Assignment Sheet Nr. 3

Paul Monderkamp, Matr.Nr. 2321677

monderkamp@thphy.uni-duesseldorf.de

Contents

1 Exercise 1

1.1 Exercise 1 (a)

3 1 a) For harm. osc.

$$Y_{n+1} = \begin{pmatrix} x_{n+1} \\ v_{n+1} \end{pmatrix} \qquad f(t_n, V_n) = f(Y_n)$$

$$= Y_n + \frac{\delta t}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right)$$

$$\begin{aligned} = Y_n + \frac{\delta t}{6}\Big(& f(Y_n) \\ & + 2 f\left(Y_n + \frac{\delta t}{2} f(Y_n)\right) \\ & + 2 f\left(Y_n + \frac{\delta t}{2} f\left(Y_n + \frac{\delta t}{2} f(Y_n)\right)\right) \\ & + f\left(Y_n + \delta t\, f\left(Y_n + \frac{\delta t}{2} f\left(Y_n + \frac{\delta t}{2} f(Y_n)\right)\right)\right)\Big) \end{aligned}$$

$$\left\{ f(Y_n) = \dot{y}(t_n) = \begin{pmatrix} v \\ -x \end{pmatrix} = \underbrace{\begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}}_{A} Y_n \right\}$$

$$\begin{aligned} = Y_n + \frac{\delta t}{6}\Big(& A Y_n \\ & + 2A\left(Y_n + \frac{\delta t}{2} A Y_n\right) \\ & + 2A\left(Y_n + \frac{\delta t}{2} A\left(Y_n + \frac{\delta t}{2} A Y_n\right)\right) \\ & + A\left(Y_n + \delta t\, A\left(Y_n + \frac{\delta t}{2} A\left(Y_n + \frac{\delta t}{2} A(Y_n)\right)\right)\right)\Big) \end{aligned}$$

$$\begin{aligned} = Y_n + \frac{\delta t}{6}\Big(& A Y_n + 2A Y_n + \delta t A^2 Y_n \\ & + 2A Y_n + \delta t A^2\left(Y_n + \frac{\delta t}{2} A Y_n\right) \\ & + A Y_n + \delta t A^2\left(Y_n + \frac{\delta t}{2} A\left(Y_n + \frac{\delta t}{2} A Y_n\right)\right)\Big) \end{aligned}$$

(1)

Figure 1.1: first part of the derivation of the formula

$$= Y_n + \frac{\delta t}{6}\Big(3A\,Y_n + \delta t\,A^2 Y_n + 2A\,Y_n + \delta t\,A^2 Y_n$$
$$+ \frac{\delta t^2}{2} A^3 Y_n + A\,Y_n + \delta t\,A^2 Y_n$$
$$+ \frac{\delta t^2}{2} A^3 \left(Y_n + \frac{\delta t}{2} A\,Y_n\right) \Big)$$

$$= Y_n + \frac{\delta t}{6}\Big(6A\,Y_n + 3\delta t\,A^2 Y_n + \delta t^2 A^3 Y_n$$
$$+ \frac{1}{4}\delta t^3 A^4 Y_n\Big)$$

$$A = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$$
$$A^2 = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}\begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}$$
$$A^3 = -\begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = -A = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$$
$$A^4 = -A^2 = \mathbb{1}$$

$$= \left(1 + \frac{\delta t^4}{24}\right) Y_n + \delta t\,A\,Y_n + \frac{1}{2}\delta t^2 A^2 Y_n$$
$$+ \frac{1}{6}\delta t^3 A^3 Y_n$$

$$= \left(1 + \frac{\delta t^4}{24}\right)\begin{pmatrix} x_n \\ v_n \end{pmatrix} + \delta t \begin{pmatrix} v_n \\ -x_n \end{pmatrix} - \frac{\delta t^2}{2}\begin{pmatrix} x_n \\ v_n \end{pmatrix}$$
$$+ \frac{1}{6}\delta t^3 \begin{pmatrix} -v_n \\ x_n \end{pmatrix}$$

which is equiv to
the formula in 2 1 a

(2)

