CE394M: FEM solvers and errors

Krishna Kumar

University of Texas at Austin

krishnak@utexas.edu

February 27, 2019

Overview

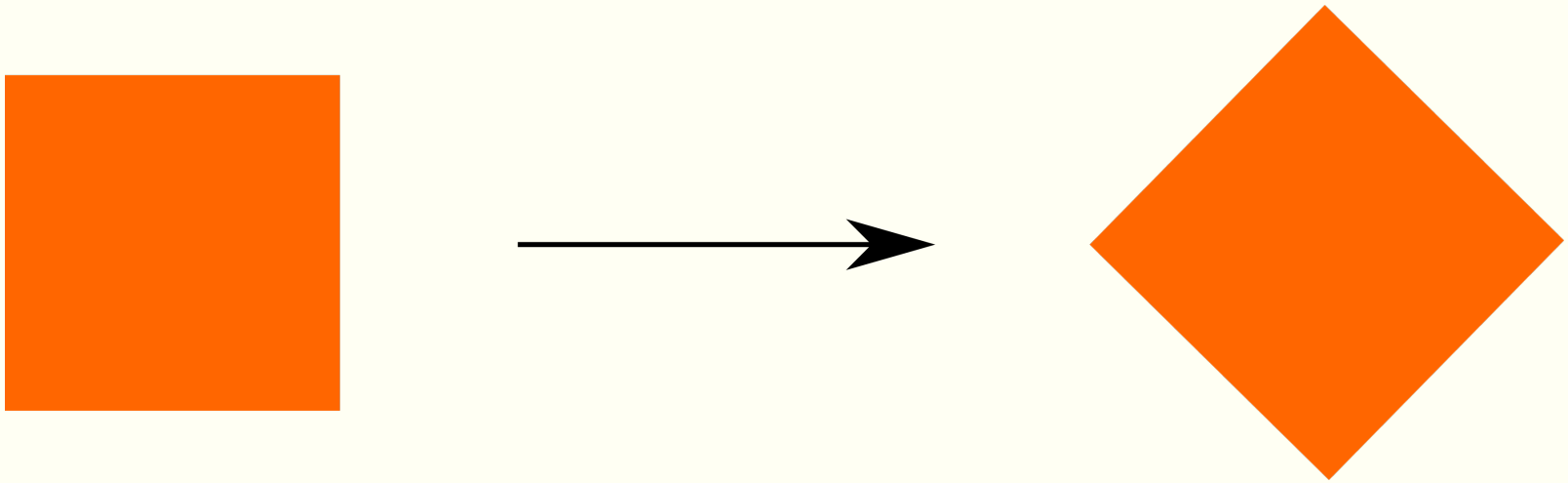
1. Solving non-linear problems
   - Iterative solvers
   - Tangent stiffness
   - Newton Raphson
2. Arc length method
3. Error estimates
   - A priori estimates
   - Posterior error estimation
4. Time-dependent problems
   - Explicit schemes
   - Implicit method

Linear and non-linear problems

- **Linear problems**

- **Non-linear problems**

Non-linearity in geotechnical engineering



No strains if linear strain-displacement relations are used.

Linear solvers

To solve a system of linear equations of the form $\mathbf{Ka} = \mathbf{b}$, there are two families of methods of solvers that can be used: direct and iterative.

Iterative solvers

The system of equations that we derived from the finite element approximation of the BVP:

