2007 European Society of Biomechanics thematic workshop on Finite Element Modelling in Biomechanics and Mechanobiology

Non-Linear Finite Element Analysis: Finite Element Solution Schemes I & II

Peter McHugh

Department of Mechanical and Biomedical Engineering
National Centre for Biomedical Engineering Science
National University of Ireland, Galway

Outline

- Basic Principles of FE
- Solid Mechanics BVP
- Linear Problems
- Non-linear Problems
- Solution Schemes
  - Implicit (Newton-Raphson)
  - Explicit Methods
  - Dynamic Explicit Methods
- Stress Update Algorithm
- Generalisations
- Summary

Introduction to FE

"The finite element method is a means of obtaining approximate numerical solutions to field problems"

- Discretise body into regions – elements
- Elements connected at special points – nodes
- Replace solution variable distribution with approximate distribution based on:
  - fixed solution "shapes" over elements
  - solution variable values at nodes

Continuous $\rightarrow$ *Discrete*

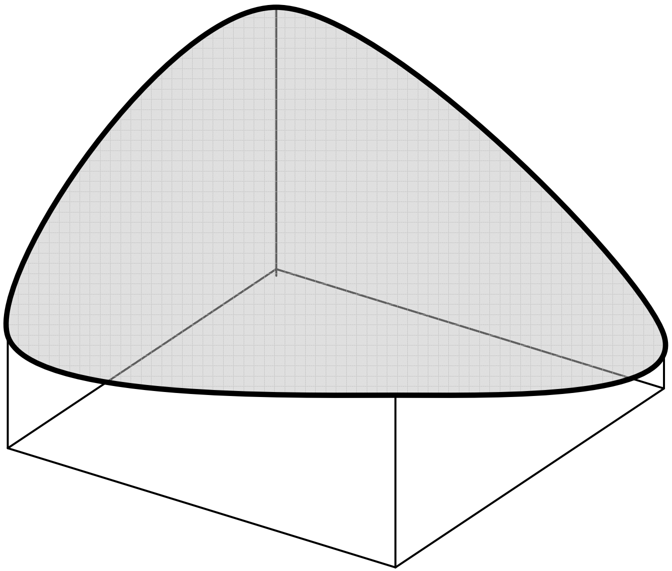
FE Approximation

- Temperature distribution $u(\mathbf{x})$
  - a single degree of freedom case (SDOF)

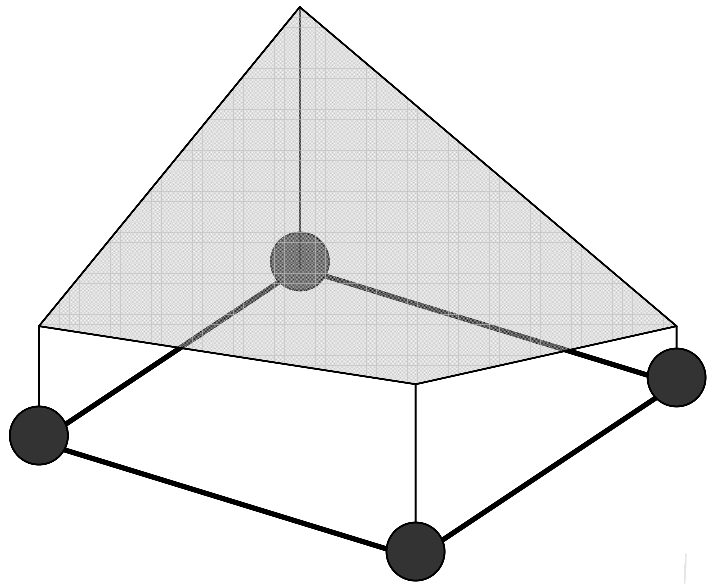
$$u(\mathbf{x}) \Rightarrow \sum_{a=1}^{n} N_a(\mathbf{x}) \cdot u_a$$

- Approximation: assumption of "shape" of solution throughout element – usually polynomial – linear, quadratic,…
- More nodes & elements $\rightarrow$ greater accuracy
- Generally quadratic better than linear

Basic Idea – One element



real



FE: 4-noded quad

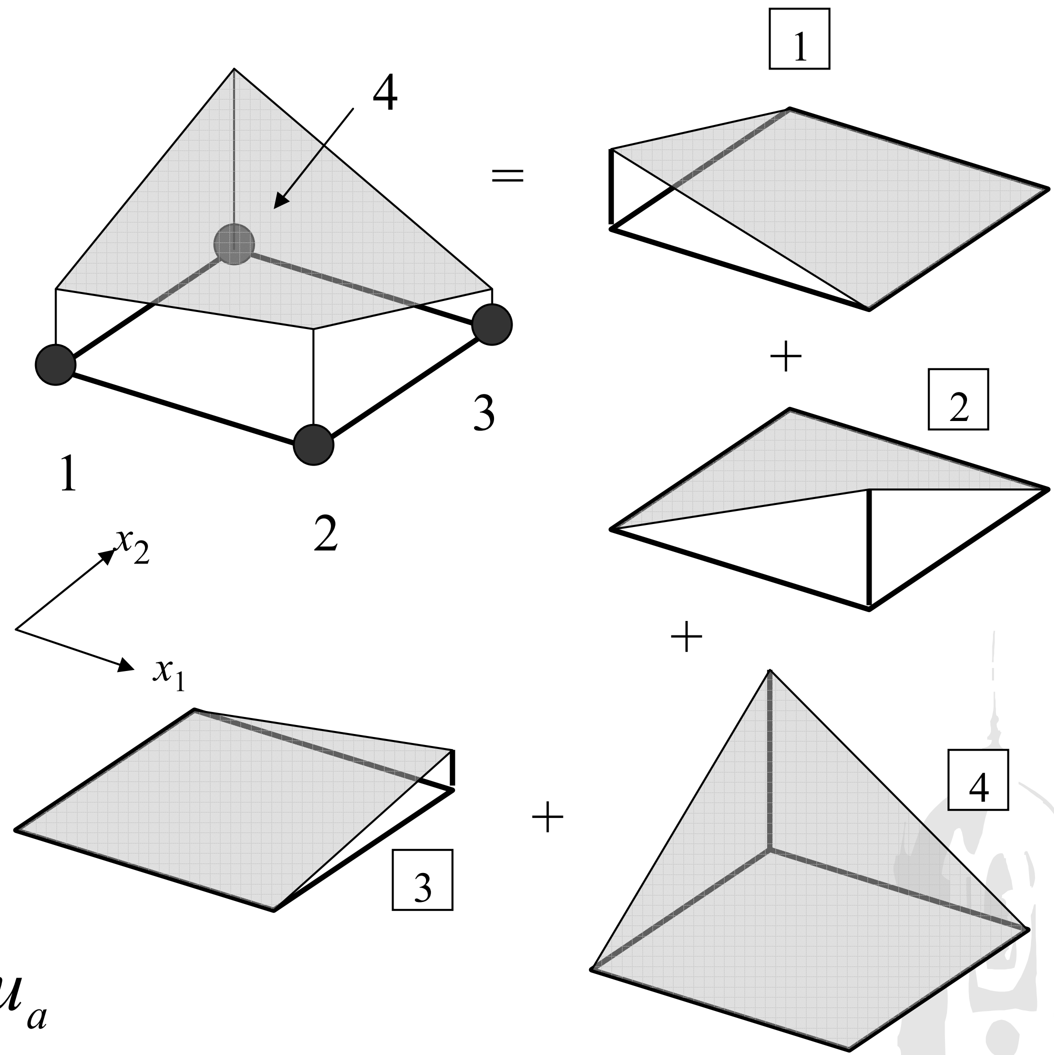
Nodal values:

