Finite differences: ODEs

Problem 6.1

In analogy to Eq. (6.10), we evaluate

$$f[x+h,y(x)+hf(x,y(x))] = f(x,y(x)) + h\frac{d}{dh}f[x+h,y(x)+hf(x,y(x))]\big|_{h=0} + \mathcal{O}(h^2)$$
$$= f(x,y(x)) + h\left[\frac{\partial f}{\partial x} + f\frac{\partial f}{\partial y}\right] + \mathcal{O}(h^2)$$

Inserting this into Eq. (6.33), we obtain

$$y(x+h) = y(x) + hf(x,y(x)) + \frac{h^2}{2}\left[\frac{\partial f}{\partial x} + f\frac{\partial f}{\partial y}\right] + \mathcal{O}(h^3)$$

Thus, the Heun method is second-order accurate.

Problem 6.2

Firstly, for future reference, the exact solution to the harmonic oscillator equation

$$\ddot{y} + y = 0 \tag{0.1}$$

with initial conditions

$$y(0) = 1, \qquad \dot{y}(0) = 0 \tag{0.2}$$

is

$$y(t) = \cos t. \tag{0.3}$$

Defining $y_1 = y$ and $y_2 = \dot{y}$, the second-order ODE (0.1) can be written as a system of first-order ODEs

$$\dot{y}_1 = y_2, \qquad \dot{y}_2 = -y_1 \tag{0.4}$$

with initial conditions

$$y_1(0) = 1, \qquad y_2(0) = 0. \tag{0.5}$$

The interval on which we wish to obtain a numerical solution is $[0,T]$, with $T = 100\pi$. We shall divide this interval into N sub-intervals and define the mesh size $\Delta t = T/N$; further, we shall define

$$t_n = n\Delta t, \qquad n = 0,1,\ldots.,N. \tag{0.6}$$

Euler method

The Euler method can then be formulated recursively

$$\begin{aligned} y_1(t_{n+1}) &= y_1(t_n) + \Delta t \cdot y_2(t_n), \\ y_2(t_{n+1}) &= y_2(t_n) - \Delta t \cdot y_1(t_n) \end{aligned} \tag{0.7}$$

with starting conditions $y_1(t_0) = 1, y_2(t_0) = 0$ in accordance with Eq. (0.5).

We have implemented this method using the Matlab function *Euler* listed below.

```
function [y1,e,eT,eTd]=Euler(N,T)

dt=T/N;
y1=zeros(1,N+1); y2=zeros(1,N+1);    % pre-allocation

y1(1)=1; y2(1)=0;    % initial conditions

for n=1:N
    y1(n+1)=y1(n)+dt*y2(n);
    y2(n+1)=y2(n)-dt*y1(n);
end

t=(0:N)*dt;
Y1=cos(t); Y2=-sin(t);        % exact solutions
e=sum(abs(Y1-y1))/(N+1);      % average error at over the interval
eT=abs(Y1(N+1)-y1(N+1));      % error in the value at the last point
eTd=abs(Y2(N+1)-y2(N+1));     % error in the derivative at the last point
```

The function Euler uses as input the number of points N, and the output time T, and outputs the average error over the interval e and the errors in the value and derivative at the end point, e_T and $e_{T,d}$.

After some numerical experimentation, it became clear that the amount of memory required to get e below 0.001 for $T = 100\pi$ was in excess of 8×10^7, causing Matlab to throw "out of memory" errors on the computer used (which has 3 Gb RAM).

Therefore, we reduced the interval by a factor 10 to $T = 10\pi$. In order to find out how many points were necessary to attain and average error $e < 0.001$, we used a trial-and-error method employing a *while* loop which incremented N with various step sizes until the above condition was fulfilled.

```
T=10*pi;
N=157200;

[y,e,eT,eTd]=Euler(N,T);

while e>0.001
    N=N+1;
    [y,e,eT,eTd]=Euler(N,T);
end
```

The loop ended at $N_{e<0.001} = 157{,}245$. Hence, the required step size is $\Delta t = \frac{10\pi}{157{,}245} \approx 2.0 \times 10^{-4}$. For this same step size, the error at the last point was $e_T \approx 0.003$ and the error in the derivative at the last point $e_{T,d} \approx 4 \times 10^{-7}$.