Tangent stiffness method

Newton Raphson method

Newton Raphson method

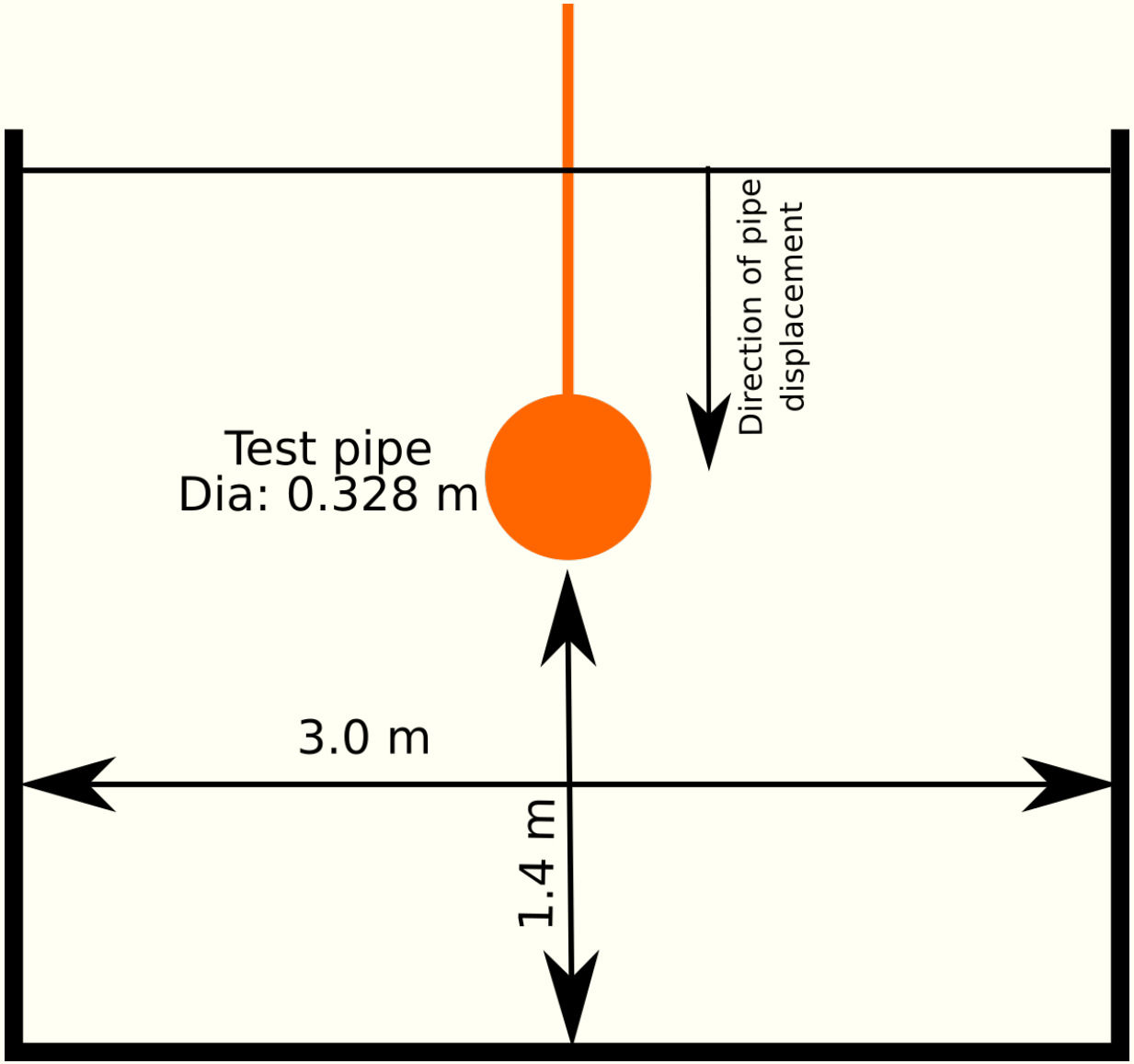
The incremental loading is expressed as follows. The external load vector F_{ext} is gradually increased from 0 in order to reach a desired value $\mathbf{F}^*$. Assuming that $\mathbf{F}^*$ itself remains constant during the analysis in terms of its 'direction' and only its magnitude is changing, we can write $F_{\text{ext}} = q$ known just to simplify our expression for the system of equations. Then we can control how the external load vector increases or decreases by introducing a scalar quantity λ and express the system as follows:

