International Journal of Computational and Experimental Science and ENgineering (IJCESEN)

Vol. 11-No.4 (2025) pp. 8336-8341
http://www.ijcesen.com

ISSN: 2149-9144

Research Article

Enhanced Algorithms for Solving Nonlinear Systems: Beyond NEWTON-RAPHSON with Example of Advanced Nonlinear Solver

Brahim Benzeghli[1]*, Salah Adoui[2]

[1]Batna 2 University, 53, Constantine Road, Fesdis, Batna, 05078, Algeria
* **Corresponding Author Email:** b.benzeghli@univ-batna2.dz - **ORCID**: 0000-0002-8986-4547

[2] Batna 2 University, 53, Constantine Road, Fesdis, Batna, 05078, Algeria
Email: s.adoui@univ-batna2.dz - **ORCID**: 0000-0002-8986-4007

Article Info:

DOI: 10.22399/ijcesen.4159
Received : 19 August 2025
Accepted : 11 October 2025

Keywords

Nonlinear System Solver
Newton-Raphson Methods
Enhanced Stability
Adaptive Algorithms

Abstract:

Solving nonlinear systems of equations is a central challenge in scientific computing, impacting a wide range of fields such as engineering, physics, and applied mathematics. Although the Newton-Raphson method is popular for its quadratic convergence near solutions, it faces notable difficulties, including reliance on the initial guess, potential failure with ill-conditioned Jacobians, and complications when multiple or closely situated roots are present. In this study, we investigate the creation of new iterative algorithms aimed at overcoming these obstacles by promoting better global convergence and improving numerical stability. The proposed approaches utilize adaptive step-size management, quasi-Newton techniques, and hybrid strategies that integrate trust-region and homotopy concepts. Results from numerical tests on standard benchmark problems show that these algorithms provide enhanced robustness for a wide array of nonlinear systems. Compared to Newton-Raphson, the new methods expand the convergence domain and frequently deliver equal or better accuracy and computational speed. This work paves the way for developing more trustworthy solvers for complex nonlinear systems in contemporary computational practice.

1. Introduction

Solving nonlinear systems of equations is a major challenge in many areas of science and engineering. The Newton-Raphson method is one of the most widely used approaches because of its rapid local convergence when conditions are favorable. However, its performance can be greatly hindered by poor initial guesses, badly conditioned Jacobian matrices, or when dealing with highly nonlinear or discontinuous systems. These limitations underscore the necessity for new algorithms that can deliver greater stability, robustness, and guarantee global convergence.

Recent advances in numerical analysis, optimization, and machine learning have led to the emergence of iterative methods that often surpass Newton-Raphson. These developments include derivative-free strategies, globally convergent approaches such as homotopy and continuation methods, trust-region and line search techniques, and adaptive or hybrid frameworks that dynamically adjust based on the system's properties.

The primary objective in designing these algorithms is to reduce sensitivity to initial conditions, enhance convergence in challenging regions of the solution space, and maintain computational efficiency. This is particularly crucial in large-scale simulations, nonlinear finite element problems, power system modeling, and machine learning contexts, where traditional methods frequently encounter convergence failures.

This work presents a framework for developing new iterative algorithms that combine mathematical rigor with practical robustness, offering a compelling alternative to the Newton-Raphson method—especially in real-world applications where stability and reliable convergence are paramount.

2. Newton-Raphson Method

The Newton-Raphson method is widely used for solving nonlinear systems, especially when the problems are smooth and well-posed. Despite its effectiveness, it faces several well-documented challenges, such as high sensitivity to the initial estimate, possible divergence, and reliance on Jacobians which can be difficult to compute or poorly conditioned. To address these issues, researchers are actively working on new or enhanced algorithms designed to surpass Newton-Raphson in terms of stability, global convergence, and robustness. The following is a summary of emerging and alternative algorithms that show promise in outperforming Newton-Raphson for nonlinear systems, along with insights into innovative ideas that could be incorporated into future algorithm designs.

