Chapter 17

Orbits

The n-body problem models planetary orbits.

The *n-body problem* in celestial mechanics is the study of a system of planets and the interaction described by Newton's laws of motion and gravitational attraction.

Over five hundred years ago, Johannes Kepler realized that if there are only two planets in the model, the orbits are ellipses with a common focus at the center of mass of the system. This provides a fair description of the moon's orbit around the earth, or of the earth's orbit around the sun. But if you are planning a trip to the moon or a mission to Mars, you need more accuracy. You have to realize that the sun affects the moon's orbit around the earth and that Jupiter affects the orbits of both the earth and Mars. Furthermore, if you wish to model more than two planets, an analytic solution to the equations of motion is not possible. It is necessary to compute numerical approximations.

Our notation uses vectors and arrays. Let n be the number of planets and, for $i = 1, \ldots, n$, let p_i be the vector denoting the position of the i-th planet. For two-dimensional motion the i-th position vector has components (x_i, y_i). For three-dimensional motion its components are (x_i, y_i, z_i). The small system shown in figure 17.1 illustrates this notation. There are three bodies moving in two dimensions. The coordinate system and units are chosen so that initially the first body, the gold planet, is at the origin,

$$p_1 = (0, 0)$$

The second body, the blue planet, is one astronomical unit, or a.u., away from the gold planet.

$$p_2 = (1, 0)$$

The third body, the red planet, is at position

$$p_3 = (2/3, 3/4)$$

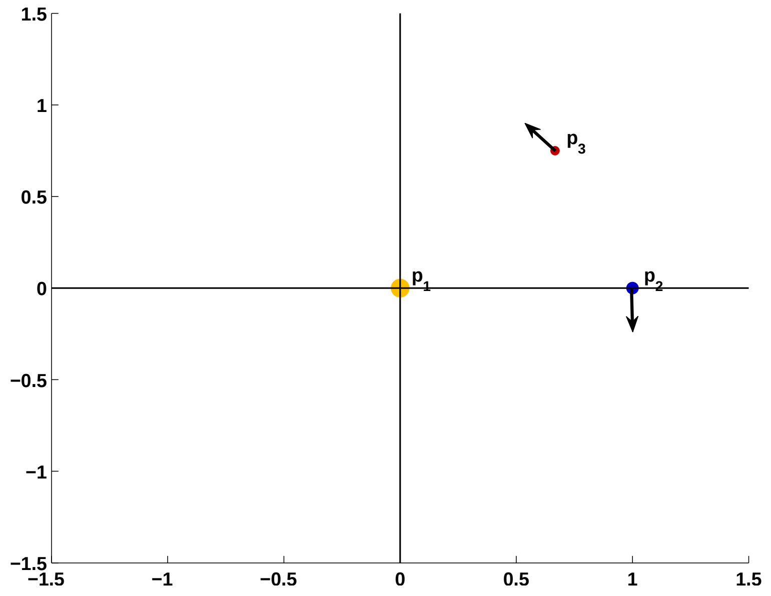

Figure 17.1. *Initial positions and velocities of a small system with three planets in two-dimensional space.*

We wish to model how the position vectors p_i vary with time, t. The *velocity* of an object is the rate of change of its position and the *acceleration* is the rate of change of its velocity. We use one and two dots over p_i to denote the velocity and acceleration vectors, $\dot{p}_i$ and $\ddot{p}_i$. If you are familiar with calculus, you realize that the dot means differentiation with respect to t. For our three body example, the gold planet is initially stationary in this coordinate system, so its velocity is a vector of zeros,

$$\dot{p}_1 = (0, 0)$$

The initial velocity of the other two planets are

$$\dot{p}_2 = (0, -1)$$

and

$$\dot{p}_3 = (-1/2, 1/2)$$

Newton's law of motion, the famous $F = ma$, says that the mass of a planet times its acceleration is proportional to the sum of the forces acting on it. Newton's law of gravitational says that the force between any two planets is proportional to the product of their masses and inversely proportional to the square of the distance between them. So, the equations of motion are

$$m_i \ddot{p}_i = \gamma \sum_{j \neq i} m_i m_j \frac{p_j - p_i}{||p_j - p_i||^3}, \quad i = 1, \ldots, n$$

Here γ is the gravitional constant, m_i is the mass of the i-th planet, $p_j - p_i$ is the vector from planet i to planet j and $||p_j - p_i||$ is the length or *norm* of that vector, which is the distance between the two planets. The denominator of the fraction involves the cube of the distance because the numerator contains the distance itself and so the resulting quotient involves the inverse of the square of the distance.

