NUMERISCHE METHODEN IN DER PHYSIK
Fourth Exercise WS 2013/2014 [C]

Numerical evaluation of initial value problems with Runge-Kutta algorithms

In this exercise we want to use the *Runge-Kutta method* to numerically solve initial value problems of the following examples:

- Satellite example from the lecture notes
  Data for comparisons can be found in the file *exakte_ellipse*.
- Trajectories of charged particles in a magnetic field
- Oscillation behaviour of van der Pol oscillators

To do that we have provided you with two programs, which you can download from our webpage:

```
runge_kutta_D.c (structure chart 26-28)

nrutil.c    exakte_ellipse
```

How to use the Runge-Kutta package:

As described in Sec. 8.5 of the German version of the lecture note, the Runge-Kutta algorithms is based on the routines ODEINT, RKQC and RK4. (ODEINT is the program that has to be called from you main program.) The user also has to provide a program called DERIVS, which defines the set of n differential equations of first order for the given problem.

```
      ----------
      | DERIVS |
      ----------
         |
.....................
.        |          .
.     -------       .
.     | RK4 |       .
.     -------       .
.        |          .
.        |          .
.     --------      .
.     | RKQC |      .   program package
.     --------      .   "runge_kutta_D.c"
.        |          .
.        |          .
.    ----------     .
.    | ODEINT |     .
.    ----------     .
.        |          .
.....................
         |
   ----------------
   | main program |
   ----------------
```

Structure of the C program:

```
// Runge-Kutta program according to structure charts 26,27,28 in the German
// lecture notes, edition WS 2002/2003.

// Attention: "local extrapolation" Glg. (8.27) INACTIV !!

#include <stdio.h>
#include <math.h>
#include "nrutil.c"


void derivs(double x, double y[], double f[])
//  Defining the functions f
{
  f[1]=.... ;   // Def. of f1(x, vec y)
  f[2]=.... ;   // Def. of f2(x, vec y)
   .
   .
}
```

```
#include "runge_kutta_D.c"  // Input of the Runge-Kutta program you can
                            // download from our webpage.
                            // The "interface"-function you have to call
                            // from the main program is named
                            // ODEINT

// void odeint(double ystart[], int nvar, double x1, double x2, double eps,
//             double h1, double hmin, int nstmax, int *nwerte,
//             double xx[], double **yy)

//  This is the driver program for the RuKu-package, which consists of
//  the functions ODEINT, RKQC, RK4 and DERIVS.
//  (see German lecture notes page 250 and following).

//  Last update: 2-10-96

//  Attention:
//  ==========================
//  There has to be a subprogram called
//        void derivs(double x, double y[], double f[])
//  which calculates for given point x known function values y[]
//  the corresponding "right hand sides" of the set of
//  differential equations f[].

//  Parameters:
//  ========================
//      Input:  ystart[]   vector with initial values of the set of diff. eq.
//              nvar       number of eq.
//              x1,x2      start- and endpoint of the integration interval
//              eps        required relative accuracy
//              h1         guessed value for the step length of the
//                         RuKu-process
//              hmin       minimal value for the work step length
//              nstmax     maximal number of points between x1 and
//                         x2, which are saved in xx[] and yy[]

//      Output: nwerte     number of saved points
//              xx[]       x values of saved points
//              yy[][]     y values of saved points
//                            first index  = index of function
//                            second index = index of point



int main()
{
```

```
    int nvar,nstmax,nwerte,... ;
    double t0,tmax,eps,h1,hmin;
    double *ystart,*tfeld,**yfeld;

    nvar=... ;
    nstmax=... ;
    y0=dvector(1,nvar);
    tfeld=dvector(1,nstmax);
    yfeld=dmatrix(1,nvar,1,nstmax);

    t0=... ;
    tmax=... ;

    ystart[1]=... ;
    ystart[2]=... ;
     .
     .

    eps=... ;
    h1=... ;
    hmin=... ;

    odeint(ystart,nvar,t0,tmax,eps,h1,hmin,nstmax,&nwerte,tfeld,yfeld);
     .
     .
    return (0);
}
```

