Faculty of Civil Engineering and Geosciences
18 April 2023, 13:30–16:30
CIEM1110 Unit 1: Numerical modeling of construction materials

General instructions
All pages of your answer sheets must contain your name and your student number. Number each page of your answer sheets and indicate the total number of pages used. Write clearly and in English. Points for each question are in the left margin brackets.

This question sheet **must be returned** before leaving.

Questions

1. Suppose you are solving a stress equilibrium problem with one-dimensional 2-node linear elements. For a certain element e in the mesh, the nodal internal force vector can be computed with

$$\mathbf{f}_e^{\text{int}} = \int_{\Omega_e} \mathbf{B}^{\mathrm{T}} \sigma \, \mathrm{d}\Omega$$

where you can assume the $\mathbf{B}$ matrix for this specific element is given by:

$$\mathbf{B} = \begin{bmatrix} -\frac{1}{2} & \frac{1}{2} \end{bmatrix}$$

and the material model follows a nonlinear law:

$$\sigma = E\varepsilon + \phi\left(\varepsilon\right)$$

with E being the Young's modulus and ϕ a polynomial correction term given by:

$$\phi\left(\varepsilon\right) = a\varepsilon^2 + b\varepsilon + c$$

Since this is a nonlinear finite element problem, the Newton-Raphson method is employed, with tangent stiffness matrix computed as:

$$\mathbf{K} = \frac{\partial \mathbf{f}^{\text{int}}}{\partial \mathbf{a}}$$

and strains are computed as $\varepsilon = \mathbf{B}\mathbf{a}_e$.

[8%] (a) Derive the material tangent stiffness D for this material model as a function of strain and the given parameters;

Solution:

$$D = \frac{\partial \sigma}{\partial \varepsilon} = E + 2a\varepsilon + b$$

[12%] (b) Assume that at a certain iteration of the simulation the nodal displacements of element e are $\mathbf{a}_e = \begin{bmatrix} 1 & 2 \end{bmatrix}^{\mathrm{T}}$. For this same iteration, compute the consistent element tangent $\mathbf{K}_e$ for this element (assume $E = 1$, $a = 2$, $b = 1$ and $c = 0.5$)

Solution: First compute strain:

$$\varepsilon = \mathbf{B}\mathbf{a}_e = 0.5$$

Then D with definition above:

$$D = 4$$

Finally $\mathbf{K}_e$, using $\frac{\partial \sigma}{\partial \mathbf{a}_e} = D\mathbf{B}$, and trivial integration for the element with constant $\mathbf{B}$-matrix:

$$\mathbf{K}_e = \int_{\Omega_e} \mathbf{B}^T D\mathbf{B}\,\mathrm{d}\Omega = \Delta x \mathbf{B}^T D\mathbf{B} = \begin{bmatrix} 2 & -2 \\ -2 & 2 \end{bmatrix}$$

with $\Delta x = 2$

2. Consider the transient (time-dependent) version of the equilibrium PDE, with an inertia term being added to the right-hand side of the PDE for elastostatics:

$$\nabla \cdot \boldsymbol{\sigma} + \mathbf{b} = \rho \ddot{\mathbf{u}}$$

where $\mathbf{u}$ and $\ddot{\mathbf{u}}$ are the displacement and acceleration fields (three components in 3D) for a domain Ω, ρ is a density parameter and linear-elasticity is assumed:

$$\boldsymbol{\sigma} = \mathcal{D} : \nabla^{\mathrm{s}} \mathbf{u}$$

where $\mathcal{D}$ is the elastic stiffness tensor, with the same relationship in matrix-vector (Voigt) notation being given by:

$$\boldsymbol{\sigma} = \mathbf{D}\boldsymbol{\varepsilon}$$

with $\mathbf{D}$ being the elastic stiffness matrix (6×6 in 3D). This PDE is solved for boundary conditions

$$\mathbf{u} = \mathbf{g} \text{ at } \Gamma_g, \quad \boldsymbol{\sigma}\mathbf{n} = \mathbf{t} \text{ at } \Gamma_h$$

which are joined by a new set of *initial* conditions:

$$\mathbf{u}(t=0) = \mathbf{u}_0, \quad \dot{\mathbf{u}}(t=0) = \dot{\mathbf{u}}_0$$

with $\mathbf{n}$ being the vector normal to the surface Γ_h. A FEM treatment for this PDE is analogous to the one employed for the *Diffusion* equation.

