ME 537
Numerical Methods in Engineering

Numerical Solutions of Nonlinear Equations: Newton-Raphson and Secant Methods

Presenter: Semih Aydemir

Instructor: Assoc. Prof. Dr. Nurettin Furkan DOĞAN

Date: Spring 2026

Presentation Outline

1. Introduction to Nonlinear Equations
•Roots of Equations & Open Methods

2. The Newton-Raphson Method
•Mathematical Derivation via Taylor Series
•Error Analysis & Quadratic Convergence
•Limitations and Pitfalls

3. The Secant Method
•Motivation & Finite Difference Approximation
•Mathematical Derivation & Superlinear Convergence

4. Method Comparison
•Convergence Order vs. Computational Cost

5. Engineering Application (MATLAB)
•Solving Nonlinear Stress-Strain Relations

6. Conclusion & Discussion

Introduction to Nonlinear Equations

-Why Do We Need Numerical Root Finding?

- • The Problem: In engineering, we frequently encounter systems governed by equations of the form f(x) = 0.
- • Analytical Limitations: For polynomials of degree five or higher, and for complex transcendental equations (involving exponentials, trigonometric functions, etc.), analytical solutions do not exist.
- • Numerical Approach: We rely on iterative numerical methods to find approximate roots within a specified tolerance (ε_s). →
- • Categories of Root Finding:
- Bracketing Methods: Bisection, False Position (Always converge, but slow).
- Open Methods: Newton-Raphson, Secant (Fast convergence, but require good initial guesses).

The Newton-Raphson Method (Concept)

Geometric Interpretation

- **Fundamental Idea:** If the initial guess at the root is xi, a tangent can be extended from the point (xi, f(xi)) down to the x-axis.
- The point where this tangent crosses the x-axis represents an improved estimate of the root,xi+1
- **The Iterative Process:**
- 1. Evaluate function and its derivative at current guess.
- 2. Draw tangent.
- 3.Find x-intercept.
- 4. Repeat until the approximate relative error $|\epsilon_a|[cite_start] < \epsilon_s$

Newton-Raphson Derivation

-Derivation Using Taylor Series Expansion

- As discussed in slide 1, the Taylor series expansion relates the value of a function at x_{i+1} to its value and derivatives at x_i:

$$f(x_{i+1}) = f(x_i) + f'(x_i)(x_{i+1} - x_i) + \frac{f''(x_i)}{2!}(x_{i+1} - x_i)^2 + \dots$$

To find the root, we want $f(x_{i+1})$ to be exactly zero.

We apply a **first-order approximation**, truncating the series after the first derivative term:

$$0 \approx f(x_i) + f'(x_i)(x_{i+1} - x_i)$$

Solving algebraically for the new estimate x_{i+1}:

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

Convergence Analysis of Newton-Raphson

-Error Analysis and Quadratic Convergence

- Let the true root be x_r .The true error at iteration i is $E_i = x_r - x_i$.
- By analyzing the remainder of the Taylor series, the true error at the next iteration, E_{i+1}, is proportional to the square of the previous error:

$$E_{i+1} \approx -\frac{f''(x_r)}{2f'(x_r)} E_i^2$$

-Quadratic Convergence: The error is roughly proportional to the square of the previous error.

-Practical Implication: The number of significant correct digits approximately **doubles** with each iteration (e.g., 1, 2, 4, 8, 16 digits).

Pitfalls of the Newton-Raphson Method
-Limitations and Divergence

- Despite its speed, the method is not foolproof.
- **1. Inflection Points:** If the initial guess is near an inflection point ($f''(x) = 0$), the tangent may shoot far away from the root.
- **2. Local Maxima/Minima:** If $f'(x_i) \approx 0$, the denominator approaches zero, causing the formula to yield massive steps or division by zero.
- **3. Oscillation:** In some functions, the estimates can bounce back and forth indefinitely between two points.
- **Requirement:** It requires an analytical expression for the derivative $f'(x)$, which is not always possible or computationally cheap for complex engineering functions.
-

The Secant Method

-Motivation and Finite Difference Approximation

- **The Core Issue:** Newton-Raphson requires evaluating the exact derivative $f'(x)$ at every step.
- **The Solution:** We can approximate the derivative using a **backward finite difference** scheme based on the two previous estimates (x_i and x_{i-1}):

$$f'(x_i) \approx \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}}$$

Substituting this approximation back into the Newton-Raphson formula yields the **Secant Method**:

$$x_{i+1} = x_i - \frac{f(x_i)(x_i - x_{i-1})}{f(x_i) - f(x_{i-1})}$$

Requirement: Unlike Newton-Raphson, the Secant method requires **two** initial estimates (x_0 and x_1).