- actual unknowns solved for in FEM since "shape" is assumed
- approximations to exact solution at nodes

Basic Idea – One element

Shape Functions

4–noded quad



$$u(\mathbf{x}) = \sum_{a=1}^{4} N_a(\mathbf{x}) \cdot u_a$$

SDOF → MDOF

Single degree of freedom case (SDOF)

$$u(\mathbf{x}) = \sum_{a=1}^{n} N_a(\mathbf{x}) \cdot u_a$$

Multi degree of freedom case (MDOF), e.g. displacements in 2D or 3D

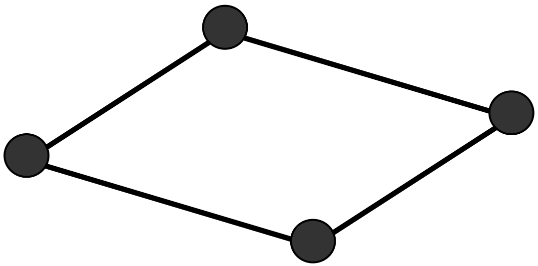
$$\mathbf{u}(\mathbf{x}) = \sum_{a=1}^{n} N_a(\mathbf{x}) \cdot \mathbf{u}_a$$

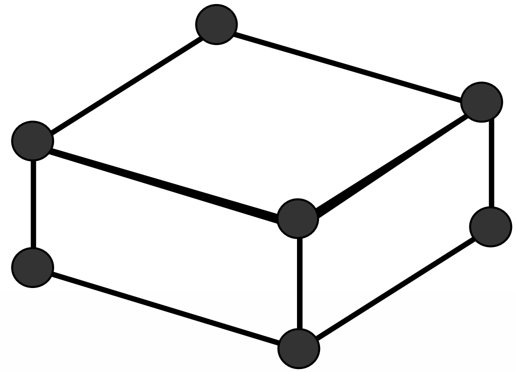
Matrix/Vector notation

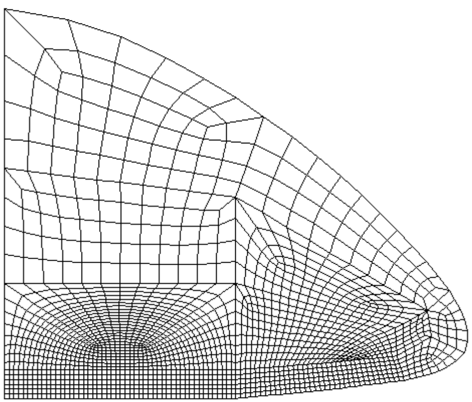
$$\mathbf{u} = \mathbf{N}\mathbf{u}_e$$

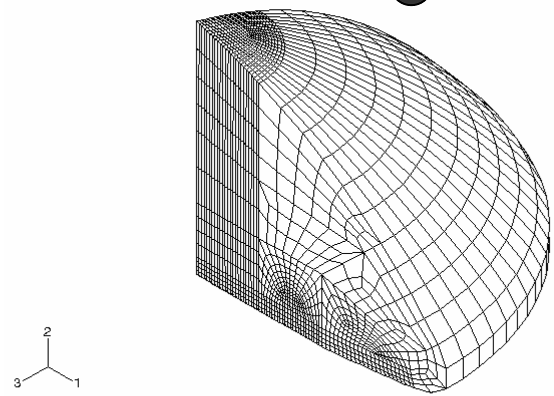
Vector of nodal displacements

One element → Mesh









- 2D: Quad/Triangle – 3D: Hex./Tetrahedral
- Good idea to keep mesh reasonably regular

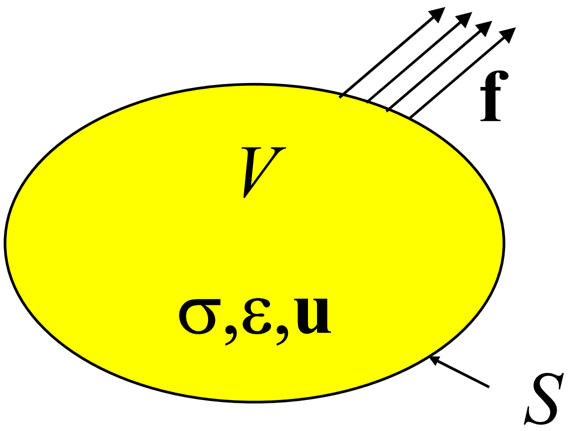
Field Problems

- Usually Field Problems are posed in the STRONG FORM
  - Partal differential equations PDEs + Boundary conditions BCs
  - PDEs + BCs $\rightarrow$ Boundary Value Problem (BVP)
  - “Pointwise” form
- FE solutions come from the WEAK FORM
  - Convert “pointwise” form to integral form over whole body
  - Principle of Virtual Work, Galerkin Method, etc.
  - Same information contained

Solid Mechanics BVP

Principle of Virtual Work (PVW)

$$\int_V \delta\boldsymbol{\varepsilon}^{\mathrm{T}}\boldsymbol{\sigma}\,dV = \int_S \delta\mathbf{u}^{\mathrm{T}}\mathbf{f}\,dS$$



