ME 471/AE 420/CSE 451: Programming Assignment 1
Spring 2017

Due: Friday, February 10, 2017 at 11:59pm (subversion)

General Instructions for Programming Assignments

To complete your submission, follow the steps below:

1. Go to your working directory (for example, cd ME471-Programming-Assignments)
2. Download assignment (svn checkout https://subversion.ews.illinois.edu/svn/sp17-me471/your_netid/01-FEA-Spring)
3. Write your FEA code
4. In case you create new files (.m, .cpp, .h), you will need to use svn add (svn add schedule files and directories in your working copy for addition to the repository.)
5. Upload the changes (svn commit -m "COMMIT_MESSAGE")

Before you commit your work, make sure all the files are following these guidelines:

1. Matlab users:
   (a) Do not change the name of the main file (for example, "MainFile.m"). The grading script will execute this file.
   (b) Do not modify the following lines in the main file:
   // ==========
   // DO NOT MODIFY THE LINE BELOW!! //Autograding script will search for this variable definition
   filename = 'input.dat';
   // ==========
   Of course, you are free to modify the name of the input file when working on your local machine, but make sure the filename variable is set to 'input.dat' before you commit.
   (c) Do not delete the contents of the C-Code folder (mainly the Makefile file)
2. For C++ users:
   (a) Do not modify the variable EXENAME inside the Makefile. The grading script will execute the file defined by EXENAME.
   (b) Do not modify the following lines in the main file:
   // ==========
   // DO NOT MODIFY THE LINE BELOW!! //Autograding script will search for this variable definition
   string filename = "input.dat";
   // ==========
   Of course, you are free to modify the name of the input file when working on your local machine, but make sure the filename variable is set to 'input.dat' before you commit.

3. Do not modify the "PrintOutput" function. This will ensure your assignment will be graded properly.

It is good practice to commit regularly and frequently. For example, commit when you are done writing a function. This allows both simpler commit messages and greater confidence in the repository.

Write your own Spring FEA code

Write a general finite element program to evaluate the response of a spring system. Your program should read a file named "input.dat" that is structured in the same way as the lecture example. You don't need to modify the "ReadInput" function. The program should also use the "PrintOutput" function to generate all the relevant output files. Submit your complete code (svn commit) following the guidelines explained above.

Tips for debugging your code

After downloading (svn checkout) your assignment, you will find two different input files in your directory: "input.dat" and "input-new.dat". The file "input.dat" is the same one used as an example in lecture 5, and refers to the configuration depicted in Fig.1. When writing your FE functions, use the file "input.dat", and check for intermediate steps along the way (for example, do you get the correct values in the EQ_NUM vector after the function "InitializeEquation"?). Don't start writing the next function until the previous one is giving you the expected results. After completing your code, you should be able to match all the results in Fig.2.

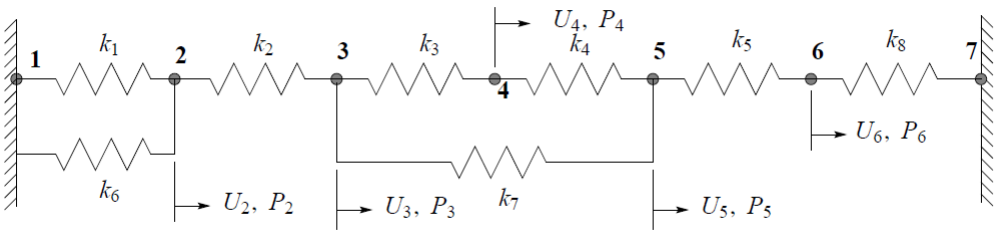

Figure 1: Spring example solved in lecture

Now rename the file "input.dat" as "input-lecture.dat" and "input-new.dat" as "input.dat"[1]. This new file has different input values for the same spring system in Fig.1. You should be able to match all the results from Fig.3 **without having to modify your code!**.

We will be checking your code using a different configuration for the spring system. To make sure your code will run successfully for any configuration, you should check your code for the spring system illustrated in Fig.4. For that, you will need to create your own "input.dat" file. You should be able to match all the results from Fig.5 **without having to modify your code!**.

