Patodia Equation Solver

Advanced Mathematical & Scientific Computing Platform

Version 4.0 Complete Features Brochure

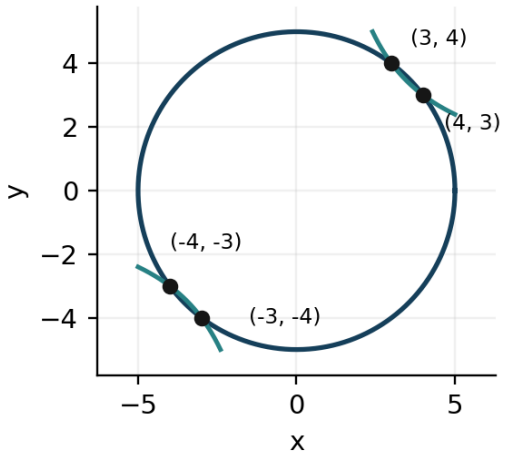
For university professors, research scholars and university students.

Turn a word problem into a mathematical model students can solve and check. Enter the equations of a supported linear or nonlinear system in any order. Patodia Equation Solver solves them simultaneously and preserves the results for teaching or research.

Find the dimensions of a rectangle

A rectangle has an area of 12 m² and a diagonal of 5 m. What are its side lengths? Let x and y be the two lengths. The area and diagonal give the equations below. The solver can return multiple real solutions without a user-supplied initial guess.

```
x^2+y^2=25
x*y=12
bound x=[-5,5]
bound y=[-5,5]
```



The model as written has four real solutions: (−4, −3), (−3, −4), (3, 4) and (4, 3). Only positive lengths fit the word problem. They give one rectangle, 3 m by 4 m.

The broader toolkit includes ODEs, boundary value problems, supported PDEs, eigenproblems, integral equations, quadrature, singularity analysis, differentiation, interpolation, limits and Compute special functions.

Multiple-root discovery is subject to the domain and resource limits. Initial conditions are still required for ODEs; nonlinear BVPs may need a starting curve. Version 4.0 adds programmable workflows for repeatable numerical studies.

EQUATION SOLVING

Algebraic equations and model formulation

Solve simultaneous real linear, polynomial, nonlinear and transcendental equations. The order of equations within a system does not change the mathematical relationships they define.

Automatic searches with explicit controls

Automatic starting points and multiple-root discovery let you explore a system without supplying an initial guess. Set bounds, time, solution cap, starts, iterations, seed and residual tolerance.

Use the stopping reason, search domain, residuals and independent checks to assess root coverage and feasibility.

Domains and constraints

Facility	What readers can do
Real and discrete variables	Combine real, integer, positive, nonnegative, negative, nonpositive and binary domains in supported models.
Bounds and feasibility	Apply inclusive variable bounds and separate <= or >= conditions to filter feasible solutions.
Expression language	Use arithmetic, powers, supported elementary functions, fixed constants, comments and explicit mathematical constants #pi and #e.
Remainders and products	Model supported integer remainder conditions and recognised product or factorisation structures.
Number styles	Use US decimal points or EU decimal commas with appropriate list and argument separators.

Make the modelling assumptions visible

Use explicit multiplication, declare variable domains and set bounds that reflect the problem. Trigonometric arguments use radians.

Optional interval verification

The optional IBEX-assisted CLI workflow can certify root enclosures and assess completeness over an eligible finite box. IBEX requires a separate installation.

Match the sums of two progressions

The second and third terms of an arithmetic progression are 14 and 16. Find a geometric progression whose ratio equals the arithmetic difference and whose first three terms have the same sum. Use a and m for the first terms and d for the common difference.

```
a + (a+d) + (a+2*d) = m + m*d + m*d^2
a + d = 14
a + 2*d = 16
```

The sum equation comes first. The reference solution is a = 12, d = 2 and m = 6. The progressions are 12, 14, 16 and 6, 12, 24. Both sum to 42.

