EAS 4510

Project #1

Fall 2022

Due: 21 September 2022

Project Rules

The rules for this project are as follows:

(1) You are not permitted to communicate with anyone about the contents of this project during the entire duration of this assignment.

(2) Aside from the Professor, you are not permitted to get assistance from anyone in any form whatsoever.

(3) Although you are not permitted to comnunicate with anyone, you are permitted to use any resources from books, lecture notes, or any online resources you find useful provided you abide by (1) above. If you use any specific external resource (aside from lecture notes or course materials), you *must provide a citation* to the source from which you obtained the information. The citation should also appear as a reference in the bibliography of your submission.

(4) All numerical solutions must be obtained using MATLAB in accordance with the instructions in the questions.

(5) You must type your solutions neatly in either Microsoft Word or LaTeX.

(6) You must be crystal clear with every step of your solution. In other words, every step in a derivation or statement you write must be unambiguous (that is, each step must have one and only one meaning). If any step is ambiguous, it will be assumed to be incorrect.

Anyone suspected of violating the assignment rules will be subject to the rules and regulations of the University of Florida Academic Honesty policy found by clicking **here**.

Point Distribution

The exam consists of a series of questions with the value of each question clearly indicated. Unless otherwise stated, full credit will be given for a proper application of a relevant concept. Contrariwise, no credit will be given for a concept applied incorrectly, ***even if the final answer is correct***.

University of Florida Honor Code

On your exam you must state and sign the University of Florida honor pledge as follows:

We, the members of the University of Florida community, pledge to hold ourselves and our peers to the highest standards of honesty and integrity. On all work submitted for credit by students at the university, the following pledge is either required or implied: On my honor, I have neither given nor received unauthorized aid in completing this assignment.

Name: **UF-ID:**

Signature: **Date:**

Requirements for Writing MATLAB Code

Great care should be taken writing the MATLAB code for this assignment. Here are some requirements for the code:

(1) Numbers should *not* be hard-wired into the code. In other words, if a symbol appears in any equation or relation, this symbol should be defined with numeric value at the top of the code, and the variable corresponding to this symbol should be used throughout the code. For example, in the description above, μ is given the value unity. Thus, the MATLAB code should read similar to the following:

```
mu = 1; % Use mu as a variable throughout the code (DO NOT USE "1").
```

(2) All code should be clearly commented. Commenting code means that every section of the code should be preceded by at least two or three sentences explaining what that section of the code does. Comments should also be included in any function that is written. This comment block should explain the purpose of the function and the comment block should appear between the "function" line and the first line of actual code in the function. Furthermore, this comment block should explain the inputs and outputs of the function.

(3) Any code that requires solving the problem for multiple values of a parameter must be coded using a loop. In other words, ***the same code should not be copied multiple times for the different values of the parameter!*** Furthermore, loops must be on ***integers*** and are ***not on real numbers!*** Given that you are asked to solve the problem using four different values of N, the loop should run from one to the length of the vector that contains the values of N. *Do not write a loop using a real-number index!* In order to assist you with a looping structure, here is a MATLAB code snippet:

```
N = [10, 15, 20, 25];
for j = 1:length(N),
% INSERT YOUR CODE THAT RUNS EACH LOOP ITERATION.
end
```

NOTE: All of the above instructions must be followed in order to obtain credit for this assignment. If any the above instructions are not followed, no extra credit will be given.

Background

It is the year 1965. It is the height of the Cold War with The United States and The Soviet Union in the midst of a Space Race. The United States Space Command (USSPACECOM) at Cheyenne Mountain, Colorado has just learned of the existence of an unidentified spacecraft in an elliptic orbit that seems to have interesting implications for reconnaissance against The United States. Being an astrodynamicist with a solid understanding of mathematics and your ability to program in MATLAB (was MATLAB available in 1965?), you have been asked to determine both a theoretical and numerical approach for determining the time taken by the spacecraft to move between two different points on the orbit. For the theoretical part of the analysis, you have been asked to derive a mathematical expression that determines the duration, $t_2 - t_1$, for the spacecraft to move between the true anomaly values $\nu(t_1) = \nu_1$ and $\nu(t_2) = \nu_2$. For the computational part of the analysis, you have been asked to write a program that approximates this integral and provide some information that will potentially be of use to USSPACECOM.