[1]You can accomplish the same thing by changing the variable filename inside the main function - filename = "input-new.dat". However, you must change the variable back to filename = "input.dat" before you commit!

Input.dat

```
7   8   3   2
1   2   2
2   3   3
3   4   3
4   5   3
5   6   4
1   2   4
3   5   5
6   7   5
3   10
4   20
5   30
1   0
7   0
```

Variables that will be printed to different output files:

$$EQ\_NUM = [\,-1 \;\; 1 \;\; 2 \;\; 3 \;\; 4 \;\; 5 \;\; -2\,]$$

$$KPP = \begin{bmatrix} 6. & 0. \\ 0. & 5. \end{bmatrix}$$

$$KPF = \begin{bmatrix} -6. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & -5. \end{bmatrix}$$

$$KFF = \begin{bmatrix} 9. & -3. & 0. & 0. & 0. \\ -3. & 11. & -3. & -5. & 0. \\ 0. & -3. & 6. & -3. & 0. \\ 0. & -5. & -3. & 12. & -4. \\ 0. & 0. & 0. & -4. & 9. \end{bmatrix}$$

$$KFP = \begin{bmatrix} -6. & 0. \\ 0. & 0. \\ 0. & 0. \\ 0. & 0. \\ 0. & -5. \end{bmatrix}$$

$$UF = \begin{Bmatrix} 4.54 \\ 13.62 \\ 17.5 \\ 14.7 \\ 6.55 \end{Bmatrix}$$

$$PP = \begin{Bmatrix} -27.2 \\ -32.8 \end{Bmatrix}$$

$$UUR = \begin{bmatrix} 0. \\ 4.54 \\ 13.62 \\ 17.5 \\ 14.7 \\ 6.55 \\ 0. \end{bmatrix}$$

$$PUR = \begin{bmatrix} -27.2 \\ 0. \\ 10. \\ 20. \\ 30. \\ 0. \\ -32.8 \end{bmatrix}$$

Figure 2: Input data and results for the spring example in Fig.1 when $P_2 = 0$, $P_3 = 10$, $P_4 = 20$, $P_5 = 30$, $P_6 = 0$, $k_1 = 2$, $k_2 = 3$, $k_3 = 3$, $k_4 = 3$, $k_5 = 4$, $k_6 = 4$, $k_7 = 5$, $k_8 = 5$.

Input-new.dat

```
7   8   3   2
1   2   10
2   3   15
3   4   5
4   5   20
5   6   30
1   2   10
3   5   5
6   7   40
2   10
5   -5
6   20
1   0
7   0
```

Variables that will be printed to different output files:

$$EQ\_NUM = [-1 \;\; 1 \;\; 2 \;\; 3 \;\; 4 \;\; 5 \;\; -2]$$

$$\boldsymbol{K}^{pp} = \begin{bmatrix} 20 & 0 \\ 0 & 40 \end{bmatrix}$$

$$\boldsymbol{K}^{pf} = [\boldsymbol{K}^{fp}]^T = \begin{bmatrix} -20 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -40 \end{bmatrix}$$

$$\boldsymbol{K}^{ff} = \begin{bmatrix} 35 & -15 & 0 & 0 & 0 \\ -15 & 25 & -5 & -5 & 0 \\ 0 & -5 & 25 & -20 & 0 \\ 0 & -5 & -20 & 55 & -30 \\ 0 & 0 & 0 & -30 & 70 \end{bmatrix}$$

$$UF = \begin{Bmatrix} 0.449 \\ 0.381 \\ 0.291 \\ 0.268 \\ 0.401 \end{Bmatrix}$$

$$PP = \begin{Bmatrix} -8.98 \\ -16.02 \end{Bmatrix}$$

$$UUR = \begin{Bmatrix} 0 \\ 0.449 \\ 0.381 \\ 0.291 \\ 0.268 \\ 0.401 \\ 0 \end{Bmatrix}$$

$$PUR = \begin{Bmatrix} -8.98 \\ 10 \\ 0 \\ 0 \\ -5 \\ 20 \\ -16.02 \end{Bmatrix}$$

Figure 3: Input data and results for the spring example in Fig.1 when $P_2 = 10$, $P_3 = 0$, $P_4 = 0$, $P_5 = -5$, $P_6 = 20$, $k_1 = 10$, $k_2 = 15$, $k_3 = 5$, $k_4 = 20$, $k_5 = 30$, $k_6 = 10$, $k_7 = 5$, $k_8 = 40$.