[20%] (a) Derive the weak form and the semi-discretized form for this PDE, arriving at an expression of the form

$$\mathbf{M}\ddot{\mathbf{a}}_n + \mathbf{K}\mathbf{a}_n = \mathbf{f}_n$$

for an arbitrary time step n and giving expressions for the integrals $\mathbf{M}$, $\mathbf{K}$ and $\mathbf{f}$. Assume interpolations are done with an arbitrary shape function matrix $\mathbf{N}$ with a corresponding $\mathbf{B}$ matrix whose components you do not need to expand. When performing integration by parts, you can make use of:

$$\int_\Omega \mathbf{a} \cdot (\nabla \cdot \mathbf{B})\,\mathrm{d}\Omega = -\int_\Omega \nabla \mathbf{a} : \mathbf{B}\,\mathrm{d}\Omega + \int_\Gamma \mathbf{a} \cdot \mathbf{B}\mathbf{n}\,\mathrm{d}\Gamma$$

where $\mathbf{a}$ is an arbitrary vector, $\mathbf{B}$ an arbitrary second-order tensor and $\mathbf{n}$ the normal to surface Γ.

Solution: See interactive textbook page on "Semi-discrete form for elastodynamics"

[10%] (b) A popular approach to solve this PDE in time is to employ a *Central Difference* scheme that approximates the acceleration at time step n by:

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

where $\mathbf{a}_{n-1}$ and $\mathbf{a}_n$ are assumed to be known. Derive a linear system of equations for time step $n+1$ of the form

$$\hat{\mathbf{M}}\mathbf{a}_{n+1} = \hat{\mathbf{f}}_n$$

and arrive at expressions for $\hat{\mathbf{M}}$ and $\hat{\mathbf{f}}$ as a function of $\mathbf{K}$, $\mathbf{M}$, $\mathbf{f}$ and other known quantities.

Solution: See interactive textbook page on "Time stepping schemes for elastodynamics"

3. Consider the following code snippet from `pyJive`:

```
1
2   # ...
3
4         model.take_action(act.GETMATRIX0, params, globdat)
5
6         model.take_action(act.GETEXTFORCE, params, globdat)
7
8         model.take_action(act.GETCONSTRAINTS, params, globdat)
9
10        Kc, fc = c.constrain(K, f)
11
12        smat = sparse.csr_matrix(Kc)
13        u = linalg.spsolve(smat, fc)
14
15        globdat[gn.STATE0] = u
16
17  # ...
18
19
```

and the following table with models and modules:

Model/module	Functionality
`BarModel`	1D equilibrium elements in extension
`PoissonModel`	Poisson/Diffusion with isoparametric elements
`SolidModel`	Continuum stress equilibrium with isoparametric elements
`DirichletModel`	Handles Dirichlet-type boundary conditions
`NeumannModel`	Handles Neumann-type boundary conditions
`SolverModule`	Solves linear quasi-static FE systems
`NonlinModule`	Newton-Raphson solver in load/displacement control
`ArclenModule`	Newton-Raphson solver with arc-length control
`TrapezoidalModule`	Trapezoidal time stepper for diffusion problems

[4%] (a) From the list of models and modules above, pick the one that you believe this snippet comes from. Provide a short motivation for your choice;

Solution: `SolverModule`, it is a code that asks for assembly of a matrix and then solves the resulting linear system of equations. No handling of history, internal force, or incrementing of the solution as would be required in the other modules listed.

[4%] (b) Imagine Line 10 of the code above was omitted and we directly used `K` and `f` for the rest of the function. What would the consequence be for the obtained solution? Provide a physical explanation for your answer.

Solution: The stiffness matrix would be singular and solving the sysmtem of equations (line 13) would result in an error or in unrealistically high values in u.