AE1106: Programming I
Graded Assignment #3

December 15, 2010

Please read these instructions first:

As an individual assignment, the code and solutions you submit should represent your own work. Discussion with others is not allowed, and neither is the use of online resources other than the standard Matlab help pages (i.e., no Wikipedia, Google searches, chat windows, etc.). During this assignment, the student assistants and the lecturer will only answer questions related to what the question is asking you to do, and will not assist you with how to actually complete the assignment. When finished, you will submit your Matlab source code file(s) to BlackBoard for grading (see below for submission instructions). **The deadline to complete the assignment is 12:00 for morning sessions, and 17:00 for afternoon sessions.** These deadlines are managed automatically by BlackBoard, so it is important that you submit your documents before the deadline in order to receive credit for the assignment (i.e., manage your time so that you start the submission process at least 15 minutes before the deadline). Those students who have been approved by the student counselors to receive extra time on their exams will receive an extra 30 minutes to complete the assignment.

This individual assignment will give you experience in working with basic matrix operations. Please follow the instructions for each question carefully in order to avoid deductions on the assignment. We recommend you also read the Grading section on Page 3 before you start work on the questions.

1. (3 pts) Write a function called "mattrans.m" that will take the transpose of an input matrix. The function should do the following:

    a. The function should take a single matrix variable A as an input, and return a single matrix B as an output.
    b. The output matrix B should contain the transpose of the matrix A.
    c. The input matrix can be of any dimension (i.e., m x n), meaning that the dimension of the output matrix should be n x m.
    d. The code to create the transpose should be done using only for-loops. As such, **the function should NOT use the built-in Matlab transpose commands, such as A', or transpose(A)**, although you may certainly use these to verify that your function generates the correct results.

    Hint: first write out how you would take the transpose manually using a small example on paper. Then later test your code on a small matrix first (e.g., a 2x2 or 3x3 case) to spot any errors.

2. (4 pts) Now write another function called "matmult.m" that computes the product of two input matrices. The function should do the following:

    a. The function should take two matrix variables (A and B) as an input, and return a single matrix C as an output.
    b. The output matrix C should contain the matrix product C = A*B.
    c. **The function should NOT use the built-in Matlab matrix multiplication commands, such as "A*B"**. The code to compute the multiplication should be done using only (nested) for-loops, and the use of the multiplication operator (*) should only happen at the level of individual matrix elements (i.e., a scalar multiplied by another scalar).
    d. The input matrices can be of any dimension, but for the multiplication to work, the number of columns of A must be equal to the number of rows of B. For example A can be 4x5, and B can be 5x3, but B cannot be 3x5. As such, you must check for this condition in your code

and display an error message if an error is found. If an error is found, the code should display an error message and exit the function without doing any further calculations.

Hints:

- You know from your linear algebra course that matrix multiplication consists of a series of vector dot-products (i.e., the dot product of a row with a column vector). So first figure out how to take the dot product of a row and column vector using a for-loop (**you cannot use the Matlab function `dot` in your final code**, but you can use it to verify your answer).
- As with the previous problem, first figure out how you would take the product of two matrices manually using a small example on paper. Then later test your code on a small matrix first (e.g., a 2x2 or 3x3 case) to spot any errors.
- You should be able to do the entire multiplication using a series of three nested for-loops (i.e., a for-loop within a for-loop within a for-loop).

3. (3 pts) Using the functions you wrote in Questions 1 & 2 (**`mattrans`**, **`matmult`**), you will now apply a simple coordinate transformation to a set of satellite orbit positions, and plot the results. The data given to you is in an Earth-Centered-Earth-Fixed (ECEF) reference frame, and you will first need to convert it to an Earth-Centered-Inertial (ECI) reference frame. You can do this conversion by multiplying the following matrix,

$$R = \begin{bmatrix} \cos(\omega t) & \sin(\omega t) & 0 \\ -\sin(\omega t) & \cos(\omega t) & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