For 2D case:

$$\delta\boldsymbol{\varepsilon} \quad \begin{pmatrix} \delta\varepsilon_{11} \\ \delta\varepsilon_{22} \\ 2\delta\varepsilon_{12} \end{pmatrix} \qquad \boldsymbol{\sigma} \quad \begin{pmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{12} \end{pmatrix} \qquad \delta\mathbf{u} \quad \begin{pmatrix} \delta u_1 \\ \delta u_2 \end{pmatrix} \qquad \mathbf{f} \quad \begin{pmatrix} f_1 \\ f_2 \end{pmatrix}$$

FE Approximation

$$\mathbf{u} = \mathbf{N}\mathbf{u}_e \qquad \delta\mathbf{u} = \mathbf{N}\delta\mathbf{u}_e$$

$$\boldsymbol{\varepsilon} = \tfrac{1}{2}\left(\frac{\partial \mathbf{u}}{\partial \mathbf{x}} + \frac{\partial \mathbf{u}}{\partial \mathbf{x}}^{\mathrm{T}}\right) = \mathbf{B}\mathbf{u}_e \qquad \delta\boldsymbol{\varepsilon} = \mathbf{B}\,\delta\mathbf{u}_e$$

Principle of Virtual Work (PVW):

$$\int_V \delta\boldsymbol{\varepsilon}^{\mathrm{T}}\boldsymbol{\sigma}\,dV = \int_S \delta\mathbf{u}^{\mathrm{T}}\mathbf{f}\,dS$$

$$\int_V \delta\mathbf{u}_e^{\mathrm{T}}\mathbf{B}^{\mathrm{T}}\boldsymbol{\sigma}\,dV = \int_S \delta\mathbf{u}_e^{\mathrm{T}}\mathbf{N}^{\mathrm{T}}\mathbf{f}\,dS$$

FE Approximation

$$\int_V \delta\mathbf{u}_e^{\mathrm{T}} \mathbf{B}^{\mathrm{T}} \boldsymbol{\sigma} \, dV = \int_S \delta\mathbf{u}_e^{\mathrm{T}} \mathbf{N}^{\mathrm{T}} \mathbf{f} \, dS$$

Eliminate virtual displacements:

$$\int_V \mathbf{B}^{\mathrm{T}} \boldsymbol{\sigma} \, dV = \int_S \mathbf{N}^{\mathrm{T}} \mathbf{f} \, dS$$

Fundamental FE Equations to solve:

$$\boxed{\int_V \mathbf{B}^{\mathrm{T}} \boldsymbol{\sigma}(\mathbf{u}_e) \, dV = \mathbf{F}}$$

Linear Problem

Linear Elasticity

$$\boldsymbol{\sigma} = \mathbf{D}\boldsymbol{\varepsilon} = \mathbf{D}\mathbf{B}\mathbf{u}_e$$

Input into FE equations:

$$\int_V \mathbf{B}^{\mathrm{T}} \boldsymbol{\sigma} \, dV = \mathbf{F} \longrightarrow \int_V \mathbf{B}^{\mathrm{T}} \mathbf{D}\mathbf{B}\mathbf{u}_e \, dV = \mathbf{F}$$

Reorganise and define stiffness matrix $\mathbf{K}$:

$$\int_V \mathbf{B}^{\mathrm{T}} \mathbf{D}\mathbf{B}\mathbf{u}_e \, dV = \mathbf{F} \longrightarrow \left(\int_V \mathbf{B}^{\mathrm{T}} \mathbf{D}\mathbf{B} \, dV \right) \mathbf{u}_e = \mathbf{F}$$

Linear Problem

$$\left(\int_V \mathbf{B}^{\mathrm{T}}\mathbf{D}\mathbf{B}\,dV\right)\mathbf{u}_e = \mathbf{F}$$

$$\downarrow$$

$$\boxed{\begin{array}{c}\text{Numerical Integration (for each element)}\\ \text{Assembly of global matrix/vectors}\end{array}}$$

$$\downarrow$$

$$\mathbf{K}\mathbf{u}_e = \mathbf{F}$$

- Solution usually “straightforward”
- Can be achieved in a single step
- Apply $\mathbf{F}$, invert $\mathbf{K}$, solve for $\mathbf{u}_e$
- Go back and determine ε and σ

Non-Linear Problems

Non-linearities can arise for many reasons, e.g.

- geometric non-linearities
  - large deformation kinematics
  - non-linear ε-$\mathbf{u}$ relationship

$$\boldsymbol{\varepsilon} = \tfrac{1}{2}\left(\frac{\partial \mathbf{u}}{\partial \mathbf{x}} + \frac{\partial \mathbf{u}}{\partial \mathbf{x}}^{\mathrm{T}} + \frac{\partial \mathbf{u}}{\partial \mathbf{x}}\frac{\partial \mathbf{u}}{\partial \mathbf{x}}^{\mathrm{T}}\right)$$

