

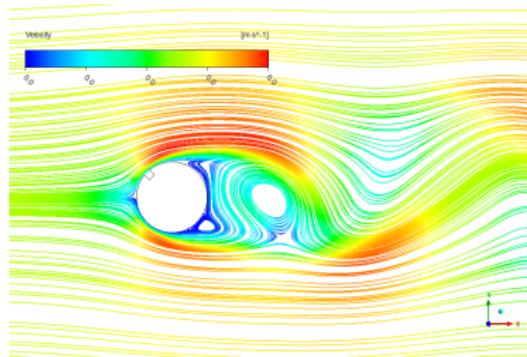
Newton's Method for Nonlinear Systems

Module 2: Nonlinear Systems and Continuum Foundations

MMAE 450

Why Nonlinear Solvers Matter

- ▶ Most engineering models are **nonlinear**
- ▶ Linear systems appear **inside** nonlinear solvers
- ▶ Newton's method underlies:
 - ▶ nonlinear finite element analysis
 - ▶ large-deformation mechanics
 - ▶ computational fluid dynamics (implicit and steady solvers)
 - ▶ multiphysics coupling (like fluid-structure interaction)



Example of a nonlinear CFD flow field

Linear solvers are a subroutine.

Residual Formulation

We seek the solution of a nonlinear system written as

$$\boldsymbol{R}(\boldsymbol{u}) = \mathbf{0},$$

where:

- ▶ \boldsymbol{u} is the vector of unknowns
- ▶ \boldsymbol{R} is the nonlinear residual vector

This formulation will appear repeatedly throughout the course.

Newton's Method: Scalar Problem

Given a nonlinear equation

$$f(x) = 0,$$

linearize about the current iterate $x^{(k)}$:

$$f(x) \approx f(x^{(k)}) + f'(x^{(k)})(x - x^{(k)}).$$

Setting the linear approximation to zero yields

$$f'(x^{(k)}) \Delta x = -f(x^{(k)}), \quad x^{(k+1)} = x^{(k)} + \Delta x.$$

Key idea: Each Newton iteration solves a **linear equation**.

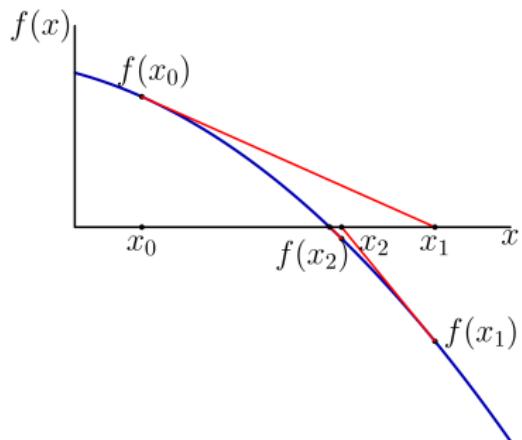
Newton's Method (Scalar Algorithm)

Newton's Method (Scalar)

1. Choose a convergence tolerance $\varepsilon > 0$
2. Choose an initial guess $x^{(0)}$
3. For $k = 0, 1, 2, \dots$:
 - ▶ Evaluate the residual $f(x^{(k)})$
 - ▶ Evaluate the derivative $f'(x^{(k)})$
 - ▶ Solve $f'(x^{(k)}) \Delta x = -f(x^{(k)})$
 - ▶ Update $x^{(k+1)} = x^{(k)} + \Delta x$
 - ▶ **Check convergence:** if $|f(x^{(k+1)})| < \varepsilon$, stop

Geometric Interpretation

- ▶ Newton's method uses the tangent line at $x^{(k)}$
- ▶ The next iterate is where the tangent intersects the axis
- ▶ Convergence is rapid when the initial guess is good
- ▶ Poor initial guesses may lead to divergence



Geometric interpretation of Newton's method

Newton's Method for Nonlinear Systems

We now consider a system of nonlinear equations:

$$\mathbf{R}(\mathbf{u}) = \begin{bmatrix} R_1(u_1, \dots, u_n) \\ \vdots \\ R_n(u_1, \dots, u_n) \end{bmatrix} = \mathbf{0}.$$

Linearizing about $\mathbf{u}^{(k)}$ gives:

$$\mathbf{R}(\mathbf{u}) \approx \mathbf{R}(\mathbf{u}^{(k)}) + \mathbf{J}(\mathbf{u}^{(k)})(\mathbf{u} - \mathbf{u}^{(k)}),$$

where the Jacobian matrix is defined by

$$J_{ij} = \frac{\partial R_i}{\partial u_j}.$$

Interpretation: \mathbf{J} is the **tangent operator**.

Newton System, Update, and Convergence

At iteration k , solve the linear system

$$\mathbf{J}(\mathbf{u}^{(k)}) \Delta \mathbf{u} = -\mathbf{R}(\mathbf{u}^{(k)}),$$

and update

$$\mathbf{u}^{(k+1)} = \mathbf{u}^{(k)} + \Delta \mathbf{u}.$$

Newton's Method (Systems)

1. Choose a convergence tolerance $\varepsilon > 0$
2. Choose an initial guess $\mathbf{u}^{(0)}$
3. For $k = 0, 1, 2, \dots$:
 - ▶ Evaluate the residual $\mathbf{R}(\mathbf{u}^{(k)})$
 - ▶ Assemble the Jacobian $\mathbf{J}(\mathbf{u}^{(k)})$
 - ▶ Solve $\mathbf{J}(\mathbf{u}^{(k)}) \Delta \mathbf{u} = -\mathbf{R}(\mathbf{u}^{(k)})$
 - ▶ Update $\mathbf{u}^{(k+1)} = \mathbf{u}^{(k)} + \Delta \mathbf{u}$
 - ▶ **Check convergence:** if $\|\mathbf{R}(\mathbf{u}^{(k+1)})\| / \|\mathbf{R}(\mathbf{u}^{(0)})\| < \varepsilon$: **stop**

Summary

- ▶ Each iteration requires solving a linear system
- ▶ Exhibits quadratic convergence when close to the solution
- ▶ Strong dependence on the initial guess
- ▶ Conditioning of the Jacobian affects robustness
- ▶ Failure modes are possible

Notebook Example

Key Takeaways

- ▶ Newton's method is a linearization engine
- ▶ The Jacobian is a tangent operator
- ▶ Linear solves dominate the computational cost
- ▶ This framework underlies the remainder of the course