EXACT ARITHMETIC

Exact integer equations and number theory

Recognised integer structures use exact arithmetic rather than rounding a floating-point answer to an integer. This is especially useful when coefficients, bounds or solutions exceed ordinary floating-point precision.

Method	Supported purpose
Linear Diophantine equations	One linear equation in two integer unknowns; extended Euclidean methods produce a parameterised family, with optional bounds and cardinality information.
Pell equations	Supported forms x^2−D*y^2=N, including the classical N=1 case and applicable signed or degenerate routes.
General quadratic equations	Eligible two-integer quadratic forms, including supported signed conics and bounded degenerate cases.
Weighted sums of squares	Supported A*x^2+B*y^2=C forms using exact arithmetic, factorisation and modular methods; signed or zero coefficients follow applicable quadratic handling.
Factor and remainder structures	Recognised integer products, eligible affine or expanded factor forms and supported modular conditions.
Very large integer values	Eligible exact routes preserve all coordinate digits and accept arbitrary-magnitude simple literal bounds such as 1e500, subject to method and resource limits.

A complete family rather than one pair

```
int x
int y
7*x+11*y=100
```

The family can be written as x=8+11*t and y=4−7*t for any integer t. Bounds restrict the permitted parameter interval. For example, 3*x+5*y=100 with both variables between 0 and 100 has seven nonnegative pairs.

Large exact power coefficients

```
int x
int y
12345^10*x-7531^11*y=1
```

The updated integer engine retains integer powers and coefficients exactly in eligible routes. Keep coefficients as integer expressions rather than passing them through a floating-point calculation.

Read exact identity verification, enumeration completion, parameter bounds and stopping reason separately. Exact returned pairs do not by themselves prove that every pair was enumerated. Compute also provides arithmetic and prime-factor functions described later.

DYNAMICS AND BOUNDARY MODELS

Ordinary differential equations and boundaries

Initial value problems

Enter a scalar ODE or coupled system, including direct derivative orders 1 through 8 and up to 64 expanded first-order states. Supply the required initial values and derivatives. Forward and backward integration are supported within the implemented explicit or affine highest-derivative structure.

Method	Typical use
RK4	Fixed-step classical Runge–Kutta for teaching stepwise integration.
RK45	Adaptive Dormand–Prince integration for suitable nonstiff problems.
SDIRK2	Implicit adaptive L-stable integration for suitable stiff problems; selected explicitly.

```
ode
y''+0.5*y'+4*y=0
initial t=0
initial y=1
initial y'=0
interval t=[0,20]
method=RK45
output step=0.05
```

Control relative, absolute and component tolerances, internal steps and step limits. Choose output points, output spacing or sample times independently of internal integration steps. Dense interpolation supports continuous sampling; optional dense coefficients can be retained in JSON.

Named events locate zero crossings, filter rising or falling directions and optionally stop integration. Inspect event times and states, trajectory plots and phase portraits.

Supported singular points

Automatic regular-series analysis can initialise or continue supported scalar linear first- or second-order IVPs at compatible singular points. A Bessel model may start at zero with its original conditions. Diagnostics distinguish incompatible, ambiguous, unsupported and unresolved cases; arbitrary singular continuation is not assumed.

Regular two point boundary value problems

Adaptive collocation handles supported linear or nonlinear BVPs, direct orders through 8 and up to 16 expanded states. Supply endpoint values, derivatives, Robin or supported coupled endpoint conditions. Configure initial curves, boundary and defect tolerances, mesh limits and Newton iterations. Different starting curves may select different nonlinear branches.

A useful classroom comparison is fixed-step RK4 versus adaptive RK45, followed by a BVP solved from endpoint data. Examine trajectory error and convergence rather than printed digits alone.

FIELDS MODES AND PARAMETER STUDIES

Partial differential equations and eigenproblems

Explore supported field models on their stated geometries, compare numerical and exact solutions, and investigate how grids and parameters affect the result.

