Jon Eble

Thomas Morrison

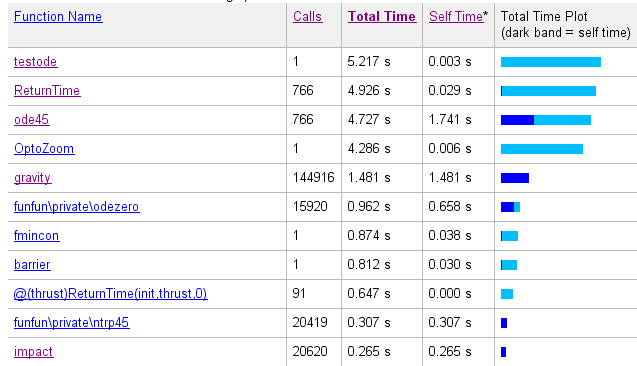
**Code Description and Design**

The script begins by defining all the initial conditions for the simulation. Once this is done, a grid of values is created corresponding to the magnitudes () and directions (θ) of possible velocity changes in the given range (0 < < 100 and 0 < θ < 360). Every value in this grid is then numerically integrated in a separate instance of ode45 in order to find the point on the grid that produces the fastest return time. Once this point has been found, a new grid of velocity values is created that is zoomed in on this point by a user-defined zooming factor. This process of finding the minimum and zooming can be repeated as many times as desired, though the values usually converge after 3-4 iterations. The final step is to take the output of the zooming algorithm (the velocity change that produces the minimum return time) and use this as the initial guess in MATLAB’s fmincon function which will output the final answer.

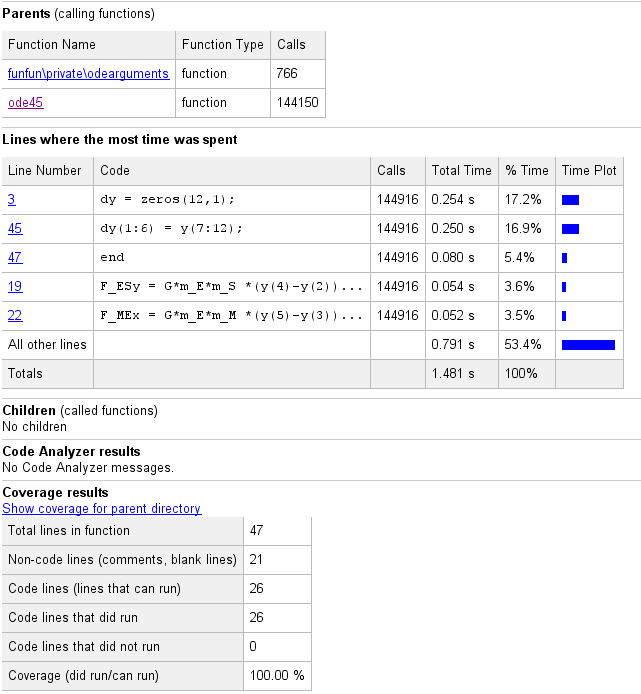
The reason that the zooming method was chosen for this problem instead of simply increasing the resolution of the grid is that computation time increases linearly with number of zooms but quadratically with the size of the grid. For example, if we have a 10x10 grid of values and wish to double the resolution of our search, changing the grid to 20x20 increases the number of calculations from 100 to 400. However, if we use the zooming algorithm and with a zoom factor of 2 (that is, the range of each axis is cut in half), then we need only compute two 10x10 grids which requires 200 calculations. This means that the zooming algorithm has achieved the same resolution as the brute force method but with half the calculations. The fmincon function was used as one final way to refine the search for the minimum value.

**Profiler Results**

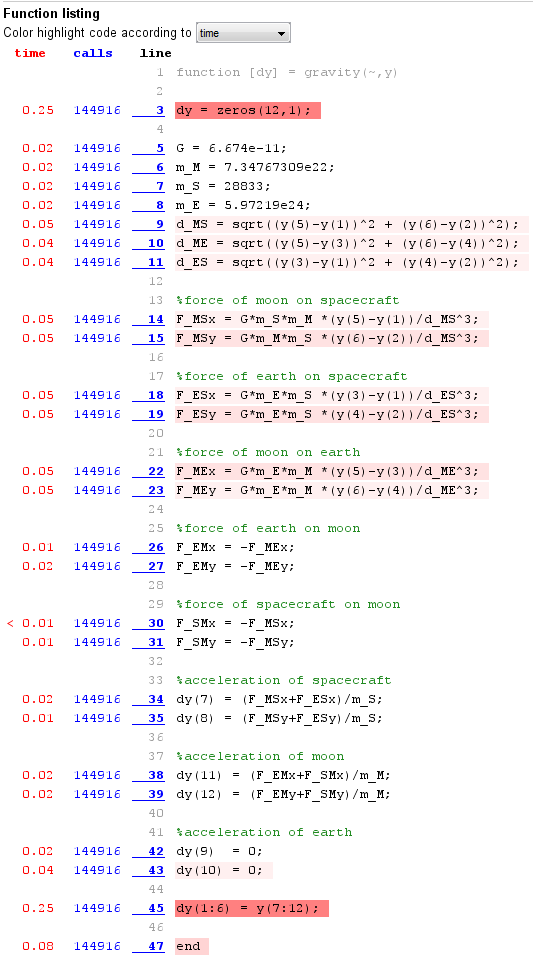
The overall profiler results show that the two most expensive functions are “ode45” and “gravity” because they have the most self-time. However, ode45 is a function provided by MATLAB, so no changes can be made to it. Instead, we can vary the number of times that ode45 is called. This can be accomplished by finding a way to decrease the range of the initial grid of values, which would allow the same accuracy to be achieved using a smaller resolution.



**Overall Profiler Results**

The profiler results for the gravity function shows that the two most expensive lines are where the output vector is initialized and where the velocity is copied from the input vector to the output. However, both of these lines are necessary and not much can be done to make them faster. This leaves us with the option of simplifying the calculations. For example, there is no need to calculate the force that the spacecraft exerts on the moon or earth because it is negligible. Similarly, the force that the moon exerts on the earth can be ignored because the earth is assumed to be stationary. Also, the constant values could be passed into the function rather than being declared each time it runs.

**“gravity” profiler results**



**“gravity” profiler results part 2**