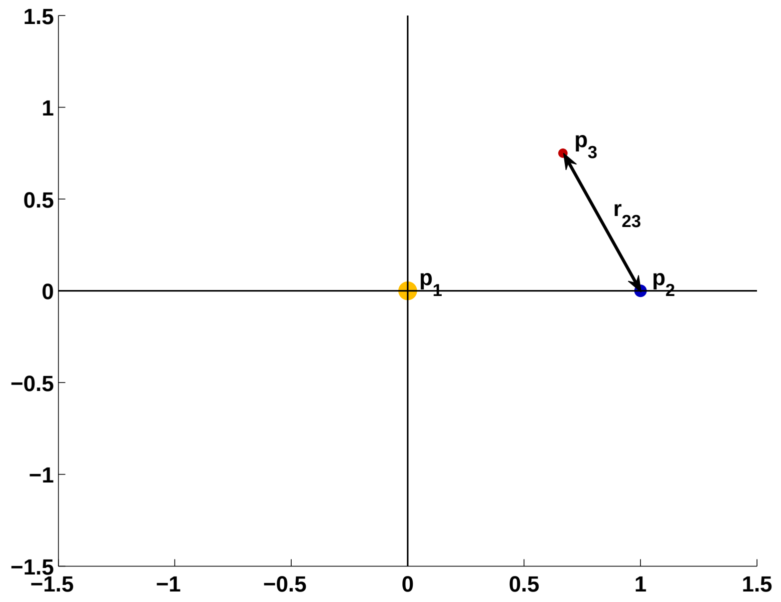

Figure 17.2. *The double arrow depicts the vectors* $r_{23} = p_3 - p_2$ *and* $-r_{32}$. *The length of this arrow is the distance between* p_2 *and* p_3.

Figure 17.2 shows our three planet example again. The length of the vector $r_{23} = p_3 - p_2$ is the distance between p_2 and p_3. The gravitation forces between planets two and three are directed along r_{23} and $-r_{23}$.

To summarize, the position of the i-th body is denoted by the vector p_i. The instaneous motion of this body is given by its velocity vector, denoted by $\dot{p}_i$. The instaneous change in the velocity is given by the acceleration vector, denoted by $\ddot{p}_i$. The acceleration is determined from the position and masses of all the bodies by Newton's laws of motion and gravitation.

The following notation simplifies the discussion of numerical methods. Stack the position vectors on top of each other to produce an n-by-d array where n is the number of planets and $d = 2$ or 3 is the number of spatial dimensions..

$$P = \begin{pmatrix} p_1 \\ p_2 \\ \vdots \\ p_n \end{pmatrix}$$

Let V denote a similar array of velocity vectors.

$$V = \begin{pmatrix} \dot{p}_1 \\ \dot{p}_2 \\ \vdots \\ \dot{p}_n \end{pmatrix}$$

And, let $G(P)$ denote the array of gravitation forces.

$$G(P) = \begin{pmatrix} g_1 \\ g_2 \\ \vdots \\ g_n \end{pmatrix}$$

where

$$g_i = \gamma \sum_{j \neq i} m_j \frac{p_j - p_i}{||p_j - p_i||^3}$$

With this notation, the equations of motion can be written

$$\begin{aligned} \dot{P} &= V \\ \dot{V} &= G(P) \end{aligned}$$

For our three body example, the initial values of P and V are

$$P = \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ 2/3 & 3/4 \end{pmatrix}$$

and

$$V = \begin{pmatrix} 0 & 0 \\ 0 & -1 \\ -1/2 & 1/2 \end{pmatrix}$$

The masses in our three planet example are

$$m_1 = 1/2, \quad m_2 = 1/3, \quad m_3 = 1/6$$

From these quantities, we can compute the initial value of the gravitation forces, $G(P)$.

We will illustrate our numerical methods by trying to generate a circle. The differential equations are

$$\begin{aligned}\dot{x} &= y \\ \dot{y} &= -x\end{aligned}$$

With initial conditions $x(0) = 0, y(0) = 1$, the exact solution is

$$x(t) = \sin t,\ \ y(t) = \cos t$$

The orbit is a perfect circle with a period equal to 2π.

