
Optimizer & Equation Solver

Numerical software for engineering and operational applications

Patodia Infotech Private Limited | OEM capabilities and benchmark evidence | 15 October 2026

Two complementary products: Optimizer searches for decisions that improve an objective while meeting constraints. Equation Solver searches for distinct feasible solutions without requiring a user-defined objective. Both expose command-line and application-library interfaces. Equation Solver binaries are available for Windows and Linux x86-64; the Linux Optimizer release is planned.

Evidence at a glance: 20 equation cases across 120 runs, with complete known-root coverage in 119 runs; TSP tests across 8 models up to 18,512 cities. Detailed results and measurement scope follow.

1. Optimizer

Mathematical optimization and dedicated operational models in one engine.

Mathematical modelling

- **Linear and mixed-integer linear optimization:** continuous, integer and binary decisions; minimization or maximization; bounds, equalities and inclusive inequality constraints.
- **Nonlinear and mixed-integer nonlinear optimization:** supported nonlinear expressions in objectives and constraints, with configurable search and feasibility controls.
- **Bounded numerical global optimization:** optional region/bound exploration and objective-gap reporting for supported models with finite bounds. This uses numerical safeguards, not exact-arithmetic certification.
- **Equation and feasibility modes:** solve represented relationships or seek feasible decisions when no optimization objective is supplied.
- **Scientific calculations:** evaluate real-valued mathematical expressions and constant-based calculations.

Dedicated application services

- **Routing:** traveling salesman problems, capacitated vehicle routing, and vehicle routing with time windows, using their respective supported formats and constraints.
- **Material utilization:** single- and multiple-stock cutting, one-dimensional bin packing, and rectangular packing/cutting.
- **Production and workforce planning:** job/machine scheduling, staff rostering, school timetabling, worker-job assignment, and supplied binary covering/partition patterns.
- **Allocation and networks:** transportation, minimum-cost network flow, and multi-resource placement/rebalancing.

Each service has its own model grammar, supported constraints and controls. A dedicated service is applicable only when it represents the actual business requirements.

Search, precision and results

- Time allowances, seed selection, and applicable iteration, start, node and gap controls.
- Native multi-seed workflows for supported services, sharing an overall time allowance.
- Double precision and true binary128 in supported builds and modes. Extended nonlinear/root mode uses ordinary-precision exploration followed by higher-precision checking/refinement; extended LP/MILP and network modes use their documented extended paths.
- Feasible candidates, objective values, stopping reasons, and applicable bounds/gaps support assessment of solution quality.
- Supported LP, MPS and NL model imports. Documented AMPL/GAMS workflows use separately installed modelling runtimes; supported Python modelling adapters provide additional entry points.

Illustrative applications: constrained formulation, process operating decisions, production scheduling, routing, cutting and resource allocation. Application-specific suitability and scale should be demonstrated on representative customer models.

2. Equation Solver

Automated search for multiple distinct real solutions of equation systems.

Core solving capabilities

- **No user-supplied initial estimate required:** automatic starting-point generation and search-range exploration; meaningful physical bounds may also be supplied.
- **Multiple solutions in one run:** search for distinct approximate real roots rather than requiring a separate manually initialized call for every root.
- **Linear and nonlinear systems:** coupled polynomial and supported transcendental equations.
- **Integer and mixed-integer systems:** real, integer and binary variables, including supported sign-restricted integer declarations.
- **Constrained equations:** variable bounds and inclusive $\leq$ / $\geq$ conditions are checked with the equations at the requested residual tolerance.
- **Feasibility sampling:** return feasible points from inequality-defined or continuous regions without an objective; this is a finite sample, not enumeration of an infinite region.

Modelling and numerical controls

- Plain-text models, inline expressions, standard input and machine-readable JSON results.
- Automatic variable detection, named constants, scientific notation, comments and implicit multiplication.
- Configurable time, starts, iterations, seed, solution count and coordinate-merging threshold.
- Double precision; hybrid binary128 refinement with `--extend`; optional binary128 discovery with `--extend --quad-search`.
- Binary128 provides 113 significand bits, approximately 33 significant decimal digits. Default residual tolerance is $1e-9$ normally and $1e-24$ in extended mode.
- Automatic ranges are interleaved in the current build so wider scales receive early search opportunities; explicit bounds remain enforced.

Discovery timings and transparent reporting

- **Per-root elapsed acceptance time:** see when each distinct returned solution first passed the solver's acceptance checks.
- **Time to the last distinct accepted root:** distinguish when the reported solution set was obtained from the total search duration.
- **Residuals and stopping status:** inspect maximum equation error/condition violation, arithmetic mode, starts used and why the run stopped.
- Discovery timings are measured within the solver run. They are not total application latency or proof that all roots have been found; later search can continue after the last accepted root.

