auditing and validation evidence

The assurance extension connects the research calculations in Chapter 23 to a declared engineering decision. It records the uncertainty sources considered, checks predeclared requirements under the specified input distribution, compares an observation vector with an explicitly stated covariance model, and assembles a hash-checked evidence bundle. Its appropriate description is a *research risk-auditing and validation workbench*. The evidence remains scoped to its inputs, model, test conditions and operating domain.

The workflow does not establish absolute verification, complete mission qualification, deployed QKD security or an authenticated metrological traceability chain. It cannot create measurements, calibration certificates, independent reviewers or a customer's acceptance decision. Even a fully documented case reaches `review_required`; an incomplete case remains `blocked`. A completed calculation and a blocked readiness assessment are compatible outcomes: the calculation may have correctly identified the missing evidence.

24.1 What is being assessed?

Keep six questions separate when reading a report:

1. Did the calculation execute and retain its inputs and outputs?
2. Do the numerical results satisfy the implemented equations and checks?
3. Do the declared uncertainty draws support the configured requirement?
4. Are the predictions compatible with the supplied held-out observations under a justified measurement and prediction covariance?
5. Have the material error sources, model limitations and residual risks been reviewed for the intended decision?
6. Can a recipient verify that the assessed bytes match the preserved evidence and an independently retained digest?

These questions cannot be collapsed into one count of passing tests. A small null-geodesic residual concerns the numerical solution. It does not measure the error of an atmospheric model, instrument calibration, clock realization or hardware prediction. A source hash identifies software content; it does not prove the source implements the right physical model.

The extension preserves all three scenarios and their scheduling relationships. It does not introduce a new PF emission law or infer polarization decoherence from coordinate labels. Scenario 3 remains a full-PF diagnostic on the physical reception events of Scenario 2. The static-GR approximation and frozen-endpoint SR covariance diagnostic retain their existing scope. The audit can expose a missing moving-receiver or GR error contribution; recording that gap does not implement the missing propagation model.

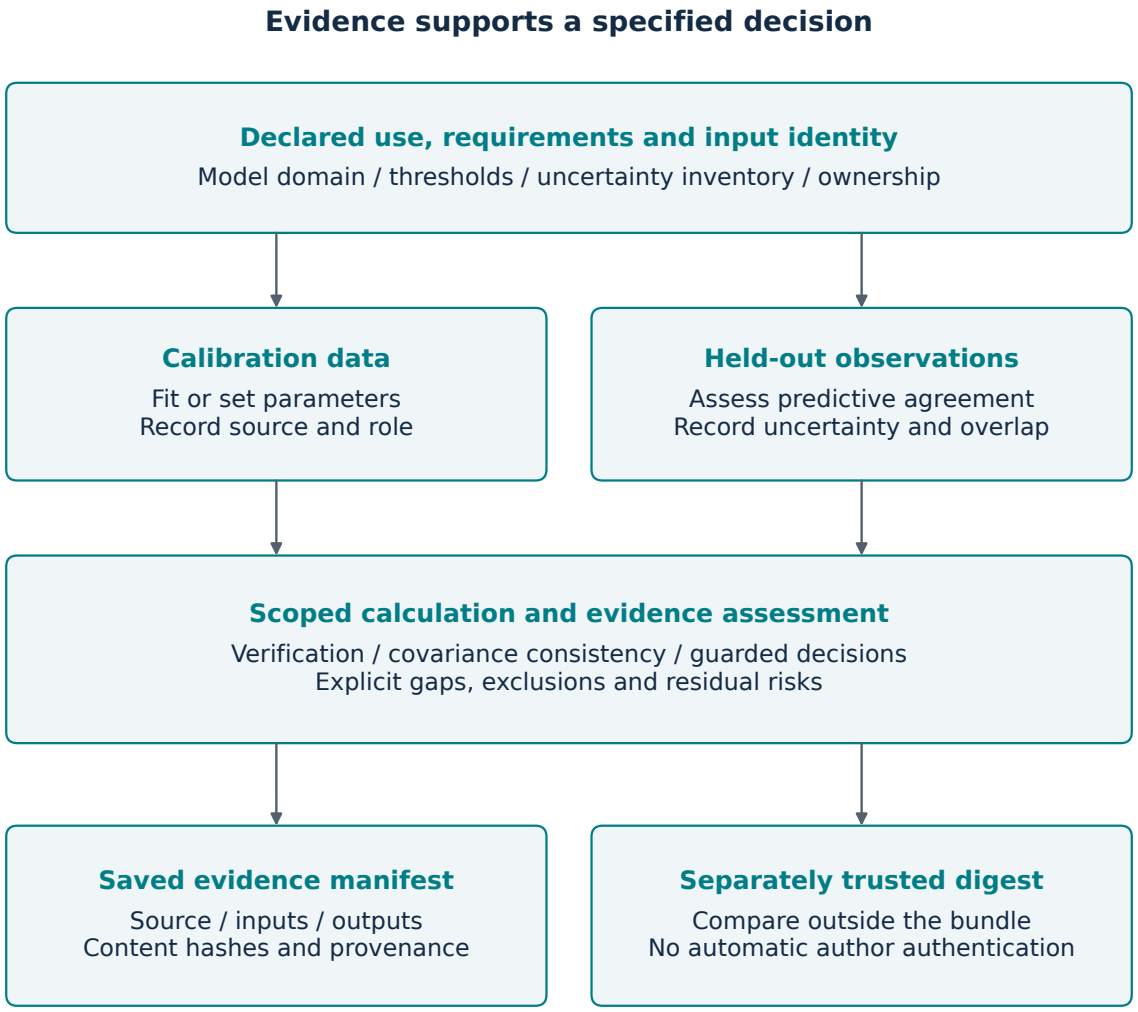

Figure 22: Conceptual evidence chain for a declared research decision. Calibration and held-out validation have different roles. A digest retained outside the bundle supports a stronger byte-identity check than a co-located manifest; it does not authenticate measurements or certify the physical model.

24.2 Run and inspect an audit

Open the existing graphical launcher and select *Risk audit: evidence and uncertainty*, whose workflow identifier is `assurance`. The supplied SR, static-GR and QKD starting configurations are respectively `config_risk_audit_sr.json`, `config_risk_audit_gr.json` and `config_risk_audit_qkd.json`. They demonstrate the workflow with explicit gaps; they do not contain independently qualified mission data. Review the ordinary physical fields, the `precision` ensemble and the `assurance` decision records before running. Use the complete JSON editor for matrix-valued covariance and record arrays.

The command-line equivalent is:

```
python run_risk_audit.py --config config_risk_audit_sr.json --output results_risk_audit_001
```

The configuration is required; output defaults to `results_risk_audit`. `--no-plots` omits the underlying precision plots. Always choose an empty output directory. The runner rejects a nonempty destination so a prior evidence bundle is not mixed with a new run. Relative input paths resolve against the configuration file's directory. The Python API is:

```
from pf_assurance.runner import run_risk_audit
report = run_risk_audit(
    config, output_dir, config_path=None, plots=True, progress=False)
```

For direct Python use, supply `config_path` to preserve the same path origin; otherwise the current working directory is used. No live acquisition is performed by this wrapper. SGP4 inputs require offline tracking, and the underlying research runner's explicit epoch, complete ensemble and expected- count constraints remain applicable.

Output	Inspection purpose
`assurance_report.json`	Execution status, intended use, `readiness.status`, blockers, source coverage, risks, uncertainty requirement results, vector comparison and next actions.
`submitted_config.json`	Submitted decision and simulation settings. The rewritten scientific configuration used for frozen inputs is retained by the precision workflow.
`precision/`	Underlying nominal scenario outputs, research tables, ensemble, effective configuration, diagnostics and optional plots.
`source_snapshot/` and `source_snapshot_manifest.json`	Scientific source snapshot and content identity. It includes root-level non-test Python files and Python/JSON files under `pf_precision`, `pf_assurance` and `vendor`.
`inputs/` and `input_snapshot_manifest.json`	Retained evidence, optional scalar/vector measurements, configured offline tracking cache, Earth-orientation and leap-second inputs, plus packaged timing fallback files. The manifest identifies origins, hashes and detected role conflicts.
`EVIDENCE_MANIFEST.json` and `EVIDENCE_MANIFEST.sha256`	Final content inventory and detached digest for independent byte checks.

The source snapshot is not a hermetic runtime image. Installed dependency versions are recorded by the research report, but the complete operating system, compiled libraries and installation environment are not bundled. Reproduction on another machine still requires an appropriate environment and comparison with the retained numerical evidence. Source and input copies are checked for changes during the run before sealing.

Inspect execution separately from readiness. Top-level `status:complete` means the calculation and evidence construction completed; it does not override `readiness.status:blocked`. Failure during execution produces a failed report and blocked readiness where possible. The runner does not rewrite an already sealed bundle to manufacture a success record.

An exact content-hash match between calibration and validation evidence is reported as a role conflict. The vector input must correspond to the retained validation evidence used to support the case. These checks detect

simple reuse or mismatched artifacts; differently serialized or overlapping datasets can still share observations and require a scientific independence review.

24.3 Configuration and intended use

The ordinary physical configuration and the `precision` section remain the source of the simulation and ensemble settings. Add a root-level `assurance` object for the decision and evidence records. The assurance schema rejects unknown fields rather than ignoring misspelled controls. Stable record identifiers use letters, digits, underscores, full stops, colons or hyphens, start with a letter or digit and are at most 128 characters long.

Field below `assurance`	Meaning
`intended_use`	Required object containing nonempty `id`, `description`, `decision`, `domain` and `owner`. The domain is an exact declared label, not an automatically inferred region of physical validity.
`uncertainty_sources`	Inventory of error sources and how each is treated. Missing categories are added as unresolved review prompts.
`uncertainty_requirements`	Predeclared ensemble requirement records; statistical fields and status rules are specified below.
`evidence`	Files with source, type, declared domain, synthetic status, independence and optional expected hash or expiry.
`vector_validation_file`	Optional path to the vector dataset described in Section 24.5. Null means no vector dataset was supplied.
`risks`	Consequences, severity, ownership, mitigation and linked evidence for the intended decision.
`review`	Named review, declared independence, disposition and supporting evidence. An omitted review defaults to pending.

Use a domain label specific enough to identify the applicable campaign or hardware configuration, and describe its physical bounds in the accompanying evidence. For example, a particular source, detector set, contact geometry and temperature range form a much narrower claim than "satellite QKD." Matching two domain strings only checks the supplied declarations; it does not establish that two experiments actually occupy the same validation domain.

24.3.1 Error-source inventory and coverage

Every uncertainty-source record contains `id`, `name`, `category`, `treatment`, `rationale`, `evidence_ids` and `parameter_paths`. The last two default to empty arrays. The eight review categories are `orbit`, `clock`, `instrument`, `environment`, `quantum_channel`, `protocol`, `numerical` and `model_discrepancy`. These categories prompt a review; they do not prove that every important error source has been identified.

Treatment	What the assessment requires and reports
`quantified`	Requires an active uncertainty-propagation binding and retained, nonsynthetic calibration evidence in the declared domain. The supported status is `documented_quantification`; otherwise it is `unresolved`. This confirms a documented linkage, not an authenticated calibration chain.
`bounded`	Requires applicable retained evidence. The status `documented_bound_requires_review` indicates a documentary bound. It is not automatically inserted into, or added to, the ensemble covariance.
`excluded`	Retains the rationale but reports `exclusion_requires_review`. The materiality of the exclusion is not automatically accepted.
`missing`	Reports `missing`. Unquantified error remains unknown; it is never assigned zero uncertainty to complete a budget.

Parameter bindings must also match the named uncertainty category; an unrelated active optical-efficiency parameter cannot close an orbit-error gap. Bindings refer to actual configured paths sampled by a completed enabled `precision.ensemble`. An inactive parameter does not supply coverage: an orbit profile is inactive for an SGP4 source or a ground endpoint, and Scenario 1 clones detector 1 rather than using detector 2's orbit profile. Disabled storage or QKD settings cannot support a coverage claim. The special binding `precisio n.timing_budget.input_covariance` refers only to an enabled, available `frozen_endpoint_sr` budget. Its presence does not cover moving-endpoint, static-GR or full Earth time-transfer errors.

For a scalar output with sensitivity vector g and joint input covariance C, the local first-order variance is

$$u_y^2 = g^{\mathsf{T}} C g = \sum_i g_i^2 C_{ii} + 2 \sum_{i<j} g_i g_j C_{ij}. \tag{24.1}$$

Diagonal and signed cross terms have different interpretations. A negative cross term can reduce the total variance without making any marginal variance negative. Do not display signed terms as ordinary nonnegative percentages. Adding a documentary bound to this expression requires a justified statistical model and dependence assumptions; the risk inventory deliberately does not perform that addition. Significant nonlinearity, a zero-key threshold or an unresolved event can invalidate a local linear approximation.

24.3.2 Evidence and review records

Each `evidence` record requires `id`, `path`, `kind`, `source`, `synthetic`, `domain` and `independent`. The kind is `calibration`, `validation`, `verification` or `review`; both flags must be explicit booleans. Optional `expected_sha256` is a 64-character hexadecimal digest or null. Optional `valid_until_utc` is a timezone-aware ISO 8601 timestamp or null. An absent expiry does not establish continuing validity.

Use the calibration kind for observations used to fit or select parameters. Use validation for a distinct prediction check; changing the word in a record does not make reused observations independent. Synthetic fixtures remain useful for testing the workflow and exposing failures, but cannot satisfy a requirement for measured hardware evidence. A source citation must describe the actual retained data and acquisition, rather than merely naming a paper that reports a similar experiment.

The `review` object accepts `reviewer`, `independent`, `decision` and `evidence_ids`. Defaults are null reviewer, false independence, `pending` decision and an empty evidence list. Decisions are `pending`, `accepted` or `rejected`. Accepted review requires a named reviewer. The intended-use owner cannot name themselves as an independent reviewer. This is a consistency check, not identity authentication; independence still needs accountable external confirmation.

24.4 Uncertainty-aware requirement decisions

The nominal checks in Chapter 23 remain useful for a single configuration. Assurance requirements instead evaluate every complete paired ensemble draw. They do not change physical inputs until the simulation passes. Each requirement contains the following fields:

Requirement field	Meaning
`id`	Unique nonempty requirement identifier.
`scenario`, `metric`, `unit`	Scenario 1, 2 or 3; one of the research aggregate metrics; its exact corresponding unit.
`operator`, `threshold`	`min` requires values at or above the threshold; `max` requires values at or below it. Equality is acceptable.
`max_exceedance_probability`	Probability limit between zero and one for violating the configured bound under the declared input distribution.
`confidence`	One-sided confidence level strictly between zero and one; default 0.95. This concerns sampling uncertainty in the estimated conditional probability.

Requirement field	Meaning
`iid_draws`	Explicit declaration that the whole draws are independent and identically distributed. Default false prevents an unsupported statistical pass. Correlated inputs *within* one draw do not preclude independent whole draws.

For example, this record tests a model-conditional minimum fidelity without claiming that 0.98 is an industry-wide threshold:

```
{
  "id": "FIDELITY_S2",
  "scenario": 2,
  "metric": "fidelity_mean",
  "unit": "1",
  "operator": "min",
  "threshold": 0.98,
  "max_exceedance_probability": 0.05,
  "confidence": 0.95,
  "iid_draws": true
}
```

Place it in `assurance.uncertainty_requirements`. The decision owner must justify both the fidelity threshold and the tail-probability allowance. The record does not enable an ensemble; configure that separately under `precision.ensemble`.

Supported metrics and units are inherited from the research extension: `fidelity_mean`, `purity_mean`, `qber_z_mean`, `qber_x_mean` and `max_ray_null_relative_residual` are dimensionless; `accepted_pair_rate_mean_hz` and `detected_pair_rate_mean_hz` use Hz; `arrival_gap_abs_max_s`, `gate_residual_abs_max_s` and `max_interception_residual_s` use seconds; `max_null_residual_m` uses metres; `secret_key_bits` uses bit, `secret_key_rate_hz` uses bit/s and `unresolved_epoch_count` uses count. The values are aggregate metrics, not arbitrary per-epoch observables.

For a predeclared total of N independent draws, let K be the number of threshold violations and p the underlying violation probability in the *configured model distribution*. At one-sided confidence $1-\alpha$, the exact Clopper–Pearson upper confidence bound used here is

$$p_{\mathrm{upper}} = \begin{cases} \mathrm{Beta}^{-1}(1-\alpha; K+1, N-K), & K < N, \\ 1, & K = N. \end{cases} \tag{24.2}$$

The inverse denotes a beta-distribution quantile. With no violations,

$$p_{\mathrm{upper}} = 1 - \alpha^{1/N}. \tag{24.3}$$

At 95% confidence, zero violations in 32 draws give an upper bound of about 8.94%; 128 draws give 2.31%; 512 draws give 0.583%. Thus a visually stable plot with no failures does not establish an arbitrarily small tail probability. For a positive target $p_\star < 1$, the reported zero-exceedance sample target is

$$N_0 = \left\lceil \frac{\log \alpha}{\log(1-p_\star)} \right\rceil. \tag{24.4}$$

This is a planning value under the zero-exceedance assumption. It is not a promise that the next run will pass, and may exceed the wrapper's 512-draw limit. A target of exactly zero cannot be demonstrated with a finite sample by this method.

