2. The update rules are very simple in this model. Extend your code so that cars move forward:

   - with a probability of 0.75 if they moved last iteration
   - with a probability of 0.5 if they did not

   Does this affect the flow of traffic at a given density ?

3. Bad driving. If you initiate an 'event' on the road, for example an obstacle that moves forward with a very low probability or stays fixed for a few iterations before it disappears, you should be able to see the resulting queue of traffic ripple backwards along the road like a wave and get longer. ← NOTE

   In real life, this 'event' may correspond to someone changing lanes without indicating causing the driver behind to brake suddenly. Quite often, slow-moving or even stationary traffic can form from one of these events.

4.5 Relevant course sections

Studying the following course sections will help you complete this checkpoint:

- Plotting using Matplotlib
- Custom Plotting

Additional material that you may find useful:

- Animation using Matplotlib

4.6 Marking Scheme

- Traffic Checkpoint Marking Scheme

5 Orbital Motion

5.1 Aim

The aim of this checkpoint is to simulate the path of the moon Phobos orbiting the planet Mars using numerical integration.

5.2 Numerical integration

Newton's second law $\vec{F}(t) = m\vec{a}(t)$ can be written as two first order differential equations:

$$\vec{v}(t) = \frac{d\vec{r}(t)}{dt} \qquad \text{and} \qquad \vec{a}(t) = \frac{d\vec{v}(t)}{dt} = \frac{1}{m}\vec{F}(t)$$

If we know the position and velocity of a body at some initial time and the form of the force acting on it, we can use numerical integration to simulate the body's future motion.

There are a variety of numerical integration schemes (such as the Euler and Runge Kutta methods), but we will use one of the simplest techniques known as the Euler-Cromer method.

The Euler-Cromer algorithm is:

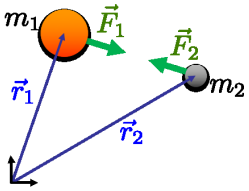
$$\begin{aligned} \vec{v}(t+\Delta t) &= \vec{v}(t) + \vec{a}(t)\Delta t \\ \vec{r}(t+\Delta t) &= \vec{r}(t) + \vec{v}(t+\Delta t)\Delta t \end{aligned}$$

where t is the current time and Δt is the small time-step.

This is a *symplectic* integrator, meaning that the total energy is preserved (although there are oscillations around the average value).

5.3 Gravitational Force Law

The gravitational force law determines the acceleration of heavenly bodies and so determines the motion of the planets and their moons (as well as the sun and its planets). Consider a 2-body system consisting of the planet Mars, of mass m_1 located at position $\vec{r}_1(t)$, and its moon Phobos, of mass m_2 located at $\vec{r}_2(t)$, as shown below.



The gravitational force $\vec{F}_2(t)$ exerted on Phobos by Mars by is:

$$\vec{F}_2(t) = G\frac{m_1 m_2}{|\vec{r}_{21}(t)|^2}\hat{r}_{21}(t)$$

where $\vec{r}_{21} = \vec{r}_1(t) - \vec{r}_2(t)$ is the position of Phobos relative to Mars, $\hat{r}_{21}(t)$ is the unit vector from the moon to the planet and G is the gravitational constant.

The acceleration $\vec{a}_2(t)$ of Phobos due this force is:

$$\vec{a}_2(t) = \frac{\vec{F}_2(t)}{m_2}.$$

Similarly, the force $\vec{F}_1(t)$ exerted on Mars by Phobos is:

$$\vec{F}_1(t) = G\frac{m_2 m_1}{|\vec{r}_{12}(t)|^2}\hat{r}_{12}(t)$$

and the acceleration $\vec{a}_1(t)$ of Mars is:

$$\vec{a}_1(t) = \frac{\vec{F}_1(t)}{m_1}.$$

← MANY-BODY PROBLEM

5.4 Checkpoint tasks

Write a PYTHON program to simulate the orbit of Phobos (one of the moons of Mars) around the planet Mars using the numerical integration scheme given above.

Initially, you may want to treat Mars as fixed, so that you only need consider the force exerted by Mars on Phobos; this is a reasonable assumption as the mass of Phobos is much smaller than that of Mars. But once you have a working code you should treat this as a "two body problem" and include the force exerted by Phobos on Mars. You should also write your code in such a way that adding another moon would not involve changing the design of your code.

← CENTRAL POTENTIAL

Your code should read in the simulation parameters (including the number of time steps and the length of the time step) and details of Mars and Phobos (including the mass and orbital radius) from file.

← INITIAL POSITIONS AN

Your code should graphically represent the motion of Phobos as it orbits around Mars. At suitable intervals it should print out the total kinetic energy of the system to check whether it is conserved in your simulation.

← INPUT PARAMETERS: M

← TOTAL KINETIC ENERG

5.5 Code design

You will need to think very carefully about the structure of your code. There is no single correct design, but one suggestion is to use two classes; one to represent the celestial bodies (in this case, Mars and Phobos) and one for the simulation. You are, of course, free to use other designs, but whatever you choose try not to overcomplicate your code and in particular think very carefully about what each of your classes is responsble for, i.e. what data it should hold (variables) and what it should do (methods). Also think carefully about what data you need to keep, e.g. in lists or numpy arrays, in order to implement the checkpoint and don't keep unneccessary data "just in case" - this will almost certainly result in ovecomplicated code that is hard to read and debug.

5.6 Optional extras

1. Modify your simulation to calculate the orbital period of Phobos. Does this match the actual value?

2. There are two Martian moons: Phobos and Deimos (named after the ancient Greek gods of fear and terror respectively). Add Deimos to your simulation and check that it still runs correctly. ← INPUT PARAMETERS: D

5.7 Relevant course sections

Studying the following course sections will help you complete this checkpoint:

- Plotting using Matplotlib
- Custom Plotting
- Animation using Matplotlib

5.8 Marking Scheme

- Orbital Motion Checkpoint Marking Scheme

1 Solar

1.1 Introduction

In 1543 Copernicus proposed that the known planets (at that time only the planets as far out as Saturn) were in circular orbit around the sun. This theory was revised by Kepler and Brahe who showed that the orbits were in fact elliptical. Since then the orbits of the planets have been very accurately calculated and can be precisely predicted far into the future. NASA uses powerful computers to determine both the position of the planets and the path of its space missions, including the Mars landing probes.

This project will aim to simulate the motion of the planets in two-dimensions and to predict launch conditions for satellites to successfully reach planets.

1.2 Numerical integration

The solar system represents a gravitational "many body problem", i.e. one that cannot be solved analytically, which means that a numerical integration scheme must be used. In this project we will use the *three-step Beeman* scheme. The Beeman algorithm is a stable method which predicts the position at the next time step by combining the current acceleration with the acceleration from the previous time step. This new position can then be used to calculate the new acceleration which, in turn, predicts the new velocity. The algorithm is given by:

$$\vec{r}(t+\Delta t) = \vec{r}(t) + \vec{v}(t)\Delta t + \frac{1}{6}[4\vec{a}(t) - \vec{a}(t-\Delta t)]\Delta t^2 \tag{1.1}$$