Nonlinear systems and iterative methods

Begin with a residual and ask which solution is accessible. Study how curvature, scaling and conditioning affect a computed root.

Contents

1. **Choosing a derivative step in a nonlinear solve**
   A smaller finite-difference step eventually makes Newton iterations worse. Derive truncation and evaluation errors, compare a complex perturbation and a directional derivative, and test which numerical derivative is meaningful for the actual residual implementation.
2. **Accelerating fixed points with differences of residuals**
   A slowly convergent physical iteration contains information in the direction of successive errors. Use a scalar extrapolation and a small residual-minimisation problem to exploit that information, then test the extrapolated state against the original equation.
3. **Finding another equilibrium after one root is known**
   Newton iteration repeatedly finds the same structural equilibrium. Derive how deflation changes its basins while preserving other solutions, use a rank-one Jacobian update, and distinguish a list of discovered equilibria from a completeness certificate.
4. **Count enclosed zeros with a contour before locating them**
   A complex residual may have several nearby zeros whose Newton basins are hard to distinguish. Use the argument principle to count them, contour moments to estimate their locations and an independent perturbation test to certify a root cluster.
5. **Where does a complex Newton step take its starting point?**
   A single cubic has three equally simple roots. Derive the iteration, colour starting values by destination and use a separate convergence history to distinguish a bad first step from slow local convergence.
6. **Certifying a Newton step before the root is known**
   A small correction is meaningful only together with a curvature estimate. Construct a ball that contains a solution and use a frozen linear solve to make damping decisions independent of equation units.
7. **Which correction does an experiment identify?**
   Separate parameter combinations visible in a nonlinear fit, pass a regular fold by continuation and decide when to refine a boundary-value mesh before trusting its Newton residual.

Nonlinear systems and iterative methods · 1. Choosing a derivative step in a nonlinear solve

A smaller finite-difference step eventually makes Newton iterations worse. Derive truncation and evaluation errors, compare a complex perturbation and a directional derivative, and test which numerical derivative is meaningful for the actual residual implementation.

Derive competing contributions to error

Suppose f is smooth and you need its slope at x to form a Newton correction. A forward difference $D_h = [f(x+h) - f(x)]/h$ has Taylor expansion $f'(x) + hf''(x)/2 + O(h^2)$. A centred difference $[f(x+h) - f(x-h)]/(2h)$ instead cancels the even terms, leaving $f'(x) + h^2 f'''(x)/6 + O(h^4)$. Both limits are exact in real arithmetic as $h \to 0$. The program, however, returns approximate residual values, and a subtraction of nearby values can lose significant digits. If each absolute evaluation error is at most δ, the derivative error from those perturbations is at most $2\delta/h$ for forward difference and δ/h for the centred one.

Thus an illustrative forward error bound has shape $ah + b/h$; minimisation gives

$$h_* = \sqrt{b/a}.$$

The centred bound $ah^2 + b/h$ has minimiser $h_* = (b/(2a))^{1/3}$. Floating-point roundoff acting on $O(1)$ residual values often gives the familiar square-root and cube-root machine-precision scales, but the physical units of x, the scale of the derivative and noise from the residual solver must be included. A nonlinear function evaluated by an iterative PDE solve can have a much larger δ than roundoff. Repeating the inner solve with tighter tolerance can shift the optimal step considerably.

For $f(x) = e^x$ at zero, the forward leading error is $h/2$, centred error $h^2/6$. If $\delta = 10^{-10}$ as an absolute per-value uncertainty, a forward worst-case bound gives optimal $h = 2 \times 10^{-5}$, while the centred model gives

$$h = (3 \times 10^{-10})^{1/3} = 6.69 \times 10^{-4}.$$

The centred step can be much larger yet retain lower truncation error. This is a scale estimate using worst-case errors, not a guarantee about the phase or stochastic correlation of actual errors. If queries share common random numbers or a warm-started solver, errors can partly cancel or acquire a systematic derivative bias.

Obtain the slope from an imaginary perturbation

For a real-valued function with an analytic complex extension near the real argument, expand $f(x + ih) = f(x) + ihf'(x) - h^2 f''(x)/2 - ih^3 f'''(x)/6 + \cdots$. Taking its imaginary part and dividing by h gives second-order accuracy with no explicit subtraction of two nearly equal real numbers. For e^x at zero, the result is $\sin h/h$. At very small representable h the imaginary signal is itself proportional to h and may still be carried accurately by analytic elementary operations. This advantage

disappears if the computation discards imaginary parts, uses nonanalytic absolute values, branches on the perturbation, or applies a black-box routine using complex arithmetic for unrelated purposes. Automatic differentiation supplies another route through differentiable implemented operations, but it too needs the program's mathematical definition to match the derivative being requested.

Audit a Jacobian action before assembling a matrix

For $F : \mathbb{R}^n \to \mathbb{R}^m$, pick direction v and examine $R(h) = \|F(x + hv) - F(x) - hJv\|$. If a proposed Jacobian action is correct and F is twice differentiable, $R(h)$ initially scales like h^2. A wrong first derivative generally produces a leading $O(h)$ term. The test checks orientation, units, signs and chain rules without constructing a dense finite-difference matrix. Test several directions and scales; a wrong matrix can annihilate the one chosen direction. At very small h a roundoff or inner-solver floor eventually hides both ideal slopes. When working with constraints use perturbations remaining inside the physical domain or give derivatives in a local feasible parametrisation.

Find the useful step before a Newton update

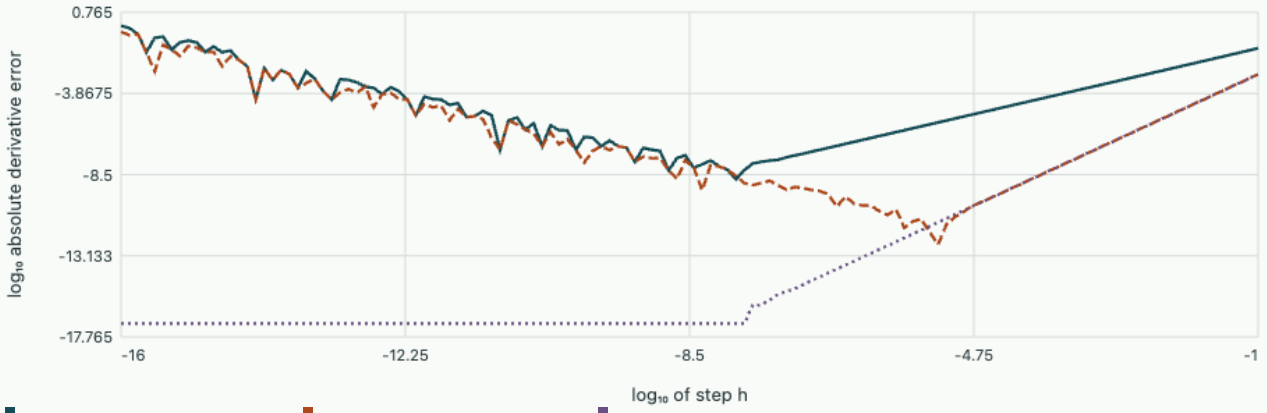
The experiment evaluates the forward, centred and complex-step slopes of a smooth elementary residual in floating-point arithmetic. It plots absolute errors on logarithmic scales and compares them with the exact derivative. Move the evaluation point and note that occasional accidental zeros are rounded at the plotting floor. Their exact location is not a robust guarantee for a different residual. Compare the range of steps with a reproducible error trend before interpreting one apparently perfect result.

Open this experiment online

Try the calculation

Evaluation point x **0**

Largest plotted $\log_{10}$ step **-1**



Forward difference · Centred difference · Complex-step imaginary part

Exact derivative exp(x) = 1.00000. Errors below 10^{-17} are drawn at the plotting floor. The complex exponential is evaluated analytically in real and imaginary parts.

Change the evaluation before changing the step

For the exponential the forward numerator has an exact factorisation $e^{x+h} - e^x = e^x(e^h - 1)$. A numerical library's `expm1(h)` evaluates the last factor accurately even for very small h, using a local series or a reduced-argument algorithm. Form the quotient as $e^x \operatorname{expm1}(h)/h$. Its mathematical truncation error is still $e^x h/2 + O(h^2)$. Its rounding behaviour differs because the two terms near e^x are never computed separately. At $x = 1$ and $h = 10^{-18}$, adding h to a binary64 value of one rounds back to one. The direct forward quotient then returns zero. The factored evaluation gives the derivative e to working precision.

This experiment diagnoses the source of an error curve. If the algebraic form is inaccessible, a finite-difference checker sees black-box function values and cannot apply the factorisation. If the entire residual is available, stable elementary functions can change the derivative accuracy of a Newton step without changing its formal order.

Tutorial · design tests for smoothness and implementation errors

Diagnose a forward-difference Newton iteration

For $f(x) = x^2 - 2$ at $x = 1.4$, obtain the exact dependence of a forward derivative estimate on h. Compare one Newton step using $h = 0.1$ with the exact Jacobian.

▼ **Work through the solution**

The difference quotient is

$$2x + h = 2.8 + h.$$

The residual is -0.04. The exact step gives

$$x_{\text{new}} = 1.4 + 0.04/2.8 = 1.414286;$$

using 0.1 gives 1.413793. Centred differences evaluate this polynomial's derivative exactly in real arithmetic for every nonzero h, but roundoff can spoil very small steps. The discrepancy at large h is a calculable systematic truncation, so merely tightening the outer Newton stopping tolerance cannot remove it.

