Classifying Solver Viability Given an Arbitrary Partial Differential Equation

Erik James Knee

January 2026

1 Abstract

Partial Differential Equations (PDEs) and their applications are one of the most important mathematical concepts applicable to the applied sciences. For some problems, there are analytical methods available to find general solutions given certain initial or boundary conditions. However, Some equations do not have such methods, which thus requires utilization of numerical methods to find solutions to equations 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. This projects aim is to record a methodology without needing excessive background knowledge in determining when common solvers will crash by analyzing the particular PDE, 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

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.

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.

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.