3. Quasi-Newton Methods with Adaptive Strategies

Quasi-Newton methods like Broadens method approximate the Jacobian instead of computing it directly. A modern enhancement could involve 'adaptive update rules', 'line search', and 'trust-region strategies'.

3.1. Improvements

- Less sensitive to initial guess
- Avoids exact Jacobian computation
- More robust for ill-conditioned systems

3.2. New Development Ideas

- Combine with machine learning to predict better Jacobian approximations.
- Use rank-one updates guided by residuals' trends.

4. Homotopy and Continuation Methods

These methods solve a difficult nonlinear system by deforming it from a simpler one (via a parameterized path).

4.1 Improvements:

- Excellent global convergence
- Can trace multiple solution branches
- Especially useful for highly nonlinear or bifurcating systems

4.2. New Development Ideas:

- Adaptive step-size and path-following strategies.
- Use machine learning to predict good homotopy paths.
- Parallelized path-tracking for multiple roots.

5. Anderson Acceleration for Fixed Point Iterations

Transforms a fixed-point iteration (like $x = g(x)$) into a faster-converging scheme
by mixing previous iterations like a multi-secant method.

5.1. Improvements:

- Can greatly enhance convergence speed.
- Black-box applicability to nonlinear solvers.

5.2. New Development Ideas:

- Combine Anderson acceleration with Newton or quasi-Newton iterations.
- Dynamically switch between acceleration and damping based on residual history.

6. Trust-Region Methods for Nonlinear Systems

Rather than taking Newton steps directly, solve a local approximation (usually quadratic) within a trust region around the current guess.

6.1 Improvements:

- Much more stable when far from the root.
- Handles ill-conditioned Jacobians better.
- Better global convergence properties.

6.2 New Development Ideas:

- Combine trust-region strategy with inexact or learned Jacobians.
- Use subspace-based trust-region refinement to reduce computational load.

7. Machine Learning-Augmented Solvers

Use trained neural networks or other learning models to:

- ❖ Predict good initial guesses.
- ❖ Estimate Jacobians.
- ❖ Select step sizes or damping parameters.

7.1 Improvements:

- Can dramatically reduce iterations
- Helps in structured problems where prior solutions exist

7.2 New Development Ideas:

- Reinforcement learning to adaptively control iteration parameters
- Meta-learning across problem families (e.g.in Multiphysics simulations)

8. Nonlinear Krylov Subspace Methods

Extend ideas from GMRES and other Krylov subspace methods into the nonlinear setting (e.g., Newton-Krylov or Inexact Newton methods).

8.1 Improvements:

- Avoids forming or inverting full Jacobians.

- Very scalable for large systems (e.g., PDE discretization).

8.2 New Development Ideas:

- Precondition nonlinear residuals using learned or physics-informed strategies.
- Dynamic subspace enrichment using residual analysis.

9. Hybrid or Switching Methods

Use multiple methods in a coordinated or adaptive way for example:

- ❖ Start with Brayden or fixed-point iteration
- ❖ Switch to Newton when close enough to the root

9.1 Improvements:

- Combines global convergence with local speed.
- Adapts to system behavior.

9.2 New Development Ideas:

- Develop "solver controllers" that choose algorithms based on residual, Jacobian norm, etc.
- Use ML classifiers to select the best solver path during iterations

10. Novel Algorithm Blueprint: "Adaptive Trust-Region Anderson (ATRA) Solver"

A new algorithm could combine:

1- Trust-region control (global stability).
2- Anderson acceleration (speed-up).
3- Inexact, learned Jacobian approximations (cheap computations).
4- A heuristic switching controller that adapts based on:
   a- Residual norm.
   b- Predicted conditioning.
   c- Iteration stagnation.

This type of solver would be particularly well-suited for stiff, ill-conditioned,
or partially known nonlinear systems, such as those in Multiphysics, nonlinear optimization, and real-time control.