$$= \mathbf{B}(\mathbf{u}_e)\mathbf{u}_e$$

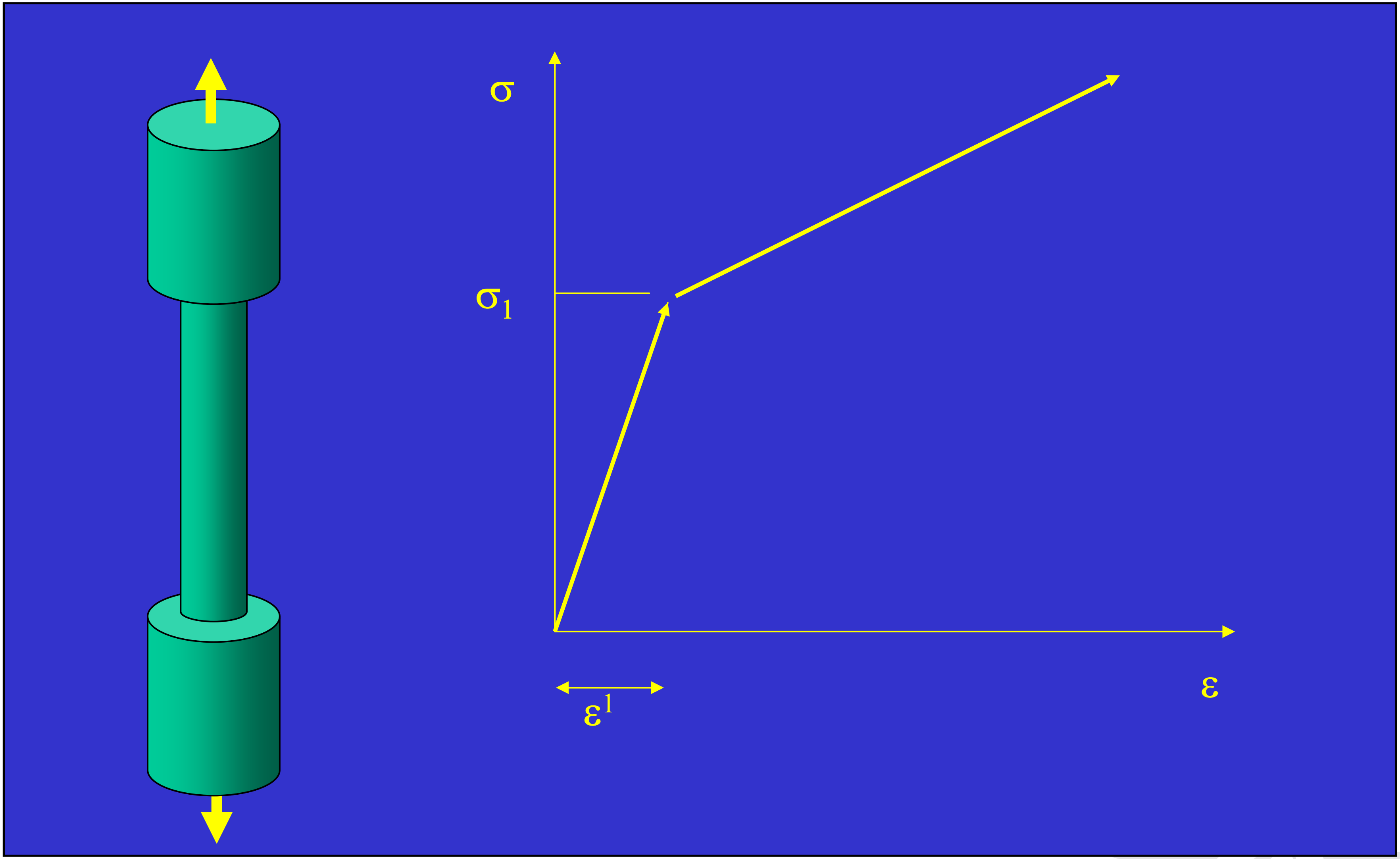
- material non-linearities
  - non-linear constitutive law
  - non-linear σ-ε relationship

$$\boldsymbol{\sigma} = \boldsymbol{\sigma}(\mathbf{u}_e)$$

- non-linear boundary conditions
  - surface contact

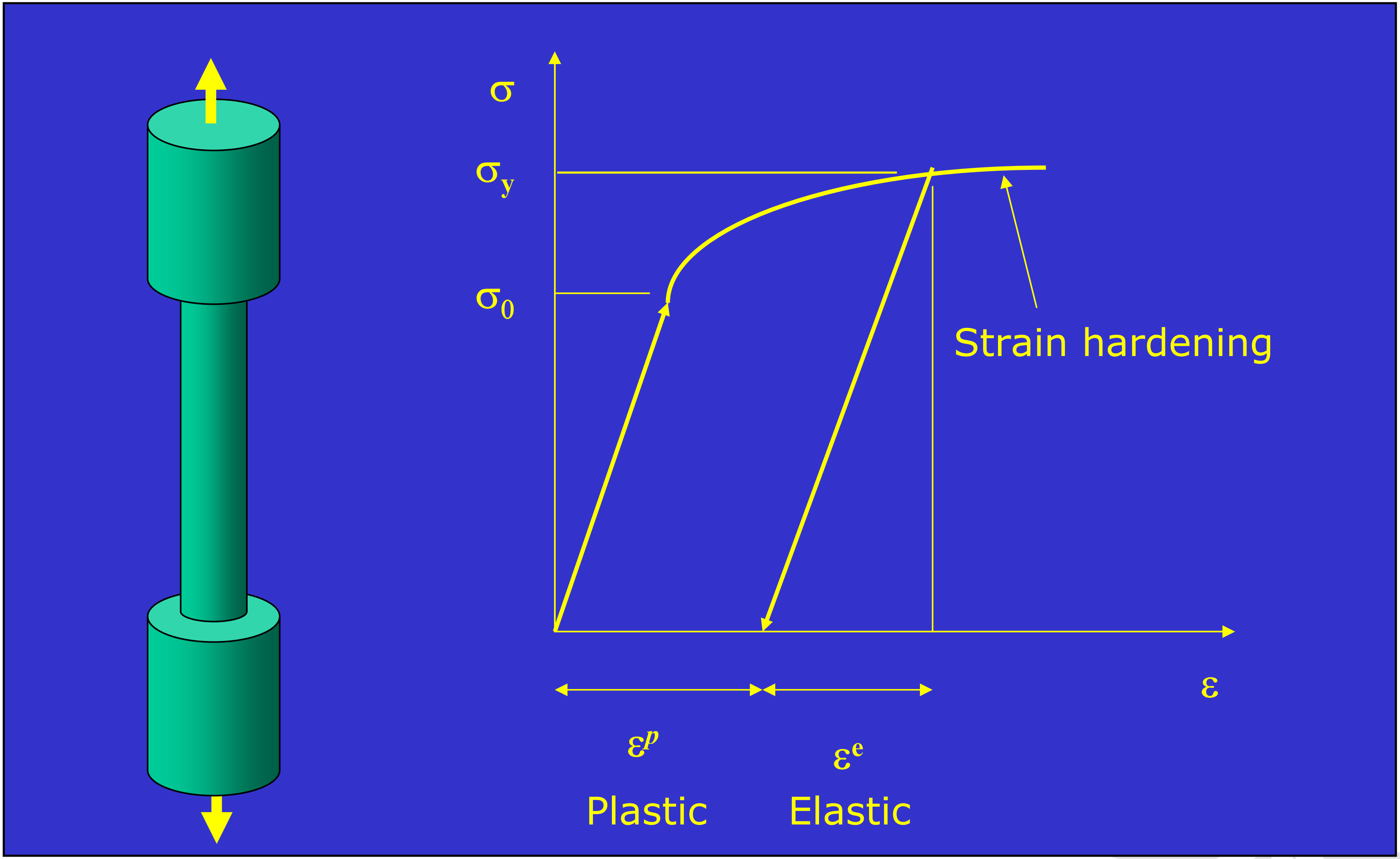
Non-linear Material

- Bi-linear elastic material



Non-linear Material

- Elastic-plastic material



Non-Linear Problem

FE equations

$$\int_V \mathbf{B}(\mathbf{u}_e)^{\mathrm{T}} \boldsymbol{\sigma}(\mathbf{u}_e) dV = \mathbf{F}$$