Model class	Implemented scope
Laplace and Poisson	Supported steady two-dimensional scalar equations on rectangles, with equal constant nonzero diffusion coefficients.
Heat and reaction diffusion	Supported one-dimensional linear diffusion, reaction and source models; Crank–Nicolson time stepping.
Advection diffusion	Supported one-dimensional linear advection–diffusion; the dedicated class uses first-order upwind spatial advection.
Wave equations	One-dimensional positive constant wave speed, initial displacement and velocity, with a checked CFL stability condition.
Sturm–Liouville eigenproblems	Regular finite-interval self-adjoint form with positive leading coefficient and weight; multiple eigenvalues and normalised eigenfunctions.

Boundary conditions and compatibility

Supported PDE and eigenproblem modes accept Dirichlet, outward-normal Neumann and nonnegative Robin conditions as appropriate. Pure-Neumann Poisson problems require a mean value and compatibility checks. Initial and boundary data must agree. A normal derivative represents flux only when the physical coefficient is included correctly.

Adaptive accuracy and exact comparisons

Automatic refinement compares successive grids and reports estimated error, internal grid sizes and observed convergence order. Supply an exact solution when available to inspect maximum and RMS errors and signed error columns. Eigenproblem acceptance targets eigenvalue accuracy; it does not certify eigenfunction error bounds.

Visual exploration and sweeps

View heat maps, contour overlays, spatial sections and computed time-slice animations. Compare numerical, exact and error fields. Export supported maps and comparison plots as PNG. Display plots use reduced precision while numeric tables and exports preserve engine strings.

PDE, integral-equation and eigenproblem modes support parameter sweeps over finite lists. Multiple directives form a Cartesian product of up to 64 cases. Each case retains its own status and diagnostics, and supported views overlay runs. A sweep explores chosen inputs; it is not optimisation or general branch continuation.

Supported classes are structured numerical models, not arbitrary-geometry or nonlinear PDE simulation. Refinement estimates rely on convergence assumptions and are not certified continuum error bounds.

INTEGRATION FACILITIES

Quadrature and integral equations

Numerical definite integration

Evaluate a known one-dimensional integrand over a finite or infinite interval. The updated quadrature engine addresses supported endpoint singularities, removable singularities, expression cancellation and recognised nonsmooth corners such as abs(x^3−x).

```
quad
expr=1/sqrt(t*(1-t))
var=t
from=0
to=0.5
```

The reference value is π/2. The integrand need not be defined at an integrable endpoint. Constants used as parameters must be assigned with const declarations.

Field	Accepted spellings
Service name	quadrature or quad
Integrand	integrand or expr
Variable	variable or var
Endpoints	interval=[a,b], range=[a,b], or the paired directives from=a and to=b

Choose one spelling per field and one endpoint form. A single quad command can combine multiple ranges or split an overall range at declared singularities. Infinite endpoints use inf or infinity with the appropriate sign. Controls include tolerances, evaluation and refinement limits, subdivisions and time allowance.

Finite integration uses double-exponential tanh–sinh methods and semi-infinite pieces use exp–sinh. Targeted fallback strategies address difficult cases. Guard arithmetic, shifted-grid checks and tail estimates support acceptance. Check the Accepted integral and Target met fields before using a result.

Generic continuation checks do not make every singularity removable. Genuine poles, unresolved domains or tails can prevent acceptance. Cauchy principal values, symbolic antiderivatives and general multidimensional quadrature are outside scope.

Integral equations for unknown functions

Solve supported linear second-kind Fredholm and Volterra equations on finite intervals. Fredholm uses a scalar Nyström discretisation and dense solve. Volterra supports scalar equations and linear coupled systems with up to eight components using forward marching and small coupled endpoint solves.