The most elementary numerical method, which we will not use, is known as the *forward* or *explicit Euler* method. The method uses a fixed time step Δt and simultaneously advances both the positions and velocities from time t_k to time $t_{k+1} = t_k + \Delta t$.

$$\begin{aligned}P_{k+1} &= P_k + \Delta t\, V_k \\ V_{k+1} &= V_k + \Delta t\, G(P_k)\end{aligned}$$

The forward Euler's method applied to the circle generator problem becomes

$$\begin{aligned}x_{k+1} &= x_k + \Delta t\, y_k \\ y_{k+1} &= y_k - \Delta t\, x_k\end{aligned}$$

The result for $\Delta t = 2\pi/30$ is shown in the first plot in figure 17.3. Instead of a circle we get a growing spiral. The method is unstable and consequently unsatisfactory, particularly for long time periods. Smaller time steps merely delay the inevitable. We would see more complicated, but similar, behavior with the n-body equations.

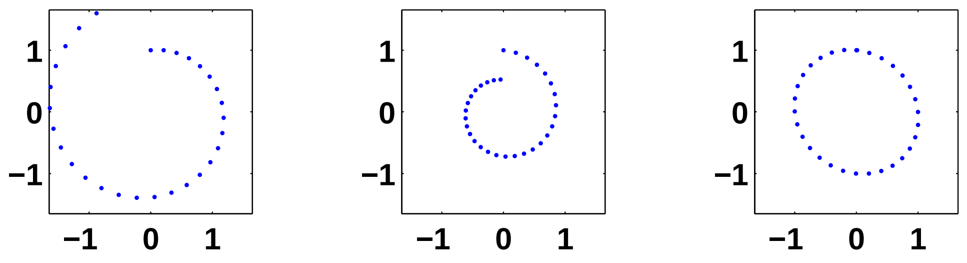

Figure 17.3. *Three versions of Euler's method for generating a circle. The first plot shows that the forward method is unstable. The second plot shows that the backward method has excessive damping. The third plot shows that symplectic method, which is a compromise between the first two methods, produces a nearly perfect circle.*

Another elementary numerical method is known as the *backward* or *implicit Euler* method. In general, it involves somehow solving a nonlinear system at each step.

$$\begin{aligned}P_{k+1} - \Delta t\, V_{k+1} &= P_k \\ V_{k+1} - \Delta t\, G(P_{k+1}) &= V_k\end{aligned}$$

Our simple circle example is linear, so x_{k+1} and y_{k+1} are easily computed by solving the 2-by-2 linear system

$$\begin{aligned} x_{k+1} - \Delta t\, y_{k+1} &= x_k \\ y_{k+1} + \Delta t\, x_{k+1} &= y_k \end{aligned}$$

The result is shown in the second plot in figure 17.3. Instead of a circle we get a decaying spiral. The method is stable, but there is too much damping. Again, we would see similar behavior with the n-body equations.

The method that we actually use is a compromise between the explicit and implicit Euler methods. It is the most elementary instance of what are known as *symplectic* methods. The method involves two half-steps. In the first half-step, the positions at time t_k are used in the gravitation equations to update of the velocities.

$$V_{k+1} = V_k + \Delta t\, G(P_k)$$

Then, in the second half-step, these "new" velocities are used to update the positions.

$$P_{k+1} = P_k + \Delta t\, V_{k+1}$$

The novel feature of this symplectic method is the subscript $k+1$ instead of k on the V term is the second half-step.

For the circle generator, the symplectic method is

$$\begin{aligned} x_{k+1} &= x_k + \Delta t\, y_k \\ y_{k+1} &= y_k - \Delta t\, x_{k+1} \end{aligned}$$

The result is the third plot in figure 17.3. If you look carefully, you can see that the orbit in not quite a circle. It's actually a nearly circular ellipse. And the final value does not quite return to the initial value, so the period is not exactly 2π . But the important fact is that the orbit is neither a growing nor a decaying spiral.

There are more complicated symplectic algorithms that are much more accurate per step than this symplectic Euler. But the symplectic Euler is satisfactory for generating well behaved graphical displays. Most well-known numerical methods, including Runge-Kutta methods and traditional multistep methods, do not have this symplectic stability property and, as a result, are not as satisfactory for computing orbits over long time spans.