$$\int_V \mathbf{B}(\mathbf{u}_e)^{\mathrm{T}} \boldsymbol{\sigma}(\mathbf{u}_e) dV - \mathbf{F} = \mathbf{G}(\mathbf{u}_e) = \mathbf{0}$$

- $\mathbf{G}$ is the out of balance/residual force
- $\mathbf{G} = \mathbf{0}$ is a set of non-linear equations in $\mathbf{u}_e$
- Solution usually by **incremental** methods
  - applying load in increments/steps: $t \rightarrow t+\Delta t$
  - stepping to final time t_{final} in time steps Δt and solving for each step
  - ***Implicit*** and ***Explicit*** methods used
  - drop "e" for convenience

Implicit

Solved for t, wish to solve for $t+\Delta t$

$$\mathbf{G}(\mathbf{u}^{t+\Delta t}) = \mathbf{0}$$

- **Implicit**: Solve for $t+\Delta t$ using state at t and $t+\Delta t$
  - don't know state at $t+\Delta t$ yet
  - Newton-Raphson method used typically – ABAQUS/Standard, ANSYS, MARC,…
  - take initial guess and **iterate** to convergence
  - end up solving "linear-like" equation for each iteration: $\mathbf{Ku} = \mathbf{F}$
  - very accurate
  - can use relatively large time steps

Explicit

- **Explicit**: Solve for $t+\Delta t$ using state at t
  - know state at t so can calculate $\mathbf{K}_t$
  - solve directly for incremental displacements:

    $$\mathbf{K}_t\,\Delta\mathbf{u} = \Delta\mathbf{F}$$
  - no iteration
  - no convergence check
  - usually used in purpose written codes
  - method is very robust
  - must use very small time steps (x10 $\rightarrow$ x1000)

Implicit: Newton-Raphson

Material non-linearity

$$\mathbf{G}(\mathbf{u}^{t+\Delta t}) = \int_V \mathbf{B}^{\mathrm{T}} \boldsymbol{\sigma}(\mathbf{u}^{t+\Delta t})\, dV - \mathbf{F} = \mathbf{0}$$

- Assume solved for state at t
- $\mathbf{u}^{t}$ are known
- Apply load increment
- Wish to update state to $t+\Delta t$
- $\mathbf{u}^{t+\Delta t}$ are main variables
- How does NR method work?

Newton-Raphson

Look at 1D: Wish to solve $f(x) = 0$

Guess at root x_i: Better guess $x_{i+1} \rightarrow$ NR formula

Method applied iteratively:

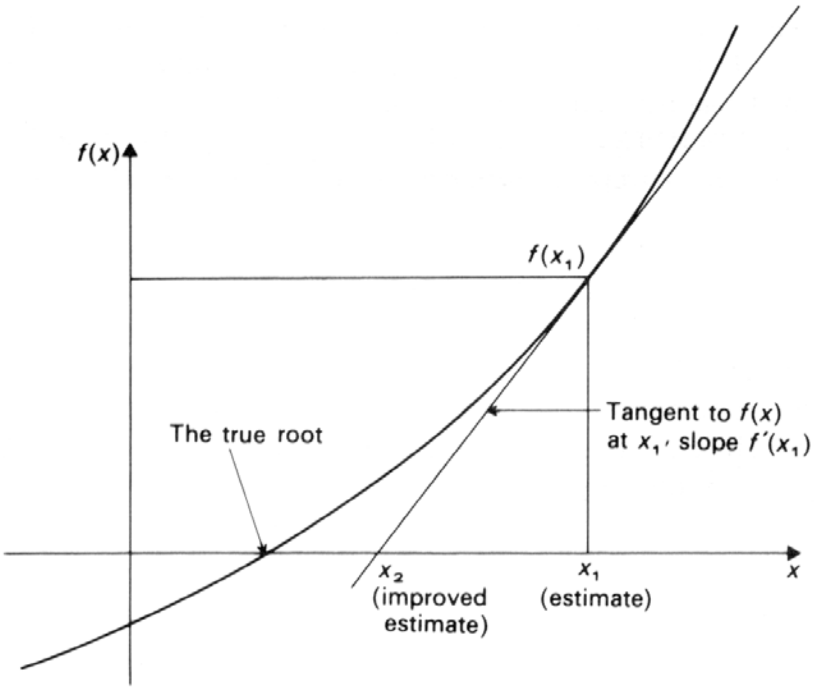
$x_{i+1} \rightarrow x_i$ and reapply NR formula

$$x_{i+1} = x_i - \left[\frac{df}{dx}\right]_{x_i}^{-1} \cdot f(x_i)$$

Continue to iterate until convergence:

$$\left|x_{i+1} - x_i\right| < Tolerance$$

$$\left|f(x_{i+1})\right| < Tolerance$$



Newton-Raphson

For FE same principle $\mathbf{G}(\mathbf{u}^{t+\Delta t}) = \mathbf{0}$

For i^{th} iteration $$\mathbf{u}_{i+1}^{t+\Delta t} = \mathbf{u}_i^{t+\Delta t} - \left[\frac{\partial \mathbf{G}\left(\mathbf{u}_i^{t+\Delta t}\right)}{\partial \mathbf{u}}\right]^{-1} \mathbf{G}\left(\mathbf{u}_i^{t+\Delta t}\right)$$

Reorganise $$\delta \mathbf{u}_{i+1} = \mathbf{u}_{i+1}^{t+\Delta t} - \mathbf{u}_i^{t+\Delta t} = -\left[\frac{\partial \mathbf{G}\left(\mathbf{u}_i^{t+\Delta t}\right)}{\partial \mathbf{u}}\right]^{-1} \mathbf{G}\left(\mathbf{u}_i^{t+\Delta t}\right)$$