Adaptive refinement and Richardson extrapolation provide convergence estimates. Supplied exact solutions enable error comparisons, and curves can be plotted and exported. These services solve for an unknown function; standalone quadrature evaluates a supplied function. First-kind, nonlinear and general singular-kernel integral equations are not claimed.

NEW IN VERSION 4.0

Automate numerical studies with workflows

Run several numerical jobs from one script. Sweep parameters, reuse models and pass results between services to repeat a study with the same calculation steps.

Combine 12 numerical services

Workflows support algebraic solving, quadrature, Compute, differentiation, limits, interpolation, ODE initial value problems, boundary value problems, PDEs, eigenvalue problems, integral equations and singularity analysis.

Sweep an integration bound

This script integrates x^2 from 0 to each of three upper bounds. It collects accepted results and reports the status of any unaccepted job.

```
workflow
model power_integral(b) as quad
    expr = x^2
    var = x
    range = [0, @b]
end
values = []
for n in 1..3
    run area = power_integral(n) within 2
    if area.accepted
        append values with area.value
    else
        report area.status
    end
end
report values
```

Expected values are approximately [0.3333333333, 2.6666666667, 9].

Control execution and inspect results

Use variables, user-defined numerical functions, if/else, for and while loops, and collections to process results. Set a whole-run time budget and use within for supported job caps. Each run supports up to 1,000 jobs and 10 million execution steps.

Run scripts in Windows Desktop or from the CLI, or call them through the DLL with an eligible licence. Inspect per-job diagnostics in Run Report or request JSON from the CLI.

NUMERICAL AND DESKTOP TOOLS

Quadrature and Desktop exploration

Split an integral at a known singularity

Use either explicit ranges or a list of singularities. Both examples below evaluate to 6/√5, approximately 2.683281573.

```
quad
expr = 1/sqrt(abs(x-0.2))
var = x
range = [0, 0.2], [0.2, 1]
```

```
quad
expr = 1/sqrt(abs(x-0.2))
var = x
range = [0, 1]
singularities = [0.2]
```

Investigate singularities and corners

The standalone singularities service investigates where an expression may be undefined or lose smoothness. Supply an interval and, optionally, candidate points.

```
singularities
expr = x * asin(2x * sqrt(1-x^2)) / sqrt(1-x^2)
var = x
range = [-1, 1]
// singularities = [-1/sqrt(2), 1/sqrt(2)]
```

Remove the comment marker to supply the two candidates. The function has finite corners at x = ±1/√2. Its formula is undefined at x = ±1, where both inward limits equal 2.

Generate nonlinear systems for teaching

Open Tools → Equation Generator to create random nonlinear systems with 1–20 variables. Choose function families and complexity, add optional bounds, and reuse the seed to reproduce a system. The generator can reveal its known solution on request and uses exact arithmetic to combine like terms and powers, factor common terms when shorter, and remove redundant parentheses.

Explore Fun with Mathematics

Find Your Number in Pi searches π for a digit sequence. The King's Puzzle asks for the smallest starting number satisfying successive divisibility conditions. Number Puzzle asks you to arrange number tiles under arithmetic rules.

Check difficult numerical results

For ODEs, BVPs, PDEs and integral equations, bounded numerical investigation can provide an additional validation step and report unresolved or ambiguous cases.

FUNCTIONS AND OBSERVATIONS

Differentiation interpolation and measured data

Differentiate expressions

Use diff or differentiate, and expr or expression. Evaluate derivatives at specified coordinates, request higher orders or mixed partials, or calculate a gradient and Hessian for multivariable models. grad and hess are short aliases for gradient and hessian.

```
diff
expr=x^2*exp(y)+sin(x*z)+y^3*z
at x=1
at y=0
at z=2
gradient=true
hessian=true
```

For a requested derivative use wrt=x and order=4, or wrt=[x,y] for a mixed partial. First derivatives use automatic differentiation; higher derivatives use numerical methods with diagnostics and error estimates. This is numerical evaluation, not a general symbolic derivative generator.

