ASEN 4057: Assignment 6

Michael Tzimourakas[a] Ross Kloetzel[b]
University of Colorado Boulder, ASEN 4057
Date: 3/16/18

[a]SID: 104805696
[b]SID: 103117373

I. Functions

A. solutions.c

This is the main script that is run and handles all of the computations. To run for each objective, the following commands should be executed:

1. chmod u+x solutions.c
2. gcc -o solutions solutions.c -lm
3. ./solutions objective clearance accuracy

The code functions as follows:

1. Declare system variables
2. Compute initial conditions for forces, velocities, and positions for all three bodies
3. Set step size for optimization based on accuracy specifications
4. Run Euler's method
5. Compare spacecraft position to end conditions
6. Once solution is found, re-run Euler's method with optimized delta-V values
7. Print optimized spacecraft position to text file

A few important things to note is the step size used in Euler's method and the accuracy. Euler's method used a 100 second time step since times smaller often led to "segmentation faults". The accuracy step for optimization depended on the user input during the function call. Optimization was done using a nested for loop of x and y delta-V values. This creates a grid of tested values with x delta-V on the horizontal and y delta-V on the vertical. The minimum delta-V or time was saved and compared to the currently calculated value. If the current value was smaller, it became the new minimum and the processed continued until the entire grid was tested. The step sizes were computed as the $\frac{accuracy}{\sqrt{2}}$ which was derived from $accuracy = \sqrt{\Delta X^2 + \Delta Y^2}$ and assumed that X and Y deltas were the same. The optimization step size played the largest role in the time for the program to run.

B. part3.sh

The shell script created is used to call all the functions with their corresponding variable inputs. The assignment calls for us to test both objectives cases using an input of 1, 10, 100, 1000, 10000, 50000, 100000 km clearance from the Moon. Shell script loops through these values and passes them in as inputs for both objective 1 and objective 2.

C. assignment6.m

The data collected from running solutions.c is outputted to a data file. This data file contains the x,y positions of both the Spacecraft and Moon. The data is collected and plotted , indicating that the resulting delta-v brings the astronauts home. This data is plotted for objective 1 and objective 2.

II. Part 1

A. Objective 1

The resulting minimum delta-V required to return to Earth can be seen below in Part 2.

B. Objective 2

III. Part 2

```
michael@michael-VirtualBox:~/Documents/Kloetzel/assignments/assignment_6$ ./solutions 1 1000 .5
X DeltaV: 0.152543 m/s
Y DeltaV: 79.445436 m/s
Magnitude: 79.445583 m/s
michael@michael-VirtualBox:~/Documents/Kloetzel/assignments/assignment_6$
```

