AA 279A – Space Mechanics Winter 2025

Problem Set 7

Wednesday, 5 March 2025
Prof. Andrew Barrows

Due: Wednesday, 12 March 2025 by 3:00pm
Topics: Single- and multi-body numerical simulations in MATLAB, orbital transfers

7.1) You will use MATLAB to simulate the motion of an object orbiting a spherically-symmetric central body by numerically integrating its equations of motion. Work in canonical units. The central body is fixed at the origin with $\underline{r}$ = (0, 0, 0) [DU] and $\mu = 1$ [DU^3/TU^2].

a) Write a MATLAB function that returns the time derivative of the orbiting object's state vector (a 6x1 matrix) as a function of state and time. Note that MATLAB's integration functions require this function to be of the form below. (Of course, you will need to add more code to make the function work.)

```
function [statedot] = yourfunc(t, state)
% state vector is [rx ry rz vx vy vz]'
%
% although required by form, input value t will go unused here

statedot = zeros(6, 1);

statedot(1:3) = state(4:6); % rdot = velocity
statedot(4:6) = ???;

end % terminates MATLAB function
```

b) Use MATLAB's **ode45** integration function (a Runge-Kutta variant) to simulate the motion of the object using the following initial conditions:

$\underline{r}$ = (4, 0, 0) [DU] and $\underline{v}$ = (0, 0.5, 0) [DU/TU] (velocity is WRT inertial space)

Note that **ode45** requires an '**@**' character to be placed at the beginning of your function name. Run the simulation using the following (default) options:

```
options = odeset('RelTol', 1e-3, 'AbsTol', 1e-6);
[t_out, y_out] = ode45(@yourfunc, [tstart:tint:tend]', y_initial, options);
```

Simulate for approximately one full orbit. Plot the orbit in the x, y plane. Choose a time interval (" **tint** ") for the output data that results in a smooth plot.

c) Extend the simulation to run for approximately 10 full orbits. Plot the orbit in the x, y plane. Comment on what you see. Is the slow change in orbit shape a numerical effect or a real physical effect?

d) Now run your simulation using a different integrator. Use MATLAB's **ode113** integration function (Adams-Bashforth-Moulton) with the following (default) options:

```
options = odeset('RelTol', 1e-3, 'AbsTol', 1e-6);
[t_out, y_out] = ode113(@yourfunc, [tstart:tint:tend]', y_initial, options);
```

Plot approximately 10 full orbits. Comment on the performance of **ode113** versus **ode45**.

e) Study the Simulink model diagram in the class handout from 3 March 2025 titled "Lecture 12 Simulink Supplement." If you wanted to reuse the "Pluto Orbiting the Sun" model for this problem, into which three Simulink blocks would you need to enter new constants?

7.2) You will create a MATLAB simulation of a simplified five-body solar system comprised of the Sun, Mercury, Venus, Earth, and the Moon. All five bodies will be allowed to move relative to the Solar System Barycenter, which serves as the origin of the simulation. Let your state vector be composed of the bodies' gravitational parameters, position vectors, and velocity vectors with respect to inertial space, all in heliocentric coordinates:

$$\begin{bmatrix} \mu_{Sun} \\ \underline{r}_{Sun} \\ \underline{v}_{Sun} \\ \mu_{Mercury} \\ \underline{r}_{Mercury} \\ \underline{v}_{Mercury} \\ \mu_{Venus} \\ \underline{r}_{Venus} \\ \underline{v}_{Venus} \\ \mu_{Earth} \\ \underline{r}_{Earth} \\ \underline{v}_{Earth} \\ \mu_{Moon} \\ \underline{r}_{Moon} \\ \underline{v}_{Moon} \end{bmatrix}$$