Question 1: 15 Points

Using the known properties of an elliptic orbit, determine a relationship between the change in time, $t_2 - t_1$, as a function of the movement of the spacecraft from an a true $\nu(t_1) = \nu_1$ at time $t = t_1$ to a final true anomaly $\nu(t_2) = \nu_2$ and time $t = t_2$. Express your result in the following form:

$$t_2 - t_1 = \int_{\nu_1}^{\nu_2} f(\nu, p, e, \mu) d\nu, \tag{1}$$

where p is the parameter (semi-latus rectum), e is the eccentricity, and μ is the gravitational parameter.

Note: the quantities p and e provides key information about the orbit. For this problem it should be possible to simplify the expression in Eq. (1) such that the function f depends only upon ν, p, e, and μ.

Question 2: 20 Points

As it turns out, the integral in Eq. (1) cannot be evaluated analytically (if the integral could be evaluated analytically, then orbital mechanics would be significantly simpler than it is). As a result, it is necessary to obtain a numerical approximation of the integral. Fortunately, a famous German mathematician named Johann Carl Friedrich Gauss[1] developed an extremely accurate approach to approximating the integral of a function. This approach, called *Gaussian quadrature*, approximates the integral of a function $f(x)$ on the interval $x \in [a, b]$ via the following approximation:

$$\int_a^b f(x) dx \approx \sum_{i=1}^{N} w_i f(x_i), \tag{2}$$

where $(x_1, \ldots, x_N)$ are the N Gauss points on the interval $x \in [a, b]$, $(w_1, \ldots, w_N)$ are the N Gauss weights that correspond to the N Gauss points, and N is the number of Gauss points. The Gauss quadrature approximation given in Eq. (2) can be used for any $N \geq 2$. Fortunately, you have been given a MATLAB code `GaussPointsWeights.m` (see Appendix) that, given a value of N, will compute the Gauss points and Gauss weights on the interval $x \in [a, b]$. The Gauss points and weights will, in turn, be used to evaluate the integral. Using the `GaussPointsWeights.m` code, as a foundation, write a `MATLAB` function that computes the approximation of the integral given in Eq. (1) using Gaussian quadrature. This `MATLAB` function should have the following inputs:

(a) a function $f(\nu, p, e, \mu)$;

(b) the initial and terminal values of the true anomaly, ν_1 and ν_2, respectively;

(c) the semi-latus rectum, p;

(d) the eccentricity, e;

[1] Johann Carl Friedrich Gauss : 30 April 1777 - 23 February 1855

(e) the gravitational parameter, μ;

(f) the number of Gauss points, N.

The output of this MATLAB function should be the approximation of the integral given in Eq. (1). The function $f(\nu, p, e, \mu)$ should be a function handle that is an input to the function. Thus, this function should look like

```
function f = timeChangeIntegrand(nu,p,e,mu)
% ------------------------------------------------------------%
% Integrand f(nu,p,e,mu) that is used to obtain the time change %
%                         deltat =t2 - t1                     %
% Inputs:                                                     %
%    nu:  true anomaly (rad)                                  %
%     p:  parameter (semi-latus rectum)                       %
%     e:  eccentricity                                        %
%    mu:  gravitational paramter                              %
% Output:                                                     %
%     f:  value of function at (nu,p,e,mu)                    %
% ------------------------------------------------------------%
```

Furthermore, the input-output structure of the function that computes the integral together with a relevant comment block should look as follows:

```
function deltat = timeChangeIntegral(f,nu1,nu2,p,e,mu,N)

% ------------------------------------------------------------%
% This function employs Legendre-Gauss quadrature to compute an %
% approximation to                                            %
%                              /nu2                           %
%                             /                               %
%                             |                               %
%      deltat =  t2 - t1  =   | f(nu,p,e,mu)dnu               %
%                             |                               %
%                             /                               %
%                            /nu1                             %
% where f(nu) is a function that used to define the change in %
% The inputs and outputs of this function are as follows:     %
% Inputs:                                                     %
%   f        = a handle to the function to be integrated      %
%   nu1      = lower integration limit                        %
%            = initial true anomaly (rad)                     %
%   nu2      = upper integration limit                        %
%            = terminal true anomaly (rad)                    %
%   p        = parameter (semi-latus rectum)                  %
%   e        = eccentricity                                   %
%   mu       = gravitational parameter                        %
%   N        = number of Gauss points & weights               %
%              used to approximate the integral               %
% Output:                                                     %
%   deltat   = Gauss quadrature approximation of              %
%              deltat, where deltat = t2 - t1 is the          %
%              time change from nu1 to nu2                    %
% ------------------------------------------------------------%
```