Catch a direction missed by one test

A proposed Jacobian differs from the correct one by $E = \left(\begin{smallmatrix} 1 & -1 \\ 0 & 0 \end{smallmatrix}\right)$. Compare tests along $(1, 1)$ and $(1, 0)$ and show which observed slope would be misleading.

▼ **Work through the solution**

The first direction lies in the nullspace of E, so its Taylor remainder can show the expected h^2 slope despite an incorrect Jacobian. The second has $Ev = (1, 0)$, producing an $O(h)$ defect. Random directions augment a set of unit- or physics-informed probes, particularly when an error affects a single column or constraint block. One successful slope does not establish correctness of an entire matrix.

Determine whether complex step applies to a norm

A residual returns $f(x) = |x|$ using a complex modulus routine. At $x = 1$, what does a complex-step call produce? Is this evidence that the real function has zero derivative there?

▼ **Work through the solution**

The routine computes $|1 + ih| = \sqrt{1 + h^2}$, purely real, so the reported complex-step slope is zero. The real derivative at $x = 1$ equals one. The implementation used a nonanalytic map of a complex variable. At a known positive branch the analytic local extension $f(z) = z$ gives the correct derivative, while the real function at the kink $x = 0$ has no classical derivative. Check both the real regularity and what complex arithmetic actually executes before applying this technique.

Further reading

- Martins, Sturdza and Alonso · complex-step derivative approximation and implementation
- ETH Zürich · numerical methods for CSE, chapter 8 on nonlinear equations
- Course investigation · inexact Newton updates
- Solve an implicit cooling step with a changing residual

Nonlinear systems and iterative methods · 2. Accelerating fixed points with differences of residuals

A slowly convergent physical iteration contains information in the direction of successive errors. Use a scalar extrapolation and a small residual-minimisation problem to exploit that information, then test the extrapolated state against the original equation.

Eliminate an unknown leading error without knowing the root

Suppose a sequence obeys $x_j = x_* + cq^j$ for an unknown limit and mode multiplier $q \neq 1$. Subtract successive terms to find

$$\Delta x_{j+1} / \Delta x_j = q.$$

Eliminating q, c yields the Aitken extrapolation $\hat{x}_j = x_j - (x_{j+1} - x_j)^2 / (x_{j+2} - 2x_{j+1} + x_j)$. For a single exact geometric error it returns the limit after observing three values, even if the multiplier is negative and the iterates oscillate. If the denominator is near zero relative to numerical uncertainty, the extrapolation can amplify noise. A nearly constant first difference may indicate almost linear drift rather than a trustworthy asymptotic geometric error.

For $g(x) = 0.95x + 0.10$, ordinary iteration from zero gives $0, 0.10, 0.195$; the fixed point is 2. The second difference is -0.005 and the formula gives exactly 2. If the last reported value is rounded to 0.196, the denominator changes to -0.004, giving an extrapolated 2.50. Substituting this candidate into the original residual $g(x) - x$ gives -0.025: it is a worse fixed point than 2 but still has a modest residual because the fixed point itself is weakly conditioned. Always evaluate the actual residual and estimate its relation to state error if a meaningful physical tolerance is required.

Choose affine weights that minimise residual history

For a vector fixed-point map g, let $f_i = g(x_i) - x_i$. Seek a combination of recent residuals with small norm while preserving a fixed point under averaging. Anderson mixing chooses coefficients α_i minimising $\| \sum_i \alpha_i f_i \|$ subject to $\sum_i \alpha_i = 1$, then evaluates a candidate $x_{\text{new}} = \sum_i \alpha_i g(x_i)$ in a basic undamped formulation. The affine constraint allows negative coefficients, which is precisely what makes acceleration beyond ordinary averaging possible. For one-dimensional affine maps the combination can eliminate a single mode exactly.

For two residual vectors write their affine combination as $f_1 + \beta(f_0 - f_1)$. Its least-squares optimum is $\beta = -f_1^T (f_0 - f_1) / \|f_0 - f_1\|^2$ if the denominator is nonzero. With several vectors a small QR or SVD solve in terms of differences is preferable to building normal equations when differences are nearly collinear. Choose component units or a weighted norm consistent with the physical equation. Otherwise a residual expressed in large numerical units can dominate the fit and accelerate one variable while neglecting another important constraint.

For a strictly affine map $g(x) = Gx + c$,

$$f(\sum \alpha_i x_i) = \sum \alpha_i f_i.$$

The predicted residual at the affine combination of old states is exactly the linear combination. The candidate formed by applying g to it has residual $G\sum \alpha_i f_i$. For nonlinear g neither equality need hold globally. An Anderson residual minimisation is a local extrapolation hypothesis, not a proof that the next nonlinear state decreases residual or stays feasible. Evaluating $f(x_{\mathrm{new}})$, accepting a damped mixture when appropriate, and dropping near-dependent history are practical ways to test that hypothesis.

Background · What happens when two residual histories almost coincide?

Take two residuals from nearly consecutive states and minimise the norm of $r_1 + \alpha(r_2 - r_1)$. Write $d = r_2 - r_1$. Differentiating the squared norm with respect to α gives

$$\alpha = -\frac{d^T r_1}{d^T d}.$$

The denominator vanishes for duplicate residuals. A tiny denominator can also demand enormous positive and negative weights that cancel in the modelled residual. For scalar residuals 1 and 0.999 those weights are about -999 and 1000. Extrapolating the nonlinear states this far relies on local behaviour far outside the sampled interval. One practical response is to discard a nearly redundant column, then evaluate the true residual of the proposed state. A small residual of the affine model alone provides no acceptance test.

Separate fast and slow modes in a two-component relaxation

Use $g(x, y) = (0.9x + 0.1, 0.2y + 0.8)$, whose fixed point is $(1, 1)$. Start at $(0, 0)$, then $x_1 = (0.1, 0.8)$. Residuals are $f_0 = (0.1, 0.8)$ and $f_1 = (0.09, 0.16)$; their difference is $(0.01, 0.64)$. The two-history coefficient above is about -0.252. The predicted residual of the affine state is approximately $(0.0875, -0.0010)$, largely eliminating the fast second component while leaving the slower first. Three appropriately independent histories can eliminate two geometrically distinct error directions for this affine example if their residual vectors affinely span zero in exact arithmetic. Near-equal decay factors can make that calculation badly conditioned. A finite memory window trades acceleration against conditioning, storage and the cost of residual evaluation.

Follow a two-mode residual through a real mixing step

Set the slow multiplier to 0.90 and the second to 0.20. The first two ordinary states coincide in both runs because the accelerated iteration has no history at the start. Thereafter the calculator solves the two-vector minimisation at each iteration, forms the proposed mixed state and evaluates its actual residual. The second component initially dominates the residual norm even though it is the more rapidly damped mode. Notice how the mixed and plain curves separate as the slow component becomes limiting. The plotted logarithmic residual measures convergence to the stated map; the physical state error in mode j is its residual divided by the magnitude of one minus that mode multiplier.

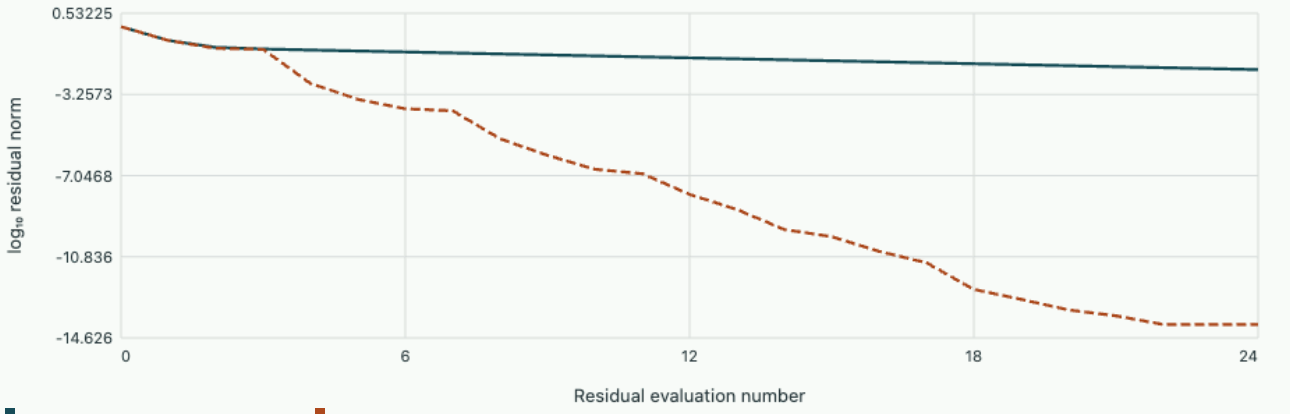
Move the two multipliers close together so the map becomes nearly a scalar relaxation, then choose a negative second multiplier so its errors alternate. Explain why affine mixing can nearly remove a single geometric mode, while nearly dependent histories make weights sensitive to evaluation noise. The implementation falls back to the ordinary map if the difference norm is already at the roundoff threshold. Very small residuals are marked at a finite plotting floor. Compare curves before that floor rather than fitting a convergence rate to the flat portion.

