Division of Engineering
Brown University

EN234: Computational methods in Structural and Solid Mechanics

Homework 6: Dynamic elasticity
Due Wednesday Nov 6, 2013

1. Consider a harmonic oscillator with equation of motion

$$\ddot{u} + \omega_n^2 u = 0$$

Suppose that the system is set in motion at time *t=0* with initial conditions $u = 0$, $\dot{u} = v_0$. The EOM is integrated using the Newmark time integration scheme with time-step Δt. Let u_1, v_1 , u_2, v_2 denote the values of $u(t), \dot{u}(t)$ at time $\Delta t, 2\Delta t$, etc. Calculate an analytical expression for u_2, v_2 in terms of $\tau = \omega_n \Delta t$ (use a symbolic manipulation program – you don't need to hand in the lengthy expression) and hence calculate an analytical solution to the normalized energy change caused by the integrator at the second time step (you can go to higher times if you wish – again no need to hand in the solutions)

$$\Delta \tilde{E}_2 = \frac{\omega_n^2 u_2^2 + \dot{u}_2^2}{v_0^2} - 1$$

Expand the energy change as a Taylor series in τ (you will need to go up to at least order τ^4 . Hence (by examining the coefficients of the terms in the series), determine (i) the value of β_1 that minimizes the energy change; and (ii) For this value of β_1, the value of β_2 for which the energy change is zero or negative (for $\tau < 1$). What combination of the parameters minimizes energy loss?

2. Write a MATLAB code that will use the Newmark method to integrate the equation of motion for the harmonic oscillator. Run the code with selected values of β that will verify the predictions of the preceding problem.

3. Implement and test a 2D dynamic element in FEACHEAP (you might actually have done the coding for this already – all you need to do is to test the element). The mass matrices are already computed in the code (you can find them in the file called usrelem.f90), so all you need to do is to compute the internal force. In class we defined the internal force as

$$\left[\int_V C_{ijkl} \frac{\partial N^a}{\partial x_j} \frac{\partial N^b}{\partial x_l} \right] u_k^b = K_{aibk} u_k^b$$

so the governing equations for linear elastic solids had the form

$$M_{ab} \ddot{u}_i^b + K_{aibk} u_k^b = F_i^a$$

FEACHEAP is coded in a simpler way, partly to solve problems with more complicated stress-strain behavior than just linear elasticity, and also to save having to compute, and store, the full stiffness matrix. The governing equations are assumed to have the form

$$M_{ab} \ddot{u}_i^b = F_i^a + R_i^a$$

where

$$R_i^a = -\int_V \sigma_{ij} \frac{\partial N^a}{\partial x_j}$$

For the particular case of a linear elastic solid, we get the usual equation by noting that

$$\sigma_{ij} = C_{ijkl} \frac{\partial N^b}{\partial x_l} u_k^b$$

In FEACHEAP a subroutine called elforce(….) is used to compute the element contribution to R_i^a. It is stored in usrelem.f90. This subroutine just calls user-defined subroutines for whatever element types or material types you might be interested in. There is an example user-subroutine called elforce_linelast_3Dbasic (in the file called el_linelast_3dbasic) for a 3D linear elastic material. All you need to do is to create a corresponding subroutine for your 2D linear elastic element.