Differentiate sampled observations

Supply inline data or CSV columns, then select derivative order and query coordinates. Local polynomial differentiation accommodates nonuniform spacing. Savitzky–Golay fitting and smoothing splines provide options for noisy observations, with controls such as window, degree and lambda.

Optional measurement standard errors support uncertainty propagation under the documented independent-error assumptions. Uncertainty from input measurements is distinct from interpolation bias, model error and heuristic stencil disagreement.

Interpolate between samples

Method	Purpose
Linear	Simple piecewise straight segments.
Local polynomial	Flexible local fits; the default auto method.
PCHIP	Shape-preserving cubic interpolation for suitable data.
Natural cubic spline	Smooth interpolation with natural endpoint conditions.
Akima	Local cubic interpolation from at least five samples.
Barycentric	Global polynomial interpolation for up to 32 samples.
Smoothing spline	Smooth noisy observations with adjustable smoothing strength.

Use interp or interpolate with data or a file and at x query coordinates. Repeated input coordinates are rejected; nonuniform spacing is allowed. Extrapolation defaults to off and is available only for supported methods. Export interpolated or derivative values as CSV. Method selection and smoothing affect the answer.

LOCAL BEHAVIOUR AND SPECIAL FUNCTIONS

Limits and Compute calculations

Investigate finite and infinite limits

Use limit or lim to examine finite points, one-sided approaches, infinite endpoints and supported multivariable problems. Reports distinguish finite limits, signed infinities, nonexistence and undetermined results.

```
limit
expr=sin(x)/x
at x=0
direction=both
```

The removable limit is 1. For abs(x)/x at zero, the left limit is −1 and the right limit is 1, so no two-sided limit exists. Use direction=left or right to investigate the approaches separately. Endpoints may be inf or -inf.

Multivariable path comparisons can disprove a common limit when paths disagree. Agreement along finitely many sampled paths is not a proof of general existence. Inspect the reported evidence and precision; undetermined is not the same as divergent.

Compute named numerical results

The standalone Compute service evaluates elementary and special functions, reuses earlier named results and supports numerical exploration. Start with compute or func and use explicit multiplication.

```
compute
precision=binary128
a=gamma(5)
b=erf(1)
c=besselj(0,2)
d=zeta(2)
total=a+b+c+d
```

Assignments are evaluated in sequence, with up to 1,000 unique result names. expr=expression provides a single calculation. Forward references and duplicate result names are rejected. Real-domain and argument checks help diagnose invalid requests.

Sample a function with Function Explorer

```
compute
function=besselj(0,x)
range=[0,20]
points=500
```

Sample 2 to 20,000 equally spaced points, including both endpoints, using x as the independent variable. View the sampled curve and table and export the data. Invalid-domain points remain undefined rather than being replaced with zero.

Compute uses 113-bit working precision by default; precision=double selects the legacy evaluator. Special functions are specific to Compute and are not automatically accepted in every other service grammar.

FUNCTION FAMILIES AND USES

Compute function catalogue

The catalogue spans elementary mathematics, scientific special functions and exact number-theory operations. Function signatures and real-domain restrictions are documented in the complete Compute reference and Desktop help.