Illustrative applications: alternative mechanism configurations, steady-state process equations, equilibrium calculations, calibration equations and mathematical research models.

Optional interval verification: the current IBEX-enabled integration can certify supported continuous square equality systems within a reported finite domain. Completeness is reported only when the required coverage and root checks succeed. Timeouts or unresolved regions leave completeness unestablished. This is a separate backend-dependent capability, not a guarantee for all supported equation models.

3. Equation Solver: measured root coverage

119 of 120 runs matched every known root in the bounded test set.

The supplied report contains 20 cases, each tested at 1, 2 and 5 seconds in normal and extended modes, using seed 42. Normal residual acceptance was $1e-9$; extended acceptance was $1e-24$. Tests include polynomial, coupled, trigonometric, exponential, nonsmooth, integer and mixed-integer examples.

Complete known-root coverage by time allowance

Allowance	Runs with complete coverage	Matched / expected root occurrences
1 second	39 / 40	1,027 / 1,028
2 seconds	40 / 40	1,028 / 1,028
5 seconds	40 / 40	1,028 / 1,028

Root occurrences sum results across cases and both arithmetic modes; they are not counts of globally distinct mathematical roots. The sole partial run returned 255 of 256 roots for the eight-variable sign-combination system in extended mode at one second. The two- and five-second runs matched all 256. The report lists zero rejected and zero unmatched returned solutions across all 120 runs.

Selected extended-mode results

Test	Known roots matched	Allowance	Last accepted root
Coupled polynomials	8 / 8	1 s	0.001317 s
Trigonometric integer-location roots	11 / 11	1 s	0.000266 s
Exponential coupling	4 / 4	1 s	0.000368 s
Six-variable sign combinations	64 / 64	1 s	0.051417 s
Eight-variable sign combinations	256 / 256	2 s	1.010291 s
Six declared integer variables	64 / 64	1 s	0.525544 s

The sign-combination cases use independent square equations and are structured coverage tests, not evidence of performance on arbitrary densely coupled systems. Selected rows illustrate different capabilities; the aggregate above includes every reported run.

What the measurements mean

Per-root and last-root timings identify when results first passed the solver's acceptance checks. The process may continue searching afterward. For the 256-root extended two-second run, last-root time was 1.010291 s, solver time 2.0002 s, and harness wall time approximately 2.1198 s. These are different measurements.

The benchmark matches coordinates against analytically known bounded sets and checks printed residuals against the requested tolerance. It does not independently recompute binary128 residuals or provide an interval completeness certificate. The attached HTML does not identify CPU/RAM or the executable hash, so timings are observations from the supplied test environment, not portable latency guarantees.

OEM relevance: automatic multiple-solution discovery, integer feasibility and high-precision refinement can be evaluated through one interface, with explicit coverage and discovery-time reporting.

Source: supplied Equation roots benchmark, report(6).html, timestamp 15 October 2026, 06:49:26 UTC. One seed per configuration; not a multi-seed statistical study. Further developer tests are not quantified here because they are not included in the supplied benchmark evidence.

4. Optimizer: measured TSP solution quality

23 usable results across 8 tour models, from 48 to 18,512 cities.

The supplied TSP report compares returned tour costs with each instance's listed optimum. Every scheduled run returned a usable result. Two of eight models matched the listed optimum; five individual runs matched it. Larger instances demonstrate useful approximate tour quality within explicit time allowances.

Best reported result per model

Model	Cities	Listed optimum	Obtained cost	Gap	Budget
att48	48	10,628	10,628	0.000%	5 s
bier127	127	118,282	118,282	0.000%	20 s
d493	493	35,002	35,069	0.191%	20 s
dsj1000	1,000	18,660,188	18,791,775	0.705%	60 s
d1655	1,655	62,128	62,712	0.940%	90 s
brd14051	14,051	469,385	478,314	1.902%	180 s
d15112	15,112	1,573,084	1,603,368	1.925%	180 s
d18512	18,512	645,238	658,945	2.124%	180 s

Gap = $100 \times (\text{obtained cost} - \text{listed optimum}) / \text{listed optimum}$. Costs use each model's own distance units and should not be added across models. Rows show the best observed result across that model's tested budgets, not averages or independently repeated best-of-seed trials.

Time-quality trade-off at large scale

For d18512, the reported reference gap decreased from 2.788% at 60 seconds to 2.124% at 180 seconds. For brd14051, it decreased from 2.449% to 1.902% over the same allowances. On d493, the 20-second run was better than the longer runs, illustrating that a longer independent heuristic run need not improve every result.