Open this experiment online

Try the calculation

Slow multiplier **0.9**

Second multiplier **0.2**



After six updates: ordinary residual 0.0531, mixed residual 0.000119. Plot floor 10^{-14}.

Tutorial · identify good and bad extrapolation data

Check an alternating scalar series

Three iterates are 1.50, 0.75 and 1.125. Infer the geometric multiplier and limit. How should the next value be checked?

▼ **Work through the solution**

The differences are -0.75 and $+0.375$, so $q = -0.5$. The Aitken prediction is

$$1.50 - 0.75^2/1.125 = 1.00.$$

If the geometric assumption persists, the next raw iterate would be $1 - 0.0625 = 0.9375$. Comparing both the new observed iterate and the actual residual at 1 tests the single-mode hypothesis. Three arbitrary numbers alone can always fit such a geometric extrapolation when the denominator is nonzero.

Show the connection to a secant root step

Use two scalar residuals $f_i = g(x_i) - x_i$. Derive the affine combination of x_0, x_1 whose predicted residual is zero and compare with a secant step for $f = 0$.

▼ Work through the solution

Solve $\alpha f_0 + (1 - \alpha) f_1 = 0$ to get

$$\alpha = -f_1/(f_0 - f_1).$$

The mixed state is $x_1 - f_1(x_1 - x_0)/(f_1 - f_0)$, the ordinary secant prediction. The combination of $g(x_i)$ is the same state because the weighted residual vanishes. For nearly equal residuals the denominator can be tiny, generating a large extrapolation. A bounded step or fallback to the base fixed-point map may be necessary if the candidate leaves the valid state domain.

Select a restart from nearly identical residual columns

Two stored vectors have residuals $(1, 0)$ and $(1, 10^{-9})$, measured with absolute component errors 10^{-7}. Can their difference support a reliable two-history fit?

▼ Work through the solution

The direction being used for the second independent sample is smaller than the evaluation uncertainty by two orders of magnitude. Its orientation and length can be dominated by numerical error, so a large coefficient in a least-squares solve carries little physical information. Drop the redundant direction or use a rank-revealing solve scaled to the expected residual error. Evaluate any retained candidate on the original model. A tiny predicted mixed residual from cancelling inaccurate large coefficients has no independent evidential value.

Further reading

- Walker and Ni · Anderson acceleration for fixed-point iterations
- Course investigation · contractive maps and error bounds
- Solve an implicit cooling step with a changing residual

Nonlinear systems and iterative methods · 3. Finding another equilibrium after one root is known

Newton iteration repeatedly finds the same structural equilibrium. Derive how deflation changes its basins while preserving other solutions, use a rank-one Jacobian update, and distinguish a list of discovered equilibria from a completeness certificate.

Prevent a known simple zero from attracting another iteration

Suppose a differentiable $F : \mathbb{R}^n \to \mathbb{R}^n$ has a known zero r, with nonsingular Jacobian $J(r)$. Near r, $F(r+h) = J(r)h + O(\|h\|^2)$, so its norm is bounded below by a positive constant times $\|h\|$ in a sufficiently small neighbourhood. Multiplying by a nonvanishing scalar multiplier away from r preserves all other zeros. Taking $M(x) = \|x - r\|^{-p} + \alpha$, $\alpha > 0$, yields deflated residual $G(x) = M(x)F(x)$ on $x \neq r$. Its norm behaves at least as a constant times $\|x - r\|^{1-p}$ near the simple root. For $p = 1$ it need not have a directional limit but cannot approach zero there; for $p > 1$ it diverges. The scalar shift α prevents the multiplier from vanishing at large distances.

If the root has multiplicity m in a scalar problem, $|F|$ can vanish like distance to the power m. An exponent $p < m$ can leave the deflated norm tending to zero, invalidating the reasoning for simple roots. In systems with singular Jacobians different directions can have different leading powers or a whole manifold of solutions can pass through r. One should identify symmetries and solve a properly constrained problem before treating every coordinate representation of a continuum of roots as a new equilibrium. Deflation changes a nonlinear solver's route; it does not make a singular physical problem isolated by decree.

Reuse a Newton correction with a rank-one change

Differentiate the scalar product to obtain

$$J_G = MJ_F + F(\nabla M)^T.$$

When J_F is invertible, let $d = -J_F^{-1}F$ be the ordinary Newton correction. Look for a deflated step $s = \beta d$. Applying J_G gives $\beta[-MF + F(\nabla M)^T d]$. Equating this to $-MF$ yields

$$s = \frac{d}{1 - (\nabla \log M)^T d}.$$

This is a convenient exact relation for a scalar deflation multiplier away from a vanishing denominator. The Jacobian of the modified problem can become singular at points where the original Newton step is perfectly well-defined. A denominator near zero can create an enormous proposed state change. A line search or trust region should verify the candidate against the chosen objective and the physically admissible variables. Once a different candidate root is located, evaluate the original F, its Jacobian conditioning and any invariant constraints independently.

See the change of basin in an analytical scalar model

Take $F(x) = x(x-1)(x+2)$ with roots $0, 1, -2$, and let $r = 0$ have already been found. On the positive half-line choose $M = 1/x$ for a transparent algebraic comparison, giving $G = (x-1)(x+2)$. This unshifted local example works on this domain; a global implementation often uses the positive shifted norm expression instead. From $x = 0.2$,

$$F = -0.352,\ F' = 3x^2 + 2x - 2 = -1.48.$$

Original Newton gives approximately -0.0378, entering the basin of the known zero. Deflated Newton has $G = -1.76,\ G' = 1.4$ and gives approximately 1.4571, entering a region directed toward the other positive root. Neither update used knowledge of the other roots to choose that direction.

For a polynomial with known exact factor, explicit algebraic division may be available. A numerical residual from a mechanics or fluid solver need not be a polynomial and generally offers no such division. Numerical deflation uses the known state and a norm to remove its attraction while continuing to evaluate the same physical governing equations. Use physical scaling in that norm. The distance between two states measured in metres and kelvins is arbitrary until component scales are chosen; an apparently huge separation in the deflation metric can represent tiny physical changes. If roots are known only approximately, multiplying a near-singular factor around a slightly wrong centre can leave the true nearby root or create severe numerical conditioning. Refine and verify each original root before including it in a list of deflation centres.

Change the Newton basin by dividing out a known root

Here the original residual is the cubic in the worked example. Its known zero at the origin is algebraically divided out, leaving a quadratic on the full line. This exact-factor experiment isolates the change in Newton geometry without introducing a singular numerical evaluation of an inverse distance. For both residuals, the calculator starts at the same chosen positive point and applies the same damping factor. At the default 0.20 the first ordinary step goes slightly negative and approaches zero; the deflated step passes one and converges back to it. The readout verifies the second candidate in the original cubic.

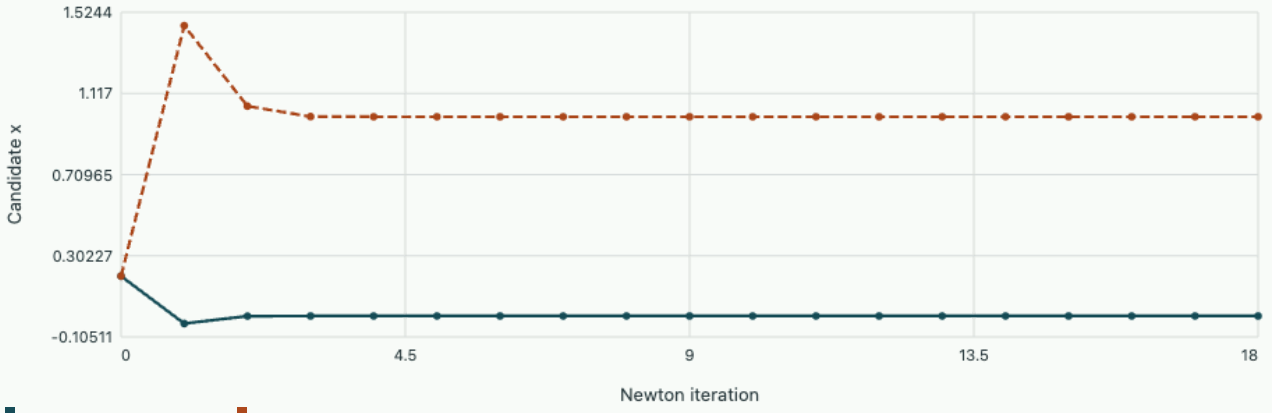
Try a smaller damping factor and compare the number of iterations needed before the two candidate sequences settle. Damping moderates each proposed correction but does not restore the removed root of the quadratic. A finite iteration count with heavy damping may end with a visible original residual; a current candidate is not an exact equilibrium merely because its curve has begun to flatten. This illustration relies on an exact polynomial factor, while deflation around a numerically inferred state requires an explicit centre tolerance and original-model verification.

Open this experiment online

Try the calculation

Initial x **0.2**

Newton damping **1**



Final candidates 0.000 and 1.000. Original cubic residual on the deflated branch 0.00e+0.

Tutorial · separate discovery, verification and completeness

Work through a simple shifted multiplier

For $F = x(x-1)$, $x > 0$, use $M = 1 + 1/x$ to deflate zero. Find the new residual, its value as $x \to 0^+$, and the deflated Newton step from $x = 0.5$.