Convergence of the Secant Method

- Superlinear Convergence

- The convergence rate of the Secant method is determined by the relation:

$$|E_{i+1}| \propto |E_i|^p \quad \text{where} \quad p = \frac{1+\sqrt{5}}{2} \approx 1.618$$

- **Superlinear Convergence:** It converges faster than linear methods (like Bisection) but slightly slower than quadratic methods (like Newton-Raphson).
- **Efficiency Note:** Although Newton-Raphson has a higher convergence order (2 vs 1.618), the Secant method only requires **one** function evaluation per iteration (since $f(x_{i-1})$ is already known from the previous step). Newton requires **two** evaluationts ($f(x_i)$ and $f'(x_i)$).

Convergence Order Comparison: Newton vs. Secant
-Which Method is Better?

- Newton-Raphson Method:
- **Convergence Order:** Quadratic $(p = 2)$
- **Function Evaluations per Iteration:** 2 (requires evaluating both $f(x_i)$ and $f'(x_i)$)
- **Pros:** Extremely fast once close to the root.
- **Cons:** Requires an analytical derivative; prone to division by zero if $f'(x_i) = 0$
- **Secant Method:**
- **Convergence Order:** Superlinear $(p \approx 1.618)$
- **Function Evaluations per Iteration:** 1 (reuses $f(x_{i-1})$ from the previous step)
- **Pros:** Does not require an analytical derivative; computationally cheaper per step.
- **Cons:** Slower convergence than Newton; requires two initial guesses.

Engineering Application Definition
-Solving Nonlinear Stress-Strain Relations

- **The Physical Problem:** In solid mechanics, materials often exhibit nonlinear behavior beyond their yield point. Hooke's Law ($\sigma = E\epsilon$) is no longer sufficient.
- **The Model:** We will use a modified Ramberg-Osgood or a power-law hardening model to describe the true stress-true strain relationship:

$$\sigma = E\epsilon + K\epsilon^n$$

- Where: σ = Applied Stress, E = Elastic Modulus, K = Strength Coefficient, n = Strain Hardening Exponent, ϵ = True Strain.
- **The Engineerina Task:** Given a specific applied stress ($\sigma_{applied}$), we need to find the resulting strain (ϵ). This requires finding the root of the nonlinear equation:

$$f(\epsilon) = E\epsilon + K\epsilon^n - \sigma_{applied} = 0$$

Preparing for Newton-Raphson

-Mathematical Formulation for the Algorithm

- To apply the Newton-Raphson method, we need our function $f(\epsilon)$ and its first derivative with respect to strain, $f'(\epsilon)$
- Function:

$$f(\epsilon) = E\epsilon + K\epsilon^n - \sigma_{applied}$$

- Derivative:

$$f'(\epsilon) = E + nK\epsilon^{n-1}$$

- The Iteration Formula:

$$\epsilon_{i+1} = \epsilon_i - \frac{E\epsilon_i + K\epsilon_i^n - \sigma_{applied}}{E + nK\epsilon_i^{n-1}}$$

MATLAB Implementation - Algorithm Flowchart

-Translating Math into Code

- Before writing the MATLAB function, we define the logical flow:
- 1-**Define Inputs**: $E, K, n, \sigma_{applied}$, initial guess ϵ_0, stopping tolerance ϵ_s, and maximum iterations.
- 2-**Initialize:** Set iteration counter to 0, and initial error to 100%.
- 3-**While Loop:** Continue while $error > \epsilon_s$ AND $iterations < max\_iterations$.
- 4-**Compute:** Calculate $f(\epsilon_i)$ and $f'(\epsilon_i)$.
- 5-**Update:** Calculate ϵ_{i+1} using the Newton formula.
- 6-**Error Calculation**: $\epsilon_a = \left|\frac{\epsilon_{i+1}-\epsilon_i}{\epsilon_{i+1}}\right| \times 100\%$.
- 7-**Store:** Save ϵ_{i+1} as the new ϵ_i for the next loop.

MATLAB Code (Required Function)