Newton Raphson method

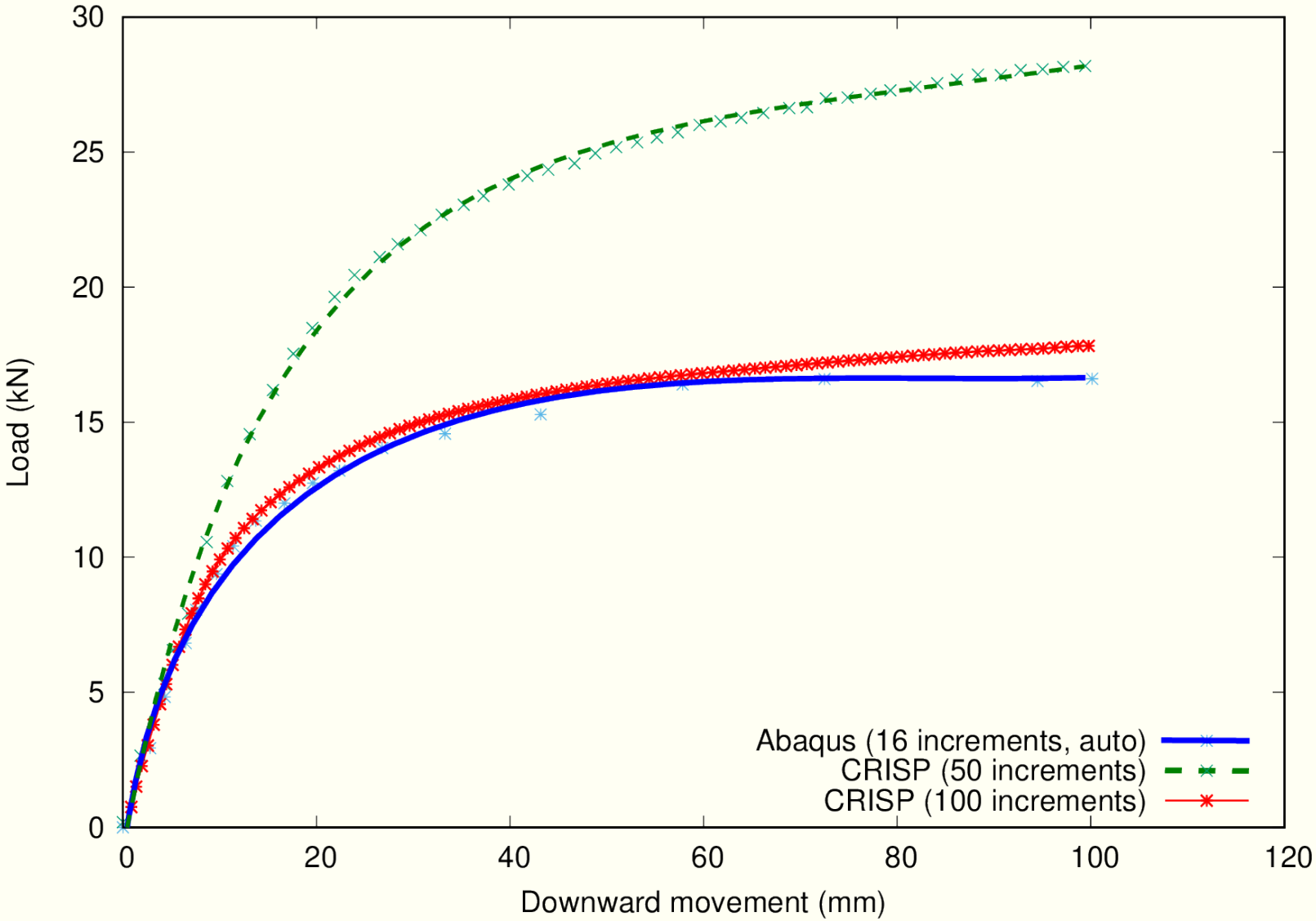
Newton Raphson method: Tolerance and Convergence

- Program - ABAQUS
- Newton-Raphson is the most standard method to solve nonlinear problems in FE.
- A large error in the initial estimate can contribute to non-convergence of the algorithm.
- **Tolerance**
  - must be small enough to ensure that the approximate solution is close to the exact mathematical solution.
  - must be large enough so that reasonable number of iterations are performed.
- **Quadratic convergence**
  - If the tangent stiffness is calculated correctly, R should reduce quadratically from one iteration to the next.

Tangent-Stiffness vs Newton Raphson



Tangent-Stiffness vs Newton Raphson



Newton Raphson iterative procedure: FE example

In Newton's method:

where $\mathbf{R}$ is the residual vector, we expand $\mathbf{R}$ in Taylor's series:

$$\mathbf{R}(\mathbf{u}^{\mathbf{r}+\mathbf{1}}) = \mathbf{R}(\mathbf{u}^{\mathbf{r}}) + \left(\frac{\partial \mathbf{R}}{\partial \mathbf{u}}\right)^{(r)} \cdot \delta \mathbf{u} + \dots$$

Omitting higher order terms (2):

$$\left(\frac{\partial \mathbf{R}}{\partial \mathbf{u}}\right)^{(r)} \cdot \delta \mathbf{u} = -\mathbf{R}(\mathbf{u}^{\mathbf{r}})$$

$$[\mathbf{T}(\mathbf{u})^{\mathbf{r}}]\, \delta \mathbf{u} = -\mathbf{R}(\mathbf{u}^{\mathbf{r}})$$

where $[T]$ is called the *tangent matrix*

Newton Raphson iterative procedure: FE example

The terms of the tangent stiffness matrix are given by

$$\mathbf{T} := \mathbf{K} + \frac{\partial \mathbf{K}}{\partial \mathbf{u}}\,\mathbf{u} \qquad \rightarrow \qquad T_{ij} := K_{ij} + \sum_{m=1}^{n} \frac{\partial K_{im}}{\partial u_j} u_m \ .$$