▼ **Work through the solution**

Here $G = (x+1)(x-1) = x^2 - 1$, tending to -1 at the excluded zero. From $x = 0.5$, $G = -0.75,\ G' = 1$, giving $x_{\text{next}} = 1.25$. The original derivative $2x - 1$ vanishes exactly at the starting point even though the deflated derivative does not. The shortcut requiring J_F^{-1} is unavailable there; directly differentiating and solving for G remains valid. This example shows why a formula derived for nonsingular J_F should not become a universal implementation assumption.

Choose a multiplier for a double root

For $F(x) = (x-1)^2(x+1)$ deflate $r = 1$ with $M = |x-1|^{-p} + 1$. Compare $p = 1$ and $p = 2$. What limits a near-root termination test?

▼ **Work through the solution**

Near 1, $|F| \sim 2|x-1|^2$. With $p = 1$, $|G| \sim 2|x-1|$ still tends to zero; with $p = 2$, it tends to two. A test accepting just a small deflated residual can therefore return very close to the old double root when

using the inappropriate exponent. A small original residual also occurs over a relatively wide neighbourhood of a multiple root. Report a root-separation test and independent bounds or refinement where multiplicity affects the required state tolerance.

Construct a genuine completeness argument in one dimension

Several initial guesses and deflations recover the three roots of $x^3 + x^2 - 2x$. What additional reasoning certifies that the list is complete, and what can be reported if only a black-box smooth system is available?

▼ **Work through the solution**

The polynomial is nonzero of degree three, so it has at most three distinct complex roots. Substitution verifies three distinct real ones and proves completeness in this case. Multiple Newton solves by themselves would not establish that count. For a smooth black-box scalar problem over a compact interval one can partition it, enclose the function and its derivative, exclude root-free pieces and isolate all retained candidates when the bounds allow. In a higher-dimensional problem a list found by starts and deflation should be reported as verified discovered roots unless an independent argument covers the entire specified domain.

Further reading

- Farrell, Birkisson and Funke · deflation for distinct nonlinear solutions
- Course investigation · continuing an equilibrium branch
- Solve an implicit cooling step with a changing residual

Nonlinear systems and iterative methods · 4. Count enclosed zeros with a contour before locating them

A complex residual may have several nearby zeros whose Newton basins are hard to distinguish. Use the argument principle to count them, contour moments to estimate their locations and an independent perturbation test to certify a root cluster.

Turn winding of a residual into a root count

Let f be analytic inside and on a positively oriented simple closed contour Γ, with no zeros on it. Suppose z_j is an enclosed zero of multiplicity m_j. Locally $f(z) = (z - z_j)^{m_j} g(z)$ with $g(z_j) \neq 0$, so $f'/f = m_j/(z - z_j) + g'/g$. Cauchy's residue theorem gives

$$N = \frac{1}{2\pi i} \oint_\Gamma \frac{f'(z)}{f(z)}\, dz = \sum_j m_j.$$

A meromorphic f instead gives number of zeros minus poles, counted with multiplicity. On the image curve $f(\Gamma)$ the same integral equals its change of complex argument divided by 2π. This provides an alternative if only complex function evaluations are available. Both versions require that the contour avoid zeros and poles. Moving the contour through one changes its integer count discontinuously; a point extremely close to the boundary demands finer sampling or a different contour.

For a circle $z = c + Re^{i\theta}$, periodic trapezoidal quadrature evaluates the integral as the mean of $(z - c)f'(z)/f(z)$. The smoothness and distance to nearby singularities govern convergence. Simply rounding the result to the nearest integer can disguise severe undersampling. Compare independent contour radii and doubled angular grids, inspect the minimum $|f|$ on the boundary, and examine unwrapped phase increments. A grid with only endpoint samples of a rapidly rotating image can miss entire turns even when successive complex values look nearly identical. An upper bound on phase speed or adaptive interval enclosure is needed for a rigorous count from discrete samples.

Locate small groups after counting them

Multiply the integrand by z^k. Its residues become $m_j z_j^k$, giving power sums $s_k = (2\pi i)^{-1} \oint z^k f'/f\, dz$. For exactly one simple enclosed root, s_1 is its location. For two simple roots z_1, z_2, obtain their sum s_1, and their product $(s_1^2 - s_2)/2$ from the identity $(z_1 + z_2)^2 = z_1^2 + z_2^2 + 2z_1 z_2$. Solve the resulting quadratic for initial guesses, then polish on the original residual and recheck the contour count. For large clusters reconstructing a high-degree polynomial from power sums can be ill-conditioned; recursively subdivide or use a contour method designed for stable extraction.

Take $f(z) = (z - 0.2)(z + 0.4)(z - 2)$ on the unit circle. Its count is two; its moments are

$$s_1 = -0.2,\ s_2 = 0.20.$$

The reconstructed product is $(0.04 - 0.20)/2 = -0.08$, giving $z^2 + 0.2z - 0.08 = 0$ and roots $0.2, -0.4$. The exterior factor changes the residual values and Newton basins but contributes no interior pole of f'/f, hence no moment. A contour enclosing all three instead changes the power sums. Severe

cancellation can occur when several large root powers sum to a small number. Shift and scale coordinates about the contour centre before evaluating high-order moments.

Use a boundary inequality to protect a count under perturbation

If an analytically tractable g and target $f = g + h$ satisfy $|h(z)| < |g(z)|$ on Γ, then the continuous homotopy $g + th,\ 0 \le t \le 1$, has no boundary zero. Its winding number around zero cannot change continuously between distinct integers, so f and g enclose the same number of zeros. This is Rouché's theorem, useful when a perturbation is small on the boundary even if it strongly splits individual roots inside. The strict bound must cover the entire curve. Observations at a handful of sample points without a bound between them are numerical evidence, not a theorem.

For $f(z) = (z-1)^2 + \epsilon z,\ |\epsilon| = 0.001$, use a circle of radius 0.1 about one. Its unperturbed modulus is 0.01 and the perturbation has magnitude at most 0.0011. Both zeros stay inside this circle. Their individual shifts can be of order $\sqrt{|\epsilon|}$ because the reference zero is double. A small coefficient perturbation can therefore leave a reliably countable cluster while making its individual members comparatively sensitive. This distinction between a cluster as a whole and a single ill-conditioned root is important when a model residual depends on uncertain fitted coefficients.

Count one movable complex zero by discrete contour integration

The experiment integrates the logarithmic derivative of a polynomial with one zero strictly inside the unit circle. Its exact count is therefore one everywhere in the slider range. The real and imaginary parts of the quadrature estimate are plotted separately against the number of uniformly spaced angular samples. At radius 0.9 on the positive real axis, eight samples give about 1.756; rounding the real part would report two zeros. Moving the root farther from the contour makes convergence faster without changing the true count.

Rotate the root slightly while keeping its radius fixed. The absolute distance from the contour is unchanged, but its phase relative to the sampling nodes changes. A nonzero imaginary estimate and oscillatory errors may appear even though the exact winding number is real. The finite-grid formula for a root at complex a is one divided by one minus a raised to the number of samples. Use it to predict locations of large errors before reading the graph. An apparent agreement for one particular sample count can be accidental; compare successive refinements and the complex error.

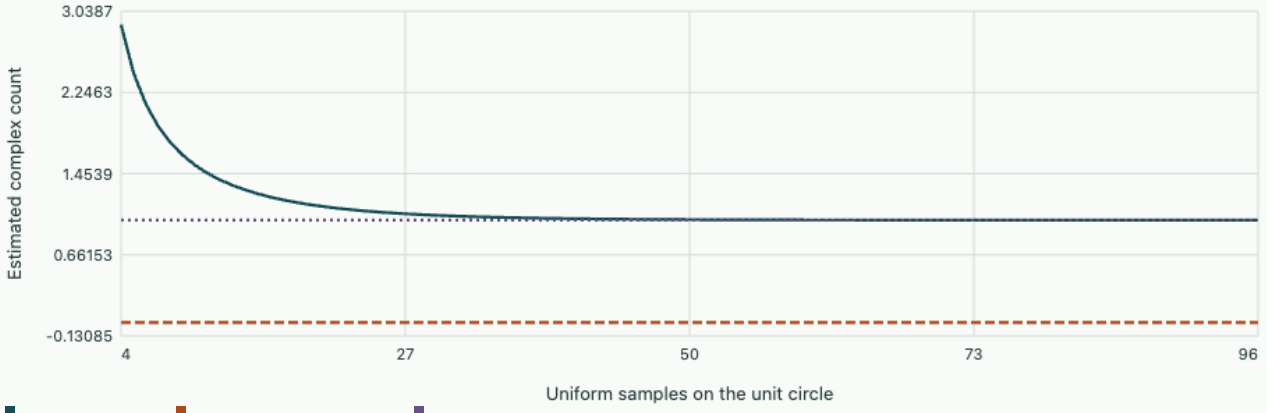
Open this experiment online

Try the calculation

Root radius **0.9**

Root angle (degrees) **0**

Inspected samples **8**



At 8 samples: 1.75583 + (-0.00000)i. True count 1.