Table 1: *Summary Table*

Methode	Globel convergence	Jacobian-Free	Good for large systems	ML Integration potential
Quasi-Newton	Moderate	Yes	Yes	Medium
Homotopy	Excellent	Yes	Medium	High
Anderson	Poor (alone)	Yes	Yes	Medium
Trust-Region	Excellent	No	Yes	Medium
Newton-Krylov	Good	Yes	Excellent	Low
ML-Augmented	Depends	Yes	Yes	High
Hybrid/Switching	Very good	Mixed	Mixed	High

11. Example of advanced nonlinear solver

Let's break an example of a solver step-by-step so we can ask about:

- ❖ Building a hybrid solver.
- ❖ Integrating Anderson acceleration.
- ❖ Designing a trust-region controller for Newton steps.

All of these are advanced nonlinear solver techniques used in optimization and numerical analysis, especially for solving systems of nonlinear equations or optimization problems.

11.1. Building a Hybrid Solver

A hybrid solver typically combines multiple strategies to improve robustness and convergence. For example:

- ❖ Newton-Raphson + Anderson acceleration.
- ❖ Newton + Brayden update.
- ❖ Switching from a quasi-Newton method to full Newton when close to solution.

This sets up hybrid strategy early iterations rely on Newton if Jacobian is available; fallback is acceleration or quasi-Newton.

```
def hybrid_solver(F,x0,jac=None,tol=1e-6,max_inter=100):
    x=x0.copy()
    for k in range(max_iter):
        # Choose methode adaptively
        if jac is not None and k<5:
            # Use Newton step with provided Jacobian
            j=jac(x)
            dx=np.linalg.solve(J,-F(x))
        else:
            # Fall back to quasi-Newton
            # or Anderson acceleration
            dx=anderson_acceleration(F,x)

            # Update
            x += dx

            # Check convergence
            if np.linalg.norm(F(x)) < tol:
                break
                return x
```

Figure 1. *Python code*

11.2. Integrating Anderson Acceleration

Anderson Acceleration (AA) is used to accelerate fixed-point iterations $x_{k+1} = G(x_k)$, often used

with nonlinear solvers. We can define Anderson acceleration like so:
We would plug this into the hybrid solver when Newton is not used.

```
import numpy as np
def anderson_acceleration(F,x0,m=5,tol=1e-8,max_iter=50):
    """Fixed-point solver with Anderson acceleration"""
    x = x0
    f = lambda x: x-F(x)
    x = []
    Fvals = []
    for k in range(max_iter):
        fval = f(x)
        if np.linalg.norm(F(x)) < tol:
            return x0

            if k == 0:
                x_new = fval
            else:
                x.append(x-x_old)
                Fvals.append(fval - f_old)
                if len(x) > m:
                    x.pop(0)
                    Fvals.pop(0)

                    dx = np.column_stack(x)
                    dF = np.column_stack(Fvals)
try:
    gamma = np.linalg.lstsq(dF, fval-f_old, recond=None)[0]
    dx=-dx @ gamma
except np.linalg.LinAlgError:
    dx=fval - x0
    x_new=fval+dx
    x_old, f_old = x, fval
    x=x_new
    return x
```

Figure 2. *Anderson Acceleration (Python - simplified)*

11.3. Designing a Trust-Region Controller for Newton Steps

Trust-region methods improve Newton robustness by limiting the step size based on the quality of the quadratic approximation.