$\mathbf{K}$ – tangent stiffness matrix

$$\delta \mathbf{u}_{i+1} = -\mathbf{K}\left(\mathbf{u}_i^{t+\Delta t}\right)^{-1} \mathbf{G}\left(\mathbf{u}_i^{t+\Delta t}\right)$$

$$\boxed{\mathbf{K}\left(\mathbf{u}_i^{t+\Delta t}\right)\delta \mathbf{u}_{i+1} = -\mathbf{G}\left(\mathbf{u}_i^{t+\Delta t}\right)}$$

Newton-Raphson

$$\mathbf{K}\left(\mathbf{u}_i^{t+\Delta t}\right)\delta\mathbf{u}_{i+1} = -\mathbf{G}\left(\mathbf{u}_i^{t+\Delta t}\right)$$

- Must be solved for each iteration for change in incremental displacements
- $\mathbf{K}$ and $\mathbf{G}$ are different for each iteration
- Same form as for linear problems: $\mathbf{Ku} = \mathbf{F}$
- Initial guess is usually $\mathbf{u}^t$

Convergence: $\left|\mathbf{G}\left(\mathbf{u}_{i+1}^{t+\Delta t}\right)\right| < Tolerance$

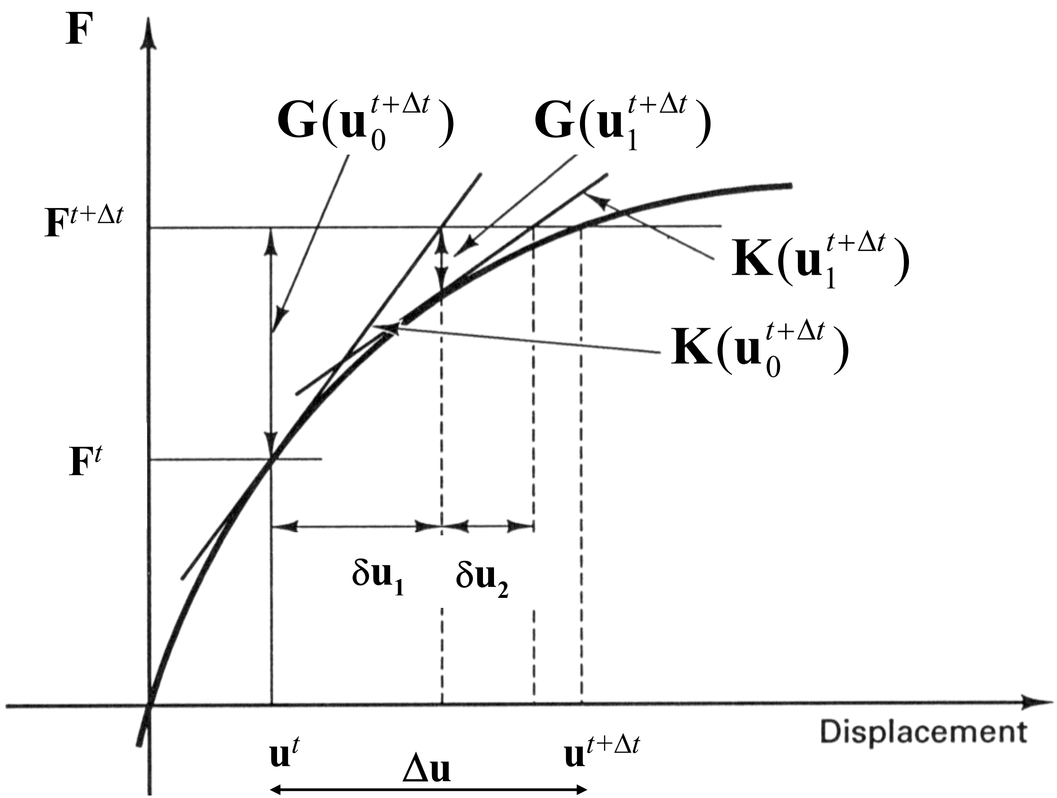
Current increment in displacements - for i^{th} iteration:

$$\mathbf{u}_i^{t+\Delta t} - \mathbf{u}^t = \Delta\mathbf{u}_i = \sum_{k=1}^{i} \delta\mathbf{u}_k$$

Newton-Raphson

$$\mathbf{u}_i^{t+\Delta t} - \mathbf{u}^t = \Delta \mathbf{u}_i = \sum_{k=1}^{i} \delta \mathbf{u}_k$$

Schematic of iteration process:



Newton-Raphson

Method requires accurate evaluation of $\mathbf{G}(\mathbf{u}_i^{t+\Delta t})$

For each iteration i $\mathbf{K}(\mathbf{u}_i^{t+\Delta t})$

$\mathbf{G}$ requires accurate σ for current estimate of $\mathbf{u}^{t+\Delta t}$

$$\mathbf{G}(\mathbf{u}_i^{t+\Delta t}) = \int_V \mathbf{B}^{\mathrm{T}} \boldsymbol{\sigma}(\mathbf{u}_i^{t+\Delta t}) dV - \mathbf{F}^{t+\Delta t} = \mathbf{0}$$

Requires a **Stress Update Algorithm (SUA)**

Newton-Raphson

Look at $\mathbf{K}$

$$\mathbf{K}\left(\mathbf{u}_i^{t+\Delta t}\right)=\frac{\partial \mathbf{G}\left(\mathbf{u}_i^{t+\Delta t}\right)}{\partial \mathbf{u}}=\int_V \mathbf{B}^{\mathrm{T}} \left.\frac{\partial \boldsymbol{\sigma}}{\partial \mathbf{u}}\right|_{\left(\mathbf{u}_i^{t+\Delta t}\right)} dV$$

$$=\int_V \mathbf{B}^{\mathrm{T}} \left.\frac{\partial \boldsymbol{\sigma}}{\partial \boldsymbol{\varepsilon}}\right|_{\left(\mathbf{u}_i^{t+\Delta t}\right)} \frac{\partial \boldsymbol{\varepsilon}}{\partial \mathbf{u}} dV=\int_V \mathbf{B}^{\mathrm{T}} \left.\frac{\partial \boldsymbol{\sigma}}{\partial \boldsymbol{\varepsilon}}\right|_{\left(\mathbf{u}_i^{t+\Delta t}\right)} \mathbf{B}\, dV$$