Tutorial · audit a contour computation

Calculate the exact aliasing of a one-root count

For $f(z) = z - a$ on the unit circle with $|a| < 1$, use M equally spaced points. Derive the trapezoidal estimate of N and evaluate it for real $a = 0.9, M = 8$.

▼ **Work through the solution**

At each point $zf'/f = 1/(1 - a/z) = \sum_{k \geq 0} a^k z^{-k}$. Averaging roots of unity cancels all terms except powers divisible by M, giving $N_M = 1/(1 - a^M)$. At $M = 8$ it is about 1.756, which would round to the wrong integer two. Increasing to $M = 64$ gives about 1.00118. This explicit example illustrates why apparent smoothness of the original analytic function does not ensure adequate sampling of its logarithmic derivative.

Recover a repeated root from moments

A contour yields $N = 2, s_1 = 1.2, s_2 = 0.72$ to high numerical accuracy. Infer the candidate roots and explain the sensitivity to a small error in s_2.

▼ **Work through the solution**

The product is $(1.44 - 0.72)/2 = 0.36$; the polynomial $z^2 - 1.2z + 0.36$ has double root 0.6. A change δ in s_2 changes the product by $-\delta/2$, giving inferred splitting $\pm\sqrt{\delta/2}$ relative to 0.6. Tiny moment noise can therefore mimic two distinct roots. Verify a multiplicity claim against derivative information on the original function and clarify whether distinct roots can actually be resolved.

Explain a zero net winding that hides a zero

A rational residual has a simple zero at 0.1 and a simple pole at -0.2 inside the unit circle. What would the count report and how could one recover the desired zero count?

▼ Work through the solution

The logarithmic derivative has residues plus one and minus one, so net winding is zero even though a zero exists. If the pole and its order are known, add the pole count to the measured winding or multiply by a known denominator that removes it and then check the resulting analytic function. A near pole–zero cancellation can be difficult to resolve numerically. Treating a rational or discretely singular residual as a globally analytic polynomial would make the count's interpretation incorrect.

Further reading

- ETH Zürich · numerical methods for CSE, polynomial roots and nonlinear systems
- NIST Digital Library of Mathematical Functions · contour integration, residues and zeros
- Course investigation · polynomial roots and conditioning
- Solve an implicit cooling step with a changing residual

Nonlinear systems and iterative methods · 5. Where does a complex Newton step take its starting point?

A single cubic has three equally simple roots. Derive the iteration, colour starting values by destination and use a separate convergence history to distinguish a bad first step from slow local convergence.

Continue the tangent step into the complex plane

A holomorphic function has a complex linear approximation $f(z+s) \approx f(z) + f'(z)s$. Requiring this approximation to vanish gives $s = -f(z)/f'(z)$ wherever the derivative is nonzero. For $f(z) = z^3 - 1$, the resulting map is

$$N(z) = \frac{2z}{3} + \frac{1}{3z^2}.$$

The second term is a pole at zero. Starting at a small positive number sends the next iterate far to the right, after which large iterates return approximately by multiplication by 2/3. At a root r, write $z = r + e$ and substitute $r^3 = 1$. Exact cancellation of the linear error gives

$$N(r+e) - r = e^2(3r + 2e)/[3(r+e)^2].$$

Sufficiently near r, the new error is about e^2/r. This argument predicts the speed after entry into a local neighbourhood. It has no information about which root a distant initial point will reach.

Let $s = \exp(2\pi i/3)$. Since $s^3 = 1$, symmetry gives

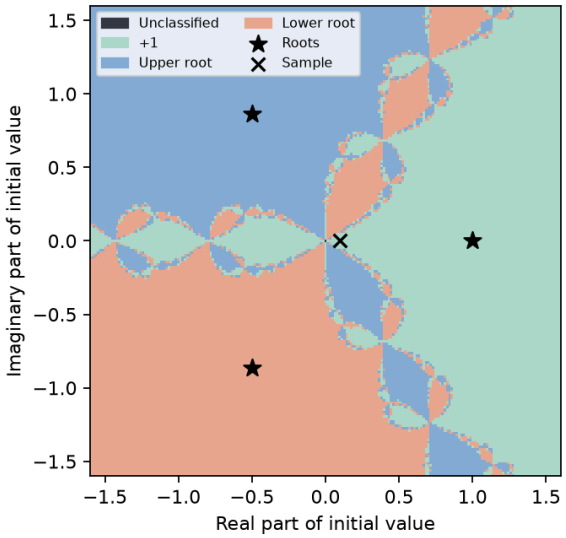
$$N(sz) = sN(z).$$

The entire sequence, including its root, rotates with the start. Conjugate starting points likewise produce conjugate histories because the coefficients are real. These are two independent checks for numerical basin maps. We classify a pixel only once its residual falls below a declared tolerance. A starting pole and a trajectory that exhausts its iteration allowance can both appear unclassified. Increasing the allowance separates these cases for many pixels.

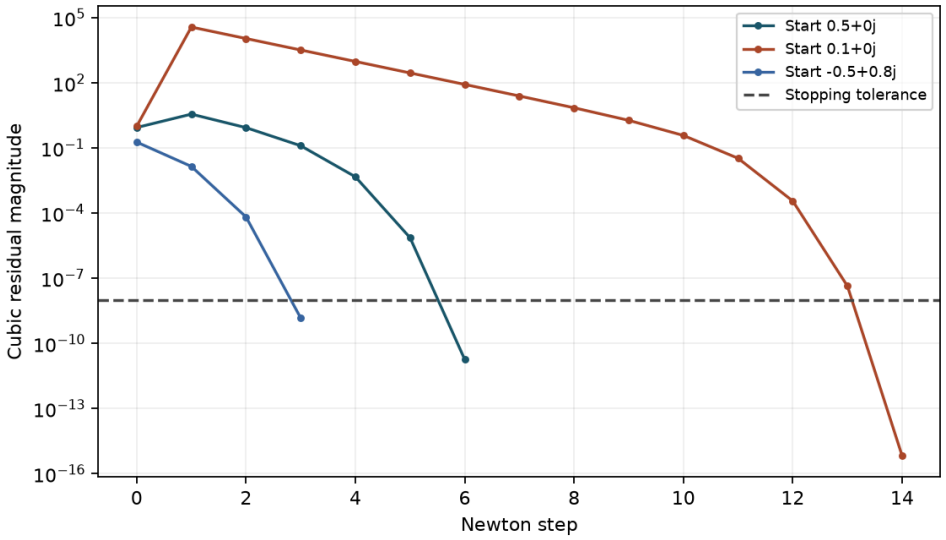
Follow a detour before quadratic convergence

From $z_0 = 1/2$ the first iterate is $1/3 + 4/3 = 5/3$. The next is $10/9 + 3/25 = 277/225 \approx 1.23111$. The error initially rises from 0.5 to 0.667. By contrast, starting at 0.1 gives $z_1 = 33.4$. The subsequent near-geometric return explains the long shoulder in the second figure. Both starts ultimately approach the real root.

Move the orange sample by writing a complex number such as $-0.5 + 0.85i$ in Python as `-0.5+0.85j`. The sample mark and the residual trace follow the same new starting point. The intricate boundaries in the image are computed from independent point trajectories. Refining the grid examines finer samples of those boundaries; the speed on the residual graph belongs to one particular start.



Each grid point starts an independent cubic iteration. Dark grey means the residual criterion was never met within the allowed steps, including the pole at zero.



The dashed line is the actual residual tolerance. A small initial magnitude can produce a large first residual followed by a long return.

Exercise · obtain two basin colours for free

Suppose $0.75 + 0.10i$ reaches the real root in five updates. What are the destinations and update counts after multiplying the start by either non-real cube root of one?

▼ Work through the solution

Write the original iterates as z_k. By induction $N^k(sz_0) = sz_k$ and $N^k(s^2 z_0) = s^2 z_k$. The destinations are s and s^2. Since $(sz_k)^3 - 1 = z_k^3 - 1$, every residual and hence the first crossing of the same threshold is unchanged. Differences in a finely sampled plot may arise because the rotated point falls between pixels.

Exercise · quantify the final two digits

For a real iterate $z = 1.01$, estimate the next error by the quadratic approximation and calculate its exact value using the error formula.

▼ **Work through the solution**

The leading term gives $0.01^2 = 10^{-4}$. The exact error is $10^{-4}(3.02)/(3 \times 1.01^2) \approx 9.8683 \times 10^{-5}$. The difference is of the next order in the initial error. The small residual after this update is about three times the error because $f'(1) = 3$.