- **Trust Region Concept**

Solve:

$$\min_{s} \frac{1}{2} s^T J^T J s + s^T J^T S \quad \textbf{Subject to } |s| \leq \Delta$$

This is the Levenberg-Marquardt idea or Dogleg method.

```
 def trust-region_newton(F, J, x, delta):
"""Compute a trust-region constrained Newton step"""
g = J(x).T @ F(x)
H = J(x).T @ J(x)
try:
s = np.linalg.solve(H,-g)
if np.linalg.norm(s) <= delta:
    return s
else:
    s = -(delta / np.linalg.norm(s)) * s
     return s
except np.linalg.LinAlgError:
return -g / np.linalg.norm(g) * delta
```

Figure 3. *Trust-Region Newton Step (Sketch)*

Trust-Region Controller Inside the solver, monitor model agreement:

```
actual_reduction = np.linalg.norm(F(x)) - np.linalg.norm(F(xns))
pretected_reduction = -g.T @ s-0.5 * s.T @ H @ setrho = actual_reduction / predicted_reduction
if rho < 0.25:
    delta *= 0.5
elif rho > 0.75 and np.linalg.norm(s) == delta:
    delta = min(2*delta, delta_max)

    if rho > eta:
        x = xns # Accept
    else:
        pass # Reject
```

Figure 4. *Trust-Region Controller*

If we're building a system, we'd integrate these into a stateful solver class, for example:

```
class HybridSolver:
    def __init__(self, F, jac=None, tol=1e-6, max_iter=50):
    self.F = F
    self.jac = jac
    self.tol = tol
    self.max_iter = max_iter

    def solve(self, x0):
        x = x0.copy()
        delta = 1.0
        for k in range(self.max_iter):
            Fx = self.F(x)
            if np.linalg.norm(Fx) < self.tol:
                break
                if self.jac:
                    Jx = self.jac(x)
                    s = trust_region_newton(self.F, self.jac, x, delta)
                    x_trial = x + s
                    x = x_trial
                else:
                    x=anderson_acceleration(self.F, x)
                    return x
```

Figure 5. *Putting It All Together*

- **Convergence and stability of Levenberg-Marquardt idea or Dogleg method**

The Levenberg-Marquardt (LM) method or the Dogleg method is both widely used algorithms for solving nonlinear least squares problems. These methods are particularly popular in optimization problems where the objective function is a sum of squares, such as in curve fitting, parameter estimation, and inverse problems. Herse a focused overview of their convergence and stability characteristics.

Levenberg-Marquardt (LM) Method:

LM blends Gauss-Newton (fast near a solution) and gradient descent (robust far from solution). It introduces a damping parameter (λ) to interpolate between them.

Update rule:

$$(J^T J + \lambda I)\delta = -J^T r$$

Where:

- J : is the Jacobian of the residuals;
- r : is the residual vector;
- δ : is the step direction.

- **Convergence:**

1- Quadratic convergence:

- near the solution, like Gauss-Newton, if the Jacobian has full rank.
- When the damping **(λ → 0)**, LM becomes Gauss-Newton, achieving fast convergence.

2- Global Convergence:

- If damping **(λ)** is large, LM behaves like gradient descent (small, cautious steps).
- Globally convergent under mild conditions e.g. objective function is continuously differentiable and bounded below.

Local

- **Requires:**

- Good damping adjustment strategy (e.g. increase (λ) when the step fails; decrease it when the step is successful).
- Jacobian needs to be well-conditioned for fast convergence.

- **Stability:**

- More stable than pure Gauss-Newton near saddle points or where Jacobian is ill-conditioned.
- Robust to poor initial guesses due to gradient descent behavior at high (λ).
- Sensitive to scaling preconditioning or normalization can help.

Dogleg Method:

The Dogleg method is used within trust region frameworks. It approximates the solution path as a piece-wise linear path (the" dogleg") between the steepest descent direction and the Gauss-Newton direction.

- **Convergence:**

1- **Local convergence:**

- Similar to Gauss-Newton when the trust region contains the Newton step.
- Quadratic convergence if the Jacobian is full-rank and the trust region is large enough.

2- Global Convergence:

- Trust region strategy guarantees global convergence to a stationary point.
- Each step minimizes a model within a constrained region helps prevent overly large, unstable steps.

- **Stability:**

- Highly stable due to the trust region control.
- Adapts well to problem geometry.
- Can be more efficient than LM when the model behaves well, because it avoids solving damped normal equations.