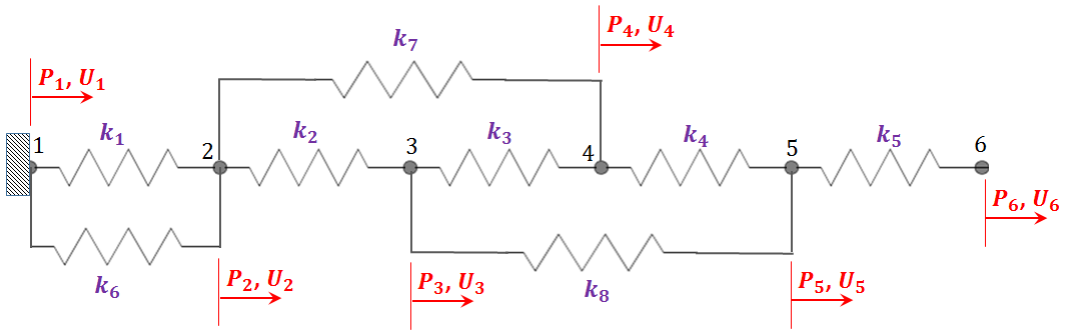

Figure 4: Another spring example for debugging. The prescribed boundary conditions are $U_1 = 0$, $P_2 = 15$, $P_3 = 0$, $P_4 = 0$, $P_5 = 25$, $U_6 = 5$ and the spring stiffnesses are given by $k_1 = 10$, $k_2 = 15$, $k_3 = 5$, $k_4 = 20$, $k_5 = 30$, $k_6 = 5$, $k_7 = 40$, $k_8 = 30$.

Variables that will be printed to different output files:

$$EQ\_NUM = [-1 \quad 1 \quad 2 \quad 3 \quad 4 \quad -2]$$

$$\boldsymbol{K}^{pp} = \begin{bmatrix} 15 & 0 \\ 0 & 30 \end{bmatrix}$$

$$UUR = \begin{Bmatrix} 0. \\ 3.269 \\ 4.178 \\ 3.779 \\ 4.699 \\ 5. \end{Bmatrix}$$

$$\boldsymbol{K}^{pf} = [\boldsymbol{K}^{fp}]^T = \begin{bmatrix} -15 & 0 & 0 & 0 \\ 0 & 0 & 0 & -30 \end{bmatrix}$$

$$\boldsymbol{K}^{ff} = \begin{bmatrix} 70 & -15 & -40 & 0 \\ -15 & 50 & -5 & -30 \\ -40 & -5 & 65 & -20 \\ 0 & -30 & -20 & 80 \end{bmatrix}$$

$$PUR = \begin{Bmatrix} -49.033 \\ 15. \\ 0. \\ 0. \\ 25. \\ 9.033 \end{Bmatrix}$$

$$UF = \begin{Bmatrix} 3.269 \\ 4.178 \\ 3.779 \\ 4.699 \end{Bmatrix}$$

$$PUR = \begin{Bmatrix} -49.033 \\ 9.033 \end{Bmatrix}$$

Figure 5: Results for the spring example in Fig.4

Grading schedule

We will start running the grading script twice daily (2:00pm and 11:59pm) on Tuesday Feb 7th. Scores and feedback should be available to you by 8am. Scores and feedback should be available to you a few hours after each run (1-5 hours depending on the assignment). In case you don't get a satisfactory score, you can change your code, make another commit, and your code will be re-graded. The grading script will run twice daily until Friday, always at the same time. You final score will be the one from the last run.