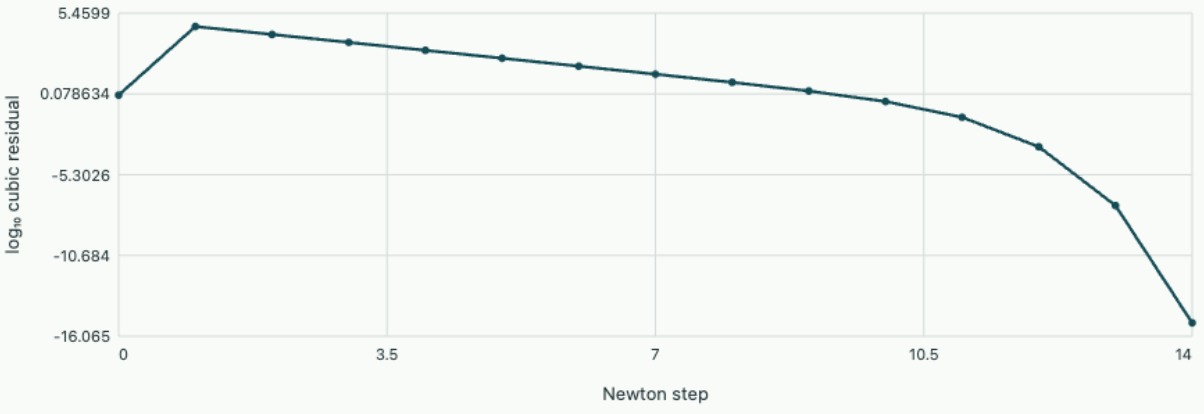
Change parameters · Explore this notebook model

Change one input at a time and compare the recalculated curves with the argument above. The browser evaluates this small model locally; the accompanying numerical values state its assumptions and reference checks. Reset restores the reference inputs, and the data can be downloaded as CSV.

Open this calculation with adjustable inputs

Try the calculation

Real initial value	0.1
Imaginary initial value	0
Maximum iterations	30



Chosen complex starting point

Converged to 1 + 0 i. Move either coordinate slightly near a basin boundary to test sensitivity.

Open editable notebook Download .ipynb

Nonlinear systems and iterative methods · 6. Certifying a Newton step before the root is known

A small correction is meaningful only together with a curvature estimate. Construct a ball that contains a solution and use a frozen linear solve to make damping decisions independent of equation units.

Bound all future corrections from initial data

Let F be differentiable on a convex neighbourhood, $J_0 = F'(x_0)$ invertible, and fix an induced norm. Suppose $\|J_0^{-1}F(x_0)\| \le b$ and $\|J_0^{-1}(F'(x) - F'(y))\| \le L\|x - y\|$. These estimates must hold on the entire proposed ball. Define the scalar majorant $g(t) = b - t + Lt^2/2$. If $2Lb < 1$, its smaller zero is $r = 2b/(1 + \sqrt{1 - 2Lb})$. Require the closed ball of that radius to lie in the domain.

Set $t_0 = 0$ and apply scalar Newton to g. Since g is decreasing and convex on $[0, r]$, the iterates increase to r. At a point no farther than t from x_0, the Neumann series for $J_0^{-1}F'(x) = I + E$, $\|E\| \le Lt < 1$, bounds $\|F'(x)^{-1}J_0\| \le 1/(1 - Lt)$. If s is an exact Newton correction, the constant and linear terms cancel. Integrating the Jacobian variation along that step gives

$$\|J_0^{-1}F(x + s)\| \le L\|s\|^2/2.$$

At the scalar Newton points exactly the same formula holds with equality for $g(t_{j+1}) = L(t_{j+1} - t_j)^2/2$. Induction now gives $\|x_{j+1} - x_j\| \le t_{j+1} - t_j$. The vector sequence is Cauchy and stays in the ball, its limit is a root and $\|x_* - x_j\| \le r - t_j$. Two roots in that ball would be joined by an averaged Jacobian differing from the identity, after scaling, by at most $Lr < 1$. It is invertible, so the roots coincide.

Use the linearised correction as the measuring stick

Premultiply the equations by any fixed nonsingular matrix S. The Newton correction and the majorant data stay the same since $(SJ_0)^{-1}S = J_0^{-1}$. The Euclidean norm of the raw residual generally changes. For a trial factor λ, form $s = -J^{-1}F(x)$ and $\bar{s} = -J^{-1}F(x + \lambda s)$ with the same factorisation. Linear prediction gives $\bar{s} = (1 - \lambda)s + e$, where local curvature bounded in the current scaled norm implies $\|e\| \le L_x\lambda^2\|s\|^2/2$. Backtrack until $\|e\| \le \lambda\|s\|/2$. Then $\|\bar{s}\| \le (1 - \lambda/2)\|s\|$. For smooth regular points sufficiently small steps pass. Rebuild the Jacobian after acceptance. Global convergence requires additional bounds keeping iterates within a regular level set.

For $F = (x_1^2 - 2, x_2^2 - 1)$ start at $(1.5, 0.9)$ in the maximum norm. Here $b = 0.19/1.8$ and $L = \max(1/1.5, 1/0.9)$. Both follow directly from the diagonal Jacobian. With $2Lb < 1$ the positive quadrant contains the certified ball.

OneSimulation experiment · Test the radius after a change of units

Replace a starting coordinate by a smaller positive value. Recheck the certificate before iterating.

Download the executable notebook · Python source

Adjust the experiment in the browser. Change the two starting coordinates for $F = (x_1^2 - 2, x_2^2 - 1)$. The plot recomputes the iterates, the maximum-norm correction β, the preconditioned derivative bound $L = \max(1/x_{1,0}, 1/x_{2,0})$, and the guaranteed radius whenever $\beta L \leq 1/2$. Move beyond the certified region and observe that failure of this sufficient test does not establish divergence. Rescale the first equation by up to six orders of magnitude: its Jacobian must receive the same scale. The browser uses the explicit diagonal solve; the notebook performs the corresponding OneSimulation dense solve.

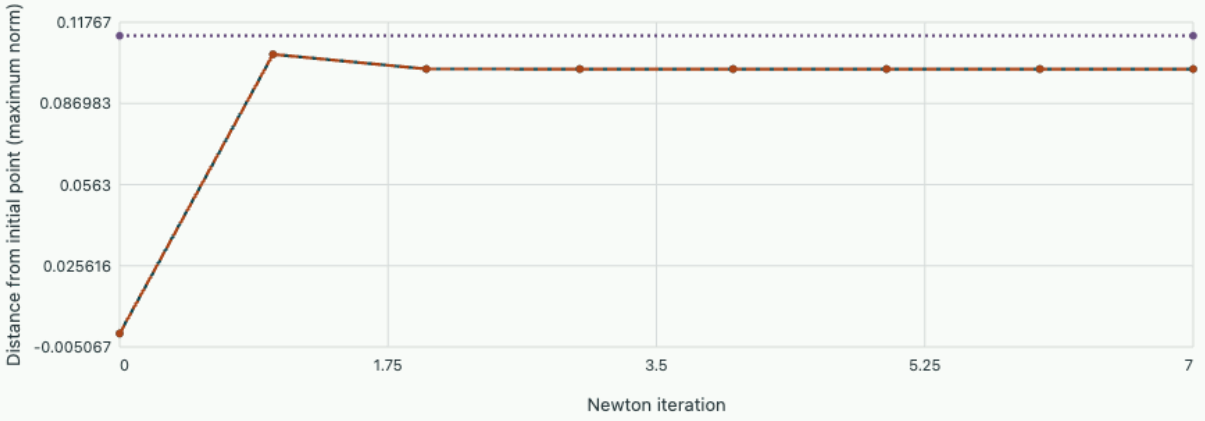
Open the live experiment

Try the calculation

Initial x_1 **1.5**

Initial x_2 **0.9**

$\log_{10}$ first-equation scale **4**



β = 0.10556, L = 1.1111, βL = 0.11728. The certified radius is 0.1126. All displayed iterates remain in that ball. Rescaling a whole equation and its Jacobian leaves the Newton step unchanged up to rounding.

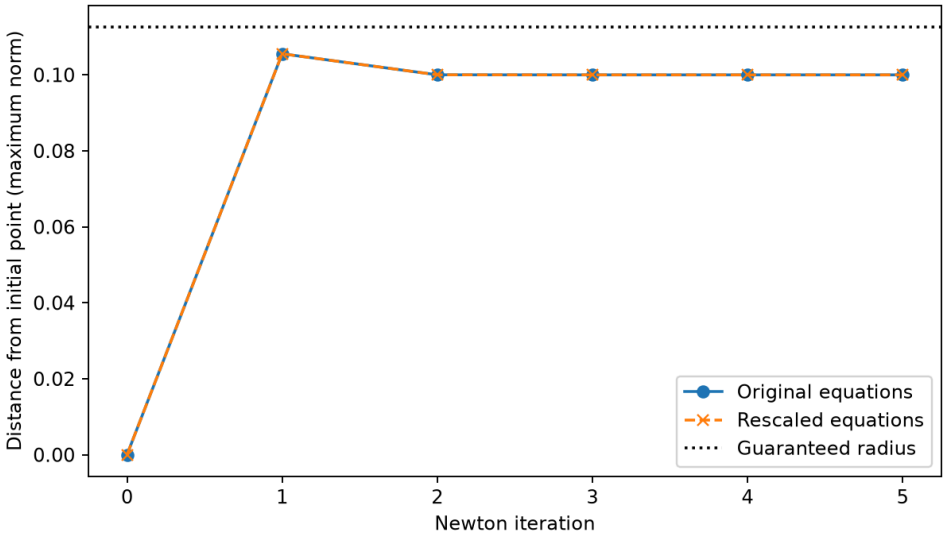
```
"""Test a Newton certificate without knowing the solution in advance."""
import matplotlib.pyplot as plt
import numpy as np
from onesim.numerics.linear.matrices.dense_linear_algebra import solve_dense

start = np.array([1.5, 0.9])
target = np.array([2.0, 1.0])
beta = np.linalg.norm((start**2 - target) / (2 * start), np.inf)
lipschitz = max(1 / start)
radius = 2 * beta / (1 + np.sqrt(1 - 2 * beta * lipschitz))
print(f"beta={beta:.6f}, L={lipschitz:.6f}, certified radius={radius:.6f}")
fig, ax = plt.subplots(figsize=(7, 4), layout="constrained")
for label, scale in [("Original equations", [1., 1.]), ("Rescaled equations", [1e4, 0.01])]:
    x = start.copy()
    distances = [0.0]
    for _ in range(5):
        residual = np.asarray(scale) * (x*x - target)
```

```
        jacobian = np.diag(2 * np.asarray(scale) * x)
        x -= solve_dense(jacobian, residual)
        distances.append(np.linalg.norm(x-start, np.inf))
    assert max(distances) <= radius and np.allclose(x**2, target)
    ax.plot(distances, "o-" if scale[0] == 1 else "x--", label=label)
ax.axhline(radius, color="black", ls=":", label="Guaranteed radius")
ax.set(xlabel="Newton iteration", ylabel="Distance from initial point (maximum norm)")
ax.legend()
fig.savefig("numerics-certified-newton.png", dpi=170)
```

Output from the reference run

```
beta=0.105556, L=1.111111, certified radius=0.112599
```



Two Newton paths coincide inside the same a priori convergence ball.