For dynamic problems it is sometimes useful to be able to define initial displacements or velocities. The code will let you do this: here's an example input file that launches two elements with a prescribed initial velocity.

```
%
%             Demonstration input file for simple general purpose FEA code FEACHEAP
%
%%%%%%%%%%%%%%%%%%%%%% MESH DEFINITION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
     MESH
%    The NODE command defines properties of the nodes.
%    The parameters are # of coords, # of DOF, and an optional integer identifier
       NODES, 3, 3, 1
%    Enter x,y,z coords of nodes.   The node number is optional, and is ignored in
%    the code.
         COORDINATES
         1,  0.d0, 0.d0, 0.d0
         2,  1.d0, 0.d0, 0.d0
         3,  1.d0, 1.d0, 0.d0
         4,  0.d0, 1.d0, 0.d0
         5,  0.d0, 0.d0, 1.d0
         6,  1.d0, 0.d0, 1.d0
         7,  1.d0, 1.d0, 1.d0
         8,  0.d0, 1.d0, 1.d0
         9,  2.d0, 0.d0, 0.d0
        10,  2.d0, 1.d0, 0.d0
        11,   2.d0, 0.d0, 1.d0
        12,   2.d0, 1.d0, 1.d0
%
     INITIAL DOF, VELOCITY
        ALL NODES, 1.d0, 0.d0, 0.d0
%
%     The ELEMENT command defines properties of elements
%     The parameters are no. nodes on the element, no. properties, total no.
%     state variables, integer identifier
%     2D solid elements should be numbered between 1 and 100; 3D solid elements
%    should be numbered between
%     1000 and 2000.
      ELEMENT, 8, 3, 0, 1001
%       Define element properties - the values are passed to user subroutine elstif
%       in the order they are listed here
%       For the example provided, the params are Youngs Modulus, Poissons ratio and
%       thermal expansion coeft
```

```
        PROPERTIES
         100.d0, 0.3d0, 0.d0
      DENSITY, 10.d0
%     Define element connectivity
%     The element number (first number in the list) is optional, and is ignored in the
%     code
        CONNECTIVITY
    1,     1, 2, 3, 4, 5, 6, 7, 8
    2,     2, 9, 10, 3, 6, 11, 12, 7
%     The ELEMENT keyword can be repeated here to define another set of elements with
different properties
%%%%%%%%%%%%%%%%%%%%%%%%%% BOUNDARY CONDITIONS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%     The BOUNDARY conditions key starts definition of BCs
      BOUNDARY CONDITIONS
%       The HISTORY key defines a time history that can be applied to DOFs or distributed
loads
        HISTORY, dof_history
          0.d0, 0.d0
         10.d0, 1.d0
        HISTORY, dload_history
          0.d0, 2.d0
         10.d0, 2.d0
%       The NODESET key defines a list of nodes
        NODESET, base
           1, 2, 3, 4, 9, 10
        NODESET, left
           1, 4, 5, 8
        NODESET, right
           9, 10, 12, 11
        NODESET, side
           1, 2, 5, 6, 11, 9
%      The ELEMENTSET key defines a list of elements
       ELEMENTSET, end_element
           2
%       The DEGREE OF FREEDOM key assigns values to nodal DOFs
%       The syntax is node set name, DOF number, and either a value or a history name.
%
%        DEGREES OF FREEDOM
%           base, 3, 0.d0
%           side, 2, 0.d0
%           left, 1, 0.d0
%           right, 1, dof_history
%        END DEGREES OF FREEDOM
%       The DISTRIBUTED LOAD key sets up prescribed tractions on element faces
%       The syntax is one of the following options:
%           element set, face #, VALUE, tx,ty,(tz)
%         (applies constant pressure to element face in direction DOF)
%           element set, face #, history name, nx,ny,(nz)
%         (time dependent pressure to element face in direction (nx,ny,nz))
%           element set, face #, NORMAL, history name
%         (applies time dependent pressure normal to element face)
%           element set, face #, SUBROUTINE, list of parameters
%        DISTRIBUTED LOADS
%           end_element, 4, Value, 10.d0,0.d0,0.d0
%        END DISTRIBUTED LOADS
%%%%%%%%%%%%%%%%%% Mesh printing, error checking %%%%%%%%%%%%%%%%%%%%%
%  Print the initial mesh to a file named initial_mesh.dat
   PRINT, initial_mesh.dat
%      The ALL NODES option forces quadratic elements to be subdivided
%      for TECPLOT plotting purposes.
       MESH, ALL NODES
%%%%%%%%%%%%%%%%%%%% Analysis %%%%%%%%%%%%%%%%%%%%%%%%%%%
%   The EXPLICIT DYNAMIC STEP key initializes a dynamic load step
```

```
     EXPLICIT DYNAMIC STEP
%         The TIME STEP key defines values of parameters controlling time stepping.
%         The parameters are time step, no. steps, no. steps between state prints, no.
steps between user prints
%         All parameters are required
           TIME STEP, 0.01, 200,10,200
%         Project element state to nodes (for contour plotting, etc).
%    The integer specifies # of states to project.
%    You can also use PROJECT STATE, # states, LUMPED PROJECTION MATRIX - this does the
%   state projection
%   using a lumped approximation to the projection matrix.  This is much faster,
%    and uses far
%         less memory, but slightly less accurate.
          PROJECT STATE, 6
%       This prints the DOF values and projected nodal state for all solid elements
%      to a tecplot readable file
%       Nodal variables are printed as
%          X, Y, (Z), Ux, Uy, (Uz), Projected states
          PRINT
            MESH, contourplots.dat
%       This activates the user subroutine controlled printing
%         PRINT
%           USER
%           List file names to contain the output.
%           FILES
%             filename1.dat
%             filename2.dat
%           PARAMETERS
%             list of parameters to be passed to the subroutine
        END EXPLICIT DYNAMIC STEP
STOP
```