Figure 1: Objective 1 output

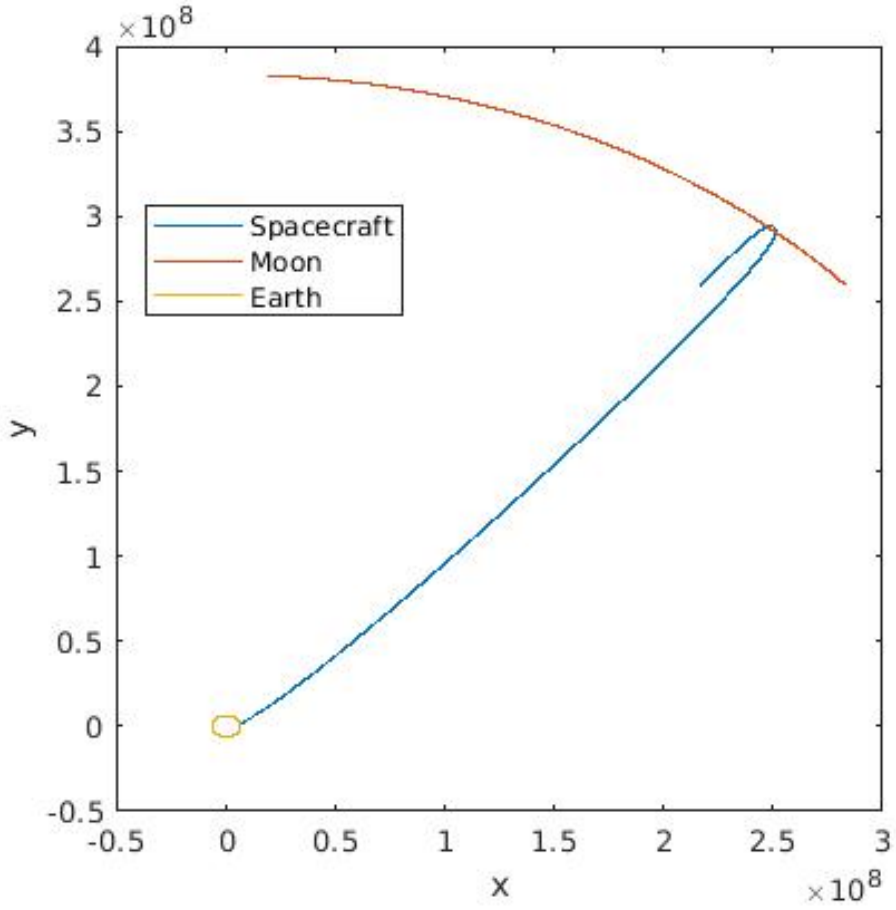

Figure 2: Objective 1 plot

```
michael@michael-VirtualBox:~/Documents/Kloetzel/assignments/assignment_6$ ./solutions 2  1000 .5
X DeltaV: -20.000000 m/s
Y DeltaV: 79.091883 m/s
Magnitude: 81.581407 m/s
Time: 81.638889 hours
michael@michael-VirtualBox:~/Documents/Kloetzel/assignments/assignment_6$
```

Figure 3: Objective 2 output

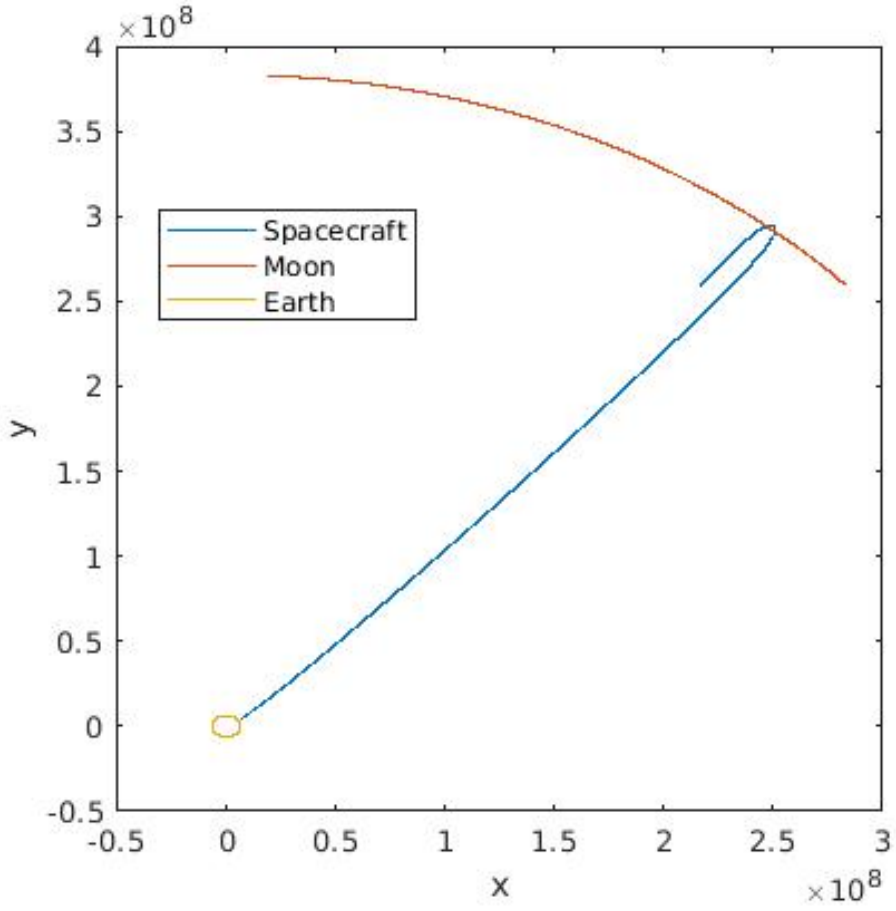

Figure 4: Objective 2 plot

IV. Part 3

part3.sh

```
1  # Michael Tzimourakas
2  # Ross Kloetzel
3  a=0.5
4  for c in 0 10 100 1000 10000 50000 100000; do
5      ./solutions 1 $c $a
6      ./solutions 2 $c $a
7  done
```

Figure 5: Bash Script

Using the bash script shown above, the C function is called and ran through the different test cases for initial guesses. Below is the output of the following code.

