Gravitational N-body Simulation in MATLAB

Alexander Johansson
Jarl Gullberg
Karl Brorsson

The Masters Programme in Spaceflight Engineering
Lule University of Technology
SE-971 87 Lule, Sweden

October 20, 2016

Abstract

N-body simulations are a classic computational problem for physicists and programmers. The simulation of an arbitrary number of bodies in a gravitational system is exponentially more taxing for the computational hardware it runs on as the number of bodies grow, and the accuracy of the simulation suffers if the simulation stepping is too low.

This report describes an implementation of a simple n-body simulation on a 2D plane written in MATLAB, which implements some common optimizations of realtime simulations. Additionally, the simulation exploits the object-oriented capabilites of MATLAB to improve code organization and implementation clarity.

Contents

1 Preface **3**

2 Introduction **3**

3 Theory **3**

3.1 Logic Flow 3

3.2 Gravitational Theory 4

4 Method **4**

4.1 Program Structure 4

4.2 Gravitational Computation 4

5 Results **4**

6 Discussion **4**

1 Preface

2 Introduction

3 Theory

The basic principle of a simulation is rather simple - a set of logical rules which determine how the simulation will progress, and a set of mathematical theories (in the form of implemented formulae) which serve to explore the intended science of the simulation.

Each works in tandem to, over time, create a simulation of available data as reasonable close to the real world as possible.

3.1 Logic Flow

In every turing-complete programming language, there exists a set of conditional and logical constructs which can be used to direct the path of the program through the source code (colloquially referred to as *control flow statements*).

MATLAB, which falls inside this set of languages, defines every required statement. A complete study of each is, however, beyond the scope of this report, and it is sufficient to mention their use.

For almost every simulation, the application of these statements and constructs follows a very similar structure. There is a main *loop* which drives the simulation forward one batch of calculations at a time, and a set of control flow statements inside it which determine what specific calculations are made.

The shell of a simulation might look something like this:

```
stopCondition = false;
while(!stopCondition)
    computationFinished = doComputation();

    if (computationFinished)
        stopCondition = true;
    end
end
```

This simulation will run indefinitely until the computation (whatever it might be) is finished, as indicated by the result of the `doComputation` function, which in turn flips the `stopCondition` and terminates the simulation.

This is, of course, a simplifed example with exaggarated control flow for clarity.

3.2 Gravitational Theory

$G = 6.67408^{-11}$
$d = \sqrt{(other.X - this.X)^2 + (other.Y - this.Y)^2}$
$m = this.Radius^2 \cdot \pi \cdot 5.5$
$F = G \cdot \frac{this.m \cdot other.m}{d^2}$

4 Method

It was clear from the beginning that, within the confines of the excercise, the bulk of the simulation would be done in a series of conditional loops wherein each object calculated the effect of each other body upon itself.

4.1 Program Structure

4.2 Gravitational Computation

5 Results

6 Discussion