Classifying Solver Viability Given an Arbitrary Function or Differential Equation

Erik James Knee

January 2026

1 Abstract

Differential equations and functions historically have been solved analytically to find existing roots, or solutions given initial or boundary conditions. However, Some equations do not have such methods of doing them by hand, which thus requires utilization of numerical methods to find solutions at predefined points of interest. These methods can vary in utilization based on the particular problem, where there are identifiable characteristics of equations that determine whether a certain method will actually converge to a solution, or will converge inefficiently compared to other methods. This projects aim is to record a methodology without needing excessive background knowledge in determining when common solvers will crash by analyzing the particular Differential Equation or function, so the reader can build intuition on what cannot work when solving a problem numerically. That way, in future research projects you are able to minimize failed attempts before arriving at a solution.

2 Introduction

To include: Newton's method, Bisection method, Line searches, Quasi-Newton, and Continuation, finite element and finite difference. Runga Kutta.

To begin our study on when certain kinds of nonlinear solvers fail, we would first like to introduce each to the reader.

Newton's Method (Root Finding)

Newton's Method is an iterative numerical technique used to approximate solutions of an equation

$$f(x) = 0,$$

where the function f is differentiable and its derivative is known.

Starting from an initial guess x_0 that is reasonably close to a true root, Newton's Method constructs a sequence $\{x_n\}$ by repeatedly using the tangent

line to f at the current approximation. The point where this tangent line intersects the x-axis is taken as the next approximation.

The iteration formula is

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}.$$

Geometrically, this formula comes from the linear approximation of f at x_n:

$$f(x) \approx f(x_n) + f'(x_n)(x - x_n).$$

Setting this approximation equal to zero and solving for x yields the Newton update formula.

When the initial guess is sufficiently close to the actual root and $f'(x) \neq 0$ near the root, Newton's Method converges rapidly, often doubling the number of correct digits at each step. However, poor initial guesses or points where the derivative is small or undefined can lead to slow convergence or divergence.

Newton's Method is widely used in scientific computing due to its efficiency and conceptual connection to tangent-line approximations [1]. ****** The main shortcoming to Newton's method however, is that you need to have an initial guess that is close enough to the solution that you are looking for. If the functions has multiple local critical points, Newton's method may converge to them instead rather than the points you are looking for.

things to consider: multiple solutions, nonlinear, hessian (jacobian) singular or ill-conditioned. Look at: steady-state reaction-diffusion equation with non-unique solutions bratu's problem finite difference

(this summary is not great. Need summary for pde rather than root finding. Try Newton-Raphson)

The Bisection Method

The bisection method is a root-finding algorithm for solving a nonlinear equation

$$f(x) = 0$$

when the function f is continuous on a closed interval $[a, b]$ and satisfies

$$f(a)\, f(b) < 0.$$

This condition guarantees, by the Intermediate Value Theorem, that at least one root lies in the interval.

The method proceeds by repeatedly halving the interval. At each step, the midpoint

$$m = \frac{a + b}{2}$$

is computed and the sign of $f(m)$ is examined. If $f(a)\, f(m) < 0$, then the root lies in the subinterval $[a, m]$; otherwise, it lies in $[m, b]$. The interval containing the root is therefore reduced by a factor of two at each iteration.

This process is continued until the interval length becomes smaller than a prescribed tolerance or until the function value at the midpoint is sufficiently close to zero. The midpoint of the final interval is then taken as an approximation to the root.

The bisection method is guaranteed to converge as long as the initial sign-change condition holds. Its convergence is linear, and although it is slower than methods such as Newton's method, it is highly reliable and insensitive to the shape of the function. For this reason, it is often used as a starting method or as a benchmark for more rapidly convergent algorithms.

Finite Difference Method

The goal of the finite difference method is to *approximate solutions to differential equations*. That is, we aim to find a function (or a discrete approximation of it) that satisfies the relationships between its derivatives over a given spatial and/or temporal domain, along with prescribed boundary conditions. Since exact, analytic solutions are rarely available, numerical methods are required.

The finite difference method works by *replacing derivatives with algebraic approximations* using values of the function at discrete points. This converts the differential equation into a *finite system of algebraic equations* that can be solved on a computer.

Before applying this to differential equations, it is helpful to consider the simpler problem of *approximating derivatives of a known function* using only its values at nearby points. This introduces fundamental concepts such as the *order of accuracy* of an approximation.