Figure 1.2: second part of the derivation of the formula

1.2 Exercise 1 (b)

1.2.1 Code

```
#include <iostream>
#include <cmath>
#include <fstream>
using namespace std;

int main()
{
double tmin     = 0.0;
double tmax     = 50.0;
const int N     = 500;
double dt       = (double)(tmax - tmin)/(N-1);

double y[N][4];
y[0][0] = 1.0;
y[0][1] = 0.0;
y[0][2] = 0.5*(y[0][0]*y[0][0]+y[0][1]*y[0][1]);
y[0][3] = 0.0;

for (int i=0;i<N-1;i++)
{
  y[i+1][0] = y[i][0] + y[i][1]*dt -0.5*y[i][0]*dt*dt-
     (1.0/6.0)*y[i][1]*pow(dt,3.0)+
     (1.0/24.0)*y[i][0]*pow(dt,4.0);
  y[i+1][1] = y[i][1] - y[i][0]*dt -0.5*y[i][1]*dt*dt+
     (1.0/6.0)*y[i][0]*pow(dt,3.0)+
     (1.0/24.0)*y[i][1]*pow(dt,4.0);
  y[i+1][2] = 0.5*(y[i+1][0]*y[i+1][0]+y[i+1][1]*y[i+1][1]);
  y[i+1][3] = dt*i;

}
ofstream outputfile;
outputfile.open("3_1_b_results.txt");

for (int i=0; i<N; i++)
{
 outputfile << y[i][0] << "  " << y[i][1]
 << "  " << y[i][2] << "  " << y[i][3] << endl;
}
outputfile.close();
return 0;
}
```

1.2.2 Results

The following results are achived via the code in the previous subsection.

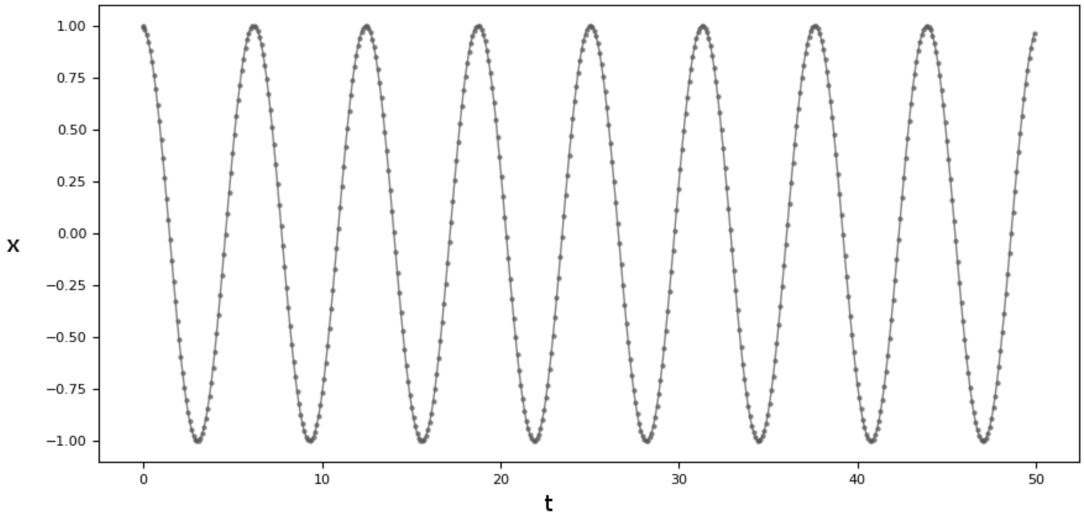

Figure 1.3: time evolution of the x coordinate of the harmonic oscillator

Figure 3.3 shows the time evolution of the x coordinate of a one dimensional harmonic oscillator solved with the Runge-Kutta 4 scheme. There is no apparent divergence from the analytical solution visible in the timeframe of the simulation.

