Ordinary differential equations

We talk about a system which is in a certain state. If the system has N degrees of freedom, the state is described by a vector $y_1, y_2, \dots, y_N$ of variables. The state will change in the course of time. In many cases the time development of the state of a system is adequately described by a system of ordinary differential equations,

$$\dot{y}_j = f_j(t, y_1, y_2, \dots, y_N) \quad . \tag{1}$$

Higher than first derivatives can always be removed. After all, the second derivative is the first derivative of the first derivative, and so forth.

Let us investigate one of the oldest physical problems, the motion of planets in the sun's gravitational field.

Recall that the conservation of angular momentum requires the planet to move in a plane. The planet's location $x_1(t), x_2(t)$ obey the following differential equation:

$$m\ddot{x}_j(t) = -\frac{GmM_\odot x_j}{(x_1^2 + x_2^2)^{3/2}} \quad . \tag{2}$$

G is the universal gravitational constant, m the planet's and $M_\odot$ the sun's mass. m drops out. By measuring x in units of $(GM_\odot)^{3/2}$ we arrive at

$$\ddot{x}_j(t) = -\frac{x_j}{(x_1^2 + x_2^2)^{3/2}} \quad , \tag{3}$$

Let us introduce $(y_1, y_2, y_3, y_4) = (x_1, \dot{x}_1, x_2, \dot{x}_2)$ such that

$$\begin{aligned} \dot{y}_1 &= y_2 && (4) \\ \dot{y}_2 &= -y_1/r^3 && (5) \\ \dot{y}_3 &= y_4 && (6) \\ \dot{y}_4 &= -y_3/r^3 && (7) \end{aligned}$$

results, where $r = \sqrt{y_1^2 + y_3^2}$.

We describe this set of four ordinary differential equations by the following derivative field:

```
% this file is kepler.m
function d=kepler(t,y);
r=sqrt(y(1).^2+y(3).^2);
d=[y(2);-y(1)./r^3;y(4);-y(3)./r^3];
```

At the command window we say

```
>> [t,y]=ode45(@kepler,[0:0.1:50],[1;0;0;0.8]);
```

A Runge-Kutta integration procedure `ode45` is invoked. It needs a derivative field, a time span, and a start vector. And this is the result of saying

```
>> plot(y(:,1),y(:,3))
>> print -deps2 kepler1.eps
```

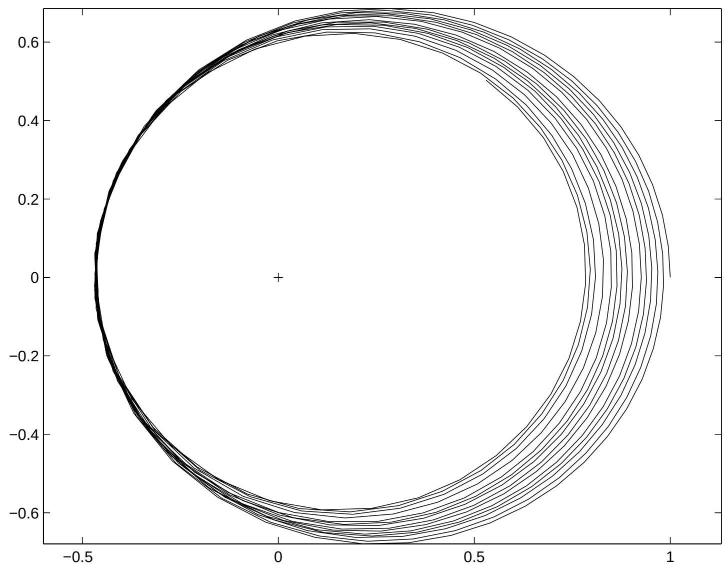

Figure 1: Planetary motion without explicite accuracy control.

Although the trajectories are ellipses, they shrink more and more. The reason is not physics, but numerics. We substantiate this remark by plotting the energy in Fig. 2

```
>> Ekin=0.5*(y(:,2).*y(:,2)+y(:,4).*y(:,4));
>> Epot=-1./sqrt(y(:,1).*y(:,1)+y(:,3).*y(:,3));
>> plot(t,Ekin+Epot);
```

We try better by setting a lower relative tolerance (the default being 0.001):

```
>> tol=odeset('RelTol',1e-8);
>> [t,y]=ode45(@kepler,[0:0.1:50],[1;0;0;0.8],tol);
```

Fig. 3 shows what we expect: closed ellipses.

There are many more options for the ordinary differential equation solver, and there are more such solvers. You should ask for details at the help desktop.

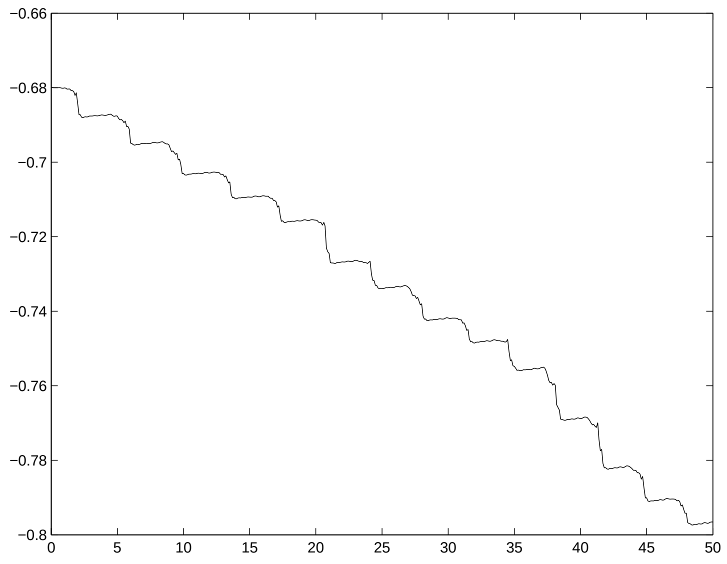

Figure 2: Total energy vs. time without accuracy control

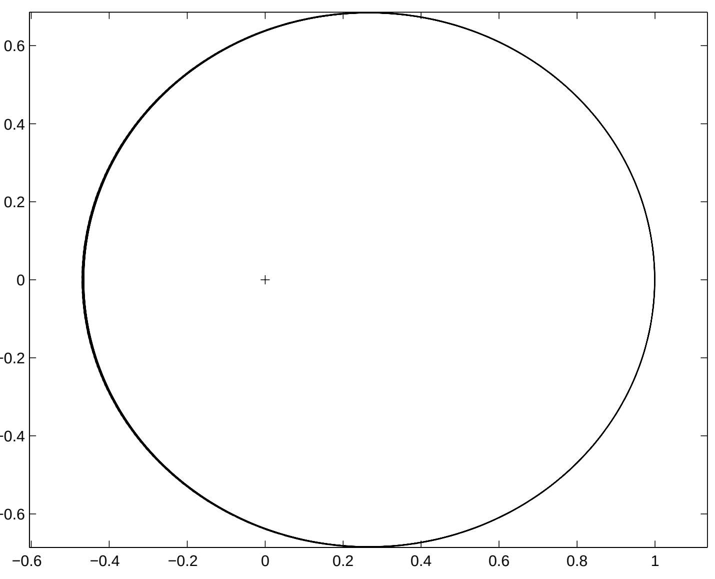

Figure 3: Planetary motion with accuracy control