For our 2×2 reduced stiffness matrix, we have

$$\begin{aligned}
T_{22} &= K_{22} + \frac{\partial K_{22}}{\partial u_2} u_2 + \frac{\partial K_{23}}{\partial u_2} u_3 \\
T_{23} &= K_{23} + \frac{\partial K_{22}}{\partial u_3} u_2 + \frac{\partial K_{23}}{\partial u_3} u_3 \\
T_{32} &= K_{32} + \frac{\partial K_{32}}{\partial u_2} u_2 + \frac{\partial K_{33}}{\partial u_2} u_3 \\
T_{33} &= K_{33} + \frac{\partial K_{32}}{\partial u_3} u_2 + \frac{\partial K_{33}}{\partial u_3} u_3 \ .
\end{aligned}$$

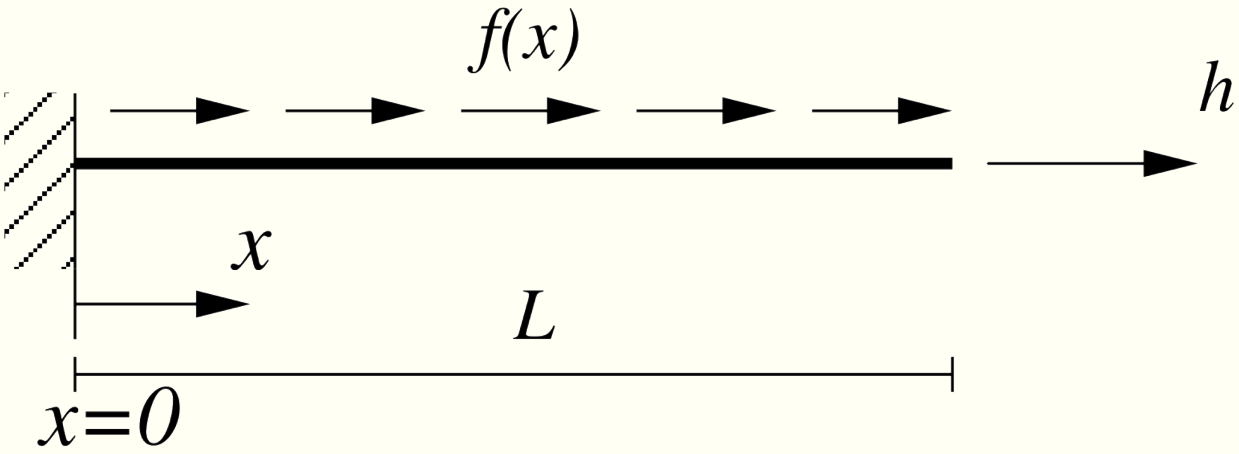
Newton Raphson iterative procedure: FE example

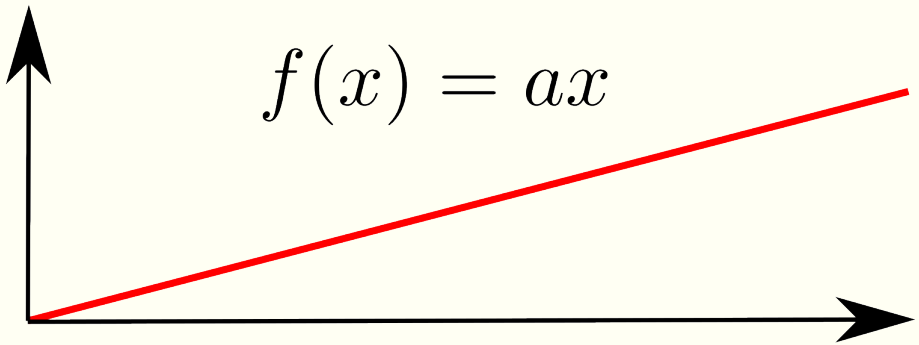
The component definition of the tangent matrix at the element level is:

$$[\mathbf{T}(\mathbf{u})^{\mathbf{r}}] = K + \frac{\partial K}{\partial u} \rightarrow \quad T_{ij} = K_{ij} + \sum_{m=1}^{n} \frac{\partial K_{im}}{\partial u_j} u_m$$

The residual vector after r iteration is:

Newton Raphson iterative procedure: FE example





The constitutive equation for this problem: $\sigma = E_0(1 - \varepsilon)\varepsilon$

Newton Raphson iterative procedure: FE example

For the axial loading of a nonlinear bar:

The weakform:

Newton Raphson iterative procedure: FE example

Multiplying the weak form by a weight function:

$$-\int_0^L w\frac{d}{dx}\left(AE\left(1+\frac{du}{dx}\right)\frac{du}{dx}\right)dx=\int_0^L wfdx$$

Applying boundary condition (Dirichlet condition of $u(0)=0$) and doing integration by parts:

$$-\int_0^L AE\left(1+\frac{du}{dx}\right)\frac{du}{dx}\frac{dw}{dx}dx=\int_0^L wfdx$$

Newton Raphson iterative procedure: FE example

The element stiffness matrix has the form

$$\mathbf{K}^e = \frac{AE_0}{h^2}(h + u_1 - u_2)\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}.$$