Evaluation scope

Runs used fresh processes with the default seed and automatic TSP settings. Total requested solver time was 27 minutes 15 seconds, plus overhead. The report is dated 8 October 2026; it does not identify CPU/RAM or the executable hash. These historical measurements are not benchmarks of the newly supplied Linux Equation Solver build or a future Linux Optimizer release.

The report compares scalar costs; it does not independently validate every tour edge or prove optimality. Matching a listed optimum is cost agreement with that reference. TSP results do not establish corresponding performance for vehicle routing, process optimization or other services. No competing solver was run in these reports.

OEM relevance: time-bounded tour optimization with measurable reference gaps provides a concrete starting point for evaluating routing-related application components. Production constraints and customer-scale requirements must be assessed on representative models.

Source: supplied TSP Benchmark Report, report(5).html, 8 October 2026. Budgets depend on instance size; all 23 scheduled results are represented in the report.

5. Shared capabilities and application integration

Supported real-valued expression functions

Arithmetic and powers; trigonometric $\sin$, $\cos$, $\tan$; inverse trigonometric $\arcsin$, $\arccos$, $\arctan$; hyperbolic $\sinh$, $\cosh$, $\tanh$; $\log/\ln$, $\log_{10}$, $\exp$; $\sqrt{}$, $\sqrt[3]{}$, abs ; $\log_{1p}$, $\expm1$; and two-argument pow , atan2 , min , max , hypot .

Functions require valid real domains. Availability is mode-specific: numerical global optimization and interval certification support narrower expression sets than ordinary solving. Angles are in radians.

Embedding and deployment

- **Command-line operation:** submit models from desktop applications, scripts and automated workflows.
- **C-compatible native library interfaces:** integrate without exchanging C++ objects across the public ABI; use matching architecture, headers and runtime dependencies.
- **Optimizer callers:** documented C, C++, Python, Java and C# examples; PHP front-end workflow.
- **Equation Solver callers:** examples supplied for C, C++, Python, C#, Java, Julia and PHP. C/C++ and Python calls were executed for the Linux release; the remaining language runtimes require customer-environment validation.
- **Windows:** executable and DLL interfaces for both products.
- **Linux Equation Solver available:** x86-64 executable and .so shared library, built for the CentOS 8 glibc 2.28 baseline, with runtime dependencies, examples and rebuild instructions. CentOS-loader checks passed; target-machine installation testing remains required.
- **Linux Optimizer planned:** compilation and validation are pending; a prebuilt Linux Optimizer binary/shared library is not yet claimed as available.
- **Inspection and automation:** preserve model inputs, options, result values and diagnostics for reproducible evaluation. Equation Solver retains high-precision values and residuals as decimal strings in JSON.

Optimizer's general DLL calls are synchronous and serialized; independent concurrent jobs use separate processes. Its current general API does not expose live progress/cancellation callbacks. Integration details differ between the two products and must follow the matching public interface.

Accuracy and scope

- Returned approximate candidates are checked against the represented requirements at the applicable tolerance. Unmodelled physical or business rules are not automatically enforced.
- A small residual does not guarantee the same number of correct digits in the unknowns. Conditioning, scaling and input accuracy remain important.
- Neither product offers arbitrary-precision or symbolic arithmetic. Equation Solver searches real roots, not complex roots.
- General root search is heuristic. No root found does not generally prove infeasibility; roots returned do not generally establish completeness. Optimizer's local convergence and best feasible objective do not automatically prove global optimality.
- Extended precision is service-dependent. Optimizer's operational services generally reject extended mode; TSP/CVRP do not perform binary128 refinement even when the flag is accepted.

Proposed technical evaluation

Supply a representative model, its domains and acceptance criteria. Compare valid solutions or objectives, residuals, discovery times, total elapsed time and integration effort on agreed hardware and settings. OEM redistribution, deployment rights, maintenance and support can be discussed after a successful evaluation.

Contact: Panna Lal Patodia, CEO
Patodia Infotech Private Limited | Bengaluru, India
Email: panna@patodiainfo.com

Basis: supplied Optimizer User Guide (8 October 2026) and Equation Solver User Guide (14 October 2026), supplemented by the current Equation Solver's verified timing, interleaved-range and optional IBEX integration features. Feature availability must be confirmed against the evaluation build. Benchmark evidence is from the two supplied HTML reports; these do not establish comparative superiority or industrial certification.

Copyright © 2026 Patodia Infotech Private Limited. All rights reserved.