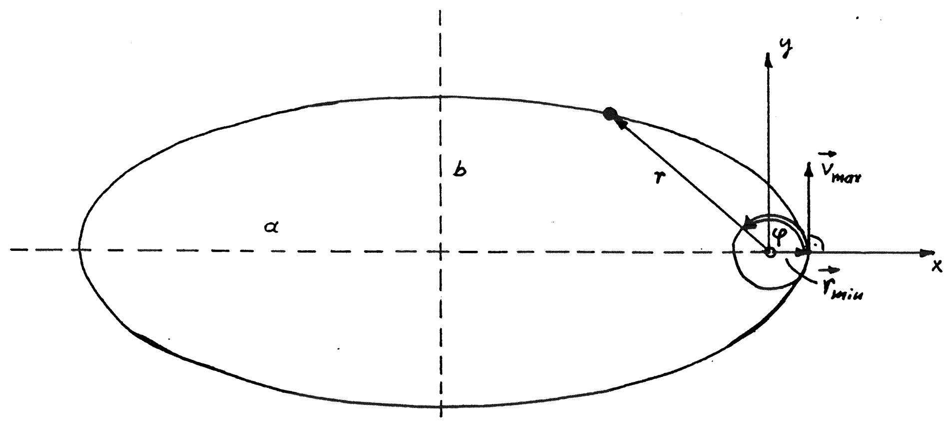

Figure 1: Trajectory of an earth satellite

1. Exercise: Satellite example from the lecture

Test the performance of your Runge-Kutta algorithm with the satellite example from the lecture (see German lecture notes starting on page 245).

Defining the problem:

A satellite is set into motion with a velocity v_{max}, tangential to the surface of the earth. We want to calculate its trajectory considering the following constraints and approximations:

- v_{max} < escape velocity.
- The earth shall be approximated as an ideal sphere with r_{min} and homogenous density.
- The influence of the earth's atmosphere is neglected.
- The influence of all other celestial bodies is neglected.

With these constraints, the *equation of motion* of the satellite reads

$$\ddot{\mathbf{r}} = -\frac{\gamma M}{r^3}\mathbf{r}\,, \tag{1}$$

where $\mathbf{r}$ denotes the vector originating from the earth's center and ending at the satellite's momentary position.

Equation (1) leads in the end to a set of two differential equations of second order for the distance r and the rotation angle φ:

$$\begin{aligned} \ddot{r} &= r\dot{\varphi}^2 - \frac{\gamma M}{r^2} \\ \ddot{\varphi} &= -\frac{2\dot{r}\dot{\varphi}}{r} \end{aligned} \tag{2}$$

Substituting

$$r \to y_1 \quad \varphi \to y_2 \quad \dot{r} \to y_3 \quad \dot{\varphi} \to y_4 \quad ,$$

one gets a set of 4 differential equations of first order:

$$\begin{aligned} \dot{y}_1 &= y_3 & y_1(t=0) &= r_{min} \\ \dot{y}_2 &= y_4 & y_2(t=0) &= 0 \\ \dot{y}_3 &= y_1 y_4^2 - \frac{\gamma M}{y_1^2} & y_3(t=0) &= 0 \\ \dot{y}_4 &= -\frac{2 y_3 y_4}{y_1} & y_4(t=0) &= \frac{v_{max}}{r_{min}} \end{aligned} \tag{3}$$

Constants and initial values: M and γ are the earth's mass and the gravitational constant

$$M = 5.977 \cdot 10^{24} \text{ kg} \qquad \text{and} \qquad \gamma = 6.67 \cdot 10^{-11} \text{ m}^3/(\text{kg s}^2)\,.$$

The satellite starts at the earth's surface ($r_{min} = 6.37 \cdot 10^6$ m) with a velocity of $v_{max} = 10.4 \cdot 10^3$ m/s.

We want to express all lengths in units of r_{min}, all velocities in units of v_{max} and all times in units of the period of revolution $T = 35705.83$.

In these units the constant $\alpha = \gamma M$ has the value 1966.39, and the intial values for $y_1(0)$ and $y_4(0)$ are 1.0 and 58.29527 respectively.