Let $u(x)$ be a smooth function (i.e., differentiable several times), and consider a point x_n. A simple finite difference approximation for the first derivative at x_n is the *forward difference*:

$$u'(x_n) \approx \frac{u(x_n + h) - u(x_n)}{h},$$

where h is a small spacing between points. This is a *one-sided* approximation because it uses only points $x \geq x_n$. Similarly, the *backward difference* uses points $x \leq x_n$:

$$u'(x_n) \approx \frac{u(x_n) - u(x_n - h)}{h}.$$

These finite difference formulas provide the basis for *replacing derivatives in differential equations* with discrete approximations, allowing the problem to be solved numerically on a grid.

Quasi-Newton Methods

As detailed by Nocedal and Wright [4], Quasi-Newton methods are iterative algorithms used to find the local minima of a function f when the Hessian

$\nabla^2 f(x)$ is unavailable or too expensive to compute. Unlike the standard Newton's method, which requires the exact Hessian to solve the system $\nabla^2 f(x_k)p_k = -\nabla f_k$, Quasi-Newton methods construct a sequence of approximations $B_k \approx \nabla^2 f(x_k)$ (or $H_k \approx [\nabla^2 f(x_k)]^{-1}$) based on information gathered from previous gradients. The search direction is defined as $p_k = -B_k^{-1}\nabla f_k$.

The update for the Hessian approximation is governed by the *secant equation*, $B_{k+1}s_k = y_k$, where $s_k = x_{k+1} - x_k$ and $y_k = \nabla f_{k+1} - \nabla f_k$. The most widely used update formula is the **BFGS** method, which maintains a symmetric positive-definite matrix and exhibits self-correcting properties. For high-dimensional problems, the **L-BFGS** (Limited-memory BFGS) variant is preferred, as it stores only a small number of vector pairs (s_k, y_k) to represent the inverse Hessian rather than a full $n \times n$ matrix. These methods are characterized by **superlinear convergence**, providing a computational middle ground between the linear convergence of steepest descent and the quadratic convergence of Newton's method.

3 Methodology

To gain an understanding of when different methods fail, we will first solve a typical question where the method is successful, and then show a contrasting example where the method may not be used.

3.1 Bisection Method

A typical problem where the bisection method works well is for the function

$$f(x) = x^3,$$

where we wish to find the root x = 0 on the interval [-1,2]. We choose this interval because using an interval such as [-1,1] is symmetric to where the root is on our function, hence it would only take one iteration which is uninteresting. This would be the same as choosing a more advanced polynomial in regards to the number of iterations, so we do this for convenience. Hence, by using the bisection method with a given tolerance of $\epsilon = 10^{-3}$, we summarize:

Table 1: Bisection Method Convergence

Iteration (n)	a_n	b_n	x_n (midpoint)	Error $\lvert x_n - x_{n-1}\rvert$
1	-1	2	0.5	–
2	-1	0.5	-0.25	0.75
3	-0.25	0.5	0.125	0.375
4	-0.25	0.125	-0.0625	0.1875
5	-0.0625	0.125	0.03125	0.09375
6	-0.0625	0.03125	-0.015625	0.046875
7	-0.015625	0.03125	0.0078125	0.0234375
8	-0.015625	0.0078125	-0.00390625	0.01171875
9	-0.00390625	0.0078125	0.001953125	0.005859375
10	-0.00390625	0.001953125	-0.0009765625	0.0029296875