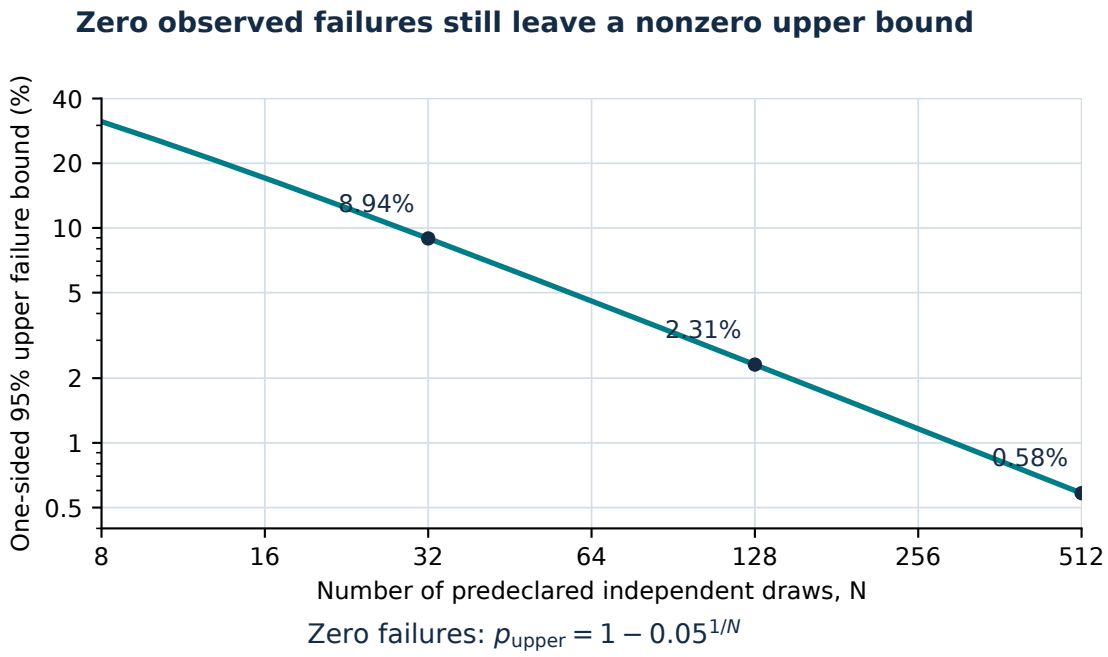

Figure 23: Exact one-sided binomial upper bound for zero observed violations. The curve is an analytic fixed-sample illustration, not measured mission reliability. The horizontal axis counts independent whole input draws, not photons, scenarios or correlated epochs within a draw.

24.4.1 Read the result without overstating it

`pass_conditional` means that the computed upper bound is no greater than the configured probability limit. `fail` means that the observed fraction exceeds that limit; this operational rule is not a confidence-level rejection test. If the observed fraction is within the limit but its upper bound is not, the status is `unresolved_sampling`. Insufficient precision is neither proof of failure nor proof of success.

Other unresolved statuses identify incomplete ensembles, unavailable metrics or undeclared draw independence. Every requested draw must be present once, with the required metric available and in the correct unit. The implementation does not discard failed draws or unavailable outputs to construct a favorable aggregate. The reported empirical quantile margin is supplementary: a positive margin alone does not supersede the upper probability bound.

The report also diagnoses the event "at least one configured requirement is violated in this whole draw." It preserves within-draw dependence rather than treating scenarios or requirements as independent experiments. No joint acceptance limit is configured by this diagnostic, and no joint PASS is inferred. Individual bounds are pointwise; they are not a familywise simultaneous confidence statement.

Choose N, thresholds, probability limits and confidence levels before running the assessment. Repeatedly increasing N, trying seeds or altering thresholds until a PASS appears invalidates the fixed-sample interpretation. The workbench does not provide an optional-stopping correction or a sequential testing procedure. More input draws reduce sampling uncertainty; they do not supply missing model mechanisms or experimental calibration.

24.5 Held-out vector comparison with covariance

A vector dataset compares several research metrics jointly, preserving their measurement and prediction correlations. It does not reconstruct a quantum channel from one density matrix and does not perform automated parameter fitting. The input JSON accepts the following fields:

Dataset field	Meaning
`schema_version`	`"1.0.0"`.
`source`, `synthetic`	Nonempty source declaration and explicit boolean synthetic flag. The routine does not authenticate either declaration.
`split`	`validation`, `training` or `calibration`. The latter two are excluded from held-out comparison.
`axes`	One to 128 unique objects with `scenario`, `metric` and `unit`, drawn from the research metric registry. Array order defines all vector and matrix axes.
`observed_values`	Finite measured or synthetic values in exactly the axis order.
`measurement_covariance`	Finite symmetric positive-semidefinite matrix R; entry ij has the product of axis-i and axis-j units.
`prediction_covariance`	Object with `mode` equal to `specified`, `ensemble` or `none`. Only `specified` accepts a `matrix`.
`cross_covariance`	Optional matrix $C = \mathrm{Cov}(\hat{y}, y_{\mathrm{obs}})$. If absent, prediction–measurement independence must be explicitly declared.
`assumptions`	Boolean declarations `gaussian_residual`, `known_covariance`, `independent_measurement_prediction` and `zero_mean_residual`; each defaults false.
`alpha`	Reference-test significance level in $(0, 1)$; default 0.05.

Let $\hat{y}$ denote the nominal prediction and y_{obs} the observation vector. With

$$r = y_{\mathrm{obs}} - \hat{y}, \qquad P = \mathrm{Cov}(\hat{y}), \qquad R = \mathrm{Cov}(y_{\mathrm{obs}}), \tag{24.5}$$

the residual covariance is

$$S = P + R - C - C^{\mathsf{T}}, \qquad C = \mathrm{Cov}(\hat{y}, y_{\mathrm{obs}}). \tag{24.6}$$

The joint matrix $\left(\begin{smallmatrix} P & C \\ C^{\mathsf{T}} & R \end{smallmatrix}\right)$ must also be positive semidefinite. Plausible marginal variances do not make an inconsistent cross-covariance valid. A nonzero C contradicts an explicit independence declaration and is rejected.

`specified` uses a supplied P after matrix checks. `none` explicitly assumes zero P; it does not establish that the prediction has no uncertainty or model discrepancy. `ensemble` estimates P from complete paired draws using the sample covariance with $N - 1$ denominator. That covariance is estimated, so the routine reports a diagnostic and withholds the known- covariance chi-square inference. The residual still uses the nominal prediction; the ensemble mean is reported separately.

24.5.1 Mixed units and singular covariance

The implementation normalizes covariance matrices by their marginal standard deviations before symmetry, eigenvalue and rank checks. This avoids declaring a physically relevant clock variance negligible merely because its numerical value in seconds squared is much smaller than a dimensionless polarization variance. An exactly zero marginal variance requires exactly zero covariance with every other axis. All matrices preserve their original values in the report; numerical symmetrization is limited to accepted eigensolver roundoff.

For positive marginal standard deviations, let $D = \mathrm{diag}(\sqrt{S_{11}}, \ldots, \sqrt{S_{mm}})$, $\widetilde{S} = D^{-1} S D^{-1}$ and $\tilde{r} = D^{-1} r$. For an exactly zero variance, the corresponding numerical scale is set to one in that axis's stated unit; the exact-zero constraint is tested separately. If the normalized matrix has eigendecomposition $\widetilde{S} = Q \Lambda Q^{\mathsf{T}}$, the diagnostic statistic is

$$T = \sum_{\lambda_j > \varepsilon} \frac{(q_j^{\mathsf{T}} \tilde{r})^2}{\lambda_j}, \qquad k = \mathrm{rank}(\widetilde{S}). \tag{24.7}$$

The report records the normalized tolerance, eigenvalues, rank and null-space residual. A residual component outside the covariance support produces `incompatible_nullspace`. A pseudoinverse must not silently discard that component. Zero residual with zero declared covariance gives `deterministic_agreement`; it supplies no statistical coverage claim.

Only when the residual is declared zero-mean Gaussian, covariance is declared known and P was not estimated from this ensemble does the routine use $T \sim \chi^2_k$. It reports the upper-tail probability $p = \Pr(\chi^2_k \geq T)$. A value at least `alpha` yields `conditional_consistency`; a smaller value yields `inconsistent`. Otherwise the status is `diagnostic_only`. Failure to find inconsistency is not proof of physical validity. The flags `source_verified` and `empirically_validated` remain false: the software cannot authenticate the experiment or certify the covariance assumptions. Estimated covariance, fitted parameters and repeated comparisons can require different reference distributions or multiplicity treatment.

24.6 Risk register and readiness

Each risk requires `id`, `description`, `consequence`, `severity`, `owner`, `mitigation`, `status`, `evidence_ids` and `requirement_ids`; the reference arrays may be empty. Severities are `low`, `medium`, `high` and `critical`. States are `open`, `mitigated` and `accepted`. These are declared organizational categories, not numerical probabilities of mission loss. The workbench does not invent a likelihood score or a monetary risk estimate.

A claimed mitigation needs retained applicable evidence and linked supported requirement checks. A claimed acceptance needs retained evidence and the documented independent review. High and critical risks without this support block readiness. Lowering severity or weakening a threshold to clear the report is not a mitigation. Record why a requirement exists, who owns the decision and which hardware or model change addresses the consequence.

The report's coverage rows, checks, blockers and recommended next actions expose what is missing. Recommendations concern evidence and model deficiencies; they are not an optimizer that promises safe mission parameters. Retain failed cases so a reviewer can inspect how the decision changed. An absent risk register, missing measurements or an unsupported calibration linkage remains visible even when every numerical verification check passes.

24.7 Evidence bundles and byte identity

The evidence sealer creates `EVIDENCE_MANIFEST.json` and `EVIDENCE_MANIFEST.sha256`. It covers every regular file recursively inside the selected bundle, except those two root-level manifest files. The digest is calculated from the exact manifest bytes. Repeated sealing of an unchanged file set is deterministic; resealing edited content deliberately produces a new identity.

The standalone verifier is read-only:

```
python verify_evidence.py --bundle results_risk_audit
python verify_evidence.py --bundle results_risk_audit --expected-manifest-sha256 DIGEST
```

Replace `DIGEST` with the 64-character manifest digest retained through a separately trusted channel. The first command reports `byte_consistent` when files match the local manifest. The second can report `externally_anchored_byte_identity` when the independently supplied digest also matches. Any detected mismatch produces `invalid`. The exit code is zero for a valid bundle and one for invalid evidence.

If an editor can replace both files and local checksums, a successful local verification does not establish which version an external reviewer previously saw. An expected digest stored only beside the editable files is not an independent trust anchor. The verifier performs no digital signing, signatory authentication or trusted timestamping. Its output explicitly keeps the flags for established authenticity, scientific validity and mission qualification false; byte identity is the property being checked.

Bundles are limited to 10,000 files, 1 GiB total regular-file content, 512 MiB per file and a 4 MiB manifest. Symlinks, traversal paths and portable-path case collisions are rejected. These checks help recipients inspect a bounded, unambiguous evidence set. They do not substitute for secure storage, controlled access or an organization's records policy.

24.8 Recommended review procedure

First write the decision and its operating domain, then specify the requirement thresholds and intended probability interpretation. Identify source, clock, orbit, instrument, environment, protocol, numerical and model-discrepancy uncertainties. Mark unresolved terms explicitly. Separate parameter-setting data from held-out observations before inspecting the validation result.

Run the configured physical model and retain the full ensemble, including its seed and prescribed sample count. Inspect numerical consistency and unavailable epochs before interpreting a covariance or tail bound. Review the vector comparison's axis order, units, dependence assumptions and residual support. Read every blocked coverage row and risk, then obtain a domain-competent review of both favorable and unfavorable findings.

Finally verify the completed evidence bundle, retain its digest independently and identify the exact version considered by the decision owner. A new model, new calibration, changed hardware, different operating regime or changed requirement may require a new assessment. Reproducibility makes the evidence inspectable; fitness for an industrial decision must still be demonstrated.

24.9 Scientific and assurance references

The implementation follows selected concepts from the sources below. This is not a clause-by-clause conformance assessment or an endorsement by the issuing organizations. The accompanying assurance research report records source dates, claim scope, competitor checks and proposed qualification work.

1. NASA-STD-7009B (2024): intended use, verification and validation domains, uncertainty and separate model-capability/results assessments. Program acceptance is specific to the decision and responsible authority.
2. ECSS-E-ST-10-02C Rev.1 (2018): requirements-based verification. Ordinary development tests are not identical to formal customer–supplier verification activities.
3. ECSS-Q-ST-80C Rev.2 (2025) and ECSS-M-ST-80C (2008): software assurance and integrated project risk management extend beyond a numerical test suite.
4. JCGM 106:2012: uncertainty-aware conformity decisions and guard bands. This workbench's conditional ensemble rule is not a calibrated operational false-acceptance guarantee.
5. JCGM 200:2012, VIM 2.41: metrological traceability concerns calibration chains and uncertainty, and is different from file provenance.
6. NIST TN 1297, Appendix A, JCGM 101:2008 and JCGM 100:2008/Amd.1:2026: sensitivity/covariance propagation, Monte Carlo and significant nonlinearity.
7. NIST FIPS 180-4 and FIPS 186-5: hashes and digital signatures serve different integrity/authentication roles. The current bundle verifier implements hash comparison, not digital signing.

Existing tools also provide substantial engineering capability. Official STK HPOP documentation describes orbit covariance and uncertain-parameter analysis; NASA's GMAT capability description includes operations support and uncertainty studies; and NetSquid models timed quantum-network behavior. The potential commercial value of PF's evidence workflow must be demonstrated against a specific customer's existing process. It should not be justified by claiming that these tools are merely isolated theoretical calculators.

The two figures in this chapter are generated by `generate_assurance_figures.py`. The diagram is conceptual and the bound curve is analytic. Neither figure is presented as satellite experimental data.

Chapter 25

Quantum protocol and component benchmarks

Edition 1.10.0 adds a separately configurable quantum-benchmark workflow. Its nine profiles cover Bell-state observables, teleportation, entanglement swapping, recurrence purification, memories, a two-link repeater, two-photon interference, a QKD loss sweep and single-qubit channel characterization. These are calculations of stated protocol and component models. They extend research coverage in areas also discussed by quantum-network and quantum-link simulation products; they are not a certification against a universal industry benchmark standard or a reproduction of proprietary vendor results.

The existing SatQuMA, laboratory BBM92, Jinan-1 and Micius workflows retain their separate numerical or empirical reference roles. No experimental dataset has been synthesized and relabelled as a new observation. A randomly sampled measurement in this chapter is explicitly a simulation generated from a model probability. Its seed, sample size and model assumptions belong to the evidence.

25.1 Choose the scientific question

Profile	Calculated result	Appropriate interpretation
`bell`	Density matrix, entanglement/state diagnostics and CHSH correlations.	Compare a configured two-qubit state with quantum identities; a predicted CHSH violation is not a loophole-free Bell experiment.
`teleportation`	Output states after Bell measurement and correction, success and transfer fidelity.	Test a specified noisy resource and protocol; classical communication is required.
`swapping`	Remote conditional state after a local Bell measurement and correction.	Inspect entanglement quality and heralding probability separately.
`purification`	Accepted-pair fidelity and survival through recurrence rounds.	Quantify conditional quality against consumed resources; purification does not create pairs for free.
`memory`	Conditional state degradation and retrieval survival versus storage time.	Test declared memory parameters; these curves do not calibrate a physical memory.
`repeater`	Event-trial waiting times, delivery rate and memory-aged state quality.	A bounded two-link model, not general network routing or a full satellite mission.
`hom`	Wavepacket overlap and two-output coincidence probability.	A specified single-photon mode model of Hong–Ou–Mandel interference.
`qkd\_sweep`	Detection/error counts and finite-key forecasts over declared loss.	Evaluate the selected existing security calculation, not operational keys.
`channel\_tomography`	Probe outputs, Pauli transfer and Choi representations.	Verify a simulated channel and optionally sample tomography measurements.

25.2 Run through the GUI or command interface

Open the local workbench and select the quantum-benchmarks workflow. Choose a profile or the all-profile starting configuration, edit its fields and run. The topology controls follow the selected workflow's applicability: a protocol benchmark is not converted into a space-to-ground pass by changing a label. This standalone entry

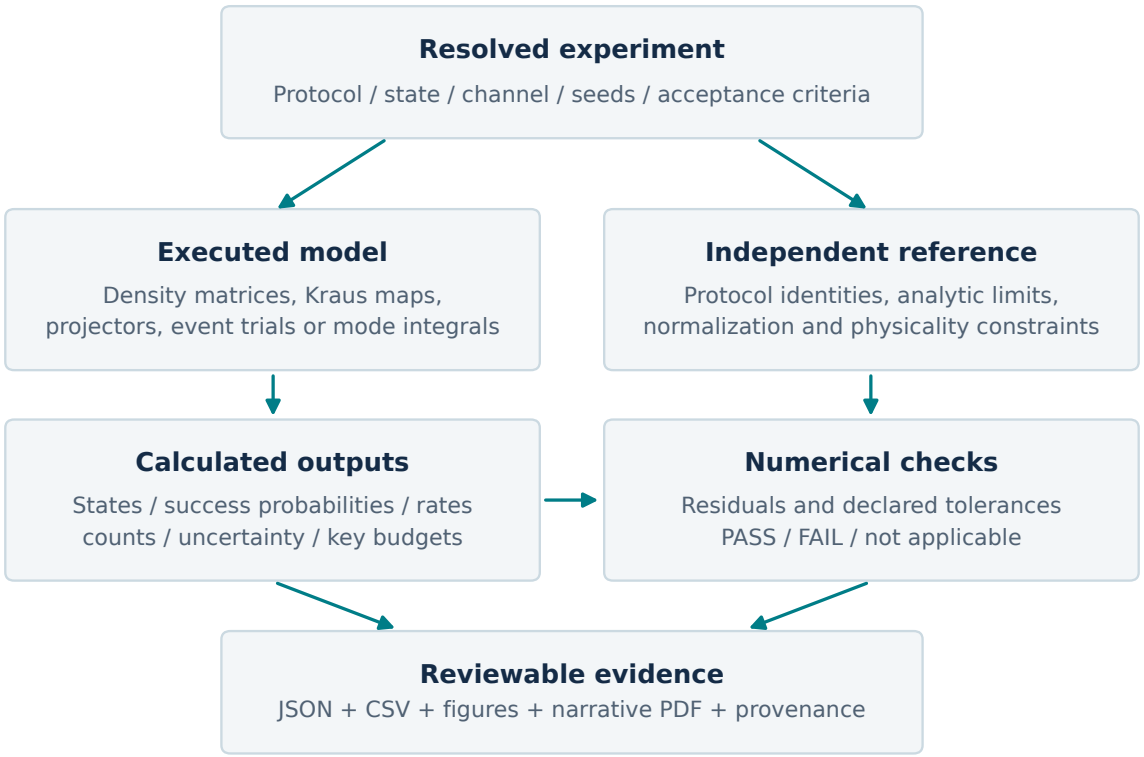

Figure 24: Quantum-benchmark evidence flow. The model is executed for the resolved configuration and compared with an independently evaluated reference or limiting identity. A passing residual check supports that numerical relationship. It does not insert an empirical measurement into the workflow or certify an apparatus.

point runs component/protocol models without a satellite trajectory. Edition 1.11.0 also adds explicit mission quantum experiments in the mission workflows; Chapter 26 specifies their apparatus, state handoff, timing and applicability. Selecting a standalone profile still does not silently enable that coupling.