Table 2: *Summary Comparison*

Feather	Levenberg-Marquardt	Dogleg Method
Approach	Damped Gauss-Newton	Trust-region based
Local convergence	Quadratic (if Jacobian full rank)	Quadratic (similar conditions)
Global convergence	Yes, with damping	Yes, via trust region framework
Stability	Good, especially with tuning	Very stable due to trust region
Step type	Solves modified normal equations	Chooses from piecewise-linear path
Computational cost	Higher per step	Lower if Gauss-Newton is expensive
Suitability	Robust to bad initial guesses	Efficient for well-scaled problem

12. Conclusion

New algorithms can indeed be developed to solve nonlinear systems that outperform the classical Newton-Raphson method in terms of stability and convergence. While Newton-Raphson is widely used due to its simplicity and quadratic convergence near the root, it suffers from limitations such as sensitivity to initial guesses, divergence in certain cases, and poor performance on ill-conditioned problems. Modern advancements in numerical methods, such as homotopy continuation, trust-region approaches, and globally convergent or hybrid methods, offer more robust alternatives. These new algorithms enhance convergence reliability, expand the domain of attraction, and improve performance in solving large-scale or highly nonlinear systems. Continued research in this area holds strong potential for more efficient and stable solvers applicable across a wide range of scientific and engineering disciplines

Author Statements:

- **Ethical approval:** The conducted research is not related to either human or animal use.
- **Conflict of interest:** The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper
- **Acknowledgement:** The authors declare that they have nobody or no-company to acknowledge.
- **Author contributions:** The authors declare that they have equal right on this paper.
- **Funding information:** The authors declare that there is no funding to be acknowledged.
- **Data availability statement:** The data that support the findings of this study are available on request from the corresponding author. The data are not publicly available due to privacy or ethical restrictions.

References

[1] Ding, Renjie & Wang, Dongling. (2025). Adaptive Residual-Driven Newton Solver for Nonlinear Systems of Equations. 10.48550/arXiv.2501.03487. DOI: 10.1016/j.anucene.2009.09.005.

[2] Gomez-Exposito, Antonio. (2014). Factored solution of nonlinear equation systems. Proceedings of the Royal Society. 470. 10.1098/rspa.2014.0236.

[3] Jiang, Yifan & Yuhong, Jin & Hou, Lei & Chen, Yi & Cong, Andong. (2025). A modified Newmark/Newton-Raphson method with automatic differentiation for general nonlinear dynamics analysis. 10.48550/arXiv.2506.13226.

[4] Tzong-Mou Wu, Solving the nonlinear equations by the Newton-homotopy continuation method with adjustable auxiliary homotopy function, Applied Mathematics and Computation, Volume 173, Issue 1, 2006, Pages 383-388, ISSN 0096-3003,
https://doi.org/10.1016/j.amc.2005.04.095.

[5] Denis Anuprienko, Comparison of nonlinear solvers within continuation method for steady-state variably saturated groundwater flow modeling, arXiv:2105.12511v1 [math.NA] 26 May 2021.

[6] Saheya, B., Chen, Gq., Sui, Yk. et al. A new Newton-like method for solving nonlinear equations. SpringerPlus 5, 1269 (2016). https://doi.org/10.1186/s40064-016-2909-7.

[7] Noor, Muhammad & Noor, Khalida & Eisa, Al-Said & Waseem, Muhammad. (2010). Some New Iterative Methods for Nonlinear Equations. Mathematical Problems in Engineering. 2010. 10.1155/2010/198943.

[8] Peter Deuflhard, Newton Methods for Nonlinear Problems: Affine Invariance and Adaptive Algorithms. Springer (2011).

[9] C. T. Kelley, Solving Nonlinear Equations with Newtons Method, SIAM (2003).

[10] Numerical polynomial homotopy continuation method to locate all the power flow solutions by Mehta, Nguyen, Turitsyn (2014).

[11] HANN: Homotopy Auxiliary Neural Network for solving nonlinear algebraic equations (2025).