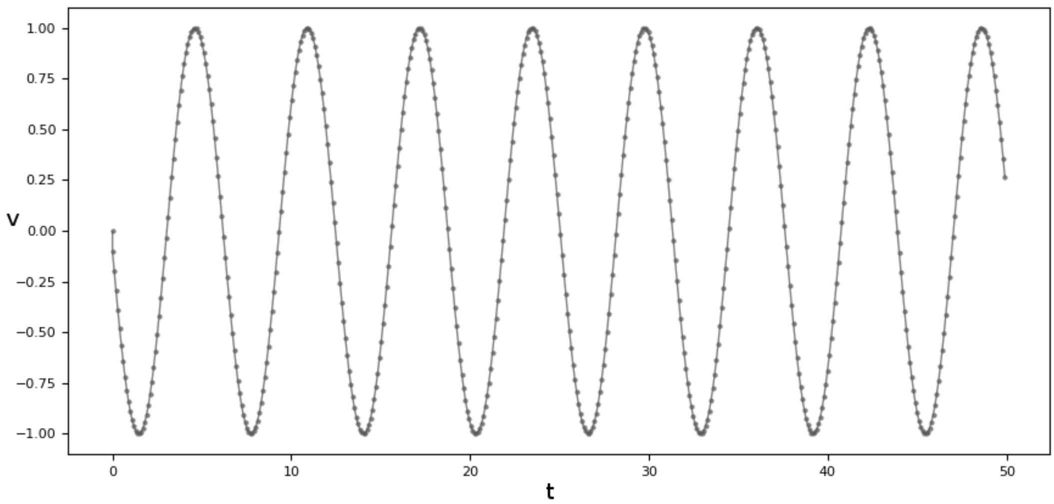

Figure 1.4: time evolution of the velocity of the harmonic oscillator

Figure 3.4 shows the time evolution of the velocity of a one dimensional harmonic oscillator solved with the Runge-Kutta 4 scheme. Similarly to the x coordinate, there is no visible instability in the timeframe of the simulation period.

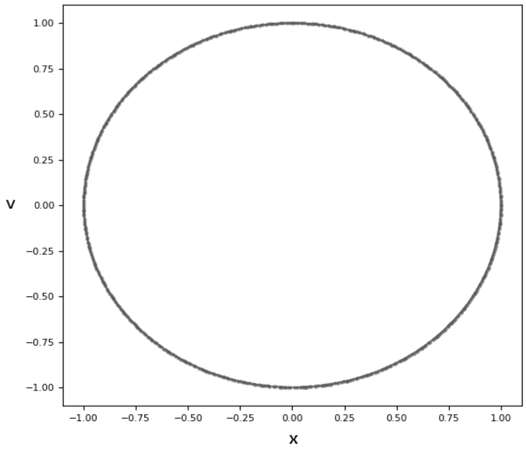

Figure 1.5: phase space trajectory of the harmonic oscillator

Figure 3.5 shows the phase space profile for the simulation results above. The trajectory seems reasonably spherical.

2 Exercise 2

2.1 Exercise 2 (a)

i) $\theta(t+\delta t) = \theta(t) + \omega(t)\,\delta t - \frac{1}{2}\sin(\theta(t))\,\delta t^2$

$\omega(t+\delta t) = \omega(t) - \sin\left(\theta(t) + \frac{1}{2}\omega(\theta)\,\delta t\right)\delta t$

to check time reversibility :

$t \leftrightarrow t + \delta t$

$\delta t \leftrightarrow -\delta t$

$\theta(t+\delta t) = \theta_{n+1}$

$\theta(t) = \theta_n$

ω analogously

$\theta_n = \theta_{n+1} + \omega_{n+1}(-\delta t) - \frac{1}{2}\sin\theta_{n+1}\,\delta t^2$

