# Newton's Method for Nonlinear Systems

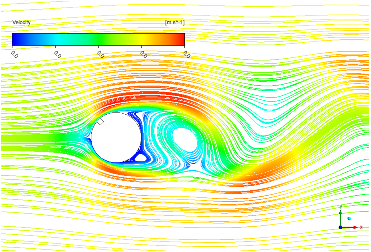
## Module 2: Nonlinear Systems and Continuum Foundations

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)

*Linear solvers are a subroutine.*




Example of a nonlinear CFD flow field

# 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

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

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

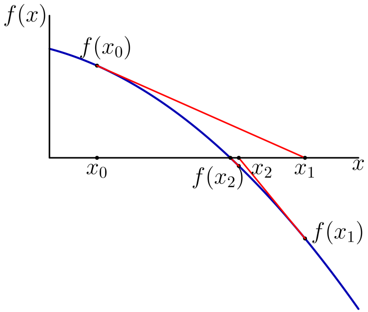
# Newton's Method (Scalar Algorithm)

**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:

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

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

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

where the Jacobian matrix is defined by

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

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

# Newton System, Update, and Convergence

At iteration $k$, solve the linear system

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

and update

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

**Newton's Method (Systems)**

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

# Summary

- Each iteration requires solving a linear system
- Exibits 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