Finally, it is noted that the Gauss quadrature approximation given in Eq. (2) can be written in vectorized form as

$$\int_a^b f(x)dx \approx \sum_{i=1}^{N} w_i f(x_i) = \mathbf{w}^{\mathsf{T}}\mathbf{F} \tag{3}$$

where

$$\mathbf{w} = \begin{bmatrix} w_1 \\ \vdots \\ w_N \end{bmatrix} \quad , \quad \mathbf{F} = \begin{bmatrix} f(x_1) \\ \vdots \\ f(x_N) \end{bmatrix}. \tag{4}$$

and $\mathbf{w}^\mathsf{T}$ is the transpose of the column vector $\mathbf{w}$ (that is, given that $\mathbf{w}$ is a column vector, the quantity $\mathbf{w}^\mathsf{T}$ is a *row* vector). It is important to understand that the function $f(\nu, p, e, \mu)$ can be vectorized in `MATLAB` and, thus, a loop is *not* required in order to compute $\mathbf{F}$ in Eq. (4).

Question 3: 60 Points

USSPACECOM has found the following through measurements that the Earth-centered inertial (ECI) position and inertial velocity at a point on the orbit are given, respectively, as follows:

$$\begin{aligned} \mathbf{r}^\mathsf{T}(0) &= \begin{bmatrix} 5634.297397 & -2522.807863 & -5037.930889 \end{bmatrix} \text{ km}, \\ {}^\mathcal{I}\mathbf{v}^\mathsf{T}(0) &= \begin{bmatrix} 8.286176 & 1.815144 & 3.624759 \end{bmatrix} \text{ km/s}. \end{aligned} \tag{5}$$

Furthermore, it has been determined that two points on the orbit of key interest are the ascending and descending nodes. Suppose that t_1 and t_2 are, respectively, the values of time when the spacecraft is located at the ascending and descending node. Using the information provided, the time has now come for you as the astrodynamicist to put your `MATLAB` code to work so that you can provide the USSPACECOM the analysis they are seeking. This work must be organized and presented in a way that USSPACECOM can clearly interpret your results. Assuming that the Earth the gravitational parameter is $\mu = 398600 \text{ km}^3 \cdot \text{s}^{-2}$ and that the radius of the Earth is $R_e = 6378.145$ km, employ the MATLAB code you have written in order to provide USSPACECOM answers to the following questions:

(a) Starting with $(\mathbf{r}^\mathsf{T}(0), {}^\mathcal{I}\mathbf{v}^\mathsf{T}(0))$ given above, perform a *series of computations* such that the following values on the orbit are obtained:

- the semi-latus rectum in *kilometers*
- the eccentricty
- the orbital period in *hours*

Each step in your series of calculations along with the semi-latus rectum, the eccentricity, and the orbital period should be printed as shown in `fprintf` statements given in the **Notes** below. [6 points]

(b) Determine the values of the true anomaly corresponding to the ascending and descending nodes, respectively. [12 Points]

(c) Using the Gauss quadrature approximation of the integral developed earlier, estimate the time elapsed on the orbit in hours from the ascending node to the descending node for the following number of Gauss points: $N = (10, 15, 20, 25)$. The results should be printed as shown in the `fprintf` statements given in the **Notes** below. [12 points]

(d) Determine the time elapsed on the orbit from the descending node to the ascending node for $N = (10, 15, 20, 25)$ Gauss points *without having to call the `MATLAB` function written for Question 2* again. The results should be printed as show in the `fprintf` statements given in the **Notes** below. [5 points]

(e) Make a plot of the orbit for this problem alongside a second plot that shows a circle of radius R_e (that is, a circle that depicts the Earth). The plot should be made using the `MATLAB` command `polarplot`. Include on the plot the ascending and descending nodes. The ascending node should be located with a marker of a circle (that is, the marker 'o' in MATLAB) while the descending node should be located on the plot with a marker of a square (that is, the marker 's' in MATLAB). Please note that this plot should contain *two* curves, one for the Earth and the other for the orbit of the spacecraft. [10 points]

(f) Explain what you find interesting about the orbit. This explanation should not be simply a single sentence. Instead, explain clearly from both the data you have acquired (through your computations above) and the plot you made in part (d) what makes the orbit interesting. [15 points]

Important: when answering part (d) above, please provide a written explanation of what is interesting about the orbital period in terms of what you know about the rotation rate of the the Earth. In other words, explain the significance of the orbital period in terms of geographic information.