Initial conditions on the state vector at the epoch time of 0000h 1 March 2025 TDB (assume TDB is essentially UTC) can be found in a file on the Canvas website called **`p_7_2_constants.m`**. This file also contains μ_{Sun}, $\mu_{Mercury}$, μ_{Venus}, μ_{Earth}, and μ_{Moon} . (This data was obtained from JPL at http://ssd.jpl.nasa.gov/horizons/ if you would like to augment your simulation with additional bodies.) For your simulations, use MATLAB's **`ode113`** numerical integrator with the following options (these tolerances are tighter than those used on Problem 7.1).

```
options = odeset('RelTol', 1e-6, 'AbsTol', 1e-9);
[t_out, y_out] = ode113(@yourfunc, [tstart:tint:tend]', y_initial, options);
```

a) Write a MATLAB function that returns the time derivative of your solar system state vector as a function of time and state. (This will be conceptually similar to the function you wrote for Problem 7.1 except that before, you had two bodies with only one of them moving; now you have five bodies with all five of them moving. Your function should also account for the fact that the μ values don't change over time and that their time derivatives are therefore zero.) Be sure to account for gravity between *all* pairs of bodies.

b) Simulate the motions for one year. Plot the shape of all five orbits in the x-y plane, all on the same plot. Be sure to label the axes and orbits.

c) Plot the Moon's position relative to the earth $\left(\underline{r}_{Moon} - \underline{r}_{Earth}\right)$ in the x-y plane with the Earth at the origin. How many times does the Moon orbit the Earth during the simulation period?

7.3) We wish to compare the cost of raising a circular equatorial 200 [km] (altitude) Earth parking orbit to geostationary versus that of injecting into a circular equatorial orbit that has the same radius as the Moon's orbit (384,400 [km]). You may ignore Earth oblateness effects and the Moon's gravity. Because numerical accuracy may be important for this problem, we suggest using MATLAB with the **`format long`** setting invoked.

a) Calculate the total ΔV required (two burns) to perform a Hohmann transfer from a circular equatorial 200 [km] parking orbit to geostationary orbit. Also calculate the transfer time.

b) Calculate the total ΔV required (two burns) to perform a Hohmann transfer from a circular equatorial 200 [km] parking orbit to a circular equatorial orbit with radius 384,400 [km]. Also calculate the transfer time

7.4) We now want to compare the Hohmann transfers examined in the previous problem with bi-elliptic transfers that use three burns, coplanar transfer orbits, and a very large intermediate radius of 1,000,000 [km]. You may ignore Earth oblateness effects and the Moon's gravity. Because numerical accuracy may be important for this problem, we suggest using MATLAB with the **`format long`** setting invoked.

a) Calculate the total ΔV required (three burns) for a bi-elliptic transfer from a circular equatorial 200 [km] parking orbit to geostationary orbit. What is the transfer time?

b) Calculate the total ΔV required (three burns) for a bi-elliptic transfer from a circular equatorial 200 [km] parking orbit to a circular equatorial orbit with radius 384,400 [km]. What is the transfer time?

c) Compare the total ΔV's and transfer times with those found for the Hohmann transfers of the previous problem. Which type of transfer minimizes total ΔV when transferring to GEO? Which type of transfer minimizes total ΔV when transferring to the Moon's orbital radius? For crewed missions, why might choosing minimum total ΔV not always be a good choice?

7.5) A spacecraft is launched due eastward from Cape Canaveral (latitude = 28.5 [deg]) into a circular 200 [km] Earth parking orbit (with inclination of 28.5 [deg] naturally). The spacecraft is to be placed into geostationary orbit using two impulsive burns: a "perigee kick" at "point A" that injects the spacecraft into an elliptical transfer orbit, and an "apogee kick" at "point B" that circularizes the orbit at GEO. Assume that the perigee kick occurs at the ascending node of the parking orbit and that it reduces the inclination by an amount Δi_A (where Δi_A is a positive number). The apogee kick brings the remainder of the inclination to zero. We do not care about the final longitude over which the satellite hovers, and you can ignore Earth oblateness effects.

a) Develop a MATLAB function that returns the total ΔV for the two burns, given Δi_A.

b) Plot total ΔV as a function of Δi_A for values of Δi_A from 0.0 [deg] (all inclination change occurs at apogee kick) to 28.5 [deg] (all inclination change occurs at perigee kick).

c) Comment on why doing all the inclination change at perigee kick might be a bad idea.

d) Using MATLAB's `fminbnd` function along with the function you wrote for part (a) (don't forget the `@` in front of your function name!), find the optimum value of Δi_A that minimizes total ΔV. What is the minimum total ΔV?

e) For the optimum transfer determine the direction of the apogee burn (expressed as components normal to the equatorial plane and in the equatorial plane).

f) How long does the transfer take?