According to eq. (3) the definition of the functions reads

```
function dy = f_satellit(t,y)

global alpha
dy = zeros(4,1);  % a column vector

dy(1) = y(3);
dy(2) = y(4);
dy(3) = y(1)*y(4)^2 - alpha/(y(1)^2); % alpha GLOBAL = 1966.390
dy(4) = -2*y(3)*y(4)/y(1);
```

Tasks:

- Solve the set of equations for a time interval from 0 to 5 revolutions of the satellite four times, each time with different tolerance values

  ```
  EPS = 0.25    0.05    0.01    0.001
  ```

- Present your results in four seperate figures depicting the calculated trajectory in xy space. Compare these trajectories to the exact, analytically calculated elliptical trajectory. You can find the exact data in the file

  ```
  exakte_ellipse
  ```

2. Exercise: Trajectory in an inhomogeneous magnetic field

Theoretical background

A particle with mass m and electrical charge q, which moves in a magnetic field (with magnetic flux density $\mathbf{B}(\mathbf{r},t)$) with a velocity of $\mathbf{v}(t)$ is subjected to a **Lorentz force**

$$\mathbf{F}_L = q(\mathbf{v} \times \mathbf{B}) \quad . \tag{4}$$

To calculate the **trajectory** of the particle we start with Newton's second law

$$m\ddot{\mathbf{r}}(t) = q\left[\dot{\mathbf{r}}(t) \times \mathbf{B}(\mathbf{r},t)\right] . \tag{5}$$

This vectorial differential equation of second order corresponds to the set of differential equations

$$\begin{aligned} \ddot{x} &= \frac{q}{m}(\dot{y}B_z - \dot{z}B_y) \\ \ddot{y} &= \frac{q}{m}(\dot{z}B_x - \dot{x}B_z) \\ \ddot{z} &= \frac{q}{m}(\dot{x}B_y - \dot{y}B_x) \end{aligned} \tag{6}$$

Taking the **initial values** for the time $t = 0$ into account

$$\begin{aligned} x(t=0) &= x_o \qquad y(t=0) = y_o \qquad z(t=0) = z_o \\ \dot{x}(t=0) &= v_{xo} \qquad \dot{y}(t=0) = v_{yo} \qquad \dot{z}(t=0) = v_{zo} \end{aligned} \tag{7}$$

we are left with a uniquely defined **initial value problem**. The solutions $x(t)$, $y(t)$ and $z(t)$ are the spatial components of the trajectory parametrised by t.

The space- and time-dependent vector of the magnetic field (more precisely of the magnetic flux density) is of the form

$$\mathbf{B}(\mathbf{r},t) = B_0 \begin{pmatrix} f_1(\mathbf{r},t) \\ f_2(\mathbf{r},t) \\ f_3(\mathbf{r},t) \end{pmatrix} \tag{8}$$

Putting this vector in equation (6) results in a common factor in all of the equations called the angular frequency qB_0/m, which in our calculations is set to the value

$$\frac{qB_0}{m} \equiv \sigma = \pi \ .$$

Your task now is to write a program that can solve numerically the initial value problem described above for arbitrary values of σ, x_o, v_{xo}, y_o, v_{yo}, z_o and v_{zo} as well as for arbitrary functions $f_i(\mathbf{r},t)$ by using the **ODEINT+RKQC+RK4** program package.

Testing the program

If the magnetic field is constant in time as well as space and points exactly in z-direction, i.e.

$$f_1(\mathbf{r},t) = 0.0 \qquad f_2(\mathbf{r},t) = 0.0 \qquad f_3(\mathbf{r},t) = 1.0\,,$$

we can solve the problem analytically and obtain the following solution

$$x(t) = -\frac{v_{yo}}{\sigma}\cos(\sigma t) + \frac{v_{xo}}{\sigma}\sin(\sigma t) + x_o + \frac{v_{yo}}{\sigma} \qquad (9)$$

$$y(t) = \frac{v_{yo}}{\sigma}\sin(\sigma t) + \frac{v_{xo}}{\sigma}\cos(\sigma t) + y_o - \frac{v_{xo}}{\sigma} \qquad (10)$$

$$z(t) = z_o + v_{zo}t\ . \qquad (11)$$

In this case, the trajectory of the particle is a helix with the symmetry axis in z-direction. Projecting the helix onto the xy plane one gets a circle with center coordinates $x_o + v_{yo}/\sigma$ and $y_o - v_{xo}/\sigma$ and a radius of $\sqrt{v_{xo}^2 + v_{yo}^2}/\sigma$. In z-direction we have a uniform motion with velocity v_{zo} independent of the magnetic field.