Figure 17.4 shows the first few steps for our example system. As we noted earlier, the initial position and velocity are

```
P =
          0         0
     1.0000         0
     0.6667    0.7500

V =
          0         0
          0   -1.0000
    -0.5000    0.5000
```

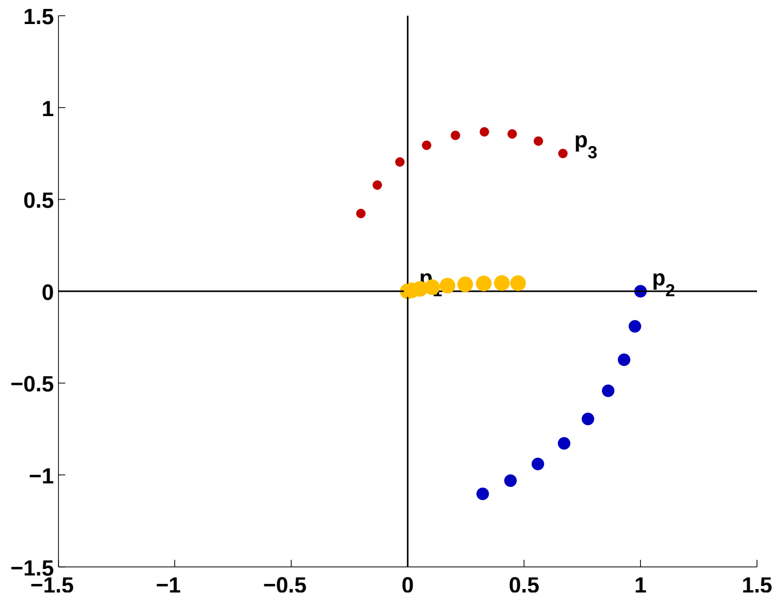

Figure 17.4. *The first few steps of our example system.*

After one step with $\Delta t = 0.20$ we obtain the following values.

```
P =
    0.0177    0.0049
    0.9760   -0.1910
    0.5615    0.8171

V =
    0.0887    0.0247
   -0.1201   -0.9548
   -0.5258    0.3353
```

To get an idea of how this works, note that the first planet accelerates towards the other two. In one step, its position changes from $(0, 0)$ to small positive values, $(0.0177, 0.0049)$. The second planet is initially at position $(1, 0)$ with velocity $(0, -1)$ in the negative y direction. In one step, its position changes to $(0.9760, -0.1910)$. The x-coordinate has changed relatively little, while the y-coorinate has changed by roughly $-\Delta t = -0.2000$. The third planet moves in the direction indicated by the velocity vector in figure 17.1.

After a second step we have the following values. As expected, all the trends noted in the first step continue.

```
P =
    0.0530    0.0129
```

```
       0.9293   -0.3725
       0.4490    0.8564

   V =
       0.1764    0.0398
      -0.2332   -0.9079
      -0.5627    0.1964
```

It turns out that since our small system lacks a massive central sun, it behaves more like three unruly asteroids than three actual planets. Figure 17.5 shows an initial section of the trajectories.

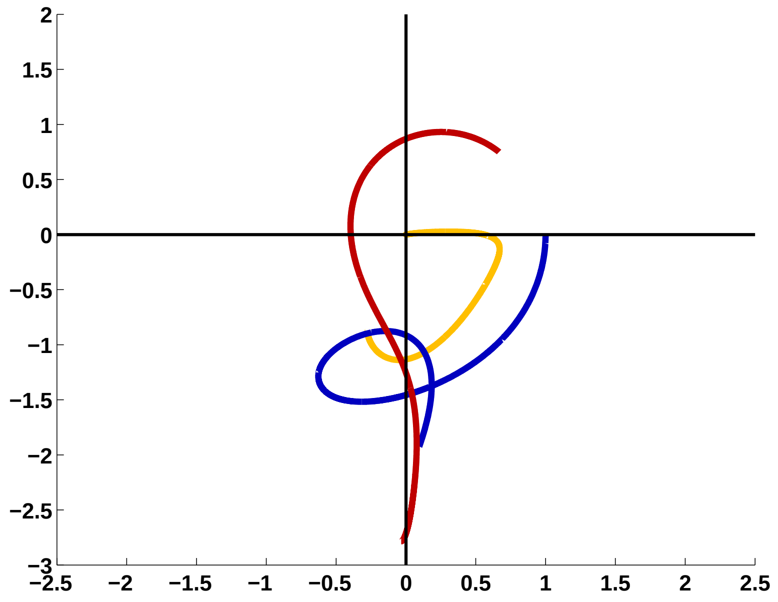

Figure 17.5. *The initial trajectories of our example system.*

Figure 17.6 shows the orbits for the four outer planets in the solar system. The orbits are actually to scale with respect to each other. But, obviously, the symbol for the sun in the center does not have the same scale. Our *Experiments* program `solar9` models nine objects in the solar system, namely the sun and eight planets. Web sources for information about the solar system are provided by the University Corporation for Atmospheric Research, the Jet Propulsion Laboratory, and the US National Air and Space Museum,

```
http://www.windows.ucar.edu
http://www.jpl.nasa.gov/solar_system
http://airandspace.si.edu:80/etp/ss/index.htm
```