The cross-platform command entry point is:

```
python run_quantum_benchmarks.py --profile all --config config_quantum_benchmarks.json --output
 ↪  results_quantum_benchmarks
python run_quantum_benchmarks.py --profile teleportation --config config_quantum_benchmarks.json
 ↪  --output results_teleportation
```

The command options are `--profile` (one profile or `all`), `--config` (suite JSON), `--output` (new evidence directory), `--write-default-config` (export defaults and exit), `--no-plots` (omit standalone figures but retain data and narrative report), and `--require-engineering-pass` (make a failed configured requirement produce a nonzero completion status). Existing nonempty output directories are rejected so a partial new run cannot mix with a prior report.

Numerical failure returns exit code 2; with the explicit engineering gate, requirement failure, an unavailable target, or no configured requirements returns 3. The suite records engineering status as `PASS`, `FAIL` or `NOT_ASSESSED`; an empty requirements list is not a successful engineering assessment. A completed valid numerical run can return zero while empirical validation remains `NOT_ASSESSED`. Parser/input errors also use a nonzero status; read the named diagnostic rather than interpreting all failures as an unfavorable physical result.

Use the package's supported Python environment. An output directory is an experiment record: retain its resolved configuration and numerical files with the figures. A graph alone does not retain enough information to reproduce the calculation. The same engine functions serve the GUI and command entry point; launching a different interface does not select a different physical model.

25.3 States, channels and protocol operations

The density operator is normalized on the explicitly stated conditional subspace. For a target pure state $|\psi\rangle$, the fidelity convention is

$$F = \langle\psi|\rho|\psi\rangle, \qquad P = \mathrm{Tr}(\rho^2). \tag{25.1}$$

A linear change of coordinates does not, by itself, dephase this state. A physical channel must specify its operators or an equivalent evolution law. A trace-preserving completely positive channel has the form

$$\mathcal{E}(\rho) = \sum_k K_k \rho K_k^\dagger, \qquad \sum_k K_k^\dagger K_k = I. \tag{25.2}$$

A measurement branch is instead trace nonincreasing: its trace is the branch probability. Normalize the branch only when that probability is positive. Conditioning on a rare successful event can yield high fidelity while the usable rate remains very low. Both numbers must therefore be reported.

25.3.1 Bell correlations and teleportation

For the two-qubit correlation matrix $T_{ij} = \mathrm{Tr}[\rho(\sigma_i \otimes \sigma_j)]$, let m_1, m_2 be the two largest eigenvalues of $T^\mathsf{T}T$. The optimized CHSH value is

$$S_{\max} = 2\sqrt{m_1 + m_2}. \tag{25.3}$$

When sampling is enabled, the report retains both the signed sum $S = E_{00} + E_{01} + E_{10} - E_{11}$ and its magnitude. For independent N shots per setting, the exact model sampling standard deviation of the signed sum is $\sqrt{\sum_{i,j}(1 - E_{ij}^2)/N}$, stored as `signed_chsh_sampling_standard_deviation`. The sampled magnitude has a folded distribution; this value is not its exact standard deviation near zero or a confidence bound certifying a Bell violation.

This is an optimization over qubit observables under the stated state model. A fixed experimental analyzer setting can yield a different value. Sampling uncertainty, detection loopholes and device-independent security assumptions are separate from computing this expression.

Teleportation transfers an input qubit using an entangled resource, a joint measurement, classical measurement information and the appropriate correction. A useful numerical check runs the actual measurement branches and compares the result with the applicable channel identity. For a depolarized Bell resource $\rho_v = v|\Phi^+\rangle\langle\Phi^+| + (1-v)I_4/4$ and ideal protocol operations, the transfer channel is depolarizing and its uniform pure-state mean fidelity is $(1+v)/2$. The configured resource uses fidelity F_{in}, with $v = (4F_{\mathrm{in}} - 1)/3$ and hence mean transfer fidelity $(2F_{\mathrm{in}}+1)/3$. The engine evolves six Pauli eigenstate inputs, enumerates measurement branches, applies the feed-forward gates and averages the resulting fidelities. This identity is restricted to that resource and operation model. The familiar classical fidelity threshold requires its input-ensemble and success conventions; it is not a universal PASS threshold for arbitrary postselected experiments.

25.3.2 Swapping and recurrence purification

Entanglement swapping operates on two input pairs and measures the middle qubits. The remote conditional state depends on the measured outcome and any feed-forward correction. The success probability must include the configured measurement-success assumptions; an ideal qubit Bell measurement and a particular linear-optical implementation do not have identical success limits. An algebraic four-outcome calculation is not a simulation of detector hardware unless that hardware model is explicitly included.

Recurrence purification consumes two noisy pairs, operates locally at each party, measures the target pair and keeps the control pair only for accepted outcomes. Report both output fidelity and acceptance. Any multi-round yield must account for two input pairs per attempted recurrence and all rejection branches. The engine supports configured local depolarization after its gates, independent classical readout errors and state-independent detector survival. The independent ideal-operation recurrence identity applies only in its stated ideal limit. Optional output twirling is an actual sum over 24 bilateral Clifford rotations. The pooled independent-copy yield is the product of each round's acceptance probability divided by two; it is not a finite memory-pool scheduler or a general fault-tolerant architecture.

25.4 Memory, waiting times and two-photon modes

A memory's conditional polarization decay and retrieval loss are different channels. The memory profile applies local zero-temperature amplitude damping plus pure dephasing to a $|\Phi^+\rangle$ resource. It requires $T_2 \leq 2T_1$ and uses $1/T_\phi = 1/T_2 - 1/(2T_1)$. For each local memory, the off-diagonal Markovian law is $\rho_{01}(t) = \rho_{01}(0)\exp(-t/T_2)$ with $T_2 > 0$; a separate retrieval survival probability accounts for erasure. Where amplitude relaxation is also included, the combined channel must respect its documented T_1/T_2 convention. A fitted device may require nonexponential decay, spectral diffusion, control-pulse errors or correlated memories. None follows merely from assigning a longer geometric separation.

The repeater profile uses independent geometric elementary-link attempts in synchronous, nonpipelined renewal cycles. Its attempt period is at least $2\max(L_A, L_B)/c_{\text{signal}}$; the link resource visibility is defined at the heralded-ready timestamp. A two-link repeater must wait for both elementary resources. The earlier resource accumulates additional storage time before swapping. This creates an explicit causal path from probabilistic link success to memory evolution and conditional state quality. The simulation records its trial convention and exposure; a finite trial estimate is not an empirical network measurement. Classical signal delay, clock convention, retry policy and loss must be kept consistent when interpreting its rate. The four-qubit state is evolved through explicit memory Kraus operators, all Bell-measurement branches and Pauli feed-forward; the reduced Bell-diagonal identity is an independent check. The configured BSM duration means processing after an instantaneous quantum measurement and is charged with final one-way notification time. Failed BSMs consume both links; a cutoff discards the older resource. This model does not implement arbitrary multihop routing, contention, purification, multiplexing, T1 relaxation or false heralds. Its standard errors describe Monte Carlo sampling under fixed inputs, not apparatus/model uncertainty. With no delivered pair, sampled delivery fidelity remains unavailable. The memory and HOM profiles separately retain their conditional theoretical quantities when retrieval or detection probability is zero.

For normalized single-photon temporal modes $f_A(t)$ and $f_B(t)$, their overlap is

$$\mu(\delta) = \int f_A^*(t) f_B(t-\delta)\,\mathrm{d}t. \tag{25.4}$$

For a lossless balanced beam splitter and ideal otherwise-indistinguishable single-photon inputs, the coincidence probability is

$$P_{\text{coin}}(\delta) = \frac{1 - |\mu(\delta)|^2}{2}. \tag{25.5}$$

Additional distinguishability, a nonbalanced splitter or configured detection survival changes the documented expression. This describes mode interference, not a general law converting a relativity-of-simultaneity offset into polarization decoherence. Multiphoton statistics, background and a detailed photodetection response require their own model.

25.5 QKD sweeps and channel characterization

The QKD loss sweep executes both the conservative and SatQuMA-compatible decoy-BB84 key-budget engines, in finite and asymptotic modes, for its resolved protocol settings at each loss point. Loss changes detection statistics, which change finite-key bounds; a rate curve cannot substitute for that calculation. Security parameters, count mode, error-correction leakage and authentication debits retain their existing interpretation. Positive forecast key length means the configured model produced a positive bound. It does not mean the program has generated or authenticated an operational secret key.

The declared expected-count model assumes phase-randomized Poisson pulses and a stationary channel for each block. With channel transmittance η, detector efficiency η_d, mean photon number μ, background probability Y_0 and signal misalignment probability e_d, it uses

$$s_\mu = 1 - \exp(-\mu\eta\eta_d), \tag{25.6}$$

$$Q_\mu = Y_0 + (1 - Y_0)s_\mu, \tag{25.7}$$

$$E_\mu Q_\mu = Y_0/2 + e_d(1 - Y_0)s_\mu. \tag{25.8}$$

Background takes precedence with a random bit when signal and background overlap. The model has no fading, dead time, afterpulsing, source drift or side channel. The engines have different statistical bounds and failure allocations: their outputs should not be expected to coincide. The conservative engine's separate parameter-estimation, privacy-amplification and authentication epsilon settings are not inputs to the SatQuMA adapter. Inspect both native reports.

The repeaterless pure-loss capacity reference is

$$K_{\mathrm{PLOB}}(\eta) = -\log_2(1-\eta) \quad \text{bits per bosonic optical mode.} \tag{25.9}$$

This is an unconstrained-energy point-to-point reference. At loss zero it is unbounded, reported as null with an explicit flag, not plotted as a finite ceiling. The sweep divides key bits by emitted pulses times the configured number of used optical modes (at least two for the modeled qubit encoding). Detector efficiency is excluded from the reference channel transmittance. A per-mode PLOB value is not a bound on an arbitrary two-arm entanglement network or a repeater delivery rate. No quantum-repeater advantage is inferred from comparing such incompatible normalizations.

The single-qubit channel profile prepares the informationally complete probes $|0\rangle, |1\rangle, |+\rangle, |+i\rangle$ and computes their output states. For the Pauli basis $\sigma_0 = I, \sigma_1 = X, \sigma_2 = Y, \sigma_3 = Z$, its transfer matrix is

$$R_{ij} = \tfrac{1}{2}\,\mathrm{Tr}[\sigma_i \mathcal{E}(\sigma_j)]. \tag{25.10}$$

The stored Qiskit Choi matrix is unnormalized with trace two; divide it by two to obtain the normalized Choi state used below. The profile checks the reconstructed representation against a direct channel representation and analytic limits. Trace preservation and Choi positivity are separate physicality checks. With normalized Choi state $J = (I \otimes \mathcal{E})(|\Phi^+\rangle\langle\Phi^+|)$, the process fidelity to the identity is $F_e = \langle\Phi^+|J|\Phi^+\rangle$. For a trace-preserving qubit channel,

$$F_{\mathrm{avg}} = \frac{2F_e + 1}{3}. \tag{25.11}$$

This average is over uniformly distributed pure qubit inputs.

The profile's dephasing parameter is a phase-flip probability, not a storage duration. Its depolarizing parameter follows $\mathcal{E}(\rho) = (1-\lambda)\rho + \lambda I/2$; its amplitude-damping parameter is the excited-state decay probability; the Z-rotation angle is in radians. Using the same numerical sweep for these parameters does not give them the same physical units or effect.

When finite-shot probe measurements are enabled, the counts are sampled from model probabilities using the recorded seed. Linear inversion can produce a non-positive estimated Choi matrix at finite sample size. The report preserves that diagnostic instead of silently projecting the estimate onto a different physical channel. Polarization-independent survival changes the number of available measurements separately from the conditional channel. These are simulated characterization checks, not calibrated experimental process tomography.

25.6 Complete configuration reference

The root object has integer `schema_version:1`, a nonempty `profiles` object keyed by the selected profile names, and optional `requirements`. An empty profile object requests its defaults. Unknown fields and invalid domains are rejected; duplicate/non-finite JSON is not accepted. The resolved configuration, rather than an earlier preset, records what executed. Numerical resource limits protect the local application; they are not asserted physical cutoffs.

Each requirement has exactly `id`, `profile`, `metric`, `aggregation`, `operator` and `value`. The ID is unique; the metric is a scalar series field; aggregation is `minimum`, `maximum` or `mean`; the operator is `>=` or `<=`; the value is finite. Requirements select an engineering question independently of numerical checks. A missing/non-numeric requested metric is not silently interpreted as zero.

The following tables list every shipped profile setting and its default. Square brackets denote a sweep/list, `null` an explicitly supported absent value, and `true/false` Boolean values. SI units apply unless the key explicitly states kilometres, decibels or hertz.

25.6.1 Bell / CHSH settings

Setting	Default	Meaning and domain
fidelity_values	[0.25, 0.5, 0.75, 0.9, 1.0]	Bell-resource input fidelity sweep; each value lies in [0,1]; at most 1001 points. This is preparation quality, not an output target or fitted observation.
local_depolarization_a	0.0	Arm A single-qubit depolarizing probability in [0,1].
local_depolarization_b	0.0	Arm B single-qubit depolarizing probability in [0,1].
phase_flip_probability_a	0.0	Arm A Z-flip probability in [0,1].
phase_flip_probability_b	0.0	Arm B Z-flip probability in [0,1].
relative_phase_rad	0.0	Physical relative phase applied to the Bell state, in radians.
measurement_angles_rad	[0.0, 1.5707963267948966, 0.7853981633974483, -0.7853981633974483]	Four Bloch analyzer angles A0,A1,B0,B1 in radians for cos(theta)Z+sin(theta)X; twice linear-polarizer orientation in H/V.
readout_error	0.0	Independent classical measurement-bit flip probability in [0,1].
shots_per_setting	4096	Simulated Born-rule measurement count per Bell setting; zero omits finite-shot estimates.
seed	20260919	Random seed for reproducible model sampling; never an experimental acquisition identifier.
numerical_tolerance	1e-10	Absolute reference-check tolerance, from 1e-14 to 1e-6; not an apparatus accuracy.

25.6.2 Teleportation settings

Setting	Default	Meaning and domain
fidelity_values	[0.25, 0.5, 0.75, 0.9, 1.0]	Bell-resource input fidelity sweep; each value lies in [0,1]; at most 1001 points. This is preparation quality, not an output target or fitted observation.
gate_depolarization	0.0	Local depolarizing probability after each supported ideal gate, including applied feed-forward gates.
readout_error	0.0	Independent classical measurement-bit flip probability in [0,1].
detector_efficiency	1.0	Detection efficiency in [0,1]; interpreted by the selected profile, not by a universal rate multiplier.
accepted_outcomes	["00", "01", "10", "11"]	Nonempty unique list of true Bell-measurement bit outcomes accepted before readout errors; two Psi outcomes are [01,11].
numerical_tolerance	1e-10	Absolute reference-check tolerance, from 1e-14 to 1e-6; not an apparatus accuracy.

25.6.3 Entanglement swapping settings

Setting	Default	Meaning and domain
`fidelity_values`	`[0.25, 0.5, 0.75, 0.9, 1.0]`	Bell-resource input fidelity sweep; each value lies in [0,1]; at most 1001 points. This is preparation quality, not an output target or fitted observation.
`link_b_fidelity`	`0.95`	Fixed Bell-resource fidelity for link B; the sweep varies link A.
`gate_depolarization`	`0.0`	Local depolarizing probability after each supported ideal gate, including applied feed-forward gates.
`readout_error`	`0.0`	Independent classical measurement-bit flip probability in [0,1].
`detector_efficiency`	`1.0`	Detection efficiency in [0,1]; interpreted by the selected profile, not by a universal rate multiplier.
`accepted_outcomes`	`["00", "01", "10", "11"]`	Nonempty unique list of true Bell-measurement bit outcomes accepted before readout errors; two Psi outcomes are [01,11].
`numerical_tolerance`	`1e-10`	Absolute reference-check tolerance, from 1e-14 to 1e-6; not an apparatus accuracy.