The tangent stiffness matrix is given by

$$\mathbf{T}^e = \mathbf{K}^e + \frac{\partial \mathbf{K}^e}{\partial \mathbf{u}}\,\mathbf{u}\,.$$

After going through the algebra, we get

$$\mathbf{T}^e = \frac{AE_0}{h^2}(h + 2u_1 - 2u_2)\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}.$$

For the two element mesh, the assembled global tangent stiffness matrix is

$$\mathbf{T} = \frac{AE_0}{h^2}\begin{bmatrix} h + 2(u_1 - u_2) & -h - 2(u_1 - u_2) & 0 \\ & 2h + 2(u_1 - u_3) & -h - 2(u_2 - u_3) \\ \text{Symm.} & & h + 2(u_2 - u_3) \end{bmatrix}$$

Newton Raphson iterative procedure: FE example

Let $A = 0.01$ m^2, $L = 2$ m, $E_0 = 100$ MPa, $R = 100$ kN. Since there are two elements, $h = 1$ m.

For the first Newton iteration, let the initial guess be

$$\mathbf{u}_0 = \begin{bmatrix} u_2^0 \\ u_3^0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Then, the first Newton iteration gives

$$\mathbf{K}_0 = 10^6 \begin{bmatrix} 2 & -1 \\ -1 & 1 \end{bmatrix}, \qquad \mathbf{T}_0 = 10^6 \begin{bmatrix} 2 & -1 \\ -1 & 1 \end{bmatrix}, \qquad \mathbf{u}_1 = \begin{bmatrix} 0.1 \\ 0.2 \end{bmatrix}.$$

The second Newton iteration gives

$$\mathbf{K}_1 = 10^6 \begin{bmatrix} 1.8 & -0.9 \\ -0.9 & 0.9 \end{bmatrix}, \qquad \mathbf{T}_1 = 10^6 \begin{bmatrix} 1.6 & -0.8 \\ -0.8 & 0.8 \end{bmatrix}, \qquad \mathbf{u}_2 = \begin{bmatrix} 0.1125 \\ 0.2250 \end{bmatrix}.$$