Family and functions	What the functions provide
Trigonometric sin cos tan cot sec csc	Circular functions in radians, including reciprocal trigonometric functions.
Inverse trigonometric asin acos atan acot asec acsc atan2	Principal inverse functions and quadrant-aware angle evaluation.
Hyperbolic and inverse sinh cosh tanh coth sech csch asinh acosh atanh acoth asech acsch	Hyperbolic functions and their real inverse branches.
Exponential and logarithmic exp expm1 log ln log10 log1p logbase logb	Exponentials and logarithms; expm1 and log1p reduce cancellation for small arguments.
Algebraic and rounding sqrt cbrt pow root abs hypot min max sign floor ceil round trunc frac	Real roots, powers, norms, extrema and integer-part operations. frac uses x−floor(x).
Factorial gamma and beta factorial fact gamma lgamma beta	Factorials, gamma, log of absolute gamma and Euler beta. Factorial accepts integers 0 through 2000.
Error and gamma derivatives erf erfc digamma polygamma	Error functions and logarithmic gamma derivatives; supported polygamma orders 0 through 8.
Bessel and Airy besselj bessely besseli besselk airy_ai airy_bi	Ordinary and modified Bessel functions and Airy solutions.
Orthogonal polynomials legendre assoc_legendre hermite laguerre assoc_laguerre chebyshev_t chebyshev_u	Classical polynomial families, including physicists' Hermite convention.
Zeta Lambert and integral functions zeta lambertw ei si ci	Riemann zeta, supported real Lambert W branches, exponential integral and sine/cosine integrals.
Elliptic integrals elliptic_k elliptic_e elliptic_f elliptic_einc elliptic_pi	Complete and incomplete elliptic integrals. The parameter is $m=k^2$, not the modulus k; elliptic_pi has complete and incomplete forms.
Integer and prime functions totient gcd lcm isprime primepi mobius divisorcount divisorsum primefactors primefactor	Totient, gcd/lcm, primality and prime counting, Möbius values, divisor counts/sums and prime-factor lists or indexed factors.

Most number-theory arguments are bounded by 10^{12}; primepi accepts 0 through 100,000,000. Restrictions differ by function. primefactors returns a sorted list with multiplicity; primefactor uses a one-based index. A list is not an ordinary scalar expression.

USE INSPECT EXPORT AND REPEAT

Desktop tools and academic adoption

A complete calculation record

Edit models with syntax highlighting, line numbers, find/replace, comments and indentation. Adjust equation and result fonts independently. Save .pdeq projects with settings and bounds, reopen recent work and import text. Built-in examples, an additional equation library and dedicated guides support learning.

Check syntax with F4 and solve with F5 or Ctrl+Enter. A separate worker keeps the interface responsive. Inspect solutions, reports and diagnostics against the completed model snapshot. Force-stopping a worker does not recover unfinished partial results.

Use trajectory and phase plots, PDE maps and animations, eigenfunction views and comparison plots. Export tables as CSV, structured results as JSON, reports as text and ODE events as CSV. Copy cells, inspect long integers and print equations or reports. Projects preserve inputs and settings; export results separately.

Automation and native integration

Desktop and the CLI run the same workflow scripts. Eligible DLL licensing enables C ABI integration, with examples for C, C++, Python, Julia and PHP. Applications should inspect result type, status, acceptance and completion fields. The native Windows engine does not require a Python or SciPy runtime.

Precision with evidence

Double arithmetic offers about 15–16 significant decimal digits; binary128 offers roughly 33–34. Recognised exact integer routes use arbitrary-precision integers independently. More printed digits do not establish accuracy. Residuals, accepted values, refinement estimates, exact checks and completeness indicators answer different questions.

Teaching and research applications

Mathematics and Applied Mathematics can explore roots, number theory and numerical analysis. Physics and engineering can investigate oscillations, boundary models, thermal fields and modes. Chemistry and Biotechnology can formulate kinetics, equilibrium or compartment models. Computer Science can study algorithms, structured results and native integration.

For a departmental pilot, choose a syllabus problem and a research model. Predict, solve and validate independently, then retain the model and exported evidence. Academic department and university licences, an automatic 14-day trial and signed-file activation support evaluation. Entitlements and deployment scope depend on the licence.

Scope: this is not a general symbolic or complex-root solver, optimiser, thermodynamic database, DAE engine or arbitrary nonlinear PDE package. Use the latest matching Desktop and engine updates for the facilities described here.

Request a demonstration with your equations

Patodia Infotech Private Limited | panna@patodiainfo.com
https://patodiainfo.com/EquationSolver/