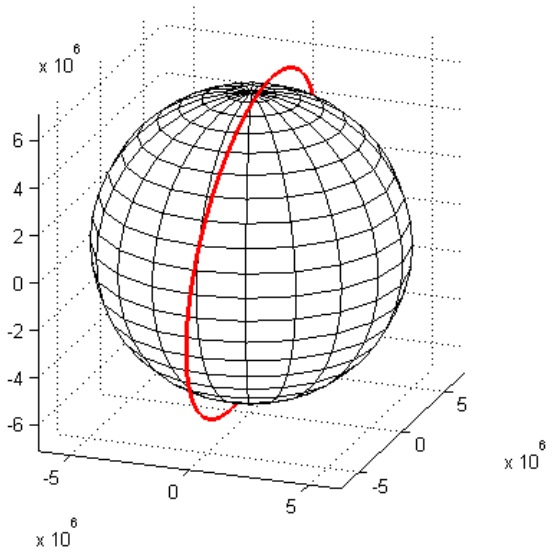
with a 3D position vector, X_{ECEF} = [x y z], where $\omega = 2\pi/86164$ (units of radians) is the rotation rate of the Earth (1 sidereal day = 86164 sec), and t is the time. For example, to convert a single position at a given time, you would perform the following multiplication:

$$X_{ECI} = RX_{ECEF}^{T} = \begin{bmatrix} \cos(\omega t) & \sin(\omega t) & 0 \\ -\sin(\omega t) & \cos(\omega t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

On BlackBoard is a text data file called "ecef.txt" which contains four columns of data: t, x, y, and z. Each row contains the time (in sec.) and ECEF coordinates (in km) of the satellite. Create a script called "orbit.m" that reads this data file and applies the coordinate transformation above to convert every ECEF position in the file to an ECI position. Note that because the positions are stored as rows in the data file, you will need to take the transpose of each position vector in order to apply the transformation. **Be sure to use your own routines for the matrix multiplication and transpose operations** (see the Grading section for instructions if you cannot get your code from questions 1 or 2 to work). Then plot the resulting ECI positions using the following plot commands:

```
Re = 6378135; % Earth radius, in m
[a, b, c] = sphere;
surf( a*Re, b*Re, c*Re, 'FaceColor', 'w', 'EdgeColor', 'k');
hold on;
plot3( ???, ???, ???, 'r', 'LineWidth', 2)
axis equal
view( 20, 20 );
```

where the "???" in the code above should be replaced by the x, y and z ECI position vectors (converted from km to m for the plot). The plot you generate should look similar to the one below:



Grading

1) The functions you write for Questions 1-3 will be tested by the graders using a script similar to the one shown below.

```
% Test a standard small case
A = round(rand(3,3)*10);
B = round(rand(3,3)*10);

C = mattrans(A);
if sum(sum(C - A')) == 0, disp('OK'); else disp('Error'); end

C = matmult(A,B);
if sum(sum(C - A*B)) == 0, disp('OK'); else disp('Error'); end

orbit
```

   Note that this script is only an example, and the actual grading script used may be different; however, you should verify that the files you upload to BlackBoard should work when the above code is run.
2) The value for each question is provided above, totaling to 10. Each question will be mostly evaluated on how well the code submitted produces the correct answers according to the test script; however, deductions will be made for code that does not run properly or produces unclear results.
3) If you cannot complete question 1 or 2, then you may use the default matrix transpose (A') and/or multiplication operator (*) to complete questions 2 and 3.
4) Only submit the following m-files for this assignment: **`mattrans.m`**, **`matmult.m`**, and **`orbit.m`**

Submission Instructions

1) When you are ready to submit your files to BlackBoard, go to the Assignments and then click on the "Graded Assignment 3" item. This will open a page where you can upload your assignment files.
2) By clicking on the "Browse My Computer" button next to the "Attach File" label, you can add one or more files to your assignment submission. You can save the current state of your submission at any time using the "Save as Draft" but your work will not be officially recorded in the system until you click the "Submit" button. After you have submitted the assignment, BlackBoard should then show you a summary page, which lets you know the submission went successfully. **Please be patient after submitting your files, as sometimes it takes a few seconds before BlackBoard registers everything properly**. We recommend that after submission, you wait 15-20 seconds, then go back to the course homepage and proceed again through to the "Graded Assignments 3" item. After doing this, you should see all of your submitted files in the summary page if everything went smoothly.
3) Remember that the submission folder automatically disappears at the deadline, so please make sure you start the submission process well before this time.
4) If you made an error and you wish to change a file after your have hit the submit button, you can go back to the "Graded Assignment 3" link and click on the "Start New Submission" link to repeat the entire process (i.e., do not just submit the updated file, but completely resubmit all files). Note that if you make more than one submission, we will only grade the last attempt.
5) As a reminder, please do NOT include special characters in the names of your files, such as "%", brackets "[ ]", underscores "_", etc., as BlackBoard may have problems with these. Also, do NOT upload a ".mat" or ".asv" file…only upload your ".m" files, or other documents (e.g., MSWord file) if requested.