25.6.4 Entanglement purification settings

Setting	Default	Meaning and domain
`fidelity_values`	`[0.25, 0.5, 0.6, 0.75, 0.9, 0.99, 1.0]`	Bell-resource input fidelity sweep; each value lies in [0,1]; at most 1001 points. This is preparation quality, not an output target or fitted observation.
`rounds`	`1`	Integer recurrence rounds from 1 to 100; no more than 10000 scan-round evaluations.
`twirl_output`	`true`	Apply ideal 24-Clifford bilateral U tensor U* random-unitary twirling after each round.
`gate_depolarization`	`0.0`	Local depolarizing probability after each supported ideal gate, including applied feed-forward gates.
`readout_error`	`0.0`	Independent classical measurement-bit flip probability in [0,1].
`detector_efficiency`	`1.0`	Detection efficiency in [0,1]; interpreted by the selected profile, not by a universal rate multiplier.
`numerical_tolerance`	`1e-10`	Absolute reference-check tolerance, from 1e-14 to 1e-6; not an apparatus accuracy.

25.6.5 Quantum memory settings

Setting	Default	Meaning and domain
hold_times_s	[0.0, 0.001, 0.005, 0.01, 0.025, 0.05, 0.1, 0.2]	Nonnegative storage durations in seconds; at most 10000 entries.
t1_a_s	0.1	Positive arm A population relaxation time T1, seconds; zero-temperature channel.
t1_b_s	0.1	Positive arm B population relaxation time T1, seconds; zero-temperature channel.
t2_a_s	0.08	Positive arm A coherence time T2, seconds; must not exceed 2 T1.
t2_b_s	0.08	Positive arm B coherence time T2, seconds; must not exceed 2 T1.
retrieval_efficiency_a	0.9	State-independent arm A retrieval probability in [0,1], distinct from conditional state quality.
retrieval_efficiency_b	0.9	State-independent arm B retrieval probability in [0,1], distinct from conditional state quality.
retrieval_lifetime_a_s	0.5	Positive arm A erasure lifetime in seconds; null disables time-dependent retrieval decay.
retrieval_lifetime_b_s	0.5	Positive arm B erasure lifetime in seconds; null disables time-dependent retrieval decay.

25.6.6 Quantum repeater settings

Setting	Default	Meaning and domain
cycles	10000	Independent renewal cycles, including failed/discarded cycles; integer from 2 to 1000000.
seed	24601	Random seed for reproducible model sampling; never an experimental acquisition identifier.
attempt_period_s	0.001	Positive synchronous attempt period; at least the two-way herald latency of the longer link.
link_length_a_km	10.0	Nonnegative elementary link A length in kilometres.
link_length_b_km	10.0	Nonnegative elementary link B length in kilometres.
attenuation_db_per_km	0.2	Nonnegative attenuation coefficient used when a link success override is null.
signal_speed_m_s	200000000.0	Positive classical/herald signal speed in metres per second, no greater than c.
source_success_a	0.5	Arm A per-attempt source success factor used for derived link probability.
source_success_b	0.5	Arm B per-attempt source success factor used for derived link probability.
detector_efficiency_a	0.8	Arm/output A efficiency in [0,1], under the selected profile convention.
detector_efficiency_b	0.8	Arm/output B efficiency in [0,1], under the selected profile convention.

Setting	Default	Meaning and domain
link_success_probability_a	null	Explicit per-attempt heralded link A probability in [0,1]; null derives it from source, loss and detection.
link_success_probability_b	null	Explicit per-attempt heralded link B probability in [0,1]; null derives it from source, loss and detection.
link_visibility_a	0.99	Werner visibility of elementary pair A, in [0,1]; fidelity is (1+3v)/4.
link_visibility_b	0.99	Werner visibility of elementary pair B, in [0,1]; fidelity is (1+3v)/4.
memory_t2_s	0.1	Positive repeater pure-dephasing coherence time in seconds; null disables this decay. This profile has no T1 relaxation.
memory_t2_by_qubit_s	null	Optional per-qubit T2 values for endpoint_a, relay_a, relay_b and endpoint_b. Missing entries inherit memory_t2_s; null disables decay for that qubit.
bsm_success_probability	0.5	Heralded Bell-state measurement success probability in [0,1]. A failed measurement consumes both links.
record_attempt_history	false	Record individual renewal-cycle link attempts, resource identities and delivery outcomes. Boolean; false omits the full history.
max_history_events	100000	Maximum stored attempt-history events, integer from 1 to 10000000. Exceeding the limit raises an error; events are never silently truncated.
ensemble_seeds	[]	Up to 100 unique additional random seeds, each from 0 to 4294967295 and different from the primary seed. Each seed executes a separate replica without full attempt history.
bsm_duration_s	1e-06	Nonnegative processing time after instantaneous quantum BSM, added to final classical notification latency.
retrieval_efficiency_a	0.9	State-independent arm A retrieval probability in [0,1], distinct from conditional state quality.
retrieval_efficiency_b	0.9	State-independent arm B retrieval probability in [0,1], distinct from conditional state quality.
cutoff_attempts	null	Nonnegative maximum waiting-attempt difference before discarding the older link; null has no cutoff.

25.6.7 HOM interference settings

Setting	Default	Meaning and domain
delays_s	[-4e-09, -3e-09, -2e-09, -1e-09, -5e-10, 0.0, 5e-10, 1e-09, 2e-09, 3e-09, 4e-09]	Relative photon arrival-delay sweep in seconds, including signed delays; at most 1000 entries.
intensity_sigma_a_s	5e-10	Positive intensity RMS duration of Gaussian photon A, seconds; not field-amplitude RMS duration.
intensity_sigma_b_s	5e-10	Positive intensity RMS duration of Gaussian photon B, seconds.
detuning_hz	0.0	Signed carrier-frequency difference, hertz.
relative_delay_jitter_s	1e-10	Nonnegative RMS Gaussian relative input-delay jitter, seconds; not detector readout jitter.
beam_splitter_reflectivity	0.5	Intensity reflectivity R in [0,1]; transmission is 1-R.
mode_visibility	1.0	Additional indistinguishability factor in [0,1], multiplying the interference term.
transmission_a	0.5	State-independent input A transmission in [0,1].
transmission_b	0.5	State-independent input B transmission in [0,1].
detector_efficiency_a	0.8	Arm/output A efficiency in [0,1], under the selected profile convention.
detector_efficiency_b	0.8	Arm/output B efficiency in [0,1], under the selected profile convention.
pair_attempt_rate_hz	100000.0	Nonnegative pair-attempt rate; converts detected coincidence probability to events per second.
quadrature_points	2049	Wavepacket quadrature grid count from 257 to 32769; numerical resource control.
jitter_quadrature_order	24	Gauss-Hermite order from 4 to 128; numerical jitter-integration control.
quadrature_tolerance	1e-07	Positive allowed absolute overlap discrepancy between independent calculations.

25.6.8 Channel tomography settings

Setting	Default	Meaning and domain
channels	["amplitude_damping", "dephasing", "depolarizing", "rotation_z"]	Nonempty selection of amplitude_damping, dephasing, depolarizing and rotation_z.
parameter_values	[0.0, 0.02, 0.1, 0.3, 0.5]	Common channel-parameter sweep: stochastic probabilities in [0,1], or Z-rotation angle in radians.
shots	0	Simulated trials per probe and Pauli setting; zero keeps exact characterization only. Up to 100000000.
seed	20260919	Random seed for reproducible model sampling; never an experimental acquisition identifier.

Setting	Default	Meaning and domain
survival_probability	1.0	Polarization-independent survival in [0,1]; affects sampled probe counts separately from exact conditional channel.

25.6.9 QKD sweep settings

Setting	Default	Meaning and domain
loss_db	[0.0, 10.0, 20.0, 30.0, 40.0, 50.0]	Stationary channel-loss sweep, 0 to 300 dB; detector efficiency is separate.
signal_intensities	[0.5]	Signal mean photon numbers; every signal/decoy combination must satisfy signal > decoy > 0.
decoy_intensities	[0.1]	Nonzero weak-decoy mean photon numbers; the third intensity is vacuum.
key_basis_probabilities	[0.8]	Common sender/receiver raw-key basis probabilities, strictly between zero and one.
block_pulses	[10000000000]	Emitted pulses per block; positive integer values no greater than 1e15.
epsilon_secrecy_values	[1e-09]	Secrecy-error targets strictly between zero and 0.5; supported engine allocation constraints also apply.
intensity_probabilities	{"signal": 0.8, "decoy": 0.15, "vacuum": 0.05}	Positive signal/decoy/vacuum choice probabilities summing to one.
detector_efficiency	0.8	Detection efficiency in [0,1]; interpreted by the selected profile, not by a universal rate multiplier.
background_probability	1e-06	Background click probability per pulse, in [0,1]; gives a random bit and takes precedence over overlapping signal.
misalignment_probability	0.015	Signal-bit error probability in [0,0.5].
pulse_rate_hz	100000000.0	Positive emission pulse rate; block duration is pulse count divided by this rate.
optical_modes_per_pulse	2	Integer mode count at least two; include every used mode in PLOB normalization.
security.epsilon_correctness	1e-12	Correctness-error target; interpreted by both native security reports.
security.epsilon_parameter_estimation	1e-10	Conservative-engine parameter-estimation allocation; not a SatQuMA input.
security.epsilon_privacy_amplification	1e-10	Conservative-engine privacy-amplification allocation; not a SatQuMA input.
security.epsilon_authentication	1e-12	Conservative-engine authentication-error allocation; not a SatQuMA input.
security.ec_efficiency	1.16	Error-correction leakage multiplier; existing engine domain checks apply.
security.authentication_bits	128	Nonnegative integer authentication debit subtracted from the block key budget.
satquma_bound	"chernoff"	SatQuMA finite-count bound: chernoff or hoeffding; asymptotic is calculated separately.

25.7 Read the result as evidence

The workflow saves a suite record, individual profile results, resolved input, CSV tables, PNG previews and vector PDFs. The narrative `quantum_benchmark_report.pdf` is a separate report: it explains statuses, metrics, assumptions and limits. A vector figure PDF contains the same plotted data as its PNG preview; it is not an additional narrative report.

A read-only saved-result validator checks artifact hashes, consistency between suite/profile/CSV records and independent state/Choi physicality:

```
python validate_quantum_benchmarks.py results_quantum_benchmarks
python validate_quantum_benchmarks.py results_quantum_benchmarks --check-current-source
```

The optional source check compares the recorded engine files with the current installation. A changed code version can legitimately fail that check for an older run. The validator prints its report and does not rewrite the sealed run. These checks establish file/calculation consistency, not empirical truth.

Review outputs in this order: identify the resolved inputs and profile; confirm which model operations ran; inspect state or channel normalization and success probability; compare the numerical result with its applicable reference; then interpret rates and any configured requirements. Keep exact predictions, sampled estimates and reference formulas distinct. A discrepancy is useful information about code, conventions, finite sampling or a violated model domain.

A reference PASS means the declared residual is within its declared numerical tolerance. A requirement PASS means a configured engineering criterion was met. Neither establishes empirical agreement. Changing a requirement or tolerance changes the question; it is not a scientific repair to a failed result. Recommended actions should identify a convention, input or numerical issue and retain the original evidence. Physical quality improvements require a new, explicit hardware or protocol assumption and a new run.

25.8 Worked results from the delivered default run

The figures below are copied from the executed all-profile default run in `validation_competitive/final_defaults`. They are model results under the shipped configuration, not physical apparatus measurements or proprietary-vendor output. Their JSON records retain the settings and calculation checks. Re-run a changed configuration into a new directory; do not overwrite this reference evidence.

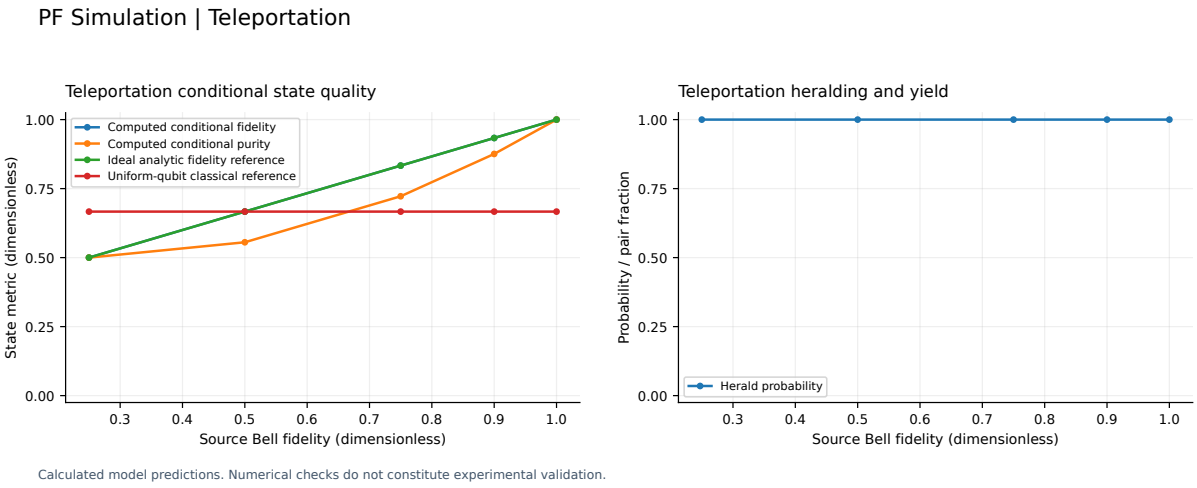

Figure 25: Executed teleportation default. The configured Bell-resource fidelity varies from 0.25 to 1 with ideal gates/readout, unit detection efficiency and all four Bell outcomes accepted. Transfer fidelity ranges from 0.5 to 1 and agrees with $(2F_{\mathrm{in}} + 1)/3$. The predicted and analytic curves overlap. The 2/3 classical line refers to deterministic uniformly distributed pure-qubit inputs; it is not a universal hardware-qualification threshold.

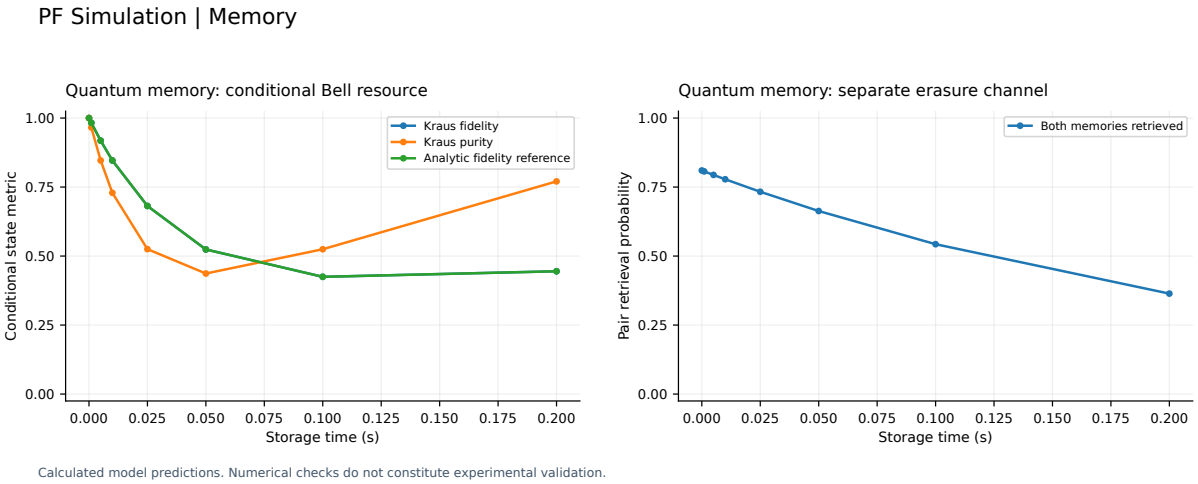

Figure 26: Executed two-memory default. Each arm has $T_1 = 0.1$ s, $T_2 = 0.08$ s, initial retrieval efficiency 0.9 and retrieval lifetime 0.5 s. The initial conditional Bell state has unit fidelity/purity while joint retrieval is 0.81. Retrieval loss remains separate from the normalized state. At long times, amplitude damping can increase purity toward a ground state without restoring Bell entanglement; higher purity alone is not higher fidelity.