Here is the core portion of our *Experiments* program. The subfunction `initialize_orbits`

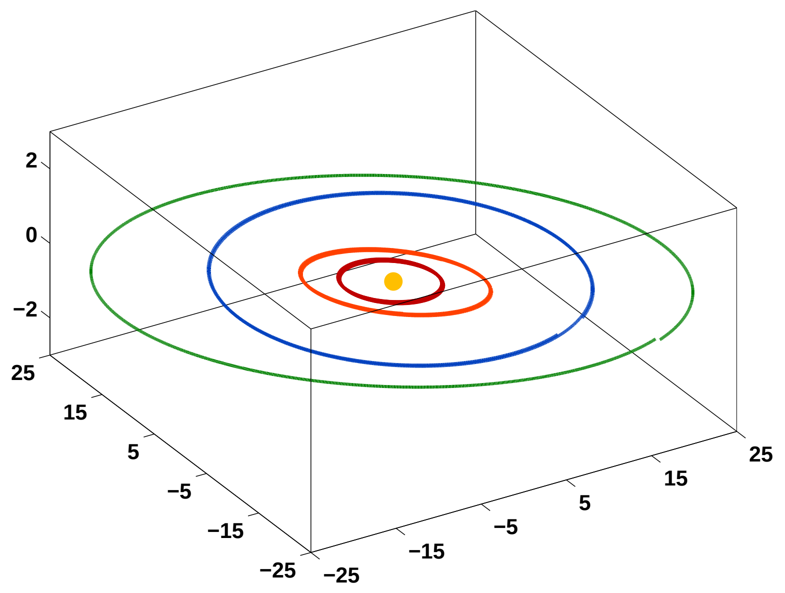

Figure 17.6. *Orbits of Jupiter, Saturn, Uranus, and Neptune.*

provides initial positions and velocities and thereby specifies the number of bodies, n, and the number of spacial dimensions, d. The other subfunctions control the plot and the graphics user interface. An exercise asks you to explain how the doubly nested `for` loops evaluate the n sums involved in Newton's law of gravitation.

```
function nbody
% NBODY  Motion of n bodies under mutual gravitional attraction.
% P = n-by-d array of position coordinates, d = 2 or 3.
% V = n-by-d array of velocities.
% M = n-by-1 array of masses.

   [P,V,M] = initialize_orbits;
   [H,stop,speed] = initialize_graphics(P);
   n = size(P,1);

   while ~get(stop,'value')
      % Advance velocities using current positions
      dt = get(speed,'value');
      for i = 1:n
         for j = i+1:n
            r = P(j,:) - P(i,:);
            g = dt*r/norm(r)^3;
            V(i,:) = V(i,:) + M(j)*g;
            V(j,:) = V(j,:) - M(i)*g;
         end
      end

      % Advance positions using advanced velocities
      P = P + dt*V;
      update_plot(H,P);
   end

   finalize_graphics(stop)
end
```

Exercises

17.1 *Newton's law.* Explain how the doubly nested `for` loops in `nbody` or `solar9` evaluate the n sums involved in Newton's law of gravitation.

17.2 *Pluto and Ceres.* Change `solar9` to `solar11` by adding the erstwhile planet Pluto and the recently promoted dwarf planet Ceres. See Wikipedia:

```
http://en.wikipedia.org/wiki/Planet
http://en.wikipedia.org/wiki/Ceres_(dwarf_planet)
```

and

```
http://orbitsimulator.com/gravity/articles/ceres.html
```

17.3 *Comet.* Add a comet to `solar9`. Find initial conditions so that the comet has a stable, but highly elliptical orbit that extends well beyond the orbits of the planets.

17.4 *Twin suns.* Turn the sun into a twin star system, with two suns orbiting each other out of the plane of the planets. What eventually happens to the planetary orbits? For example, try

```
sun1.p = [1 0 0];
sun1.v = [0 0.25 0.25];
sun1.m = 0.5;
sun2.p = [-1 0 0];
sun2.v = [0 -0.25 -0.25];
sun2.m = 0.5;
```

Try other values as well.