Differential Equations

Differential computation of functions is an example of solving a *differential equation*. In this case, we do this because it is faster than direct computation from the formula.

We often want to go the other way — we start with the differential equation, and want to find the formula that solves it, or at least be able to compute the numerical solution.

Some phenomena that can be modelled using differential equations:

- Changes in the populations of predators and prey over time.
- Changes in the economy, that result in changes in employment, investment, etc.
- Most of physics: eg, the orbit of a satellite under the influence of gravity.

A Simple Example:
Exponential Growth of a Population

Growth of organisms in an environment with plenty of resources can be described by the following differential equation:

$$\frac{dx(t)}{dt} = ax(t)$$

Here, t is time, and $x(t)$ is the population at time t. The constant a represents how rapidly the population grows.

This differential equation can be solved symbolically. Maple knows how, at least if we plug in a specific number for a (here $a = 3$):

```
> dsolve( { x(0) = 5, diff(x(t),t) = 3*x(t) }, { x(t) } );

                                 x(t) = 5 exp(3 t)
```

Here, $x(0) = 5$ specifies the initial population.

How believable is this solution as a model of growth in real populations?

A More Complex Example: Modeling an Orbit Around a Planet

People have been trying to build mathematical models for the motions of the planets and moons since ancient times.

Johannes Kepler found a good *descriptive* model for planetary motion. Isaac Newton explained how this model could be derived from his laws of motion and law of gravity.

We'll look at using Newton's laws to model the motion of an object orbiting a planet. We assume that we know:

- The mass of the planet.
- The initial position of the object.
- The initial velocity of the object.

We want our model to tell us the path of the object for some future period of time.

The Mathematical Model

In our model, the velocity of the object determines how its position changes with time. In Cartesian coordinates:

$$\frac{dp_x}{dt} = v_x, \quad \frac{dp_y}{dt} = v_y$$

Here, t is the time, p_x and p_y are the position coordinates, and v_x and v_y give the velocity.

Change in velocity depends on acceleration:

$$\frac{dv_x}{dt} = a_x, \quad \frac{dv_y}{dt} = a_y$$

Finally, the acceleration comes from the gravitational force of the planet (assumed to be at the origin). By using $F = ma$ and $F = GMm/d^2$, where $d = \sqrt{p_x^2 + p_y^2}$, and remembering that the force is toward the planet, we get

$$a_x = -\frac{GM\,(p_x/d)}{d^2}, \quad a_y = -\frac{GM\,(p_y/d)}{d^2}$$

Simplifications in the Model

Is this a good model? A bit of thought shows that it's not a *perfect* model of how a real object orbits the Earth (say).

- It ignores all the other objects in the universe. They all have some gravitational (or other) effect.
- It assumes that the planet is stationary — not a good assumption if the orbiting object is as massive as the planet itself.
- It assumes that Newton's law of gravity is correct. Einstein showed that it's only an approximation.
- It assumes that the object is rigid. Objects that can deform will absorb a small amount of energy as they orbit.

Is it good enough? That depends...

Solving the Model Numerically

This system of differential equatons was solved symbolically by Isaac Newton over 300 years ago, but it's too difficult for Maple to do.

If it's also too difficult for us, we might try to solve it numerically for a particular case.

We have to provide specific numbers for the initial position and velocity of the object, as well as the mass of the planet.

We don't get a formula in the end — just a list of numbers giving the x and y coordinates of the object at various times in the future. These numbers will be *approximations* to the true answer.

This seems much inferior to a symbolic solution. The big plus is that we can almost always try to find a numeric solution (though sometimes our attempt may not work well).

