ME538 Fall 2020 – Finite Element Analysis
Due: Wednesday, November 11, 2019 by 12 PM
Programming Assignment #2: Linear dynamic 1D matlab finite element code

The assignment:
Given the shell of a 1D finite element matlab code, implement the internal force, mass matrix, time integration algorithm and impose boundary conditions, all for a 2-node linear element. The element integrals should be evaluated by first constructing the parent coordinates, then mapping those to the actual coordinates. Also calculate a lumped mass matrix for usage in some of the problems below.

Code given:
The shell of the matlab code will be given in the folder 1D_Dyn_FEM_Heaviside. The executable file is dynamic1D.m.

The problem:
Once the programming has been performed, test the code on a simple 1D problem consisting of 20 elements between x=0 and x=100. Fix the node at x=100, and apply an external force at x=0 of f=1000 (i.e. a compressive force)– this force should be applied at all times. Track the displacement and velocity versus time for a node at x=50. What do you notice? Is the displacement or velocity smoother? Why? What interesting behavior do you notice in the velocity plots?

Use more elements, but make sure you increase the simulation time proportionally so that the wave travels the same distance back and forth across the 1D bar each time. What happens to the displacements and velocities at x=50 when you use 50 elements, or 100, or 150 elements? For the 20 and 50 element cases, use time steps that are 10 times smaller than the initial time step – what differences do you observe now? (Remember that you have to run the simulation 10 times longer when you reduce the time step to get the same total simulation time). Do you think it is better to reduce the time step or use more elements?

Implement a lumped mass matrix option. Is there any difference to the solution for the displacements and velocities for the 20 and 50 element cases? If so, what, and why?

The details:
1. Implement the element mass matrix subroutine femass.m. Calculate the internal force due to the stiffness.
2. Within the main code, 1Ddynamic.m, assemble the element mass matrix Ma into the global mass matrix M.
3. Calculate the total force vector f (i.e. f = fexternal – finternal)
4. Write the time integration loop in dynamic1D.m
5. Impose boundary conditions during the time integration subroutine
6. Impose the loading of fexternal=1000 on the node at x=0 while fixing the node at x=100.

7. Implement a lumped mass matrix, compare solutions for the questions listed above to the consistent mass matrix
8. Compare the stable time step you can get with a lumped vs. consistent mass matrix for the 20 element case. For the lumped mass matrix case, use a critical time step of 1.0XX times the critical one, where the XX is the last 2 digits of your social security number. What happens when the time step is larger than the critical value? Is the ratio of the critical time step for the lumped vs. consistent mass matrices the same as theoretically derived in class? What happens if, for the lumped mass matrix, you use a time step that is exactly the critical value?
9. Which solution is smoother, the displacement or velocity? Why?

What to turn in:

1. The code written to calculate the element mass matrix (femass.m)
2. The code written for the time integration
3. Representative plots of displacement vs. time and velocity vs. time for the node at x=50 for the three different meshes and time steps suggested above. (Note: representative means you do not have to turn in graphs for every simulation case you run).
4. Comparisons between the results obtained using lumped and consistent mass matrices for the problems mentioned above.
5. A brief discussion summarizing what you learned, and answer the questions posed above.

Grading Criteria (100 points total):

1. Mass matrix code: 30 points
2. Time integration code: 40 points
3. Writeup, analysis and discussion: 30 points