Runge-Kutta method

Equations (0.4) can be written in the general form

$$\begin{aligned}\dot{y}_1 &= f(t, y_1, y_2) = y_2,\\ \dot{y}_2 &= g(t, y_1, y_2) = -y_1.\end{aligned} \tag{0.8}$$

The Runge-Kutta method for this system is as follows[1]. Defining

$$\begin{aligned}
k_1 &= \Delta t \cdot f\big(t_n, y_1(t_n), y_2(t_n)\big) = \Delta t \cdot y_2(t_n)\\
l_1 &= \Delta t \cdot g\big(t_n, y_1(t_n), y_2(t_n)\big) = -\Delta t \cdot y_1(t_n)\\
k_2 &= \Delta t \cdot f\left(t_n + \frac{\Delta t}{2}, y_1(t_n) + \frac{k_1}{2}, y_2(t_n) + \frac{l_1}{2}\right) = \Delta t\left(y_2(t_n) + \frac{l_1}{2}\right)\\
l_2 &= \Delta t \cdot g\left(t_n + \frac{\Delta t}{2}, y_1(t_n) + \frac{k_1}{2}, y_2(t_n) + \frac{l_1}{2}\right) = -\Delta t\left(y_1(t_n) + \frac{k_1}{2}\right)\\
k_3 &= \Delta t \cdot f\left(t_n + \frac{\Delta t}{2}, y_1(t_n) + \frac{k_2}{2}, y_2(t_n) + \frac{l_2}{2}\right) = \Delta t\left(y_2(t_n) + \frac{l_2}{2}\right)\\
l_3 &= \Delta t \cdot g\left(t_n + \frac{\Delta t}{2}, y_1(t_n) + \frac{k_2}{2}, y_2(t_n) + \frac{l_2}{2}\right) = -\Delta t\left(y_1(t_n) + \frac{k_2}{2}\right)\\
k_4 &= \Delta t \cdot f\left(t_n + \frac{\Delta t}{2}, y_1(t_n) + \frac{k_3}{2}, y_2(t_n) + \frac{l_3}{2}\right) = \Delta t\left(y_2(t_n) + \frac{l_2}{2}\right)\\
l_4 &= \Delta t \cdot g\left(t_n + \frac{\Delta t}{2}, y_1(t_n) + \frac{k_3}{2}, y_2(t_n) + \frac{l_3}{2}\right) = -\Delta t\left(y_1(t_n) + \frac{k_3}{2}\right)
\end{aligned} \tag{0.9}$$

and

$$\begin{aligned}k &= \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4),\\ l &= \frac{1}{6}(l_1 + 2l_2 + 2l_3 + l_4)\end{aligned} \tag{0.10}$$

we have the following recursive equations:

$$\begin{aligned}y_1(t_{n+1}) &= y_1(t_n) + k,\\ y_2(t_{n+1}) &= y_2(t_n) + l.\end{aligned} \tag{0.11}$$

We have implemented in this into the following Matlab code:

```
function [y1,e,eT,eTd]=RK4(N,T)

dt=T/N;
y1=zeros(1,N+1); y2=zeros(1,N+1);    % pre-allocation

y1(1)=1; y2(1)=0;    % initial conditions

for n=1:N
    k1=dt*y2(n);
    l1=-dt*y1(n);
    k2=dt*(y2(n)+l1/2);
    l2=-dt*(y1(n)+k1/2);
    k3=dt*(y2(n)+l2/2);
    l3=-dt*(y1(n)+k2/2);
    k4=dt*(y2(n)+l3/2);
    l4=-dt*(y1(n)+k3/2);
    k=1/6*(k1+2*k2+2*k3+k4);
    l=1/6*(l1+2*l2+2*l3+l4);
    y1(n+1)=y1(n)+k;
    y2(n+1)=y2(n)+l;
end
```

[1] http://www.nsc.liu.se/~boein/f77to90/rk.html

```
t=(0:N)*dt;
Y1=cos(t); Y2=-sin(t);       % exact solutions
e=sum(abs(Y1-y1))/(N+1);     % average error at over the interval
eT=abs(Y1(N+1)-y1(N+1));     % error in the value at the last point
eTd=abs(Y2(N+1)-y2(N+1));    % error in the derivative at the last point
```

Using an interval of $T = 10\pi$ and the same trial-and-error / brute force method as before – to be specific, with the following code –

```
T=10*pi;
N=26200;

dt=T/N;

t=(0:N)*dt;

[y,e,eT,eTd]=RK4(N,T);

while e>0.001
     N=N+1;
     [y,e,eT,eTd]=RK4(N,T);
end
```

we found that the average error over the interval e dropped below 0.001 first for N = 26,208, which corresponds to $\Delta t = \frac{T}{N} = \frac{10\pi}{26{,}208} \approx 0.0012$. The error in value at the last point was $e_T \approx 0.003$, and the error in the derivative at the last point $e_{T,d} \approx 2 \times 10^{-6}$.

Problem 6.3

Since the RK4 method is fourth-order accurate, we expect the local error to decrease by a factor of 10^4 for each reduction in the step size by a factor of 10.