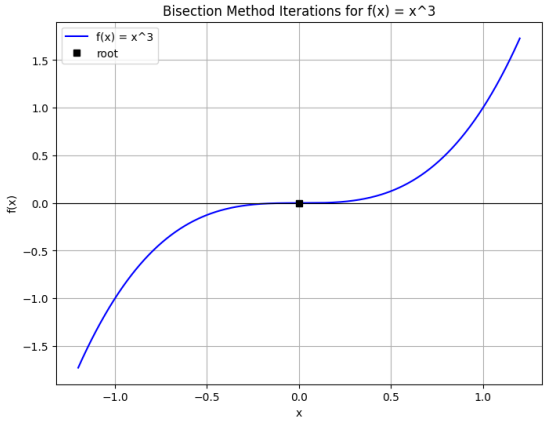

Figure 1: Function and desired root

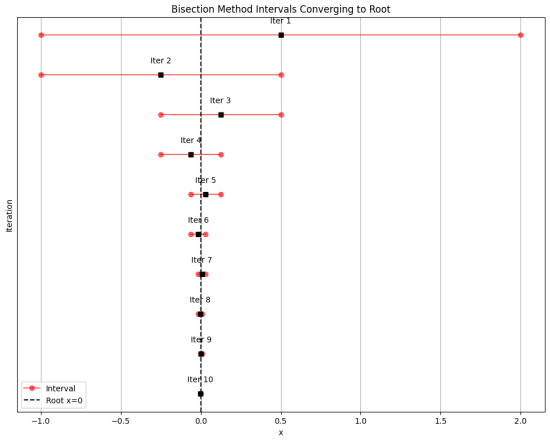

Figure 2: Iteration progress

An overly simple way we can get this algorithm to fail is to violate its necessary condition of the function needing to satisfy $f(a)f(b) < 0$. For a function that is positive everywhere besides the root such as $f(x) = x^2$, the function only touches the x-axis rather than crossing it. This means we cannot establish the 'bracketing' required to decrease the size of the interval in which the solution can be found. If we do try anyway, this is the result:

Table 2: Bisection Method Convergence

Iteration (n)	a_n	b_n	c_n (midpoint)	Error $\lvert c_n - c_{n-1} \rvert$
1	-1	2	0.5	–
2	0.5	2	1.25	0.75
3	1.25	2	1.625	0.375
4	1.625	2	1.8125	0.1875
5	1.8125	2	1.90625	0.09375
6	1.90625	2	1.953125	0.046875
7	1.953125	2	1.9765625	0.0234375
8	1.9765625	2	1.98828125	0.01171875
9	1.98828125	2	1.994140625	0.005859375
10	1.994140625	2	1.9970703125	0.0029296875

As we can see, the method incorrectly finds the solution to be $x = 2$, because the midpoint can never switch signs to 'bracket' itself into the correct interval for which the solution exists.

To illustrate a more non-obvious example, we have to be more creative. The bisection algorithm is very robust in that it is guaranteed to converge given the necessary conditions for convergence is fulfilled. However, in practice, many problems in science do not have as strong of properties that the bisection method

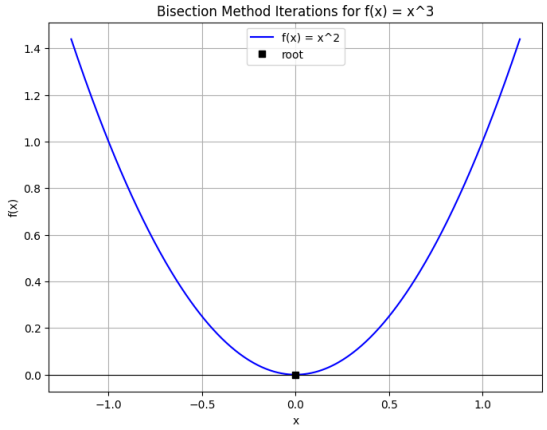

Figure 3: Function and desired root

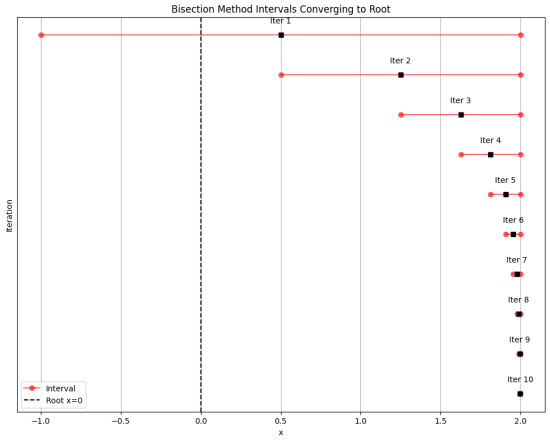

Figure 4: Function and desired root

requires of being continuous everywhere on a closed interval, so it is important to consider the functions in science that could have point discontinuities that are the true global extrema (maybe include example here?).

