STABLEORBIT

COLLIN SCHROEDER

1. Physics

1.1. **Force.** Given Newton's 2nd law we can write down the forcing equation for the ith particle in our system.

$$\sum_{n=0}^{N} \vec{F}_n = m_i \vec{a_i} \tag{1}$$

Using Newton's law of gravity we can rewrite this as

$$\sum_{n=0}^{N} \frac{G m_i m_j}{\vec{r}^2} = m_i \vec{a_i} \tag{2}$$

It follows that to compute the force for every particle in the system we must do the following.

$$\sum_{i=0}^{N} \sum_{j=0, j\neq i}^{N} \frac{G m_i m_j}{\vec{r}^2} = m_i \vec{a_i} \tag{3}$$

However, since these forces are third law partners we may greatly reduce our computation by incoorporating this into the algorithm. The sum becomes

$$\sum_{i=0}^{N} \sum_{j=0}^{i} \tag{4}$$

Or in pseudo code

```
for i in range(0,N):
    for j in range(0,i):
        a = computeAccel(body[i],body[j])
        bodyAccel[i] += a
        bodyAccel[j] -= a
```

1.2. **Integration.** Having computed the acceleration due to gravity on all the particles we simply have to update our velocity and position. We do this by using the Leapfrog Euler algorithm, a well developed numerical method that is excellent for N-Body integration. Solving differential equations numerically we first define a time-step dt, the larger this is the faster our simulation proceeds, however, error is also introduced as dt becomes large. A typical way of observing if a numerical algorithm is to remain stable is to compute the systems total energy and ensure that it remains at a constant value within some range δ. To start the Leapfrog algorithm we compute the acceleration and then update the velocities of the particles with a half step through time.

$$velocity(t_1) = velocity(t_o) + acceleration(t_o)\frac{dt}{2} \tag{5}$$

Next, we update the position

$$position(t_1) = position(t_o) + v(t_1)\ dt \tag{6}$$

After the first step the algorithm proceeds as

$$velocity(t_{n+1}) = velocity(t_n) + acceleration(t_n)\ dt \tag{7}$$

$$position(t_{n+1}) = position(t_n) + velocity(t_n)\ dt \tag{8}$$

The first step in this algorithm introduces a method to deal with error that is not incorporated into the typical Euler method. This algorithm proves to work well at conserving energy for long periods of time on the order of 10^9 years.