Tutorial on certification and damping

Compute the first majorant step and use it as a bound

▼ Work through the solution

Scalar Newton at zero gives $t_1 = b$. The vector correction has size b. After that move the distance to the root is at most $r - b$, which can be evaluated without substituting a known root.

Change the starting point to a nearly singular configuration

▼ Work through the solution

For $(1.5, 0.1)$, the second correction is $0.99/0.2 = 4.95$ and $L = 10$. The majorant gives no conclusion. For that coordinate a full step reaches 5.05, whereas halving the step decreases its size.

The failure of the sufficient inequality leaves open whether an actual iteration converges.

Deuflhard · local and global Newton methods

Nonlinear systems and iterative methods · 7. Which correction does an experiment identify?

Separate parameter combinations visible in a nonlinear fit, pass a regular fold by continuation and decide when to refine a boundary-value mesh before trusting its Newton residual.

Solve only for directions constrained by data

Given covariance $C > 0$, whiten the residual $r = C^{-1/2}(f(x) - d)$ and its Jacobian J. Gauss–Newton minimises $\|r + Js\|^2/2$. Write $J = U\Sigma V^T$. Components orthogonal to its range cannot be changed to first order. Each reliably measured direction obeys $v_i^T s = -u_i^T r/\sigma_i$. A direction in the nullspace is free. Setting its correction to zero selects the minimum Euclidean norm in the chosen parameter units.

$$s_\tau = -\sum_{\sigma_i > \tau} (u_i^T r/\sigma_i) v_i, \qquad \|s_\tau\| \le \|r\| / \min_{\sigma_i > \tau} \sigma_i.$$

If the forward derivative carries error e_J, singular values below $\|e_J\|$ cannot be distinguished reliably from zero. Also compare predicted changes to measurement noise over physically allowed parameter displacements. Damping with a penalty gives factors $\sigma_i/(\sigma_i^2 + \lambda^2)$ and is continuous in the singular values. A penalty adds assumptions about scales.

Take two equal-noise readings whose model rows depend only on $a + b$, with sensitivities $(1, 1), (2, 2)$. At $a = b = 0$ readings $(3, 6)$ demand $a + b = 3$. Minimum norm assigns $a = b = 1.5$. Changing parameter units can change that individual assignment while retaining the identifiable sum. A new observation of $a - b$ adds an independent singular direction.

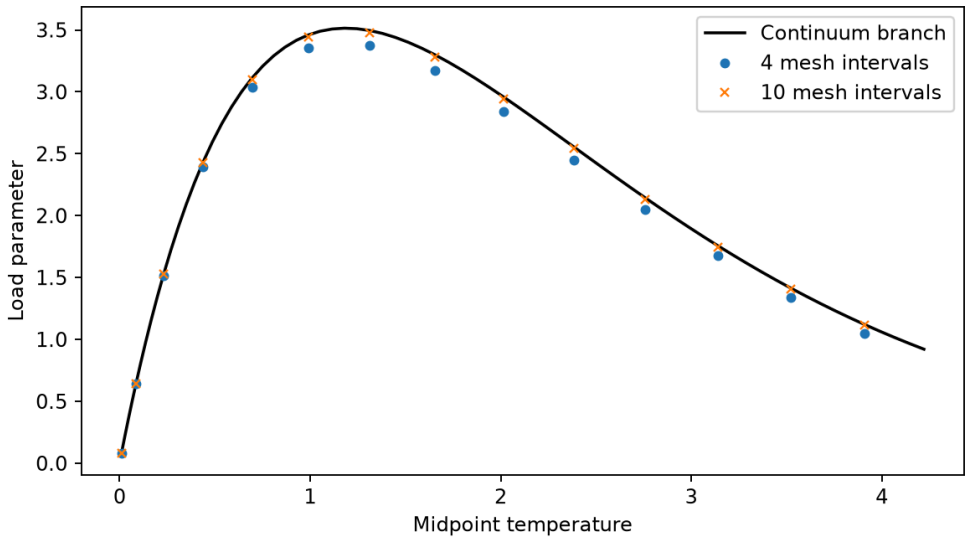
Exchange a failing parameter for a curve coordinate

Let $F(u, p) = 0$ be n equations in $n + 1$ unknowns and suppose $[F_u, F_p]$ has row rank n. Its nullspace supplies a unit tangent t oriented with the preceding tangent. Predict $z_p = z_k + ht$. The corrector solves $F(z) = 0,\ t^T(z - z_p) = 0$. At z_k the augmented derivative is invertible since its nullspace would lie both along t and perpendicular to it. This remains true locally for sufficiently small prediction steps, including a simple fold with singular F_u.

For a boundary-value application consider $u'' + \lambda e^u = 0,\ u(0) = u(1) = 0$. A centred grid gives $F_i = (u_{i-1} - 2u_i + u_{i+1})/\Delta x^2 + \lambda e^{u_i}$. The augmented rows have tridiagonal state derivative with diagonal $-2/\Delta x^2 + \lambda e^{u_i}$ and parameter column e^{u_i}. Use grid-weighted state inner products for tangents so refinement does not change the meaning of arclength. Halve h and repeat the prediction if corrector displacements are too large relative to h. If halving reduces the displacement by about four, the quadratic predictor defect is a geometrical effect.

How can a perfectly converged branch still predict a wrong maximum load?

Newton controls the residual of the chosen grid only. Correct the same branch point on a grid of half the spacing, matching arclength or a monotonic observed amplitude rather than load at the fold. In the regular second-order regime divide the fine–coarse output difference by three to estimate the fine-grid error. Refine the grid when this exceeds the output tolerance. Tighten the augmented linear and nonlinear solves enough that their estimated effect on that output is smaller.



A small nonlinear residual places a point on its discrete branch. Increasing the number of intervals changes the predicted turning load even after the nonlinear solves converge.

Tutorial on a fold with an exact reference

Find a mesh-independent check for the boundary-value branch

▼ **Work through the solution**

Set $u(x) = 2\log[\cosh(a/2)/\cosh(a(x-1/2))]$. Then $u'' = -2a^2\operatorname{sech}^2(a(x-1/2))$ and $e^u = \cosh^2(a/2)\operatorname{sech}^2(a(x-1/2))$. Hence $\lambda = 2a^2/\cosh^2(a/2)$ and the end values vanish. This parametric curve can test either side of the turning point.

Locate the fold and choose a comparison observable

▼ **Work through the solution**

Differentiating $\log\lambda$ gives $2/a - \tanh(a/2)$. Solve $a\tanh(a/2) = 2$, yielding a near 2.40 and λ near 3.51. The midpoint temperature $2\log\cosh(a/2)$ is strictly increasing for positive a, making it a well-defined matching observable across meshes.

Deuflhard · Gauss–Newton and continuation

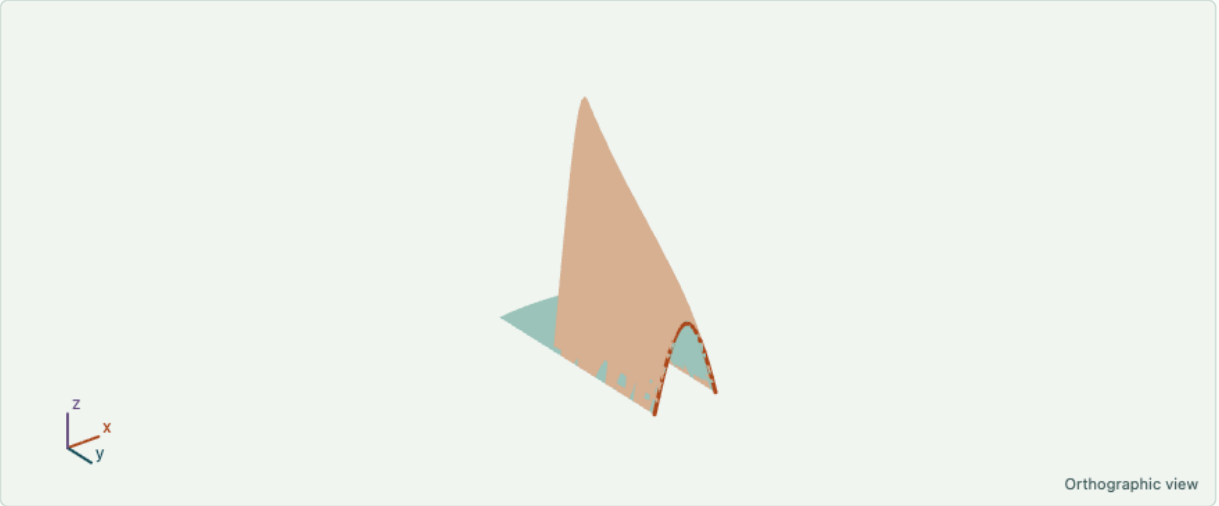
Change parameters · Which correction does an experiment identify?