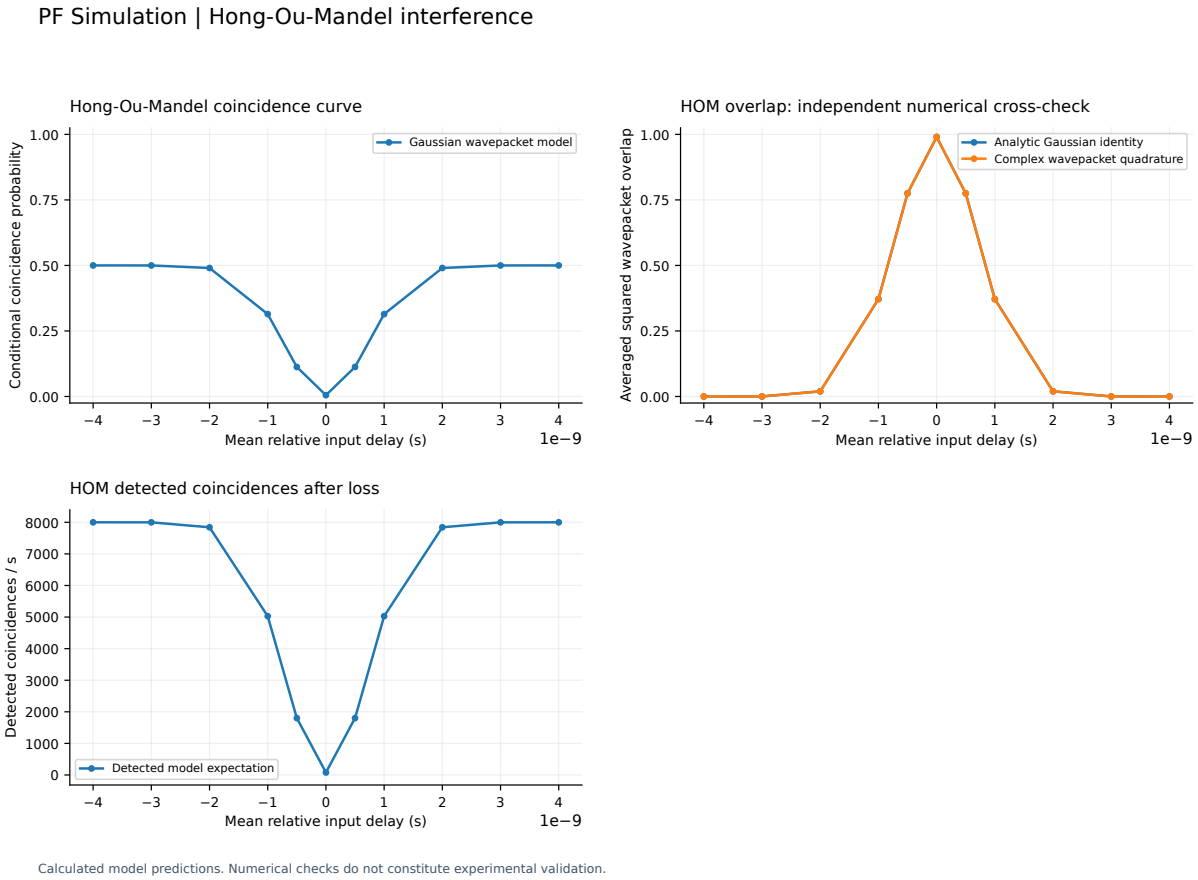

Figure 27: Executed Gaussian two-photon default. Equal intensity RMS durations of 0.5 ns, relative arrival jitter of 0.1 ns, zero detuning, a balanced beam splitter and unit additional mode visibility give a minimum conditional coincidence probability of approximately 0.004926. The model's nonzero dip floor comes from the declared input jitter. Transmission and detection survival multiply the counted rate separately. Numeric wavepacket/jitter quadrature checks the Gaussian overlap identity.

25.9 Scope of competitive comparison

These profiles make PF Simulation's coverage inspectable in several protocol and component areas described by quantum-network simulators. They do not implement every advertised feature of BQP, Aliro, NetSquid, SeQUeNCe or other products. A feature name alone does not specify a shared mathematical problem, implementation, parameter domain or accuracy target. Proprietary algorithm performance, raw vendor test data, pricing and full platform access are not established by this release.

For a defensible head-to-head comparison, fix the problem and input state, channels, topology, protocol, success convention, numerical tolerances and hardware resources. Obtain authorized numerical exports with versions and logs. Compare common observables with a stated independent reference. A runtime ratio for a different task is not a quantum-link accuracy result. Agreement between two simulators is useful cross-validation but not a replacement for measured apparatus data.

The supported description is *configurable quantum protocol and component benchmarks with explicit numerical references and execution evidence*. Quantum hardware execution, universal industry-standard compliance, independent product certification, empirical PF superiority and complete commercial-platform parity are not claimed.

25.10 Primary references and source register

The dated product and scientific-source register is `COMPETITIVE_BENCHMARK_SOURCES.md`. It separates vendor feature statements from primary protocol research and records what each source can support. Useful primary foundations include the following; the engine and configuration remain authoritative for the narrower implemented model.

- Bennett et al., quantum teleportation (1993), doi:10.1103/PhysRevLett.70.1895.
- Zukowski et al., entanglement swapping (1993), doi:10.1103/PhysRevLett.71.4287.
- Bennett et al., recurrence entanglement purification (1996), doi:10.1103/PhysRevLett.76.722.
- Horodecki et al., the two-qubit CHSH criterion (1995), doi:10.1016/0375-9601(95)00214-N.
- Hong, Ou and Mandel, two-photon interference (1987), doi:10.1103/PhysRevLett.59.2044.
- Nielsen, average gate fidelity (2002), doi:10.1016/S0375-9601(02)01272-0.

Chapter 26

Mission quantum experiments and physical apparatus

Edition 1.11.0 separates three choices that previously shared a workflow menu: what the study asks, where the apparatus is located, and which quantum experiments it executes. The standalone benchmark runner remains available. Mission experiments are an explicit additional calculation enabled on a mission; they do not silently import a trajectory into a standalone reference experiment.

26.1 Study purpose, topology and experiments

Study purpose selects the calculation and evidence question. **Physical topology** selects the mission's node arrangement. For a published reference or standalone experiment, the same interface instead shows **Reference applicability** or **Apparatus scope**. **Quantum experiments** selects the enabled mission modules. **Starting configuration** remains an editable preset, not an assertion that its hardware has been measured or that a physical mission has been validated.

Study purpose	Physical interpretation
Simulation	One declared mission: trajectories, photon reception, states, optical collection, timing and optional QKD/quantum experiments.
Orbital mechanics	The same mission under matched SGP4 and numerical orbital predictions, with downstream result differences.
Research precision	Mission predictions with declared input uncertainty, requirements and supplied observation comparisons.
Risk audit	Requirements, uncertainty, evidence and risk assessment for the configured mission.
Published benchmarks	Particular SatQuMA numerical, BBM92 laboratory and Jinan-1 satellite reference comparisons. Each retains its own applicability.
Quantum benchmarks	Standalone component/protocol apparatus with explicitly configured state, channel, timing or mode parameters.
Micius reference	Selected published entanglement-distribution observables and a controlled comparison.
Micius integration	A controlled engine-consistency fixture based on the Micius reference, rather than an arbitrary mission geometry.

An inter-satellite mission places receivers on satellite worldlines. A satellite–ground mission includes configured ground receivers; a mixed receiver arrangement can retain one satellite endpoint. These geometric roles do not, by themselves, install a memory, second source, joint analyzer or classical feed-forward system. The **Scenario and apparatus** card identifies those roles, the selected module prerequisites, model-derived quantities and declared apparatus assumptions. Nonmission workflows do not expose the live-tracking shortcut or orbit viewer as if they were sources of their calculations.

26.1.1 Simulation versus matched SGP4 comparison

Simulation asks what a declared mission predicts. The orbit-comparison workflow starts SGP4 and a numerical orbital model from matching position and velocity at one epoch, evaluates them at common times, and runs the mission calculation for both. It asks how changing the orbital predictor changes the geometry and the

downstream timing, state, collection and configured protocol outputs. SGP4 is a predictive reference, not measured spacecraft truth. Newtonian central-gravity/J2/drag choices are separate from the SR/static-GR photon and clock options. Enabling static-GR photon timing does not change the numerical orbital force law into a general-relativistic orbit integrator.

26.1.2 Qiskit is a representation, not a location

A qubit register can represent photons at different satellites, a stationary memory or a local protocol input. Storing the joint state in one classical simulation does not place that physical apparatus inside one quantum computer. The package uses Qiskit state/channel operations and Aer where applicable, plus explicit numerical orbital, optical, density-matrix, stochastic and security calculations. A simulated circuit is not a submission to quantum hardware. The physical scenario follows from the declared nodes, operations and event history, not the choice of mathematical library.

26.2 Preserve the complete received state

The mission interface carries the full complex density matrix. It does not replace that matrix with a depolarized Bell or Werner-type state reconstructed from a scalar fidelity. Two states can have the same fidelity while having different local populations, phase and two-qubit correlations. Subsequent measurement branches, entanglement diagnostics and protocols can therefore produce different results.

For a concrete example, in the ordered basis $|00\rangle, |01\rangle, |10\rangle, |11\rangle$,

$$\rho_a = |00\rangle\langle 00|, \qquad \rho_b = \tfrac{1}{2}|\Phi^+\rangle\langle\Phi^+| + \tfrac{1}{6}(I_4 - |\Phi^+\rangle\langle\Phi^+|). \tag{26.1}$$

Both have $\langle\Phi^+|\rho|\Phi^+\rangle = 1/2$, but $\mathrm{Tr}(\rho_a^2) = 1$ and $\mathrm{Tr}(\rho_b^2) = 1/3$. A fidelity-only reconstruction would erase that distinction before the experiment had even executed. This example is algebraic; it is not a plotted or observed mission result.

The serialized state contains separate real and imaginary matrix entries and explicit basis metadata. Protocol qubits use Qiskit's little-endian convention: $|q_1 q_0\rangle$, with q_0 Alice and q_1 Bob. The full-state protocol API validates normalization, Hermiticity and positivity; it does not silently normalize, repair, twirl or fit an imported state. Operations that change the state, including optional purification twirling, are explicit model operations.

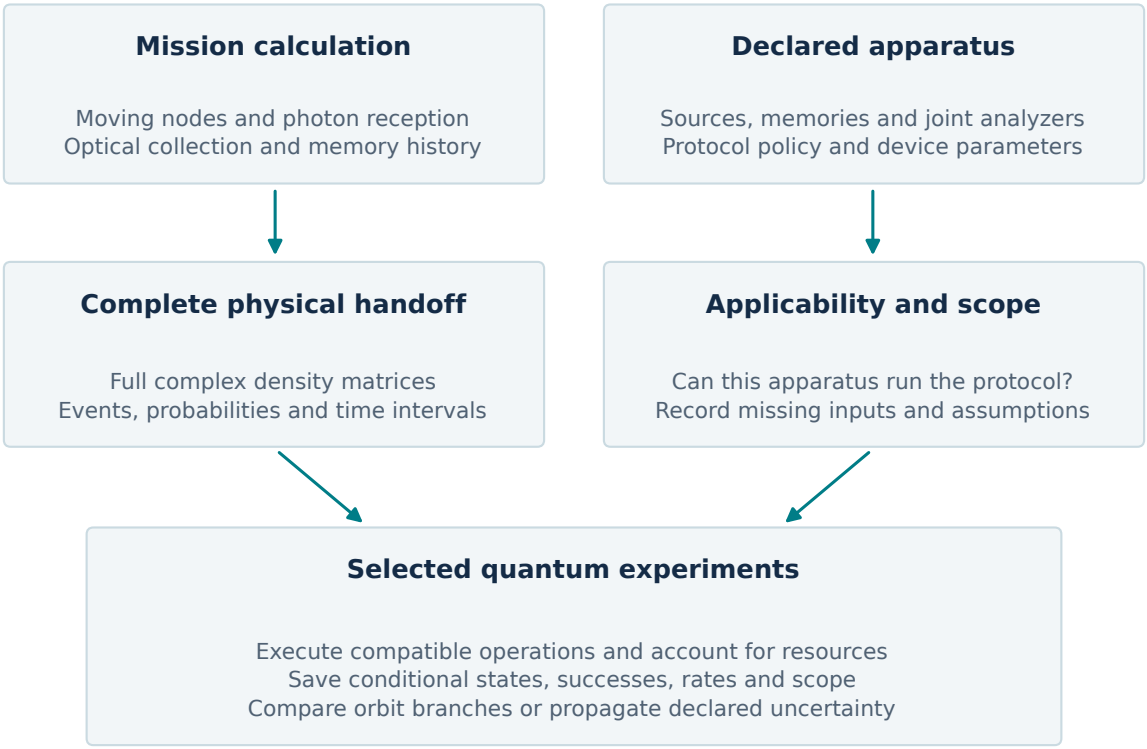

Figure 28: Conceptual interface between the mission calculation and selected quantum experiments. The complete state and physical event history enter with declared apparatus assumptions. Applicability checks precede interpretation of the results. This figure contains no measured or simulated numerical data.

The programmatic entry point for the Bell, teleportation, swapping and purification state operations is:

```
from pf_benchmarks.protocols import run_state_protocol
result = run_state_protocol(name, rho, config=None, second_state=None)
```

It accepts a complex matrix or the documented real/imaginary JSON representation. Swapping and purification require an actual second pair state. Taking a tensor product declares independence of the two input pairs; a general correlated four-qubit resource is outside this API. Purification executes one recurrence round unless a caller supplies and schedules further resources. Teleportation can use an explicit input density matrix or six Pauli probe states; partial Bell-measurement acceptance can depend on that input and the fidelity is reported with its heralding convention.

Bell and memory fidelity use the mission source target. Teleportation and network instruments use the standard Φ^+ correction convention; a singlet resource is not silently converted. A different convention needs an explicit physical correction model. Standalone fidelity sweeps retain their declared source assumptions, but scalar source parameters cannot replace received states in the complete-state mission protocol entry point.

26.3 Understand the nine physical apparatus models

Experiment	Underlying apparatus	Mission interpretation
Bell/CHSH	Two parties measure a distributed entangled pair.	Evaluate the received two-qubit state and retain availability separately from conditional correlations.
Teleportation	Input qubit, shared pair, sender joint measurement and receiver correction.	Declare input preparation, analyzer and receiver memory; account for classical notification and storage.
Swapping	Two elementary pairs meet at a central joint-measurement node.	A second physical source/link resource and central apparatus must exist. One emitted pair split between two receivers is not two pairs.
Purification	Two shared pairs, local operations, measurements and communication.	Pair consumption, waiting and acceptance belong to the resource budget; higher fidelity alone is insufficient.
Memory	Local storage channels and a retrieval process.	Evolve the actual mission resource for declared holding intervals; do not apply emitter storage noise twice.
Two-link repeater	Elementary-link attempts, memories, joint measurement and notification.	Use the declared scheduling approximation and link histories; this remains a bounded two-link model.
HOM interference	Two photons arrive at the same beam splitter.	Require a common receiving node and compatible modes. Spatially separated detections cannot interfere merely because a frame makes their times equal.
QKD sweep	The standalone sweep uses a prepare-and-measure decoy-state source.	The mission module preserves its BBM92 or decoy-BB84 architecture and actual detection history.
Channel characterization	Known inputs probe a specified channel.	Characterize the declared mission two-qubit channel from its model; a single received pair state cannot identify an arbitrary channel.

Selecting all nine experiments does not create the hardware or resources they require. Missing prerequisites produce explicit applicability explanations. Geometric visibility, successful resource delivery, a valid conditional state, numerical verification and empirical validation are distinct questions.

26.4 Enable and configure mission modules

In a mission study, select **Run selected experiments on this mission**, choose modules and use **Declare installed apparatus**. Defaults are an inactive experiment layer and false apparatus flags. The complete `mission_quantum` object remains editable. This fragment belongs inside a complete mission configuration and enables actual-state Bell characterization:

```
"mission_quantum": {
  "enabled": true,
  "modules": ["bell_chsh"],
  "apparatus": {"bell_analyzers": true}
}
```

It does not replace the trajectory, source or detector configuration. A disabled section preserves the original mission behavior.

Field	Default	Meaning
`enabled`	false	Explicitly enable mission experiments.
`modules`	All nine	Selected IDs, subject to prerequisites.
`collection_model`	geometry	Supplied geometry efficiencies, or explicit optical capture model.
`exposure_s`	null	Exposure per independent snapshot; otherwise midpoint cells span the mission interval.
`seed`	20260920	Resource-scheduling seed.
`max_events`	100000	Event budget; exhaustion fails rather than silently truncating.
`apparatus`	All false	Equipment declarations, not calibration records.
`memory.t1_s`, `memory.t2_s`	null	No damping/dephasing in that declared term. Finite T_1 requires finite $T_2 \leq 2T_1$.
`memory.hold_s`	0	Additional coordinate hold; proper duration is integrated along the worldline.
`memory.retrieval_efficiency`	1	Retrieval survival per stored qubit.
`classical.speed_m_s`	c	Positive signal speed no greater than c.
`classical.processing_delay_s`	0	Apparatus processing delay.
`classical.route_latency_s`	null	Compute direct route; a value declares externally routed latency.
`secondary_link.config`	null	Complete second mission configuration.
`secondary_link.independent_source`	false	Declare independent elementary sources.
`protocol_options`	Empty	Module-specific options; unsupported options fail.

Module IDs are `channel_tomography`, `bell_chsh`, `teleportation`, `entanglement_swapping`, `purification`, `memory_decay`, `repeater_chain`, `hom_interference` and `qkd_sweep`. Apparatus flags are `bell_analyzers`, `teleportation`, `memories`, `swapping`, `purification`, `hom`, `channel_probes` and `heralded_capture`. The last declares nondestructive heralded loading for memory protocols. Destructive photon detection cannot leave the same photon available for storage.

26.4.1 Mission channel and optical diagnostics

The mission channel diagnostic applies the executed two-qubit map to 16 operator probes, exporting its 16×16 Pauli-transfer and Choi matrices. It checks trace preservation and agreement with the mission state, using the executed source depolarization, correlated Gaussian phase, deterministic phase and emitter-storage metadata. It is exact model characterization, not measured process tomography. Optical erasure remains separate from the conditional map.

HOM options are `sigma_a_s`, `sigma_b_s`, `detuning_hz`, `relative_delay_jitter_s`, `mode_visibility` and `reflectivity`. Default detuning/jitter are zero, visibility is one and reflectivity is one half; absent widths come from the mission photon model. The complete incoming polarization state contributes $\mathrm{Tr}(\mathrm{SWAP}\,\rho)$, preserving symmetric/antisymmetric behavior. Polarization and temporal sectors are assumed factorizable.

Both input paths must reach one declared beam-splitter receiver worldline. The actual arrival difference and received Gaussian widths use the same calibrated coordinate-equivalent timebase. Detector readout jitter is not optical jitter, and Doppler compensation is not inferred. Counts are before an electronic output gate; the input-path gate is not reused as an output-port gate.