Jacobian of constitutive law

Consistent Tangent Matrix

$$\mathbf{D}^{tan}=\left.\frac{\partial \boldsymbol{\sigma}}{\partial \boldsymbol{\varepsilon}}\right|_{\left(\mathbf{u}_i^{t+\Delta t}\right)}$$

$$\boxed{\mathbf{K}\left(\mathbf{u}_i^{t+\Delta t}\right)=\int_V \mathbf{B}^{\mathrm{T}} \mathbf{D}^{tan} \mathbf{B}\, dV}$$

Form of $\mathbf{K}$

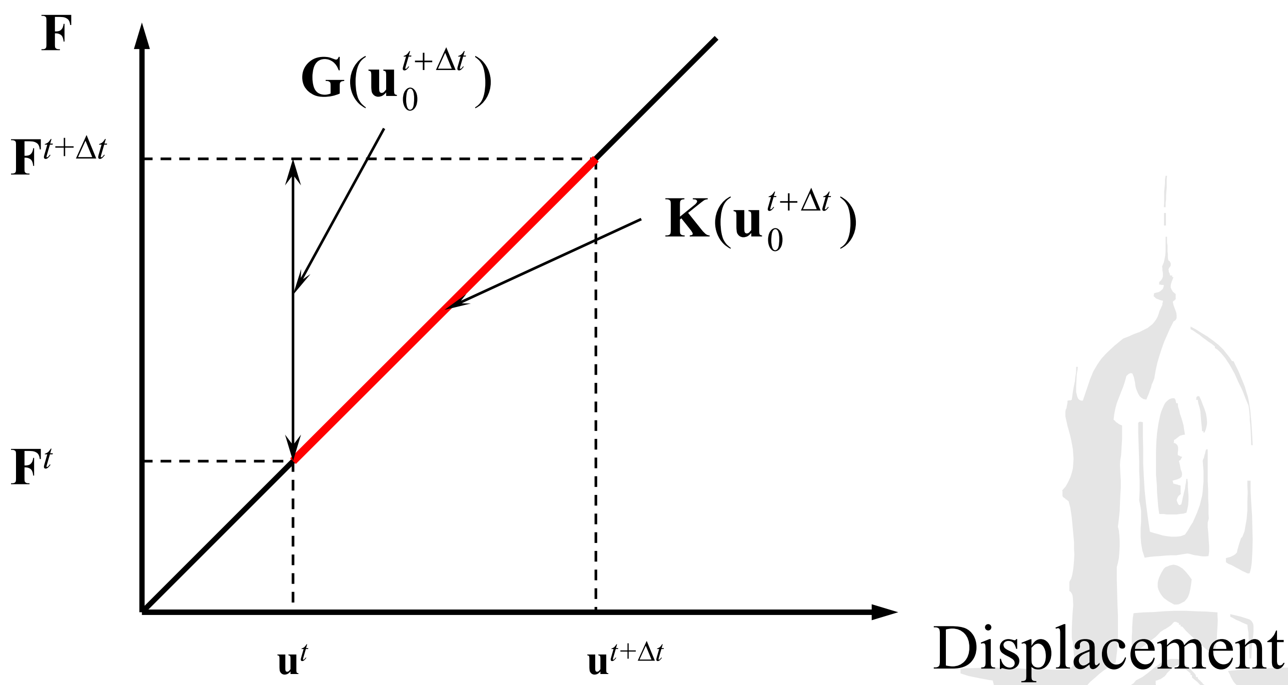
- Same form as for linear problem
- Different for each iteration
- Consistent tangent matrix can be difficult to evaluate for complex constitutive laws

Newton-Raphson Re-cap

- Load applied incrementally
- For each increment, iteration is performed until convergence is achieved
- Need to be able to calculate $\mathbf{K}$ and $\mathbf{G}$ accurately

Application: Linear Elasticity

- $\mathbf{D}^{tan} = \mathbf{D}$ (constant)
- $\mathbf{K}$ constant for each iteration
- Convergence reached in 1 iteration
- Can apply full load in one increment
- Same as simple linear one-step solution



Newton-Raphson

- Accurate and displays rapid convergence
- However there are modified methods used
  - Simplified methods:
    - Constant $\mathbf{K}$ – from first iteration in increment
    - Initial stress method – $\mathbf{K}$ from first increment
  - Complex methods: BFGS, etc.
- Can modify $\mathbf{K}$ but still must calculate $\mathbf{G}$ accurately (SUA)

Non-Linear Solution Methods

- Implicit methods: Newton-Raphson
  - NR: Gold Standard
  - Rigorous convergence criterion
- Explicit methods: $\mathbf{K}_t \Delta\mathbf{u} = \Delta\mathbf{F}$
- Both involve formation and inversion of the global stiffness matrix $\mathbf{K}$
- Major computational chore – processing and storage
  - Huge efforts made in developing efficient storage and processing methods
  - Skyline solvers, element by element solvers, etc.
- Alternative?

Dynamic Explicit Methods

- Problems reformulated as dynamic
- Include nodal velocities, accelerations
- Include inertia $\rightarrow$ mass matrix $\mathbf{M}$

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{G}(\mathbf{u}, \dot{\mathbf{u}}) = \mathbf{0}$$

- Problems solved incrementally
- No need to form or invert $\mathbf{K}$ at all!
- LS-Dyna, ABAQUS/Explicit,…
- Method is very robust – great for highly non-linear problems
- No convergence check – must be careful about accuracy and stability
- Must use very small time steps (x10 $\rightarrow$ x1000)
- Algorithms for determining Δt