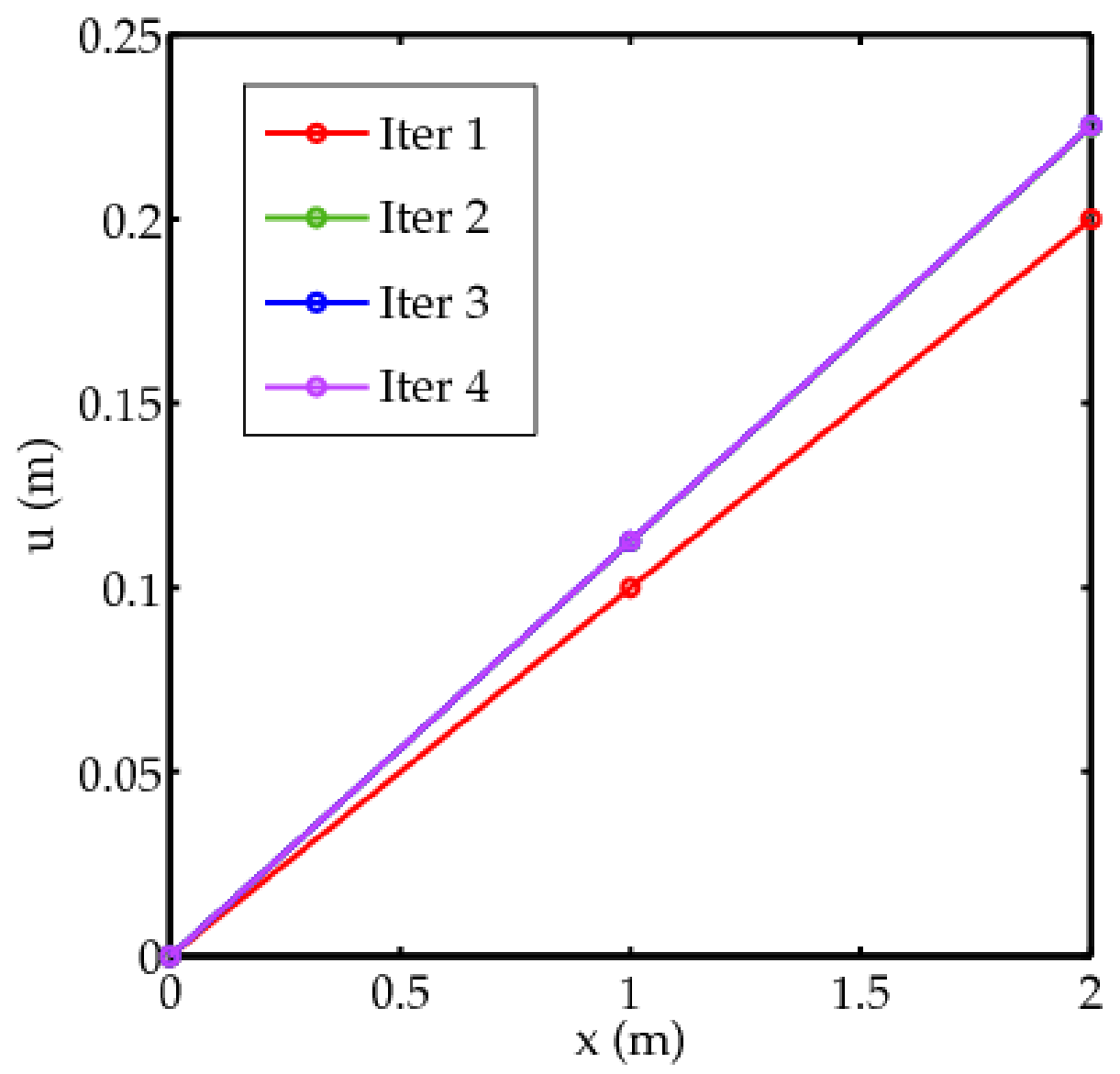
The third Newton iteration gives

$$\mathbf{K}_2 = 10^6 \begin{bmatrix} 1.775 & -0.8875 \\ -0.8875 & 0.8875 \end{bmatrix}, \qquad \mathbf{T}_2 = 10^6 \begin{bmatrix} 1.55 & -0.775 \\ -0.775 & 0.775 \end{bmatrix}, \qquad \mathbf{u}_3 = \begin{bmatrix} 0.1127 \\ 0.2254 \end{bmatrix}.$$

The fourth Newton iteration gives

$$\mathbf{K}_3 = 10^6 \begin{bmatrix} 1.7746 & -0.8873 \\ -0.8873 & 0.8873 \end{bmatrix}, \qquad \mathbf{T}_3 = 10^6 \begin{bmatrix} 1.5492 & -0.7746 \\ -0.7746 & 0.7746 \end{bmatrix}, \qquad \mathbf{u}_4 = \begin{bmatrix} 0.1127 \\ 0.2254 \end{bmatrix}.$$