26.4.2 Mission QKD histories

The QKD module reuses executed detection statistics for the configured BBM92 or decoy-BB84 protocol. BBM92 retains the complete detected state including modeled accidentals; decoy BB84 retains its independent pulsed source. Optical losses come from the actual mission links. The optional `protocol_options.qkd_swe ep.security_overrides` array scans conservative security assumptions on those same counts; count mode

cannot change. SatQuMA parameters belong in `qkd.satquma_security`. The combined mission forms one security block: do not reconstruct it by summing independently penalized per-epoch keys. The scan fabricates neither new counts nor a constant-loss experiment.

26.5 Time and resource accounting

Endpoint storage acts on the actual received state after its configured emitter-storage/propagation effects, without repeating those channels. Proper holding duration is integrated along the actual memory worldline. A saved matrix alone cannot reconstruct missing worldline or clock history. Classical readiness and feed-forward use computed moving-node routes and visibility. A declared external routed latency is checked against causal lower bounds, but its relay geometry is not solved or independently established. Teleportation's independent-pair throughput assumes sufficient memory slots; it is not a finite-capacity queue model.

States and optical transmission are piecewise constant within snapshot cells. With two or more epochs, midpoint cells span exactly the sampled interval; no extra final exposure is appended. A single epoch needs explicit exposure for integrated counts. Explicit `exposure_s` applies per independent snapshot; overlapping exposures are not disjoint pass intervals.

Each selected module is a separate experiment on the mission ensemble. Multiple selections do not clone one physical pair across incompatible measurements. Their totals, or totals from alternative scenarios, cannot be summed as one jointly scheduled experiment. Conditional quality, availability, operation acceptance, retrieval survival and output rate remain separate. A high conditional fidelity can coexist with zero usable events.

26.5.1 Two-resource scheduling

Swapping and the two-link repeater require primary `detector2` and secondary `detector1` to share the relay worldline. Purification requires both pairs to share both endpoint worldlines. The independently executed histories need matching epoch grids, UTC epochs and spacetime. Nested secondary experiments remain disabled; a copied fidelity is not a second resource.

One pair slot per link and serial Bernoulli attempts define the scheduler. The period respects both source period and loading acknowledgements. Physical photon paths are retraced at actual serial emissions. Timing gates use those events when the required apparatus metadata is available; the source is idle in zero-capture cells. Loading and release acknowledgements reach each source before its next applicable attempt. Purification sends a readiness trigger before the bilateral operation. Resources age while waiting for the other link and for classical messages. Operations consume two pairs, including failures; unused captured pairs and emission attempts are reported.

The output rate from summed conditional success probabilities is conditional on one sampled resource history, not an unreported ensemble mean. Sampled successes and sampled rate are separate. Event state predictions may be present for a sampled failed operation. Epoch quality uses success-probability weights; no usable opportunity leaves it unavailable. Event-budget exhaustion raises an error; overlapping explicit exposures are rejected for this scheduler.

`entanglement_swapping` and `repeater_chain` use the same bounded serial two-link scheduler and swap operation with separate experiment seeds. They are not different routing architectures. Purification executes one recurrence round. General multihop routing, multiplexing, arbitrary contention and automatic multiround distillation are outside this model.

26.6 Saved evidence and analysis axes

Enabled scenarios save `scenarioN_mission_quantum.json`, `scenarioN_mission_quantum_epochs.csv` and `scenarioN_mission_quantum_metrics.csv`. With plotting enabled they also save `scenarioN_mission_quantum_report.pdf` and module PNG previews. The JSON retains states, assumptions and source-result identity. CSV contains nested JSON records and uses `\N` for missing values. Plots show missing outcomes as gaps, not zero.

Status	Meaning
COMPLETED	The declared calculation produced results; inspect epoch coverage and reasons.
NOT_APPLICABLE	Topology or apparatus cannot perform the selected experiment.
UNAVAILABLE	Required state, events, route or another outcome could not be obtained.

Execution status is not empirical validation. New mission experiment provenance records empirical NOT_ASSESSED. Invalid inputs fail explicitly. A valid zero-event calculation is distinct from an unavailable state; neither is replaced with fabricated success.

Matched-orbit differences require corresponding available module/epoch results. A secondary mission in that comparison must explicitly use SGP4, the same spacetime and its own matched/frozen orbital inputs. Research precision retains full states/provenance for nominal and ensemble runs. Epoch metrics use mission_quantum.<module>.<metric>_mean; resource summaries use mission_quantum.<module>.<summaryfield>. Means require complete coverage; unresolved epochs are not silently averaged away. Requirements and risk review can use these axes without turning model predictions into observations. S2/S3 remain alternative descriptions of their shared physical history, not additional exposure or an automatic scheduling advantage.

See MISSION_QUANTUM_GUIDE.md for module-specific operating details and current runnable example configurations. The example values are declared research assumptions, not calibrated hardware parameters.

26.7 Optical capture choices and runnable examples

collection_model:geometry uses supplied scalar collection efficiencies, mission gates and emitter-memory survival. The alternative optical model uses both explicitly configured qkd.optical_links arms evaluated on actual mission rays; QKD itself can remain disabled. Local polarization rotations act on the complete state. Optical efficiency replaces the geometric collection factors rather than multiplying them twice. Background clicks are not invented as stored entangled resources; QKD retains its separate detection and accidental-count model.

The optical resource adapter requires zero dead_time_s in both arms. Its capture model does not include detector dead time, separately from the network scheduler's finite memory slots. Direct moving-node classical null routes support flat SR and static GR. Static-GR signaling below c requires a declared external route latency; its subluminal trajectory is not integrated.

The following full-package configurations supply runnable examples:

- config_mission_quantum_protocols.json, config_mission_quantum_network.json and config_mission_quantum_purification.json: single-pair operations, relay swapping/repeater experiments and same-endpoint purification.
- config_mission_quantum_comparison.json: explicitly matched SGP4/numerical comparison, including an independent secondary pair source.
- config_mission_quantum_optics_hom.json: common receiver worldline, complete-state HOM and mission channel characterization.
- config_mission_quantum_optics_qkd.json: mission downlink and actual configured QKD security model.
- config_mission_quantum_ground.json and config_mission_quantum_ground_gr.json: satellite distribution to two ground nodes with complex pair state, optical capture, Bell, channel, teleportation and memory experiments in SR or static GR.

The ground examples declare a 0.01 s external routed communication delay; they do not reconstruct a ground-fiber route. Example apparatus parameters are illustrative inputs, not calibrated hardware values. Run each complete configuration through its matching study purpose; see MISSION_QUANTUM_GUIDE.md for command equivalents and result inspection.

Chapter 27

Standalone and hosted SaaS editions

The hosted edition adds accounts, organizations, projects, managed inputs and durable job records around the existing scientific engines. The standalone launcher and command-line runners remain available without a hosted account. This chapter explains the user-facing distinction. Installation, environment variables and AWS infrastructure belong to `SAAS_DEPLOYMENT.md`; the operating walkthrough is `SAAS_USER_GUIDE.md`.

27.1 One scientific application, two execution modes

Area	Standalone	Hosted SaaS
Access	Local launcher and loopback browser service.	Authenticated account and organization membership.
Scientific calculation	Existing local Python runners.	The same runners in a worker execution environment.
Inputs	Local configuration and referenced files.	Authorized project uploads and selected packaged data.
Saved work	Local run folders.	Project-associated jobs and private stored artifacts.
Execution control	Local queue and process lifetime.	Durable status, worker claims, cancellation and configured budgets.
Commercial policy	No SaaS account required for the standalone workflow.	Operator-configured entitlements and usage records.
Deployment	Supported local Python installation.	Separate web service, database, storage and workers.

The deployment mode does not select a new physical model. The worker receives the declared effective configuration and invokes the allowed scientific runner. Complex density matrices, basis metadata, phases, correlations, seeds and mission apparatus assumptions remain part of the calculation. Passing through a queue or private object store is not a reason to reduce the state to one fidelity number.

Matching configuration, inputs and runtime permit numerical comparison between editions. Different dependency versions, thread arrangements or numerical libraries can produce justified floating-point differences. Compare recorded quantities with a stated tolerance; do not infer physical equivalence merely from a matching screenshot or require byte-identical PDFs from different hosts.

27.2 Organizations, projects and access

Sign in to the hosted application and select the intended organization and project before opening the workbench. A project groups its uploaded inputs, job records and saved results. Organization membership determines access to its projects; a project name alone does not create an additional membership boundary inside the same organization.

An owner administers the organization. Administrative, editing and viewing roles give progressively narrower access. Use a viewing role for a colleague who should inspect results without starting calculations or uploading inputs. Invitations and account recovery follow the configured identity and email service. A development email outbox records a message locally; it does not prove that an external mailbox received the message.

Sharing a job identifier or artifact address does not itself grant access. The recipient must have the required membership and project context. Public sharing and anonymous result publication are separate features and

are not created by an ordinary download link. Export a deliberate run bundle when a reviewer needs a portable copy outside the hosted application.

27.3 Managed input files and reproducibility

Upload a file to the intended project and use its returned asset reference in the configuration. The hosted edition distinguishes `asset://` project references from `bundle://` references to allowed application data. Neither spelling authorizes arbitrary server files. A local Windows path or absolute Linux path from a desktop configuration cannot identify a hosted input; upload the corresponding data and replace the reference explicitly.

This applies to primary and secondary mission tracking caches, Earth-orientation and leap-second data, benchmark reference tables, measurement CSV files, assurance evidence and replay inputs. Directory-shaped inputs use the supported bounded archive mechanism. Preserve the documented relative layout inside an archive, and do not supply symbolic links or parent-directory paths.

An imported configuration is editable scientific data, not executable code. The service does not accept a browser-selected Python interpreter, shell command or arbitrary script. Online orbit acquisition uses the service's approved data-source policy. Offline replay must use captured inputs and must not silently refresh them to make a historical comparison appear current.

Keep the saved request, effective configuration, input hashes, runtime identity, random seeds and result manifests with a study. An uploaded evidence document retains its declared provenance. Storage integrity does not authenticate its issuer or establish the independence of a calibration or validation source.

27.4 Run and inspect a study

The hosted workbench retains the existing study-purpose, topology and quantum experiment controls. Selecting all nine quantum modules still does not install their required apparatus. The mission applicability rules in Chapter 26 apply in both editions.

1. Select an organization and project, then open the workbench.
2. Choose the study purpose and starting configuration. Review the physical topology, apparatus and full configuration.
3. Attach any required project assets. Validate the configuration and review the service's resource limits before submitting the job.
4. Inspect the queued/running status and available log. A closed browser tab does not cancel a hosted job.
5. Open the completed results, numerical checks, figures and source records. Download the run bundle when a portable record is required.

Execution completion, numerical consistency, empirical agreement and research readiness retain separate meanings. A finished job can correctly report zero key generation, a physically blocked link, an inapplicable module or inadequate empirical evidence. Hosted execution does not convert those results into a successful physical mission.

Cancellation and interruption preserve their status. An incomplete artifact is not a completed scientific result. A retry must have explicit attempt history; it must not append unlabelled samples to an earlier calculation or replace a failed physical trial with a favorable outcome.

27.5 Service budgets are not model parameters

The operator configures queue capacity, concurrency, runtime, storage and upload limits. Usage accounting supports service administration and subscription entitlements; it is distinct from photon counts, memory time or quantum resource consumption in a scientific report.

If a requested calculation exceeds a service limit, the application rejects, queues or terminates it with an explicit reason. A resource policy must not silently shorten a mission, reduce sampling, omit a selected module, replace the full state with a proxy or fabricate a completed result. A user may choose a smaller study, but that change belongs in a new declared configuration.

Billing requires an operator-configured provider and valid service credentials. A disabled payment integration does not imply that a subscription payment was accepted. Development plans and webhook test cases are implementation evidence, not a record of a real purchase or a live payment-provider certification.

27.6 Local testing and production operation

The local SaaS environment permits account, project, storage, queue and actual scientific-job testing without public hosting credentials. It is separate from the single-user desktop launcher. Production operation requires the documented identity, database, storage, TLS, worker and secret configuration.

The deployment package includes an AWS target, but infrastructure definitions are not evidence that a cloud account has been provisioned. Real AWS, external identity, email delivery and payment behavior must be verified in the intended deployment. Local tests, service doubles and infrastructure syntax checks have their own recorded scope. They do not establish public availability, incident recovery, load capacity, operating cost or a security certification.

The release validation records distinguish executed local studies from configured external integrations. Existing scientific validation limits remain in force: no SaaS feature establishes an empirical PF advantage, calibrated quantum hardware performance, operational cryptographic keys or mission qualification.

Chapter 28

Research data, study design and independent validation

Release r22 addresses the immediate data-interchange, sampling and evidence limitations identified in the r21 academic assessment. The changes improve what a researcher can configure, preserve and independently inspect. They do not create measured acquisitions, prove a PF advantage, authenticate a dataset's provenance or establish market superiority. The existing standalone runners and separate hosted application use the same scientific implementation.

28.1 Choose the evidence question before the workflow

Question	Starting point	Meaning of a successful calculation
Does an imported quantum state survive the mission handoff?	`config_mission_quantum_full_state.json`	Full-state propagation and numerical-reference agreement within the declared source/channel model.
Are conditional predictions stable to sampling and epoch resolution?	`config_precision_research_299.json`	Prespecified numerical checks and finite-sample capability, without empirical coverage.
Are decision margins supported by complete evidence?	`config_risk_audit_research_299.json`	A scoped risk/evidence assessment; missing calibration or observations can still block readiness.
How do heterogeneous memories and resource availability affect a two-link protocol?	`config_mission_quantum_network_research.json`	Sampled results for the supported serial apparatus, with resource history and sampling limits.
Does a frozen downlink model predict independent acquisitions?	`config_research_validation_import.json`	A per-acquisition comparison under declared provenance and uncertainty assumptions.

The historical quick precision presets remain available and are explicitly labelled demonstrations. The word "research" in an older filename does not make sixteen draws sufficient for a one-percent probability decision. The synthetic research-validation demonstration exercises the workflow and is always blocked for empirical claims.

28.2 Import full quantum states and local channels

28.2.1 Explicit subsystem order closes the memory interchange defect

The r21 standalone memory exporter used left-to-right tensor order A, B, whereas mission protocols interpreted B, A. Fidelity and purity can be identical under that permutation, so neither scalar detects a swapped physical arm. New standalone memory export rows use B, A and carry explicit `state_metadata`. The historical Python `pf_benchmarks.network.memory_state()` default remains A, B for existing callers; request `subsystem_order=('B','A')` for a mission matrix. Historical files are not silently relabelled.

The version-one `pf.quantum.state` document contains the full real and imaginary 4×4 matrix, tensor-factor order, dimensions $[2, 2]$, basis labels, conditioning and provenance. The rightmost factor varies fastest. An

explicit A, B document is converted using permutation $[0, 2, 1, 3]$ and the conversion is recorded; missing or ambiguous order is rejected. Inputs must be finite, Hermitian, unit-trace and positive semidefinite within the stated numerical tolerance 10^{-10}. The importer does not normalize, clip, project, replace with a Werner state, or reconstruct a matrix from scalar fidelity.

28.2.2 Configure the source and maps

Place the state document inline at `quantum_channel.source_state`. Explicitly set `source_depolarizing`: use zero when an additional source-depolarizing map is not intended. `bell_state` then identifies the fidelity reference, not a replacement preparation. Provenance identifies an input as theoretical, simulated or a measured reconstruction. Propagating any of these inputs produces a simulated prediction; a measured-input declaration is not independently authenticated by PF.

Optional `quantum_channel.local_channels.a` and `.b` accept version-one `pf.quantum.local_kraus_channel` documents. Each identifies its physical subsystem and complex 2×2 Kraus operators. The check $\sum_j K_j^\dagger K_j = I$ enforces trace preservation within 10^{-10}; loss/erasure belongs to the separate resource model. At most 256 operators are accepted as a resource bound. Use the helper functions to create correctly labelled documents:

```
from quantum_data_contract import export_state, export_local_channel
source = export_state(rho_BA, provenance={
    "data_kind": "measured_reconstruction",
    "source_id": "your dataset DOI and reconstruction identifier"})
channel = export_local_channel(kraus_A, "A", provenance={
    "data_kind": "measured_reconstruction",
    "source_id": "your process reconstruction identifier"})
config["quantum_channel"].update(
    source_state=source, source_depolarizing=0.,
    local_channels={"a": channel})
```

The actual Aer density-matrix circuit executes source preparation, source depolarization, imported local maps, correlated Gaussian polarization phases, fixed A-arm phase and the configured memory dephasing. A separate matrix/characteristic-function calculation checks general-source results. For these runs, Bell-family analytic fields are null; inspect `reference_kind`, `reference_fidelity`, `reference_purity` and `density_matrix_reference_error`. Shared imported maps are inputs to both implementations, so this check does not validate the experimental map itself.

The supplied full-state preset is a satellite/ground model fixture with an asymmetric phase-bearing source and local map, not measured apparatus data:

```
python run_privileged_frame_simulation.py --scenario 2 \
  --config config_mission_quantum_full_state.json \
  --output full_state_run --no-sampling --no-controls
```

The source and maps remain fixed across mission epochs. This extension does not supply time-dependent process tomography, non-Markovian baths, a multiphoton source or a source-calibration fit. The complete field contract and legacy conversion procedure are in `R22_STATE_CONTRACT.md`.

28.3 Use the rate for the physical population being compared

The scenario `research_data.rate_products` and precision exports now record the quantity, unit, population, conditioning, denominator, time coordinate, provenance and availability. The following quantities must not be interchanged simply because an idealized preset produces equal values.