output.txt

```
X DeltaV: 0.152543 m/s
Y DeltaV: 79.445436 m/s
Magnitude: 79.445583 m/s
X DeltaV: -20.000000 m/s
Y DeltaV: 79.091883 m/s
Magnitude: 81.581407 m/s
Time: 81.638889 hours
X DeltaV: 0.152543 m/s
Y DeltaV: 79.445436 m/s
Magnitude: 79.445583 m/s
X DeltaV: -20.000000 m/s
Y DeltaV: 79.091883 m/s
Magnitude: 81.581407 m/s
Time: 81.638889 hours
X DeltaV: 0.152543 m/s
Y DeltaV: 79.445436 m/s
Magnitude: 79.445583 m/s
X DeltaV: -20.000000 m/s
Y DeltaV: 79.091883 m/s
Magnitude: 81.581407 m/s
Time: 81.638889 hours
X DeltaV: 0.152543 m/s
Y DeltaV: 79.445436 m/s
Magnitude: 79.445583 m/s
X DeltaV: -20.000000 m/s
Y DeltaV: 79.091883 m/s
Magnitude: 81.581407 m/s
Time: 81.638889 hours
X DeltaV: 0.152543 m/s
Y DeltaV: 79.798990 m/s
Magnitude: 79.799136 m/s
X DeltaV: -20.000000 m/s
Y DeltaV: 79.445436 m/s
Magnitude: 81.924217 m/s
Time: 81.861111 hours
X DeltaV: -6.211418 m/s
Y DeltaV: 80.506097 m/s
Magnitude: 80.745361 m/s
X DeltaV: -20.000000 m/s
Y DeltaV: 80.506097 m/s
Magnitude: 82.953189 m/s
Time: 82.583333 hours
X DeltaV: 0.152543 m/s
Y DeltaV: 81.920310 m/s
Magnitude: 81.920452 m/s
X DeltaV: -20.000000 m/s
Y DeltaV: 81.566757 m/s
Magnitude: 83.982950 m/s
Time: 83.305556 hours
```

Figure 6: Output From Bash Script

V. Profile Report

Note that for the following profile reports, no call graph was generated since solutions.c did not call any other source files.

```
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls  Ts/call  Ts/call  name
100.10      0.46     0.46                             main

 %         the percentage of the total running time of the
time       program used by this function.

cumulative a running sum of the number of seconds accounted
 seconds   for by this function and those listed above it.

 self      the number of seconds accounted for by this
seconds    function alone.  This is the major sort for this
           listing.

calls      the number of times this function was invoked, if
           this function is profiled, else blank.

 self      the average number of milliseconds spent in this
ms/call    function per call, if this function is profiled,
       else blank.

 total     the average number of milliseconds spent in this
ms/call    function and its descendents per call, if this
       function is profiled, else blank.

name       the name of the function.  This is the minor sort
           for this listing. The index shows the location of
       the function in the gprof listing. If the index is
       in parenthesis it shows where it would appear in
       the gprof listing if it were to be printed.
FF
Copyright (C) 2012-2015 Free Software Foundation, Inc.

Copying and distribution of this file, with or without modification,
are permitted in any medium without royalty provided the copyright
notice and this notice are preserved.
```

Figure 7

The following profile report is created for objective one. The total time is 100.10 seconds to run the code and display results.

```
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls  Ts/call  Ts/call  name
100.10      0.48     0.48                             main

 %         the percentage of the total running time of the
time       program used by this function.

cumulative a running sum of the number of seconds accounted
 seconds   for by this function and those listed above it.

 self      the number of seconds accounted for by this
seconds    function alone.  This is the major sort for this
           listing.

calls      the number of times this function was invoked, if
           this function is profiled, else blank.

 self      the average number of milliseconds spent in this
ms/call    function per call, if this function is profiled,
       else blank.

 total     the average number of milliseconds spent in this
ms/call    function and its descendents per call, if this
       function is profiled, else blank.

name       the name of the function.  This is the minor sort
           for this listing. The index shows the location of
       the function in the gprof listing. If the index is
       in parenthesis it shows where it would appear in
       the gprof listing if it were to be printed.
^L
Copyright (C) 2012-2015 Free Software Foundation, Inc.

Copying and distribution of this file, with or without modification,
are permitted in any medium without royalty provided the copyright
notice and this notice are preserved.
```

Figure 8

The following profile report is created for objective two. The total time is 100.10 seconds to run the code.