CEE 235A Advanced Structural Analysis **November 6, 2024**
Programming Project

Due Dates:
Interim Submission 1: Monday, November 18 at 11:59 p.m.
Interim Submission 2: Friday, November 29 at 11:59 p.m.
Interim Submission 3: Monday, December 9 at 11:59 p.m.
Final Program Submission: Monday, December 16 at 11:59 p.m.

You are welcome and encouraged to work on this project in teams of two people. Individual teams may discuss the project with other groups, but **under no circumstances should teams share or copy source code from one another.**

OVERVIEW

This handout describes the requirements of the interim and final submissions for the computer programming assignment to write an analysis program for three-dimensional framed structures. The program that you are to write should have the following features capabilities:

- Linear-elastic analysis of three-dimensional structures for any geometry that can be input into the MASTAN2 pre-processor.
- Definition of free or fixed support conditions as permitted by MASTAN2, including the ability to analyze cases with specified support displacements.
- Prismatic beam-column elements *including* flexural and shear deformations. Note – you should assume that all structures are input and specified as rigid FRAMEs, i.e., you do not need to deal with TRUSS structures or pin-ended members. However, you will receive extra credit (20%) if you develop a program that can accommodate pin-ended members.
- Concentrated nodal forces and moments defined in the global coordinate system.
- Uniform member loads defined in the member coordinate system.

PROGRAMMING ENVIRONMENT

You will write the program in the MATLAB programming language and make use of the pre- and post-processing features of MASTAN2. The upper most level of your program should be called "*ud_3d1el.m"* and should conform to the input/output variable definitions required by MASTAN2. The name of this function stands for "User Defined 3D 1st-order Elastic". This upper level function will in turn call other functions that you write. All of your functions should begin with your own unique identifier (e.g., your initials), to distinguish them from other functions that are part of MASTAN2 or the code of other students in the class.

You should write your code in a well-structured and modular fashion making appropriate use of comment lines (%) and sub-functions that are called by your main program. Each of your files should begin with a header similar to that of the function *ud_3d1el.m* that lists the function purpose, author/date, input-output arrays, functions called, etc. Modularity and structured programming are important for several reasons. First, they will help facilitate debugging since you can test individual sub-portions of your code before assembling the overall program. Second, they will allow you to divide tasks between you and your partner.

INTERIM SUBMISSIONS

You have four weeks to complete the entire programming project. This is much more time than you should need, but you should plan and use your time wisely. Interim submissions are required to help guide you through the project and avoid procrastination. Requirements for these submissions are included below, and the due dates are listed above.

Note – the interim assignments are not balanced in terms of workload. A rough breakdown of the estimated workload for each interim assignment is given below. Therefore, don't become complacent after completing assignments 1 and 2. The interim submissions will be reviewed for general conformance with the assignments and good programming conventions and will be considered in your final grade.

Interim Assignment 1 – 10%
Interim Assignment 2 – 20%
Interim Assignment 3 – 40%
Final Submission (including report) – 30%

Interim Submission 1:

A) Write a short MATLAB program that will calculate the 12 x 12 element stiffness matrix for the beam-column element in global coordinates (including shear deformations). The program should be written in three separate modules, each of which will be a *.m file. One *.m will be a simple script file that will serve to test the two function files described below. This script file is just for testing purposes and will eventually be replaced by your main analysis program. The two function files should be written so that they can later be incorporated within your *ud_3d1el.m* program that is called from MASTAN2.

yourinitials_estiff.m: Write a function called **yourinitials_estiff.m** to calculate a 12 x 12 element stiffness matrix in local coordinates. The first line of **yourinitials_estiff.m** should read:

function [elk] = yourinitials_estiff (A, Izz, Iyy, J, Ayy, Azz, E, v, L)

Given the input data (in parenthesis), this function should return a 12 x 12 array called "elk", short for element "k" matrix.

yourinitials_etran.m: Write a function called **yourinitials_etran.m** to calculate a 12 x 12 element transformation matrix to go from local to global coordinates. The first line of **yourinitials_etran.m** should read:

function [gamma] = yourinitials_etran (coordi, coordj, webdir)

Given the input data (in parenthesis), this function should return a 12 x 12 array called "gamma". The input data is assumed to be three vectors that include coordinates of the i-th and j-th node and the member unit web direction.

To run these, you will need to type in arrays with the input variables. It's suggested that you use data from either an old homework or from an example problem in the textbook. Also, the TA may post a test case that you can run.

Note – your function and variable names need not be exactly the same as those given above. But your names should be representative of the functions and, where appropriate, the variable names should follow those you will later be calling from ***ud_3d1el.m****.*

B) Edit the file ***ud_3d1el.m*** to include statements that will print the input variables for ***ud_3d1el.m*** when the MASTAN2 program is run. You don't need to include formatted print statements, rather simple lines listing the variables will do. These print statements should come before the executable statement "AFLAG = inf;". So, the additions you will be making to ***ud_3d1el.m*** will look like the following:

```
nnodes
coord
etc…
AFLAG = inf;
```

Then you should test that these changes work by inputting the sample structure from the lecture on the project into MASTAN2 and running a USER DEFINED first-order elastic analysis (linear elastic). The MASTAN2 program should return with the statement that no analysis code is available, but the input variables should appear in the MATLAB window. Print out the input variables and compare these to the example analysis model geometry and compare them to make sure that you understand how the input arrays are organized.