Quantity	Interpretation
`controlled_pair_rate_hz`	Base collection expectation, including its configured gate, memory erasure and occlusion; it does not automatically include the separate QKD optical budget.
`optical_signal_pair_rate_hz`	True detected entangled-pair expectation from the QKD optical/count model.
`detected_total_coincidence_rate_hz`	Expected true pairs plus expected accidental coincidences, before basis sifting.
`sampled_detected_coincidence_rate_hz`	Actually sampled Monte Carlo coincidence count divided by positive modeled epoch exposure; not measured counts.
`detected_pulse_rate_hz`	Weak-coherent-pulse detection expectation; not an entangled-pair coincidence rate.
`secret_key_rate_hz`	Protocol security-bound output per security-block exposure; not operational key delivery.

The legacy `expected_accepted_pair_rate_hz` is retained with its explicit controlled-base meaning. Mean-rate metrics carry corresponding names in precision results. Null means unavailable or inapplicable; zero is a calculated or sampled zero. Zero exposure supplies no sampled rate. Do not sum repeated pair statistics across the two arm rows of a key-epoch table, and do not add alternative scenarios as if they were disjoint key blocks.

28.4 Run a prespecified numerical study

The new precision and assurance research presets use 299 independent whole input draws for each of three declared seeds and nominal epoch grids of 3, 5 and 9 points. Their three scenarios require 8,124 scenario-epochs. The primary seed supplies the fixed-sample risk calculation; other seeds diagnose stability and are not pooled into a more favorable bound.

```
python run_precision_analysis.py \
  --config config_precision_research_299.json --output study_299
python run_risk_audit.py \
  --config config_risk_audit_research_299.json --output audit_299
```

For a predeclared sample of N IID whole draws and zero exceedances, the exact one-sided upper bound at confidence c is

$$p_{\text{upper}} = 1 - (1 - c)^{1/N}.$$

At 95% confidence, a bound at or below 1% requires at least 299 draws. Sixteen zero-exceedance draws instead give about 17.1%. This is a *capability* calculation: observed failures, incomplete draws or an unavailable metric can prevent the actual requirement from passing. The probability describes the supplied model-input distribution, not real mission risk. Current example distributions are explicitly uncalibrated.

Declare `precision.study_design` before execution. Its metrics specify scenario, metric, unit, absolute tolerance and relative tolerance. All seed pairs are checked on mean, standard deviation and 2.5/50/97.5 percentiles. Adjacent nominal grids are executed by the engine and compared at the same start/end times. The example screens Scenario 2 fidelity and purity at one absolute percentage point and maximum arrival gap at 100 ns plus 1% relative tolerance. These are visible example numerical criteria, not experimentally justified hardware tolerances or industry standards. Failed criteria are not relaxed to create a favorable report.

`study_design_report.json` separates sampling capability, seed stability, nominal-grid stability and selected covariance support. Additional seeds are saved under `study/seed_<seed>.json`; grid metrics and full additional nominal scenario records are also saved. A complete process can produce an insufficient design or failed scientific criterion. Nominal arithmetic-grid sensitivity is not adaptive solver convergence or convergence of the complete uncertainty distribution.

Covariance diagnostics normalize varying outputs before SVD and distinguish the sample cap $N - 1$ from further dependence among outputs. Exact constants and supported singular covariance are disclosed; no

artificial diagonal noise is inserted. The selected-axis screening rule defaults to ten draws per nonconstant variable and is not a covariance-accuracy theorem. Full rank is optional because derived observables can be dependent. Residual components outside a declared covariance's support remain incompatible rather than being silently discarded by a pseudoinverse.

Epistemic inputs, aleatory inputs, Monte Carlo sampling and nominal grid sensitivity are separately labelled. Measurement uncertainty and model discrepancy are not supplied by a larger ensemble; discrepancy remains unquantified unless separate evidence supports a model for it. `R22_STUDY_DESIGN.md` describes the full configuration and interpretation.

28.5 Trace resources in the two-link network model

`config_mission_quantum_network_research.json` adds heterogeneous T1/T2 and retrieval efficiencies for the four stored qubits, optional attempted emissions, stable resource identifiers, buffer/disposition records and five seeds in total. Primary arm B and secondary arm A share the relay worldline for swapping. Missing memory settings inherit the existing defaults; explicit null T1/T2 disables the corresponding component subject to the stated complete-positivity constraint.

The optional `max_ready_age_s` rejects a pair of resources when an operation would start. Both then receive physical release messages before new generation. This is a ready-to-operate admission cutoff, not an autonomous memory TTL; buffers remain occupied through the decision and release. Attempt logging preserves failed signal-capture trials and does not change the random sequence. An `output_resource_id` exists only for a sampled successful operation. A conditional matrix associated with a failed operation describes the counterfactual accepted ensemble, not a delivered pair.

The standalone repeater adds per-qubit T2, consumed-attempt history and state-record references. Its existing elementary state family and local Markovian dephasing assumptions remain. Mission protocols continue to use full complex matrices. Repeated-seed rate intervals and renewal-reward rate intervals are approximate; exact cycle-success intervals are not exact pairs-per-second intervals because elapsed time is random. Rare zero-delivery runs remain insufficient. Inspect `R22_NETWORK_GUIDE.md` for minimum-event reporting rules, resource limits and the separation of sampled states, accepted ensembles and unavailable fidelity.

These extensions retain a serial two-link apparatus with one pair slot per link. They do not add arbitrary routed/multiplexed networks, shared resources across independently selected modules, detector false-herald or multipair physics. Additional lifecycle data make the supported model inspectable; they do not change those physical boundaries.

28.6 Freeze a model before independent acquisition validation

The new research-validation workflow evaluates the existing Gaussian-aperture weak-coherent-pulse downlink count/QBER model. Imported range and operating histories are exogenous inputs. Observed counts are never substituted for model predictions. This adapter does not forecast future weather or orbit truth, calculate a finite secret key, or simulate an arbitrary network.

28.6.1 Prepare an acquisition manifest

Start with `config_research_validation_import.json`, which intentionally has no selected dataset. The self-contained JSON manifest declares independent acquisition/session/pass identities, apparatus/calibration identity, calibration or validation role, measured/synthetic provenance, source URI, license, collector, UTC windows, raw-source hash and processing. Each acquisition's observation hash is mandatory. Optional complete-file hashing binds the manifest bytes; hashes do not authenticate the collector.

Use seconds, metres, hertz, counts and QBER as a fraction. Unknown values and unknown uncertainties remain null. Each observation has contiguous range and operating-state segments. The supported states are transmitting, shutter closed and detector off. Detector exposure is the sum of enabled segment lengths and is never inferred from the observed count. Predicted QBER is accepted-photon-rate weighted across the exact row support. The existing sparse deadtime approximation rejects occupancy at or above 0.01.

Count observables must match the declared pre-gate or accepted contract. QBER uses the supported intensity-independent sifted sample; incompatible postselection is not relabelled for import. A doubled-resolution calculation reports count and QBER refinement and gates the acquisition using explicit tolerances. This remains a numerical diagnostic, not a physical error bound.

The optional CSV converter accepts single-segment observations and records the CSV hash. It does not infer missing provenance or uncertainty:

```
python -m pf_research.import_csv --manifest my_manifest.json \
  --acquisition-id pass-2027-01 --csv my_counts.csv \
  --output manifest_with_counts.json
```

For full schema, supported columns and multi-segment histories, use `RESEARCH_VALIDATION_GUIDE.md`. Its demonstration observations are synthetic, so copy the schema rather than the observations or provenance.

28.6.2 Prospective calibration and frozen validation

1. Declare apparatus, acquisition identities/windows, observables, parameter bounds, thresholds and an external protocol reference when available. Complete calibration acquisitions; planned validation observations may be empty at this stage with the documented empty-content hash.
2. Run calibration alone and preserve the frozen artifact before collecting or inspecting future validation outcomes:

   ```
   python run_research_validation.py --config my_protocol.json \
     --stage calibrate --output frozen_protocol
   ```

3. Add actual validation observations and hashes to the already declared acquisitions. Update the optional whole-manifest hash, then apply without fitting:

   ```
   python run_research_validation.py --config my_protocol.json \
     --stage validate --frozen-model frozen_protocol/frozen_model.json \
     --output prospective_validation
   ```

4. Review every acquisition and failed gate. Preserve the original frozen artifact, observed data and rejected attempts with independent protocol records.

The frozen identity binds fitted parameters, calibration data, protocol and thresholds, production-model source and relevant runtime versions. Changing these requires a new protocol/model freeze. Prospective gates require the freeze to precede validation acquisition windows and calibration to precede the freeze. Computer timestamps and user declarations are not independently authenticated. Convenience `--stageall` cannot prove that a person had not previously inspected the holdout outcomes.

Checks reject declared same-session leakage, reused passes, overlapping same-apparatus windows and copied observations despite several metadata renamings. A shared raw archive hash is disclosed because one archive may contain multiple passes. False metadata, transformed copies or undocumented support can defeat such checks; independent provenance review remains necessary.

28.6.3 Read the scientific status

Inspect `research_validation_report.json`, its Markdown summary, `predictions.csv` and the plotted observations/predictions. The count relative MAE is mean absolute rate error divided by mean observed rate for each acquisition, not mean absolute percentage error. QBER MAE is a fraction: 0.005 means 0.5 percentage points. No favorable pooling hides a difficult pass.

Marginal residuals within 1.96 reported standard uncertainties are a diagnostic conditional on Gaussian measurement error. They exclude parameter uncertainty, range uncertainty and model discrepancy and are not predictive interval coverage. Unknown uncertainty blocks that diagnostic. Local fit rank does not prove physical identifiability.

`blocked` means a necessary gate remains unmet. `scoped_observational_agreement` means the imported observations meet the declared conditional gates for these acquisitions and this model. It does not authenticate provenance or establish broader quantum-network validity. Synthetic observations always block empirical readiness. The `academic_commercial_claims` status remains `not_established`: independent researcher replication and actual adoption trials are separate. The blank researcher-trial form records task, baseline, resources, analyst time, assistance, failures and reproducibility; it contains no trial outcomes.

28.7 Preserve and verify the research archive

Scenario, precision and standalone benchmark outputs include a versioned research archive in addition to the existing JSON/CSV products. The archive has `manifest.json`, `records.jsonl`, `states.jsonl` and `index.sqlite`. Run/record identities are distinct from content identity; state identity binds both complex entries and their original metadata, so identical arrays with different tensor order are not conflated.

```
python -m research_data verify results/scenario1_research_RUNID
python -m research_data archive --source results/scenario1.json \
  --scenario 1 --output converted_archive
python -m research_data restore converted_archive \
  --output restored_scenario1.json
```

New destinations protect existing evidence. Readers verify inventories, sizes, hashes, referenced states and restored payload identities. Corruption raises an error; missing states are not synthesized. Iterator writing and the disk-backed state index limit archive-writer memory, but do not make an upstream in-memory simulator streaming. Legacy conversion and full scenario restoration intentionally materialize their source/result in memory. `R22_DATA_CONTRACT.md` gives the lazy Python interface and rate semantics.

28.8 Standalone, hosted operation and permitted claims

The workbench exposes the new study, state, network and research-validation controls. Hosted uploads remain project assets rather than arbitrary server paths; states/maps are inline documents. Hosted calibration and validation select the declared stage and frozen-model asset. Repeated seeds and grids count toward the job budget, and an exhausted service budget does not authorize silently simplifying the physical run. Standalone users retain the same command-line procedures without a hosted account.

AWS configuration now makes the Cognito tier explicit when security-audit mode is enabled. Provider credentials, deployment, identity, billing, delivery, capacity and operational recovery still require live verification. See the SaaS guides for current configuration and Chapter 27 for the shared-core model.

Release execution evidence is recorded under `validation_research_r22`. Consult the final manifest and status for the exact executed scope instead of inferring a new result from an older benchmark figure. New data structures, additional modeled histories and software verification strengthen reviewability. They do not replace independent experimental acquisitions, externally repeated studies, researcher usability trials or evidence of commercial adoption.

Chapter 29

Compare saved simulation runs

Release r23 adds a read-only comparison from completed runs in the workbench. Use it to inspect the effects of parameter changes, examine repeat runs, and retain a portable record of the differences. The standalone and hosted editions share the same comparison calculations and presentation. Comparing does not rerun a simulation or overwrite either run's saved inputs and outputs.

29.1 Choose related runs

Open **Runs and live progress**, find a completed reference run in **Run history**, and select **Compare** beside its completion status. The reference is run A. Choose run B from the list of related completed runs, then generate the comparison. Labels, creation times and identifiers help distinguish repeated runs with similar names. In a long hosted history, select **Load older runs** to continue searching earlier pages of history. Close the dialog to return to history; JSON and CSV downloads preserve the comparison for later review.

The grouping rule requires the same saved **Study purpose**, **Physical topology** (or apparatus/reference scope), and **Starting configuration**. The last field is the recorded preset identity. Parameters and run overrides may differ: displaying those changes is one purpose of the feature. Grouping by a common preset is not a claim that the two runs represent the same physical experiment.

Unfinished, failed, cancelled, unrelated or unavailable runs do not provide comparison candidates. A completed process can still contain failed scientific targets; their recorded statuses remain visible. Older runs are supported when their saved requests contain the required grouping fields. Missing provenance is not inferred from numerical similarity. A raw imported configuration is labelled custom; a GUI export can preserve a recognized preset identity.

29.2 Read configuration changes before result changes

The comparison distinguishes identical saved configuration and run options from changed saved inputs. Content hashes identify the compared request data. Identical request data do not establish identical external acquisitions, software versions, live epochs, or random realizations. Inspect changes in spacetime, source state, channel parameters, seeds, sample counts, timing and security assumptions before attributing an output change to a particular cause.

For a recorded numerical metric x, the displayed difference is

$$\Delta x = x_B - x_A.$$

Where $x_A \neq 0$ and the calculation is finite, the percentage is

$$100\,\frac{x_B - x_A}{|x_A|}.$$

A zero reference leaves the percentage unavailable. Missing, ambiguous or nonfinite values remain unavailable; they are never supplied as zero. Units must agree. A numeric-looking string is not silently converted into a measurement. Exact values remain available in exports and numeric tooltips.

Status comparisons retain execution and scientific-evidence distinctions. A favourable numerical difference does not imply a more accurate model, an experimentally supported result, statistical significance or a PF advantage.

29.3 Series, tables and scope

Named series use their saved coordinate and unit labels. Only exact common coordinates are compared; there is no interpolation or matching by row number. Unmatched points remain identified. Index-only axes, duplicate coordinates, ambiguous series identities or incompatible time origins can prevent a series comparison. Overlays and difference plots describe the matched saved points, not a new trajectory calculation or a new experimental acquisition.

Recognized statistical tables use semantic row identities, such as scenario, metric, acquisition and observable. Summary-preview limits remain explicit. The feature compares supported summary metrics, series and tables, rather than all fields in every output artifact. Original artifacts remain available from each run's results.

In particular, equal fidelity, purity or other displayed scalar metrics do not establish equality of full quantum states. Phase and correlations can differ. The comparison does not reconstruct a state from fidelity; inspect the preserved full-state records when the research question depends on matrix elements or subsystem conventions. The state-import and export contracts in Chapter 28 still apply.

29.4 Hosted access and portable records

The hosted candidate list is limited to the currently selected project. Both run identifiers are authorized before comparison data are read. Read-only project members may inspect comparisons, but an identifier by itself grants no access. The report identifies the completed execution attempts used. Deleted outputs, unavailable summaries and unsuccessful attempts cannot supply results. Saved JSON artifacts are checked against the attempt manifest before use.

JSON and CSV exports retain the selected differences and interpretation scope. CSV text that could be interpreted as a spreadsheet formula is escaped. Content hashes help check recorded bytes; they do not authenticate an external source or certify a scientific claim. Read the release validation record for the executed software checks and their limits. Independent measurements and researcher replication remain separate evidence requirements.

Chapter 30

Research integrations and public-source validation

Revision 25 extends the optional **Research integrations** study purpose in both the standalone and hosted workbench. Approved FMI models, recorded Swabian event files, and MATLAB/Simulink worker routing join the shared runner. CSV replay remains available. Each scientific experiment has an identified engine, explicit inputs, detailed outputs, and traceable checks. Selecting a research backend does not silently replace the physics of an older mission.

30.1 Enable the optional research environment

Use Python 3.12 for the research workers. The existing Python 3.8 standalone core remains separate. On Windows, run `setup_research_worker.bat` and then `Launch_PF_Research.bat`. The research launcher opens the same `PF_Simulation_GUI.pyw` in the research environment. A direct double-click on that file follows Windows' Python association and may select a different environment. On Linux or macOS use `setup_research_worker.sh`. Orekit also requires Java 17 or later.

The administrator may select a worker interpreter using `PF_RESEARCH_PYTHON`; scientific configurations cannot choose an executable. MATLAB requires a compatible installed license and Engine API. Swabian binary replay needs the vendor Time Tagger SDK. CSV replay does not require that SDK. The setup script does not supply proprietary software or licenses.

In the workbench choose Research integrations, select a preset, inspect its parameters and launch the study. The local and hosted runners use the same saved scientific configuration and adapters. Each capability reports its configuration and execution outcomes separately. A configured route is not a verified license, a decoded acquisition or a passed numerical experiment.

30.2 Hosted capability workers

A SaaS user's browser does not need a Python installation. The server supplies the research environment. The research Docker image includes the optional scientific libraries and Java; it does not bundle proprietary vendor products. The approved packaged Dahlquist FMU can execute in the default research worker. MATLAB and Swabian binary replay require administrator-controlled profiles.

Set `PF_SAAS_RESEARCH_PROFILES_FILE` to an administrator-owned JSON file. Revision 25 introduced capability keys `fmu`, `matlab`, and `swabian`. Chapter 31 documents the r26 numerical-relativity route. A profile has an explicit revision and uses the same execution backend as the service. The supplied development, Docker and AWS example profiles are under `deploy/saas/`. Complete fields and operational commands are in `R25_DEPLOYMENT_GUIDE.md`.