A Numerical Solution in Maple

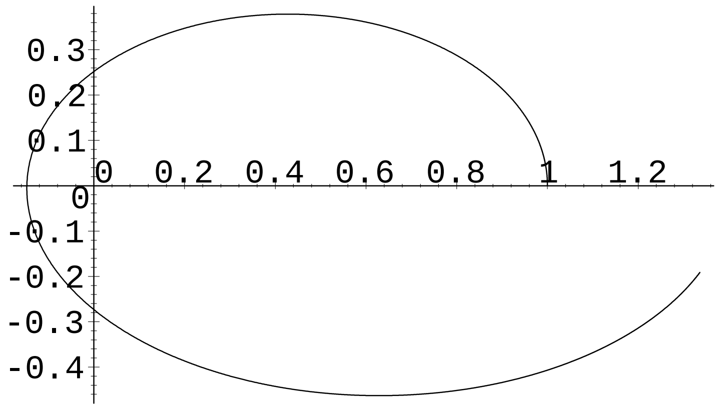
```
numerical_orbit := proc (GM, start_xpos, start_ypos,
                         start_xvel, start_yvel,
                         step, n_steps)

  local xpos, ypos, xvel, yvel, xaccel, yaccel,
        distance, t, i;

  xpos := array(0..n_steps);
  ypos := array(0..n_steps);
  xvel := array(0..n_steps);
  yvel := array(0..n_steps);

  xpos[0] := evalf(start_xpos);
  ypos[0] := evalf(start_ypos);
  xvel[0] := evalf(start_xvel);
  yvel[0] := evalf(start_yvel);

  for t from 1 to n_steps do

    distance := sqrt(xpos[t-1]^2 + ypos[t-1]^2);

    xaccel := - xpos[t-1] * GM / distance^3;
    yaccel := - ypos[t-1] * GM / distance^3;

    xvel[t] := xvel[t-1] + step*xaccel;
    yvel[t] := yvel[t-1] + step*yaccel;

    xpos[t] := xpos[t-1] + step*xvel[t-1];
    ypos[t] := ypos[t-1] + step*yvel[t-1];

  od;

  [ [ xpos[i], ypos[i] ] $ i=0..n_steps ]

end:
```

Disappointing Results

If we have this procedure in the file `orbit.mp` we can try it out as follows:

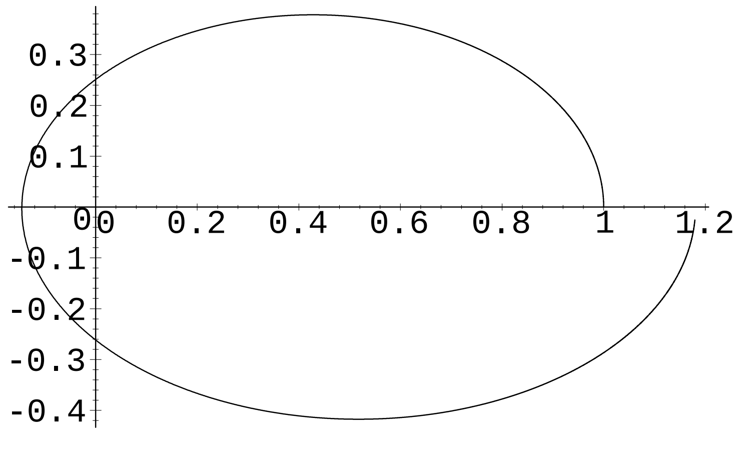
```
   > read ‘orbit.mp‘;
   > n1a := numerical_orbit(1,1,0,0,0.5,0.001,3000):
   > plot(n1a,scaling=constrained);
```

The result looks like this:



Not very good! Try again with a smaller step:

```
> n1b:=numerical_orbit(1,1,0,0,0.5,0.0005,6000):
> plot(n1b,scaling=constrained);
```



What's the Problem?

We have found that the numerical solution we get is very inaccurate.

In this type of problem, inaccuracy could be due to any of:

1. An inadequate mathematical model.
2. An inaccurate approximation to the model — here in terms of little time steps.
3. The approximate representation of numbers in the computer.

In this case, (2) seems most likely.

We could overcome this problem by using a very small time step — but this might take a very long time.

Instead, we might try to think of a better way of doing the approximation.

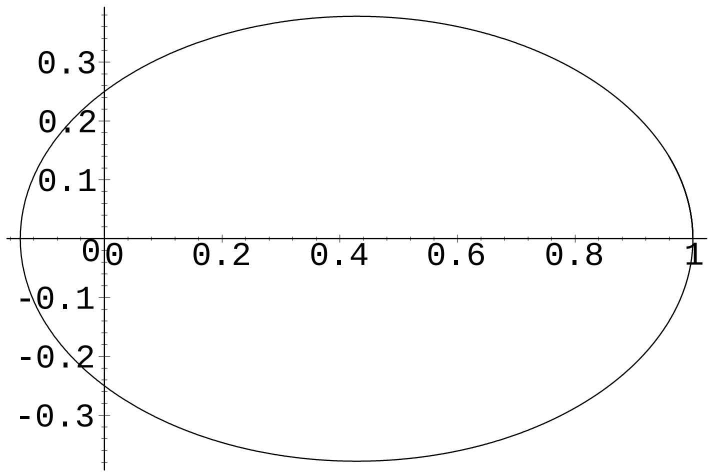
A Better Numerical Solution

```
numerical_orbit_2 := proc (GM, start_xpos, start_ypos,
                           start_xvel, start_yvel,
                           step, n_steps)

  local xpos, ypos, xvel, yvel, xaccel, yaccel,
        distance, t, i;

  xpos := array(0..n_steps);
  ypos := array(0..n_steps);
  xvel := array(0..n_steps);
  yvel := array(0..n_steps);

  xpos[0] := evalf(start_xpos);
  ypos[0] := evalf(start_ypos);
  xvel[0] := evalf(start_xvel);
  yvel[0] := evalf(start_yvel);

  for t from 1 to n_steps do

    distance := sqrt(xpos[t-1]^2 + ypos[t-1]^2);

    xaccel := - xpos[t-1] * GM / distance^3;
    yaccel := - ypos[t-1] * GM / distance^3;

    xvel[t] := xvel[t-1] + step*xaccel;
    yvel[t] := yvel[t-1] + step*yaccel;

    xpos[t] := xpos[t-1] + step*xvel[t];
    ypos[t] := ypos[t-1] + step*yvel[t];

  od;

  [ [ xpos[i], ypos[i] ] $ i=0..n_steps ]

end:
```