Newton Raphson iterative procedure: FE example



Newton Raphson iterative procedure: FE example

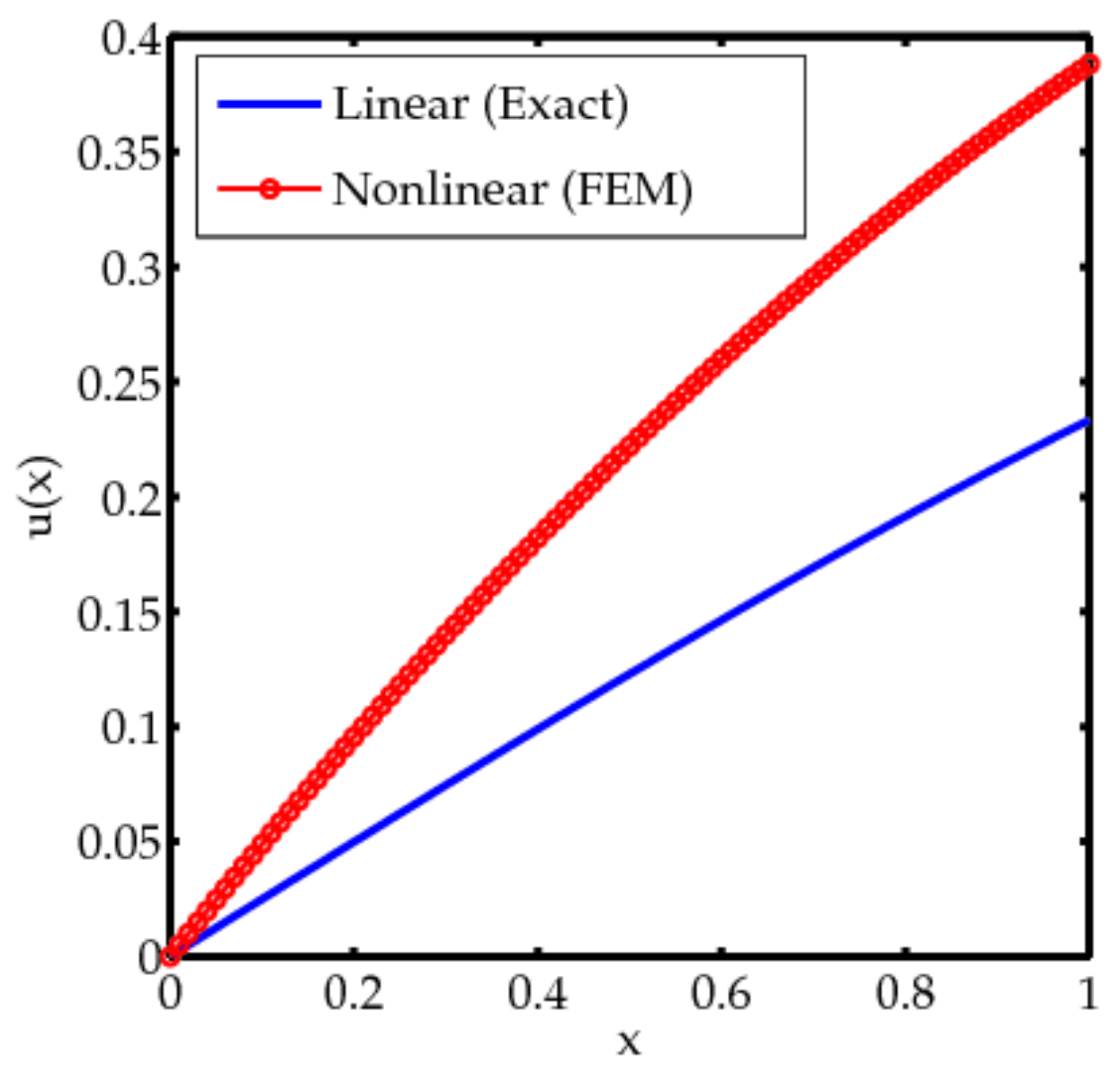


Fig. FEM displacement solution for nonlinear bar under distributed axial load. We have used 100 linear elements in these calculations.

Newton Raphson: Drawbacks

Newton Raphson: Drawbacks

Arc length method

System of the non-linear (in general) equations to solve:

$$F_{\text{int}} - F_{\text{ext}} = 0 \rightarrow F_{\text{int}} - \lambda q = 0$$

Suppose (u_0, λ_0) satisfy the system of equations and thus belongs to the 'equilibrium path'. Arc Length postulates a simultaneous variation in both the displacements Δu and the load vector coefficient $\Delta\lambda$.

Arc length method

The Arc length equation:

$$(\Delta \mathbf{u} + \delta \mathbf{u})^T \cdot (\Delta \mathbf{u} + \delta \mathbf{u}) + \psi^2 (\Delta \lambda + \delta \lambda)^2 (\mathbf{q}^T \cdot \mathbf{q}) = \Delta l^2$$

where ψ and Δl are user defined parameters. In a sense Δl defines how far to search for the next equilibrium point and it is analogous (but not directly equivalent) to the load increment $\Delta \lambda$ we used in Newton's method.

Arc length method

Arc length method: Drawbacks

A priori estimates

Functional norms

Before considering the error, we need to decide how it should be measured. The question is how to measure a function? The answer to this is to use function norms. A common norm is known as the L^2 norm. For a function on a domain Ω, the L^2 norm is defined as:

This norm provides a measure of 'how big' a function is. A measure of the difference between two functions u and v is provided by:

Functional norms

The L^2 norm measures the 'magnitude' of a function, but does reflect any details of the derivatives. Other norms are the H^1 norm:

We will use function norms to measure the difference between the exact solution and the finite element solution. For example, we can define a particular measure of the error e as:

How good is the FE solution?

Inevitably, the finite element solution is not exact. But how do we know that it bears any relation to the exact solution? If the mesh is refined, will the error reduce? We may say intuitively yes, but this is not very satisfactory. This is where a priori analysis helps. A priory error estimate:

$$\|u - u_h\|_s \leq Ch^\beta \|u\|_{k+1}$$

- s is the norm of interest,
- k is the polynomial order of the shape functions (complete polynomials),
- $\beta = min(k + 1 - s, 2(k + 1 - m))$,
- m is the highest order derivative appearing in the weak form
- C is an unknown constant that does not depend on h or the exact solution u.

What is of key interest is the exponent β.