The main purpose of this part of the assignment is to begin familiarizing you with how your program will interact with and be part of MASTAN2. While you are doing this, you should experiment with the MATLAB debugger, i.e., set some break points in your ***ud_3d1el.m*** file and use the "step" command to march through your code line by line.

SUBMISSION: For this first interim submission, you should hand in the following: (1) pdf of your source code for the three routines in part A, and (2) pdf of the edited version of ***ud_3d1el.m*** *from part B.*

Interim Submission 2:

A) Write the required Matlab functions and modifications to ***ud_3d1el.m*** that will do the following:

(1) create an array that lists the internal degree of freedom numbers for each element using the member connectivity data input from MASTAN2 into ***ud_3d1el.m*** (i.e., the data in the arrays "ends" and "fixity"). The array you create is essentially what is referred to as "memb_id (nele, 12)" in the lecture notes on the programming assignment.
(2) create an array of dof loads (the applied load vector), from the concentrated load data input from MASTAN2 into ***ud_3d1el.m*** (i.e., input data in the array "concen"),

You should check that your program develops the member DOF numbering and load arrays correctly by inputting the frame example that we went through in class. You should also check it with another example of your own that includes more members and applied nodal loads.

B) Flow Chart: Prepare a draft flow chart with a plan for the variable declarations and procedural instructions for your program.

SUBMISSION: For this second interim submission, you should hand in the following:

- *draft flow chart*
- *pdf of your source code for your Matlab functions, including edited version of* ***ud_3d1el.m*** *(note – at this point you can remove the print statements that you added to* ***ud_3d1el.m*** *for interim submission 1)*
- *pdf of the examples and output that you used to confirm that your routines work.*

Interim Submission 3:

Write the required Matlab functions and modifications to ***ud_3d1el.m*** that will do the following:

(1) assemble the global $[K_{ff}]$ and $[K_{fs}]$ matrices.,
(2) modify the array of dof loads from submission 2 to include the contributions from distributed member loads (i.e., using the member load data from the input array "w"),
(3) solve for the displacements at the free dof,
(4) solve for reactions at the support dof including the required modifications to account for any distributed member loads,
(5) check the accuracy of the calculated displacements by comparing back-multiplied dof loads to the original load vector for the free DOF (i.e., compare [P] = [Kff] [Δ] to the input [P] vector).

SUBMISSION: For this third interim submission, you should hand in the following:

- *pdf of your source code for your Matlab functions, included the edited version of* ***ud_3d1el.m***
- *pdfs of two examples that you've used to check that your calculated displacements and reactions are correct. You can obtain the correct solutions from either worked examples in the book and/or using the regular features of MASTAN2.*

Final Submission:

For your final submission, your program should have essentially all of the functionality of the standard 1st-order elastic analysis routines in MASTAN2 (see above). For this submission you should hand in the following:

(a) A short (1/2 page) written description of your program summarizing its features and the overall structure of the program. The description should describe some of the material in the flow chart (next item).
(b) A flow-chart describing the logic, structure and variables (data flow) in your program.
(c) A pdf listing of your program. We expect that the main program file ***ud_3d1el.m*** will include comment statements that define and describe all the main variables and functions in your program. Each sub-function should also include appropriate comment statements.
(d) An electronic copy of your source code. We will be running you program to check that it really works, so you should ZIP your files and upload them to Bruin Learn (specific instructions to follow).
(e) A summary of results from the example/verification problems, which will be distributed shortly.
(f) A short statement on how work and responsibility was divided between the two partners on the project team. The extent to which each team member contributes to the project will be reflected in your individual project scores.
(g) A copy of the completed grading sheet with your self-assessment of your effort (see attached sheet).

Grading: The programming assignment will be graded on the following basis:

1) [50 %] Does the program include all the required features, and does it run correctly?

2) [30 %] Is the code well organized (modular, fairly efficient) with clear comment and variable definition statements? The key point here is that someone who is familiar with structural analysis and programming concepts should have no problem understanding your code.

3) [20%] Quality and clarity of presentation of the final report submission, including (a) flowchart and accompanying brief narrative of the program organization, (b) summary of verification problems and brief comments on agreement (or lack thereof) of results between your program and MASTAN2, (c) comments on the assignment, including how responsibilities were shared with your partner, what you learned from the assignment, and ways that the assignment could be improved in the future.

A NOTE ABOUT PROGRAM FLOWCHARTS: The main purpose of developing the program flowchart is to help organize your code, including (a) the overall logic and steps in the analysis, (b) modularity and how the required calculations will be organized in Matlab Functions, and (c) data flow and variable names. The flowchart is also essential to explain your code organization to others, such as the TA or me if you need help debugging. There are many resources about flow charts available on the internet, but many of these go overboard (for this project application) in formalizing the syntax, symbols and other conventions used for flowcharts. Moreover, while you are welcome to create your flowchart using computer drawing programs, this is NOT necessary. I would much rather see a neat handwritten flowchart that is genuinely useful to you when planning your program than to wait and create an elaborate computer-generated flow chart after your program is completed. It is also fine (even preferable) to write your flowchart in pencil, so that you can erase and update it as your program evolves.