Another thing to consider about bisection is its efficiency. The bisection method is guaranteed linear convergence if the problem fulfills its requirements, —x**3-x-1 over [1,2]— but can be out-competed by Newton's method from quadratic convergence. For example, if we look at the same function of $f(x) = x^3$ only this time we are interested in a much more global interval such as $[-10^{50}, 10^{49}]$, we see that Newton's method reaches the required solution much quicker, even with a really poor initial guess of $x = 10^{12}$ and same tolerance of 1e-3 (there might be something funny going on in the tolerance cutoff of table 3):

Table 3: Bisection Method Convergence (Selected Iterations)

Iteration (n)	a_n	b_n	c_n (midpoint)	Error $\lvert c_n - c_{n-1}\rvert$
1	-1.0×10^{50}	1.0×10^{49}	-4.5×10^{49}	–
2	-4.5×10^{49}	1.0×10^{49}	-1.75×10^{49}	2.75×10^{49}
3	-1.75×10^{49}	1.0×10^{49}	-3.75×10^{48}	1.375×10^{49}
4	-3.75×10^{48}	1.0×10^{49}	3.125×10^{48}	6.875×10^{48}
5	-3.75×10^{48}	3.125×10^{48}	-3.125×10^{47}	3.4375×10^{48}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
~ 50	-4.20×10^{0}	5.21×10^{0}	5.05×10^{-1}	~ 2.35
~ 60	-1.85	5.05×10^{-1}	-6.71×10^{-1}	~ 1.18
~ 70	-8.32×10^{-2}	5.05×10^{-1}	2.11×10^{-1}	$\sim 2.94 \times 10^{-1}$
~ 80	-9.66×10^{-3}	6.38×10^{-2}	2.71×10^{-2}	$\sim 3.68 \times 10^{-2}$
177	-4.76×10^{-4}	6.73×10^{-4}	9.87×10^{-5}	5.74×10^{-4}

Table 4: Newton's Method Convergence (Initial Guess $x_0 = 10^{12}$)

Iteration (n)	x_n	Error $\lvert x_n - x_{n-1}\rvert$
0	$1.0000000000 \times 10^{12}$	–
1	$6.6666666667 \times 10^{11}$	$3.3333333333 \times 10^{11}$
2	$4.4444444444 \times 10^{11}$	$2.2222222223 \times 10^{11}$
3	$2.9629629630 \times 10^{11}$	$1.4814814814 \times 10^{11}$
4	$1.9753086419 \times 10^{11}$	$9.8765432108 \times 10^{10}$
5	$1.3168724280 \times 10^{11}$	$6.5843621395 \times 10^{10}$
$\vdots$	$\vdots$	$\vdots$
20	$3.0072865982 \times 10^{8}$	$1.5036421141 \times 10^{8}$
30	$7.8226425763 \times 10^{6}$	$3.9133212881 \times 10^{6}$
40	$9.0437726838 \times 10^{4}$	$4.5218993419 \times 10^{4}$
50	$1.5683285455 \times 10^{3}$	$7.8416427255 \times 10^{2}$
60	$2.7197216389 \times 10^{1}$	$1.3598608064 \times 10^{1}$
70	4.0795824584	2.0397912290
75	1.5917895280	$7.9589476402 \times 10^{-1}$
80	$1.3974558271 \times 10^{-1}$	$6.9872791354 \times 10^{-2}$
84	$1.6156014688 \times 10^{-3}$	$8.0780073454 \times 10^{-4}$

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

3.2 Newton's Method

Now that we have contrasted how the bisection method can sometimes be less useful than Newton's method, it is natural that we next summarize why Newton's method sometimes is not the best choice either.

3.3 Finite difference method

counterxample: consider the stiff ode:

$$\frac{du}{dx} = -1000\,u + 3000 - 2000\,e^{-x}, \quad 0 \le x \le 1, \quad u(0) = 0$$

The exact solution is:

$$u(x) = 3 - 0.998\,e^{-1000x} - 2.002\,e^{-x}.$$

TO DO:

- talk about initial guess have to be good for newton
- Find textbooks or research papers on Newton's Method
- Find textbook on Bisection
- Find textbook on quasi-Newton

References

[1] Edwin Herman and Gilbert Strang, *Calculus Volume 1*, OpenStax, Houston, TX, online edition, 2016. Updated 2026. Section 4.9: Newton's Method. Available at `https://openstax.org/details/books/calculus-volume-1`. ISBN 978-1-938168-02-4.

[2] K. E. Atkinson, *An Introduction to Numerical Analysis*, 2nd ed., John Wiley & Sons, New York, 1989.

[3] R. J. LeVeque, *Finite Difference Methods for Ordinary and Partial Differential Equations: Steady-State and Time-Dependent Problems*, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2007.

[4] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer, New York, second edition, 2006.