$\omega_n = \omega_{n+1} + \delta t\,\sin\left(\theta_{n+1} - \frac{1}{2}\omega_{n+1}\,\delta t\right)$

it is impossible to solve either one of these eq. ~~of~~ for θ_{n+1} or ω_{n+1} and thus old eq. cannot be obtained from them

Figure 2.1: time reversibility for Runge-Kutta 2 scheme

$$\theta_{n+1} \leftrightarrow \theta_n \quad ; \quad \delta t \rightarrow -\delta t$$

(i)

$$\theta_n = \theta_{n+1} + \omega_{n+1}(-\delta t) - \frac{1}{2}\sin\theta_{n+1}\,\delta t^2$$

$$\omega_n = \omega_{n+1} + \frac{1}{2}\delta t\,[\sin\theta_{n+1} + \sin\theta_n]$$

$$\cancel{\Rightarrow \theta_{n+1} = \theta_n + \omega_{n+1}\delta t +}$$

$$\omega_{n+1} = \omega_n - \frac{1}{2}\delta t\,[\sin\theta_{n+1} + \sin\theta_n] \quad \checkmark$$

in first eq.:

$$\theta_{n+1} = \theta_n + \delta t\left(\omega_n - \frac{1}{2}\delta t\,[\sin\theta_{n+1} + \sin\theta_n]\right) + \frac{1}{2}\sin\theta_{n+1}\,\delta t^2$$

$$= \theta_n + \delta t\,\omega_n - \frac{1}{2}\delta t^2 \sin\theta_n \quad \checkmark$$

$\Rightarrow$ time reversible

Figure 2.2: time reversibility for velocity Verlet algorithm

2.2 Exercise 2 (b)

2.2.1 Code

```
#include <iostream>
#include <cmath>
#include <fstream>
#include <string>

using namespace std;

int main()
{
string timeinterval = "0_005";
string outputfilename = "3_2b_euler_dt_" + timeinterval + ".txt";

double tmin = 0.0;
double tmax = 50.0;
double dt = 0.005;

int N = (int)(tmax - tmin)/dt + 1;

double y[N][3];

y[0][0] = (7.0/360.0)*2.0*M_PI;
y[0][1] = 0.0;
y[0][2] = 50.0*y[0][1]*y[0][1]+100.0*(1.0-cos(y[0][0]));

ofstream out;
out.open(outputfilename);
out << y[0][0] << "  " << y[0][1] << "  " << y[0][2] << endl;

for (int i=0;i<N;i++)
  {
    y[i+1][0] = y[i][0] + y[i][1]*dt;
    y[i+1][1] = y[i][1] - dt*sin(y[i][0]);
    y[i+1][2] = 50.0*y[i+1][1]*y[i+1][1]+100.0*(1.0-cos(y[i+1][0]));

    out << y[i+1][0] << "  " << y[i+1][1] << "  " << y[i+1][2] << endl;
  }


out.close();
return 0;
}
```

The code above is the program with which the differential equation

$$\dot{\theta} = \omega$$
$$\dot{\omega} = -\frac{g}{L}\sin(\theta) \tag{2.1}$$

with $g = L = 10$ is solved according to the Euler algorithm. The Euler-Cromer algorithm follows the same scheme except the for loop with the iteration looks as follows:

```
for (int i=0;i<N;i++)
  {
    y[i+1][1] = y[i][1] - dt*sin(y[i][0]);
    y[i+1][0] = y[i][0] + y[i+1][1]*dt;
    y[i+1][2] = 50.0*y[i+1][1]*y[i+1][1]
      +100.0*(1.0-cos(y[i+1][0]));

    out << y[i+1][0] << "   " << y[i+1][1]
        << "   " << y[i+1][2] << endl;
  }
```

2.2.2 Results