Notes

Print all results neatly using the command `fprintf` in MATLAB in the format shown below and include this block of text output in your submission. The quantities to be printed are those asked in Question 3. The format of the `fprintf` statements is shown below.

```
fprintf('-------------------------------------------------------\n');
fprintf('-------------------------------------------------------\n');
fprintf('                        Part (a):                      \n');
fprintf('-------------------------------------------------------\n');
fprintf('-------------------------------------------------------\n');
fprintf('Quantity 1 [units]:                          \t\t %16.8f\n',q1);
fprintf('-------------------------------------------------------\n');
fprintf('Quantity 2 [units]:                          \t\t %16.8f\n',q2);
fprintf('-------------------------------------------------------\n');
fprintf('                             .                         \n');
fprintf('                             .                         \n');
fprintf('                             .                         \n');
fprintf('-------------------------------------------------------\n');
fprintf('Quantity Q [units]:                          \t\t %16.8f\n',qQ);
fprintf('-------------------------------------------------------\n');
fprintf('Semi-Latus Rectum (p) [units]:               \t\t %16.8f\n',p);
fprintf('-------------------------------------------------------\n');
fprintf('Eccentricity (e) [units]:                    \t\t %16.8f\n',e);
fprintf('-------------------------------------------------------\n');
fprintf('Orbital period [hours]:\t\t\t\t\t %16.8f\n',           tauhr);
fprintf('-------------------------------------------------------\n');
fprintf('-------------------------------------------------------\n');
fprintf('   Part (c): Time change from nu1=%i deg to nu2=%i deg    \n',nu1deg,nu2deg
   );
fprintf('-------------------------------------------------------\n');
fprintf('-------------------------------------------------------\n');
for i=1:length(N),
    fprintf('Time elapsed (%i,%i) deg [hours] (N=%i):%16.8f\n',nu1deg,nu2deg,N(i),
       deltat1(i));
end
fprintf('-------------------------------------------------------\n');
fprintf('-------------------------------------------------------\n');
fprintf('   Part (d): Time change from nu2=%i deg to nu1=%i deg  \n',nu2deg,nu1deg);
fprintf('-------------------------------------------------------\n');
fprintf('-------------------------------------------------------\n');
for i=1:length(N),
    fprintf('Time elapsed (%i,%i) deg [hours] (N=%i):%15.8f\n',nu2deg,nu2deg+180,N(i
       ),deltat2(i));
end
fprintf('-------------------------------------------------------\n');
fprintf('-------------------------------------------------------\n');
```

```
fprintf(' Part (e): Please provide a written explanation indicating\n');
fprintf(' anything that is interesting regarding the orbital period\n');
fprintf(' that was computed in part (b) of this question.  If more \n');
fprintf(' lines of fprintf statements are needed, simply copy and  \n');
fprintf(' paste more fprintf statements to your code and continue  \n');
fprintf(' with your explanation.                                   \n');
fprintf('----------------------------------------------------------\n');
fprintf('----------------------------------------------------------\n');
```

Appendix: MATLAB Code for Computing Gauss Points and Weights

```
function [x,w] = GaussPointsWeights(a,b,N)

% ------------------------------------------------------------%
% This function computes the N Gauss quadrature points and the  %
% corresponding N Gauss quadrature weights on [a,b].            %
% Inputs:                                                       %
%   a = lower limit of integral to be evaluated                 %
%   b = upper limit of integral to be evaluated                 %
%   N = Number of Gauss Points and Weights to Be Computed.      %
% Output:                                                       %
%   x = Column Vector with the N Gauss Points on [a,b].         %
%   w = Column Vector with the N Gauss Quadrature Weights.      %
% ------------------------------------------------------------%
%         This code uses the Golub-Welsch algorithm.            %
% ------------------------------------------------------------%

beta = .5./sqrt(1-(2*(1:N-1)).^(-2)); % 3-term recurrence coeffs
T = diag(beta,1) + diag(beta,-1);     % Jacobi matrix
[V,D] = eig(T);                       % Eigenvalue decomposition
x = diag(D); [x,i] = sort(x);         % Gauss points on [-1,+1]
w = (2*V(1,i).^2).';                  % Gauss weights on [-1,+1]
x = (b-a)/2*x + (b+a)/2;              % Gauss points on [a,b]
w = (b-a)/2*w;                        % Gauss weights on [a,b]

end
```