- The development subprocess route selects an absolute local Python interpreter. It is a same-host development route.
- Docker profiles select a compatible worker image on the Docker host.
- AWS Batch profiles select a capability-specific queue and job definition. The operator must configure images, resources and the dispatcher's permissions.

Profiles do not provision remote infrastructure. No arbitrary remote HTTP execution endpoint is exposed. Clients cannot submit an image, interpreter, license value, function name, executable model or queue as a scientific input. The workbench obtains a sanitized capability summary without exposing the administrator's paths or license settings. Existing project authorization, managed inputs and job state remain in force.

The implemented MATLAB route uses MATLAB Engine with the fixed supplied clock function or Simulink model. A compiled MATLAB Runtime or MATLAB Production Server adapter is not included. The free Runtime alone cannot execute the Engine-based adapter. Test actual license checkout and the required toolbox inside the selected worker before describing the vendor route as operational.

FMI studies accept the approved model and verify its bytes. Arbitrary uploaded native models are not accepted. The worker must provide compatible native libraries and job-specific temporary storage. An unsupported platform or a failed engine is reported explicitly, rather than substituted by the analytic reference used to check it.

30.3 Recorded inputs and exact timestamps

Hosted measurements use project-managed input records. CSV replay remains available; direct Swabian recording replay additionally uses its configured SDK worker. Keep numbered segments in their original numerical order, preserve required segments, and declare the acquisition window, channel map and calibration. Original input hashes and recording provenance accompany the analysis.

Time tags use exact integer picoseconds. Preserve integers during conversion, upload, decoding and export; floating-point spreadsheet rounding can change large timestamps. The supplied conversion workflow exports a compatible CSV and provenance sidecar; `R25_TIME_TAGS.md` specifies exact fields and conversion commands. A source label supplied by an uploader does not independently authenticate an experimental acquisition.

Recorded replay does not connect live hardware. Instrument control, real-time acquisition and its clock calibration require a separately operated instrument computer and acceptance records. Test decoding with genuine vendor recordings; a mocked SDK verifies adapter behavior only.

30.4 Models and research products

Integration	Scientific role and outputs
QuTiP	Explicit two/four-subsystem dynamics, complex state histories, observables, independent matrix-equation comparisons, and a bounded bath-memory reference.
Qiskit Experiments	State and process tomography with raw measurement counts and separate raw/physical reconstruction records. Simulated execution remains simulated evidence.
SeQUeNCe	Native hardware, resource and reservation protocols, finite memories, competing requests, delivery events, and a published tutorial reproduction. Native representation approximations remain explicit.
Orekit	Two-body and J2 trajectories, reference-frame conventions, convergence, PF-engine residuals and published SGP4 verification vectors.
Detector replay	Integer-time event records, explicit coincidence pairing, lag distributions, calibration metadata, and declared finite-window normalization.
PyMC and SALib	Calibrated memory-parameter inference, posterior predictions, independent quadrature checks, and sensitivity of the actual PF memory channel.
EinsteinPy	Symbolic Schwarzschild identities and checks against PF metric, clock and radial photon calculations.
Engineering	Fixed MATLAB/Simulink clock equations and a pinned FMI decay reference; no general apparatus calibration is inferred.
Published data	Recorded equation comparisons and fixed-model residuals against a public superconducting-detector dataset.

Detailed model conventions and parameters are supplied in these files:

- `R24_QUANTUM.md` and `R24_NETWORK.md`;
- `R24_ORBIT_ENGINEERING.md`;
- `R24_OBSERVATIONS_UQ.md`;
- `R24_PUBLIC_RESEARCH_REVIEW.md`.

The source registry records which publications supply equations, reference code, simulated outputs or experimental measurements. These categories support different claims.

30.5 Run, inspect and reproduce

```
python run_research_integrations.py --list
python run_research_integrations.py --config config_r24_orbit_j2.json --output results_j2
python run_research_integrations.py --verify results_j2 --compare-source
```

An availability listing does not execute validation. Each study must start in an empty output directory so files from prior runs cannot be incorporated. `report.json` and `report.md` retain scope, checks, engine, sources and limitations. Backend products preserve the actual state, event, trajectory, measurement or inference data used in the study. The saved configuration and environment record the reproducibility inputs. The evidence manifest binds output and production-source bytes and identifies source changes during a run. Hashes attest to integrity; they do not establish acquisition authenticity.

The workbench shows execution, numerical checks, experimental qualification, external replication and commercial benefit separately. A comparison with a paper is interpreted under its assumptions. A missing licensed dependency is reported as blocked, not completed. Failed checks remain failed. Measured residuals without a justified acceptance criterion remain descriptive.

30.6 Experimental and operational interpretation

An experimental-data adapter needs actual acquisition records and calibration. The supplied uncertainty and inline raw-event demonstrations are explicitly synthetic. Vendor recording fixtures, when present, retain their source and acquisition scope in the release evidence. Public figure data are useful when their metadata support the selected observable; they are not automatically raw detector streams. Calibration and evaluation acquisitions remain distinct, with limitations on identifiability, uncertainty and model discrepancy recorded in the result.

The public detector comparison evaluates a fixed published response model and retains its discrepancies. It does not certify a universal detector model or a satellite forecast. A quantum processor experiment, if separately performed, would characterize that processor and circuit; it would not validate free-space optical loss or PF timing. Similarly, a standard-metric GR check does not establish a new privileged-frame selection theorem.

The historical record `validation_r25/RELEASE_VALIDATION.json` identifies executed studies, software tests and blocked checks. Historical records keep their original scope. Broad experimental accuracy, independent external replication and commercial superiority require separate evidence. Local API tests do not establish Docker execution or a live cloud deployment. The workflow is designed to preserve that distinction throughout execution and export.

Chapter 31

Numerical relativity and connected research workflows

Revision 26 adds Einstein Toolkit and kuibit as distinct research components. **Einstein Toolkit evolves numerical spacetime; kuibit reads and analyzes its outputs; EinsteinPy supplies separate symbolic and analytic references.** A successful import of one package does not establish execution of another. This chapter describes the implemented bridge and its scientific boundaries. The release evidence identifies which numerical studies actually executed.

31.1 Select the physical question before selecting an API

Research tools apply to different observables. A scalar curvature reference is not a quantum channel, a detector time stream is not a spacetime solution, and a tomography reconstruction does not measure a satellite orbit. Select an adapter because its declared inputs and outputs match the study. Explicit companions can independently check a saved parent result; they do not silently change the parent's physics or overwrite its evidence.

Component	Useful inputs and products	Interpretation boundary
Einstein Toolkit	Registered numerical-relativity studies; metric and constraint outputs	Executed thorns, parameters and grid resolutions define the qualified scope.
kuibit	Supported scalar, grid, wave and horizon output families; plots and exports	Availability depends on the fields actually written by the simulation.
EinsteinPy	Declared mass, radii and metric conventions; standard Schwarzschild checks	Does not evolve the Einstein Toolkit's numerical spacetime or establish PF selection.
QuTiP	Explicit density operators and channel parameters; state and observable histories	A channel requires a physical noise or receiver model, not merely a coordinate shift.
Qiskit Experiments	A defined parent state and measurement configuration; sampled tomography	Aer is simulated acquisition. It does not become a satellite measurement.
Orekit and SeQUeNCe	Orbit/time conventions or network resources; native simulation products	Their research studies retain their own supported input contracts.
PyMC and SALib	Declared likelihoods, priors and parameter ranges; inference or sensitivity	Model-conditional uncertainty does not quantify every source of discrepancy.
Engineering and replay	Approved models or exact recorded events; native outputs and checks	Licenses, vendor SDKs, calibration and acquisition provenance remain required.

31.2 Einstein Toolkit execution and kuibit analysis

Einstein Toolkit is a native computational-relativity framework, not a Python package installed by `pip install einsteinpy`. The selected native build, thorn list, compiler settings, parameter files and run commands must be recorded. An administrator registers executable studies; a scientific request selects a registered identifier and permitted numerical parameters. Client configurations do not choose arbitrary executables or upload runnable thorn code.

The GaugeWave reference exercises a nontrivial coordinate representation of flat spacetime. Its exact solution permits comparisons of the evolved metric and convergence with resolution. It is not a measured gravitational

wave, a satellite gravitational field, or evidence for a physical privileged frame. Convergence requires an explicit norm, aligned comparison domain and time, and multiple resolutions. A smaller residual at one resolution alone is not a convergence-order demonstration.

kuibit reads the simulation outputs that are actually present. Scalar time series support evolution and constraint plots. Grid data support field profiles and slices. Wave-extraction and apparent-horizon products require the corresponding output families. A menu or parser for a family does not certify that every thorn, mesh structure, extraction convention or restart layout has been validated. Missing products remain unavailable; they are not filled with synthetic values.

The public GW150914 reference uses an attributed subset of an existing Einstein Toolkit simulation archive. Its scalar, grid, wave and horizon outputs support kuibit analysis and documented final-remnant comparisons. Reading that archive is not a new merger evolution or a new analysis of detector strain. The preset `config_r26_et_gw150914.json` retains that distinction. Strain reconstruction from Ψ_4 requires a declared integration convention and frequency cutoff; the resulting curve is not independent observational data.

The native GaugeWave preset instead evolves the reviewed McLachlan BSSN configuration at 32, 64 and 128 interior longitudinal points through one code time unit. The amplitude is 0.01 and the wavelength is one code length. This is a bounded adaptation of the standard numerical-relativity test, not the complete long-duration stability study from a paper.

31.3 From numerical geometry to a quantum link

The bounded numerical-metric bridge uses sampled metric data with explicit coordinates, time samples, units and provenance. It calculates null propagation, clock proper time and endpoint-measured frequency before supplying a defined quantum-channel calculation. Each stage preserves its inputs and outputs so that a change in geometry can be traced to the final observable.

For a photon wave vector k^μ and an observer four-velocity u^μ, the measured frequency is proportional to $-g_{\mu\nu}u^\mu k^\nu$. Coordinate frequencies alone do not establish an observable redshift. Likewise, clock comparison requires proper-time integration along specified observer paths. The common normalization and unit convention must be preserved when data move between numerical-relativity output and SI-valued link calculations.

The optical stage uses a constant endpoint-frequency factor for its declared Gaussian receiver mode, not a full wavepacket field evolution through arbitrary time-varying spacetime. The initial numerical-metric bridge is restricted to its declared plane-metric geometry. It does not establish general three-dimensional ray tracing through an evolving, rotating, nonspherical Earth spacetime. It must reject an unsupported metric contract, insufficient time coverage or invalid interpolation domain. There is no analytic fallback advertised as native Einstein Toolkit output.

A frequency shift can reduce a specified receiver's spectral-mode overlap, but it is not automatically irreversible decoherence. The reference quantum channel states the encoding, mode definition, receiver acceptance and any compensation. Its result concerns that channel. A different detector or compensation strategy constitutes a different comparison. The relativistic quantum reference uses single-rail vacuum/one-photon states; those labels do not denote the polarization qubits in the normal mission model. QuTiP beam-splitter evolution and a partial trace describe loss into unmatched spectral modes under that explicitly stated encoding.

31.4 Research companions and shared evidence

A companion identifies a completed parent study and the exact artifacts it consumes. The parent keeps its original execution and qualification status. The companion records parent hashes, the derived configuration, its own backend/environment and its separate comparison results. A successful child does not retroactively validate every observable produced by its parent.

The applicability matrix distinguishes direct execution, an explicit parent companion, a reference study and an unsupported combination. Where an API cannot consume a workflow's actual scientific outputs, the interface must say so. Merely launching the same independent preset next to every workflow would not establish cross-workflow coupling.

The implemented memory companion recomputes the relevant declared channel with QuTiP. The tomography companion consumes an explicit parent density matrix and uses simulated measurements. The EinsteinPy companion takes the parent's applicable gravitational mass and radii rather than silently substituting unrelated reference conditions. The orbit companion derives osculating elements from the selected saved Cartesian state, then reorients the reference orbit at its own epoch; it is not an exact replay of the parent trajectory. Their results remain conditional on the supported parent fields and assumptions.

The fifth companion, `mission_spectral_channel`, consumes an actual saved mission ray: emission/reception endpoints, covectors, clock rates and body parameters. It reruns the boundary-value problem and applies the explicitly selected optical receiver-mode assumptions. Select arm `a` or `b`. The defaults are a 700 THz carrier, 1 MHz spectral-intensity standard deviation, zero residual receiver-proper delay and no compensation. These are new visible optical assumptions, not values inferred from an unrelated parent bandwidth. `center` aligns the carrier; `full` assumes ideal inverse spectral dilation. This single-rail calculation does not apply an extra gravitational dephasing term to the parent's polarization state. Full contracts are in `R26_WORKFLOW_INTEGRATIONS.md`.

31.5 Use companions in the workbench

Each workflow setup includes an expandable **Scientific API applicability for this workflow** table. Its statuses are `supported`, `reference_only`, `inapplicable`, and `unimplemented`. These are operation/applicability labels, not a claim that a runtime is installed or the current job has passed a numerical test.

Open a completed run's Results and find **Scientific companion studies**. Select a saved scenario output and its zero-based epoch index. The interface reports actual prerequisites and exposes the bounded options for that companion. Run the selected study and inspect its separate report, parent lineage and figures. For hosted jobs, authorization and the committed parent attempt scope which artifacts the companion may consume.

The command-line equivalent is:

```
python run_workflow_integrations.py --capabilities
python run_workflow_integrations.py --request parent_request.json --parent-output completed_parent
 ↪ --list
python run_workflow_integrations.py --request parent_request.json --parent-output completed_parent
 ↪ --selection companions.json --output results_companions
```

The selection file has a `studies` list containing `companion_id` and bounded `options`. Use the eligibility listing's actual artifact/epoch rather than assuming the first epoch contains a complete state or successful ray. The CLI also permits an explicitly configured contextual reference where the applicability matrix allows it; this is labelled separately from data derived from the parent.

31.6 Visualization and reporting

Inspect the numerical data alongside each plot. Reports should identify the plotted field, component, coordinate or physical units, sample selection, resolution and comparison reference. Constraint histories, metric profiles, convergence curves, wave products, horizon diagnostics and quantum-state products represent different quantities and should not share an unlabeled quality score.

Every research API study writes a portable `report.html` with its image gallery, numerical checks, summary, parent lineage where present, source references and links to CSV/JSON outputs, environment and evidence manifest. Use the hosted **Download run** action to obtain the complete directory; downloading HTML alone omits its relative images and data. Keep the report beside those artifact paths when copying the result. The workbench displays supported generated artifacts and preserves their data for export. A plot is linked to a completed analysis or reports that its required inputs are absent. Static publication figures and interactive inspection are complementary; neither replaces the underlying arrays and check records. The report separates configuration availability, native execution, numerical agreement, empirical qualification and commercial outcomes.

31.7 Reproduction and acceptance

Install the Python dependencies in `requirements_numerical_relativity.txt` and configure the separately built native executable before a fresh evolution. The reviewed recipe and pinned source identities are in `pf_integrations/models/einstein_toolkit/`. For example:

```
python release_tools/et_build.py --workspace /opt/pf-et --jobs 4
export PF_ET_EXECUTABLE=/opt/pf-et/Cactus/exe/cactus_pf_gauge
python run_research_integrations.py --config config_r26_et_gauge_wave.json --output results_et
python run_research_integrations.py --verify results_et --compare-source
```

The Linux native build needs its documented compiler and development dependencies; a Python-only Windows installation does not supply them. Data-only analysis uses `config_r26_et_analysis.json`; a fresh native run is a different action. The `config_r26_et_metric_link.json` preset exercises the bounded numerical-metric bridge. The separate relativistic-quantum presets describe Schwarzschild links and Gaussian receiver modes.

The SaaS profile file gains keys `einstein_toolkit` and `kuibit`. Only the native capability permits administrator variables `PF_ET_EXECUTABLE` and `PF_ET_STUDIES_FILE`. Each profile uses the service's execution backend. The supplied container recipe requires an actual build and target-platform acceptance before operational use.

Use the setup and registered-worker instructions in `R26_START_HERE.md` and `R26_DEPLOYMENT_GUIDE.md`. `R26_COMPLETENESS.md` maps implemented operations to executed evidence and remaining acceptance requirements. The release record under `validation_r26/` is authoritative for actual run outcomes.

Preserve the native source revision, thorn list, compiler and libraries, parameter files, output frequency, numerical resolutions, coordinate/unit conventions, complete logs and raw output hashes. Preserve analysis versions, interpolation policies and tolerances with the derived products. A rerun with a changed thorn list or tolerance is a new qualification, even if its figure looks similar.

The public-source review distinguishes original method papers, documented software examples, machine-readable numerical references and experimental measurements. Reproducing a well-specified numerical result supports that calculation under its stated assumptions. Experimental accuracy still needs suitable calibrated data and uncertainty analysis. Independent replication, PF operational advantage and commercial claims require their own evidence.

31.8 Primary references and their roles

- Einstein Toolkit download and requirements: installation context. The packaged lock file fixes source identities.
- Löffler et al. (2012): the Einstein Toolkit framework; no PF-specific validation is inferred.
- Alcubierre et al. (2004): GaugeWave verification. The release records its own short-time parameters.
- Bozzola (2021): kuibit analysis and scientific-output conventions.
- GW150914 simulation archive: attributed numerical outputs and rounded remnant references, not measurements.
- Bruschi et al. (2014): curved-spacetime spectral modes under explicit receiver assumptions.

`R26_RESEARCH_REFERENCES.md` records further sources, retrieval identities and numerical ambiguities. No scaling factor is adjusted to force a discrepant printed estimate. High-precision equation targets remain generated references. Background citations do not independently support PF selection, operational advantage or commercial claims.

Topic index