Central Difference Method

- No formation of **K,** but accuracy in **G** still required
- **M** in diagonal form
- Method works in "half increments"

  i-1/2, i, i+1/2, i+1,…
- Solution "marches through time"
- For increment i:

$$\ddot{\mathbf{u}}_{\mathrm{i}} = -\mathbf{M}^{-1}\mathbf{G}_{\mathrm{i}}$$

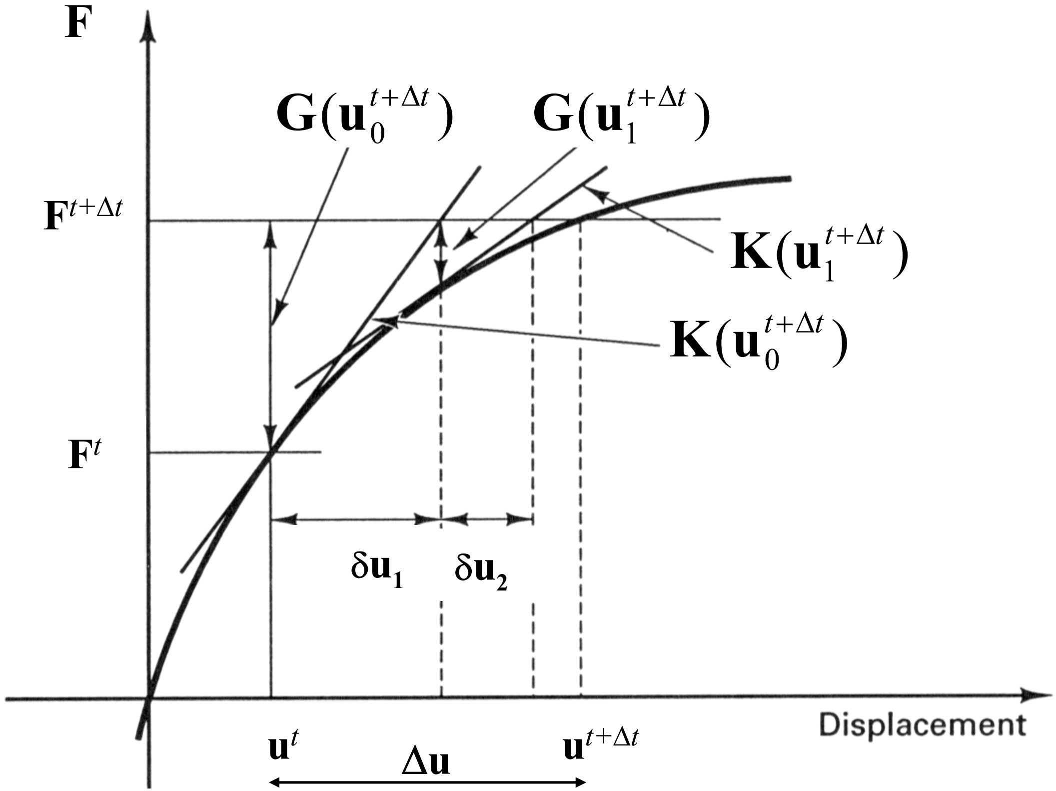
$$\dot{\mathbf{u}}_{\mathrm{i}+\frac{1}{2}} = \dot{\mathbf{u}}_{\mathrm{i}-\frac{1}{2}} + \frac{\Delta t_{\mathrm{i}+1} + \Delta t_{\mathrm{i}}}{2}\ddot{\mathbf{u}}_{\mathrm{i}}$$

$$\mathbf{u}_{\mathrm{i}+1} = \mathbf{u}_{\mathrm{i}} + \Delta t_{\mathrm{i}+1}\dot{\mathbf{u}}_{\mathrm{i}+\frac{1}{2}}$$

Stress Update Algorithm

- To get accurate solution for any iteration/increment, need accurate $\mathbf{G}^{t+\Delta t}$

$$\mathbf{G}(\mathbf{u}^{t+\Delta t}) = \int_V \mathbf{B}^{\mathrm{T}} \boldsymbol{\sigma}(\mathbf{u}^{t+\Delta t}) dV - \mathbf{F}^{t+\Delta t} = \mathbf{0}$$



Stress Update Algorithm

- Stress can depend on many variables/phenomena
  - displacement/strain, temperature/heat flux, diffusion, evolving porosity, etc…
  - relevant material properties for each phenomenon
- Need $\sigma^{t+\Delta t}$ to be accurately calculated as a function of changes in the independent variables: $t \rightarrow t+\Delta t$
- Not trivial for very complex (multi-physics) or non-linear systems

Stress Update Algorithm

Commerical codes (ANSYS, ABAQUS, MARC,…)

- Using standard material models available
  - elasticity, visco-elasticity, plasticity,….
  - accuracy usually “guaranteed”

Why Emphasis Here?

Commerical codes (ANSYS, ABAQUS, MARC,…)

- Using User Material modules (ABAQUS-UMAT)
- Now common in mechanics and biomechanics
- Great freedom in describing stress dependence on different variables – σ(mech, therm, chem, bio)
  - Bio: protein synthesis, actin fibre/bundle formation,…
- Allow material properties to evolve through time

- ABAQUS: $\Delta t, \mathbf{u}^{t+\Delta t} \rightarrow$ UMAT
- UMAT: $\sigma^{t+\Delta t} \rightarrow$ ABAQUS
- ABAQUS believes correct – it does not check!!

Why Emphasis Here?

Many constitutive laws are in rate form & non-linear

$$\dot{\boldsymbol{\sigma}} = \mathbf{f}(\boldsymbol{\varepsilon}, \dot{\boldsymbol{\varepsilon}}, T, \dot{T}, ...)$$

$$\boldsymbol{\sigma}^{t+\Delta t} = \boldsymbol{\sigma}^{t} + \Delta\boldsymbol{\sigma}$$

Not trivial to determine $\Delta\sigma$ based on Δt, $\mathbf{u}^{t+\Delta t}$

- Algorithms: Simple Euler, Backward Euler, Central Difference, Radial Return,…
- User need to ensure UMAT performing accurately ***before*** use

Other Observations 1

- Considered solid mechanics situation
  - Dealing with σ
  - Although generalised to multi-physics problems
- However, general methods and cautions hold true for other problem types
  - Thermal: heat flux and temperature
  - Convection+diffusion: mass transport and concentration

Other Observations 2

- Incremental solution methods vital for non-linear problems
- However, also very important for any time/rate-dependent problem
  - Both linear and non-linear
  - Track how state is changing over time (transient)
  - Same methodologies used
  - Visco-elasticity
  - Creep and visco-plasticity

Summary

- Introduced FE - Linear & Non-linear
- Linear – single $\mathbf{K}$ matrix inversion
- Non-linear – incremental methods
  - Implicit: Newton-Raphson – iteration – gold standard
  - Dynamic Explicit: No $\mathbf{K}$ – no iteration – small time steps
  - ***Must*** have accurate stress update algorithm
  - User modules for com. codes → great flexibility to deal with multi-physics problems
  - General principles applicable to other problem types – thermal, mass transport, etc.
- Incremental methods
  - necessary for time dependent problems