Change one input at a time and compare the recalculated curves with the argument above. The browser evaluates this small model locally; the accompanying numerical values state its assumptions and reference checks. Reset restores the reference inputs, and the data can be downloaded as CSV.

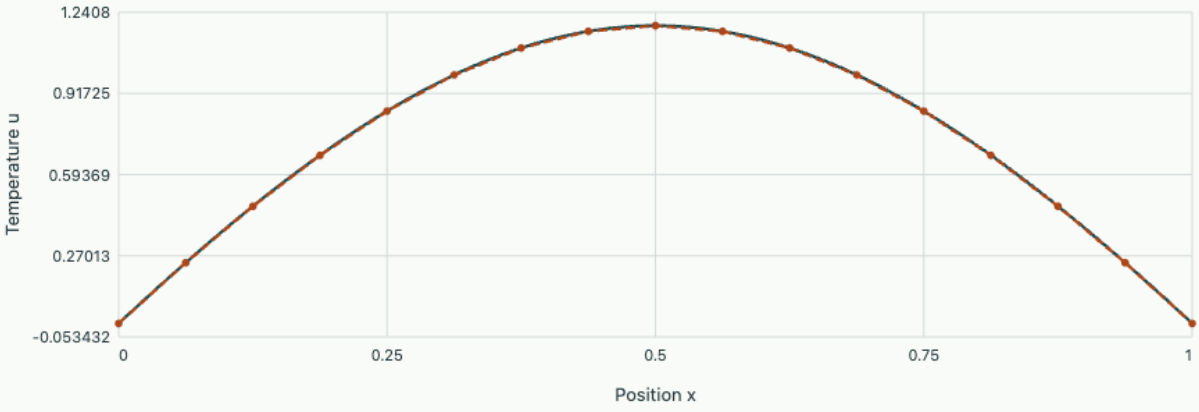
Open this calculation with adjustable inputs

Try the calculation

Branch parameter a	2.4
Mesh intervals	16



A three-dimensional graph of a family of one-dimensional boundary-value solutions. The folded sheet shows two temperature profiles at some loads; its vertical coordinate is temperature. The highlighted line is the computed finite-difference branch at the selected midpoint temperature. Axes: x = x; y = load λ; z = temperature u.



Analytical fixed-midpoint branch Augmented finite-difference solve

Exact load λ = 3.5138, discrete λ = 3.5067, augmented residual 5.4589e-13. Matching midpoint temperature allows comparison on both sides of the fold.

Download the executed OneSimulation notebook · Python source

The notebook reproduces the reference case and exposes the mesh or geometry parameters. Its saved output below comes from the actual OneSimulation solver.

```
"""Follow both Bratu branches using OneSimulation augmented Newton solves."""
import argparse
import matplotlib.pyplot as plt
import numpy as np
```

```
from onesim.numerics.linear.matrices.dense_linear_algebra import solve_dense

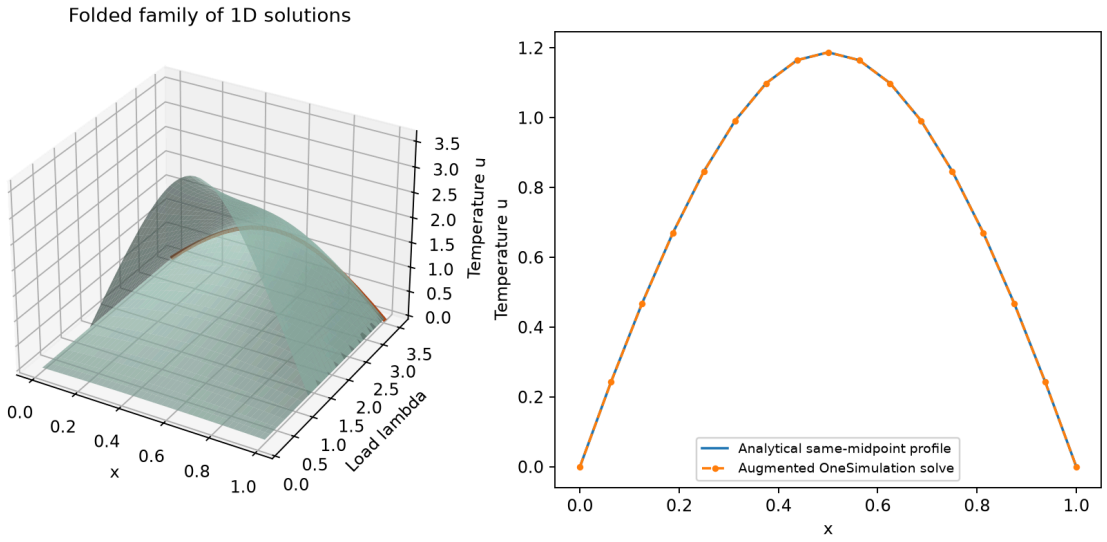

def branch(parameter=2.4, intervals=16):
    if parameter <= 0 or intervals < 4 or intervals % 2:
        raise ValueError("Use a positive branch parameter and an even interval count >= 4")
    x = np.linspace(0, 1, intervals+1)
    exact = 2*np.log(np.cosh(parameter/2)/np.cosh(parameter*(x-.5)))
    reference = 2*parameter**2/np.cosh(parameter/2)**2
    state, load = exact[1:-1].copy(), reference
    h = 1/intervals
    for iteration in range(15):
        full = np.r_[0., state, 0.]
        residual = np.r_[(full[:-2]-2*state+full[2:])/h**2 + load*np.exp(state),
                         state[intervals//2-1]-exact[intervals//2]]
        if np.linalg.norm(residual) < 1e-10:
            break
        jacobian = np.zeros((intervals, intervals))
        jacobian[:-1, :-1] = (np.diag(-2/h**2 + load*np.exp(state))
                              + np.diag(np.ones(intervals-2)/h**2, 1)
                              + np.diag(np.ones(intervals-2)/h**2, -1))
        jacobian[:-1, -1] = np.exp(state)
        jacobian[-1, intervals//2-1] = 1
        correction = solve_dense(jacobian, -residual)
        state += correction[:-1]
        load += correction[-1]
    assert np.linalg.norm(residual) < 1e-8
    return x, np.r_[0., state, 0.], load, reference, np.linalg.norm(residual)


def experiment(parameter=2.4, intervals=16):
    x, state, load, reference, residual = branch(parameter, intervals)
    print(f"Analytical load: {reference:.9f}; discrete load: {load:.9f}")
    print(f"Augmented Newton residual: {residual:.3e}")
    refined = branch(parameter, 2*intervals)
    assert abs(refined[2]-reference) < abs(load-reference)
    print(f"Load error after doubling the mesh: {abs(refined[2]-reference):.3e}")
    fig = plt.figure(figsize=(10, 4.6), layout="constrained")
    view = fig.add_subplot(121, projection="3d")
    xx, aa = np.meshgrid(np.linspace(0, 1, 45), np.linspace(.3, 5, 45))
    lam = 2*aa**2/np.cosh(aa/2)**2
    field = 2*np.log(np.cosh(aa/2)/np.cosh(aa*(xx-.5)))
    view.plot_surface(xx, lam, field, color="#9dc6b9", alpha=.7, linewidth=0)
    view.plot(x, np.full_like(x, load), state, color="#ad491d", lw=2.5)
    view.set(xlabel="x", ylabel="Load lambda", zlabel="Temperature u", title="Folded family of 1D
solutions")
    ax = fig.add_subplot(122)
    ax.plot(x, 2*np.log(np.cosh(parameter/2)/np.cosh(parameter*(x-.5))), label="Analytical same-
midpoint profile")
    ax.plot(x, state, "o--", ms=3, label="Augmented OneSimulation solve")
    ax.set(xlabel="x", ylabel="Temperature u")
    ax.legend(fontsize=8)
    fig.savefig("complete-numerics-bratu.png", dpi=170)
    return load, residual


if __name__ == "__main__":
    parser = argparse.ArgumentParser(description=__doc__)
    parser.add_argument("--parameter", type=float, default=2.4)
    parser.add_argument("--intervals", type=int, default=16)
    args, _ = parser.parse_known_args()
```

```
    experiment(args.parameter, args.intervals)
    plt.show()
```

```
Analytical load: 3.513830356; discrete load: 3.506676842
Augmented Newton residual: 5.459e-13
Load error after doubling the mesh: 1.786e-03
```



A folded family of one-dimensional Bratu temperature profiles; the highlighted curve is computed with an augmented OneSimulation Newton solve.