-MATLAB Function Implementation

```
function [root, ea, iter] = newton_strain(E, K, n, sigma, e0, es, max_it)
    % Initialization
    iter = 0;
    ea = 100; % Initial approximate relative error (%)
    e_current = e0;

    % Newton-Raphson Iteration Loop
    while (ea > es) && (iter < max_it)
        iter = iter + 1;

        % Function and Derivative evaluation
        f_val = E * e_current + K * (e_current^n) - sigma;
        f_der = E + n * K * (e_current^(n-1));

        % Check for zero derivative
        if f_der == 0
            error('Derivative is zero. Method fails.');
        end

        % Calculate next estimate
        e_next = e_current - (f_val / f_der);

        % Calculate approximate relative error
        if e_next ~= 0
            ea = abs((e_next - e_current) / e_next) * 100;
        end

        % Update for next iteration
        e_current = e_next;
    end
    root = e_current;
end
```

```
% Şekil Değiştirme (Strain) aralığı
epsilon = linspace(0.0001, 0.01, 200);

% Fonksiyon: f(e) = E*e + K*e^n - sigma_app
f_val = E.*epsilon + K.*(epsilon.^n) - sigma_app;

% Gerçek kökü bulalım (işaretlemek için)
f_func = @(e) E*e + K*(e^n) - sigma_app;
true_root = fzero(f_func, 0.005);

% ---- GRAFİK ÇİZİMİ ---- %
figure('Color', 'w'); % Beyaz arka plan
plot(epsilon, f_val, 'b-', 'LineWidth', 2.5);
hold on;

% Sıfır ekseni (Eski sürümler için yline yerine plot kullanıldı)
plot([0 0.01], [0 0], 'k--', 'LineWidth', 1.5);

% Kökü kırmızı nokta ile işaretle
plot(true_root, 0, 'ro', 'MarkerSize', 8, 'MarkerFaceColor', 'r');

% Eksenler, Başlık ve Grid
xlabel('True Strain (\epsilon)', 'FontSize', 12, 'FontWeight', 'bold');
ylabel('Function Value f(\epsilon) [MPa]', 'FontSize', 12, 'FontWeight', 'bold');
title('Root of Nonlinear Stress-Strain Equation', 'FontSize', 14);
grid on;

% Açıklama (Legend)
legend('f(\epsilon) Curve', 'Zero Axis (Target)', 'True Root', 'Location', 'northwest');
hold off;
```

Executing the Engineering Problem
-Numerical Results and Iteration Table

- Numerical Results and Iteration Table
- Test Case Parameters (e.g., Aluminum Alloy):

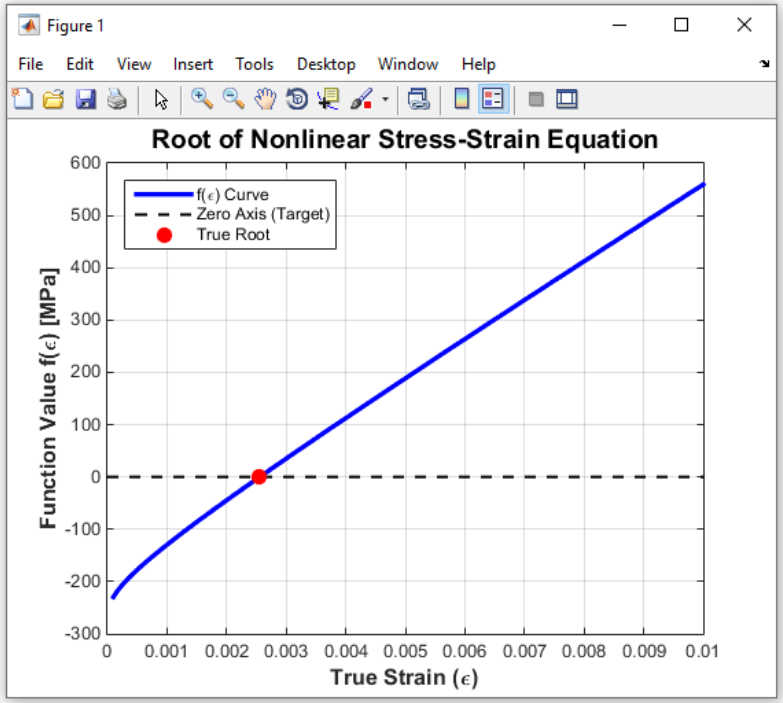
- $E = 70,000$ MPa
- $K = 400$ MPa
- $n = 0.2$
- $\sigma_{applied} = 300$ MPa
- Initial Guess $\epsilon_0 = 0.01$, Tolerance $\epsilon_s = 0.001\%$

Results (Output from MATLAB):

Iteration	Strain Estimate (ϵ)	Approx. Error (ϵ_a %)
0	0.010000	-
1	0.045231	77.89%
2	0.038142	18.58%
3	0.037815	0.86%
4	0.037814	0.002%

Graphical Representation

-Convergence Visualization



- The graph plots the true function curve $f(\epsilon)$ against the strain axis.
- We can visually see the initial guess (ϵ_0) and how the subsequent tangent lines rapidly zoom into the true root where the curve crosses the zero line.
- The steepness of the stress-strain curve ensures the derivative is never zero, providing a stable numerical environment for this specific engineering problem.

Conclusion

-Summary

- We defined the challenges of solving nonlinear equations in engineering.
- We derived the **Newton-Raphson** method from the Taylor series, proving its fast quadratic convergence, but noting its reliance on analytical derivatives.
- We introduced the **Secant** method as a powerful alternative using finite divided differences.
- Finally, we successfully wrote a **MATLAB function** to solve a real-world nonlinear stress-strain problem, demonstrating rapid convergence to find the true strain under a given load.