Error in the temperature/displacement field

For a steady heat conduction or elasticity problem, for the error in the temperature/displacement field:

Therefore, for $k \geq 1$,

Error in the strain/flux field

For a steady heat conduction or elasticity problem, for the error in the strain/flux field we have:

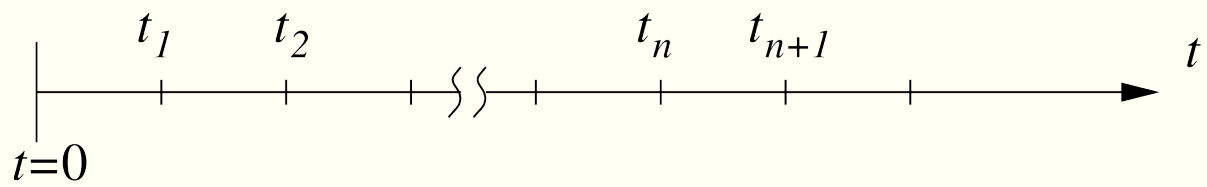
Therefore, for $k \geq 1$,

Posterior error estimation and adaptivity

A *posterior error* estimation involves quantification of the error after a simulation has been preformed.
If we have an a posterior error estimator for a particular problem, a simulation can been performed, the error estimated in various part of the domain and the mesh and finite element type can be adjusted in order to reduce the error to a prescribed value. This is known as *adaptivity*.
The two obvious strategies are known as:

Time-dependent problems

We wish to develop schemes to deal with semi-discrete finite element equations in a step-by-step fashion. Starting at time $t = 0$, we want to solve a problem up to time $t = t_{final}$. The time interval $(0, t_{final})$ will be 'chopped' into increments Δt:



What is needed is a strategy to advance the solution from t_n to t_{n+1}. There are many different techniques for this. We will consider some simple examples and the most commonly used methods.

Forward Euler

Given an equation of the form:

$$\dot{y} = f(y, t)$$

for which we know everything at time $t_n (y_n \mathrm{and} f(y_n, t_n))$, we want to advance to time t_{n+1} and find y_{n+1}. The simplest approach is the forward Euler method. It involves:

Forward Euler

Forward Euler

We can apply the forward Euler scheme to the nodal degrees of freedom a by replacing y with $\mathbf{a}$ and f with $\dot{\mathbf{a}}$,

We would like to eliminate the time derivative term $\dot{\mathbf{a}}$ from the semi-discrete heat equation, $\mathbf{M}\dot{\mathbf{a}} + \mathbf{K}\mathbf{a} = \mathbf{b}$, so that we are left with only $\mathbf{a}$. In the forward Euler expression, we have $\dot{\mathbf{a}}_n$, therefore we consider the heat equation at time step t_n,

Forward Euler

We have a system on linear equation that we can solve to find $\mathbf{a}_{n+1}$. Now the benefit of a lumped mass matrix can be seen. If $\mathbf{M}$ is lumped, there is no need to solve a system of equations and finding $\mathbf{a}_{n+1}$. is trivial.
System of equations for a lumped mass matrix:

Forward Euler

The time step at which a method becomes unstable is known at the *critical time step*. Conditionally stable methods are stable for:

- The critical time step often scales with eigenvalues of the finite element system, which in turn depend on the element type and size.
- As the element size is reduced, the critical time step reduces.
- The heat equation is a so-called stiff equation. This means that the restriction on the time step for explicit schemes is particularly severe. It scales according to

where h is a measure of the size of an element. This is a very strong restriction, and means that explicit schemes are not usually practical for the heat equation.

Second-order Central difference method

$$\dot{\mathbf{a}}_n = \frac{\mathbf{a}_{n+1} - \mathbf{a}_{n-1}}{2\Delta t}$$
$$\ddot{\mathbf{a}}_n = \frac{\mathbf{a}_{n-1} - 2\mathbf{a}_n + \mathbf{a}_{n+1}}{\Delta t^2}$$

This algorithm is explicit – it is possible to express $\mathbf{a}_{n+1}$ in terms of quantities at n and earlier times. This scheme is $O(\Delta t^2)$ accurate and it is conditionally stable. It is stable for:

$$\Delta t \leq \frac{2}{\omega_{h,max}}$$

where $\omega_{h,max}$ is the highest natural frequency. Therefore:

$$\Delta t \leq \frac{h}{c}$$

where c is the dilatation wave speed ($c = E/\rho$) and h is the length of an element, which is the Courant, Friedrichs and Lewy (CFL) stability condition required for a wave to travel through an element.

Backward Euler

Note now that f is evaluated at y_{n+1} and not at y_n. However, we do not know y_{n+1}; it is what we want to find.

Since the time derivative term is evaluated at t_{n+1}, we pose the semi-discrete heat equation at t_{n+1},

Backward Euler

Collecting the unknown terms on the left-hand side and the known terms on the right- hand side:

- Now, even if $\mathbf{M}$ is a diagonal matrix, we still need to solve a system of equations to find $\mathbf{a}_{n+1}$ because $\mathbf{K}$ is not diagonal.
- A system must be solved because the backward Euler method is an *implicit method*.
- In stepping from t_n to t_{n+1}, it relies on data at t_{n+1} which is what characterises implicit methods.
- The consequence is that a system of equations must be solved.

Summary of time-stepping schemes