Better Results

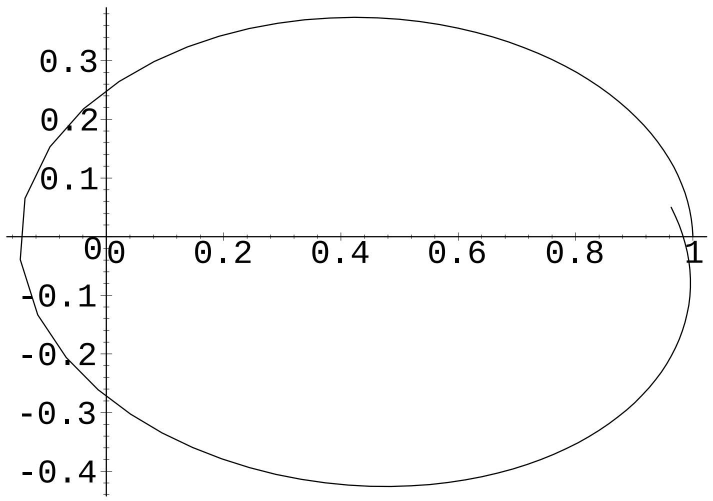
Results are much better now:

```
> n2a := numerical_orbit(1,1,0,0,0.5,0.001,3000):
> plot(n2a,scaling=constrained);
```



Encouraged, we can try taking bigger steps:

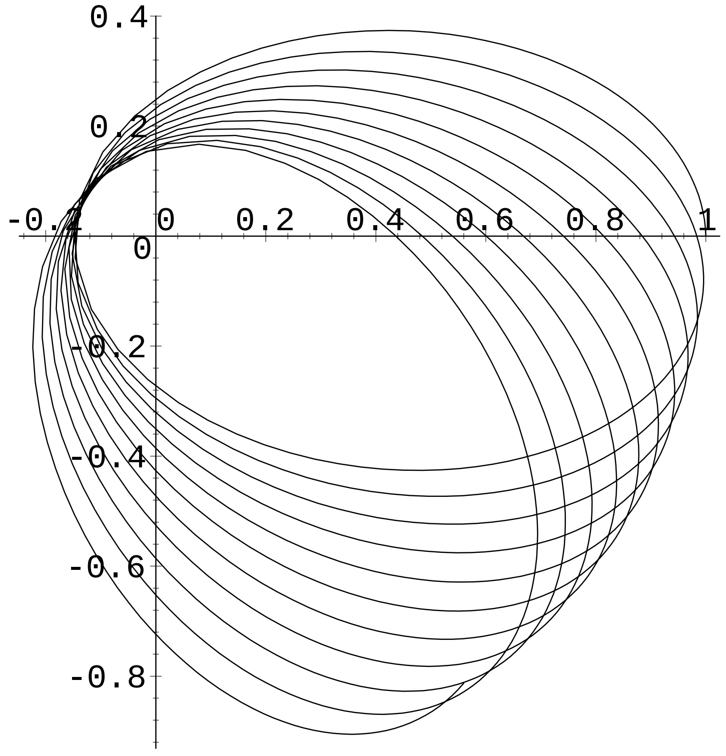
```
> n2b:=numerical_orbit_2(1,1,0,0,0.5,0.03,100):
> plot(n2b,scaling=constrained);
```



Looking Farther Into the Future

What happens if we try to compute the orbit for farther into the future?

```
> n2bb:=numerical_orbit_2(1,1,0,0,0.5,0.03,1000):
> plot(n2bb,scaling=constrained);
```



This isn't quite right — in the true answer (assuming the model is correct), the object retraces the same ellipse exactly.

On the other hand, it's not too bad. The object doesn't spiral into the planet, or wander off to distant space.