



Division of Engineering
Brown University

EN234: Computational methods in Structural and Solid Mechanics

Homework 5: Special elements, Time Dependent Problems Due Thursday Oct 30, 2013

1. Modify the 2D element you coded in FEACHEAP last week to function as a B-bar element. You can follow the MATLAB code provided on the homework page to see the extra lines of code that correct the stiffness to avoid volumetric locking. This should require quite minor edits to your existing code (if you prefer, you could set up a 3D code). Test the modified element on some suitable boundary value problem that will illustrate that the modification successfully avoids volumetric locking.
2. **Optional:** Write an augmented Lagrangean hybrid element for FEACHEAP that will solve near incompressible linear elasticity problems. You can write a 2D or 3D code, as you prefer (or both, if you really have nothing better to do). You will need to use the following general approach. There is a sample MATLAB code online that you can use as a guide. The MATLAB code retains the pressure degrees of freedom in the global stiffness matrix, however. In FEACHEAP it is better to eliminate the pressure degrees of freedom inside the subroutine that calculates the element stiffness rather than create pressure nodes in the mesh (because you would have to write special routines to print this type of mesh for TECPLOT to read – if you really want a challenge you could try this – you would have to use the user-defined print subroutine in usprn.f90)

- The mesh in the input file should look exactly the same as for regular linear elasticity elements
- The element stiffness should be constructed as follows. First, assemble the matrices $\mathbf{K}, \mathbf{Q}, \mathbf{\Pi}$ as follows

$$k_{aibk} = \int_{V_e} C_{ijkl} \frac{\partial N^a(\mathbf{x})}{\partial x_j} \frac{\partial N^b(\mathbf{x})}{\partial x_l} - K \frac{\partial N^a(\mathbf{x})}{\partial x_i} \frac{\partial N^b(\mathbf{x})}{\partial x_k} dV \quad q_{aib} = \int_V \frac{\partial N^a(\mathbf{x})}{\partial x_i} M^b(\mathbf{x}) dV$$

$$\pi_{ab} = \int_{V_e} \frac{1}{K} M^a(\mathbf{x}) M^b(\mathbf{x}) dV$$

where $K = C_{ppqq} / 9$ is the bulk modulus (this assumes 3D – in 2D the expression is $K = C_{\alpha\alpha\beta\beta} / 4$).

Here, N^a are the usual element shape functions, and M^a are interpolation functions for the pressure. These should be one order lower than the displacement interpolations – if you are using linear displacement interpolation then there is only one pressure node, and M is simply equal to 1. If you are using quadratic interpolation for displacements then M should be linear shape functions. The integrals defining k_{iakb} may be evaluated using the full integration scheme. The remaining integrals can be evaluated using one order integration lower (but the method should still work if you use the same integration scheme for all integrals – you can try this in the matlab code).

- Finally, compute the element stiffness as $\mathbf{K}^{el} = \mathbf{K} - \mathbf{Q}\mathbf{\Pi}^{-1}\mathbf{Q}^T$ (this step is not done in the matlab code)
- Test your code on some interesting problem!

3. Write an element for FEACHEAP that will solve the transient diffusion equation. Start with the governing equations for concentration and flux

$$\frac{\partial c}{\partial t} + \frac{\partial j_i}{\partial x_i} = 0 \quad j_i = -D \frac{\partial c}{\partial x_i} \quad j_i n_i = j^* \quad \text{on } S_2 \quad c = c^* \quad \text{on } S_1$$

Taking c as the unknown variable, the weak form can be expressed as

$$\int_V \frac{\partial c}{\partial t} \delta c + \int_V D \frac{\partial c}{\partial x_i} \frac{\partial \delta c}{\partial x_i} + \int_{S_2} j^* \delta c dA = 0$$

We can regard this equation as a first order differential equation of the form

$$\frac{dc}{dt} = f(c)$$

and evaluate the time integral using a generalized Euler scheme

$$\frac{\Delta c}{\Delta t} = \theta f(c) + (1 - \theta) f(c + \Delta c)$$

where $0 < \theta < 1$ is an adjustable parameter (you can recover both a forward-Euler and backward-Euler scheme, as well as a mid-point trapezoidal scheme with an appropriate choice). If we introduce the usual finite element interpolation for c the finite element equations for an increment in concentration Δc^a can be expressed in the form

$$K_{ab} \Delta c^b = R_a + F_a$$

where **K**, **R**, and **F** are obtained by summing the element stiffness matrix, residual and external force vectors

$$k_{ab}^{el} = \int_{V_e} \left(\frac{1}{\Delta t} N^a N^b + D(1 - \theta) \frac{\partial N^a}{\partial x_i} \frac{\partial N^b}{\partial x_i} \right) dV \quad r_a^{el} = - \int_{V_e} D \frac{\partial N^a}{\partial x_i} \frac{\partial N^b}{\partial x_i} c^b dV \quad f_a^{el} = - \int_{V_e} j^* N^a dA$$

Implement these expressions in FEACHEAP, as follows:

- It is best to start with the new_user_element_stub.f90 code for this problem rather than copy one of the linear elasticity elements.
- Note that the nodes in your mesh will contain only 1 degree of freedom – make sure to specify this in the input file.
- The variables DULOC and UTLOC in the element stiffness subroutine will contain the concentration increment Δc (which is not required in your calculations) and c at the nodes. You will need to use c to compute the r_a^{el} vector.
- The time variable and time increment variable are stored in the Globals module and called TIME and DTIME, respectively. To access these you need to put a `use Globals` statement at the top of your code (this is already in the element template)
- The values of θ, D are properties for your element (define them in the input file)
- The element stiffness can be evaluated using the usual procedure – but note that you have to use a higher order integration scheme because the shape functions are not differentiated.
- The components r_a^{el} should be stored in the vector called resid(), and can be evaluated using the standard integration scheme.

- FEACHEAP will do the book-keeping necessary to model the evolution of concentration with time – you don't need to add any special code to deal with this.
- Start by solving a problem with prescribed concentrations on the boundary, so you don't need to code the element force vector f . You can add this later (of course the pre-coded 'DLOAD' subroutines won't work for this problem).
- If you like, you could use the 'new_user_element_elstat' subroutine to project the flux vector components (or the magnitude of the flux) as nodal state variables, but you can comment out the PROJECT STATE command in the input file to avoid having to do this.
- FEACHEAP should have no problem printing the nodal concentrations and mesh to a tecplot readable file in the usual way (the code will detect that nodes have only 1 DOF and will not attempt to plot a deformed mesh).
- You should use a LINEAR solution procedure in your input file. Note that even though you compute a residual for this element the CHECK STIFFNESS code will not work (it works only when an element is coded for NONLINEAR solution procedures, but we are still working on linear problems...).
- Test your code by simulating a 1D problem that has an exact solution. Start by running the code with $\theta=0$ (this is the most stable solution – in fact with $\Delta t \rightarrow \infty$ the steady state solution is computed directly) and a small time-step. Then explore other values of θ , the influence of time step size, etc, on the accuracy and stability of the solution.
- **OPTIONAL** Once the code works with prescribed concentrations, you could code a 'new_user_element_dload' subroutine to apply the prescribed flux boundary condition.