Use these considerations to test your program, i.e. compare your results for $0 \leq t \leq 10$ s with the analytic results from equations (9-11) for following initial values:

$$x_o = 0 \qquad y_o = 0 \qquad z_o = 0 \qquad v_{xo} = 0 \qquad v_{yo} = 2\pi \qquad v_{zo} = 0.25$$

EPS for all calculations $= 10^{-5}$.

Graphic representation:

1. Present the three functions $x(t)$, $y(t)$ and $z(t)$ in one figure (see fig. 2).

2. Show the calculated trajectory in 3D-space, as for example shown in fig. 3.

   You can make use of the very convenient plotting function *plot3(vec-x,vec-y,vec-z)* in Matlab for these kinds of trajectories $[x(t), y(t), z(t)]$. There are also similar functions available for other graphic libraries (gnuplot,...).

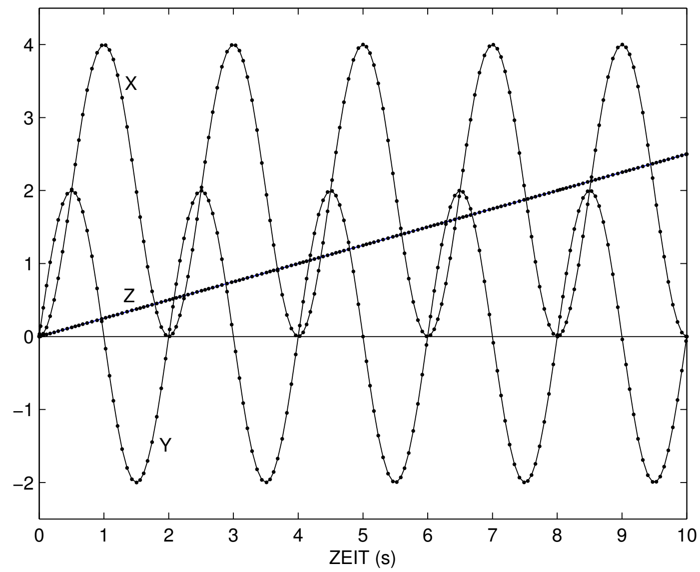

Figure 2: The functions $x(t)$, $y(t)$ and $z(t)$ in the test case.

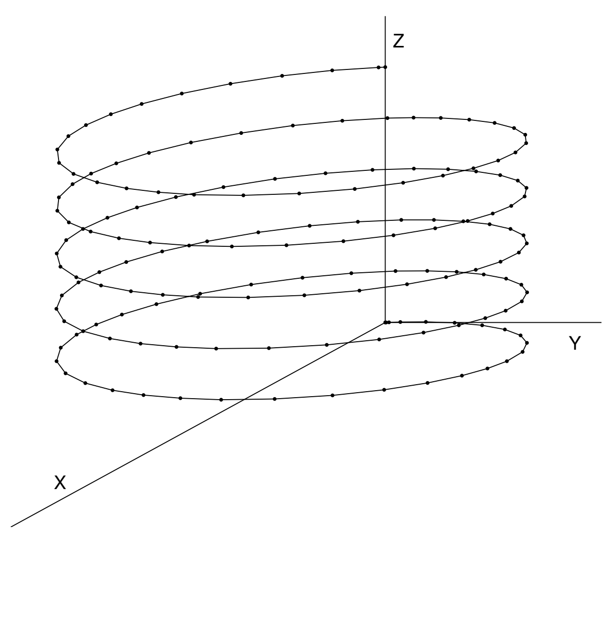

Figure 3: Trajectory of the charged particle for the test example.

Application example

Use the following inhomogeneous magnetic fields:

Example: Magnetic field, inhomogeneous in time:

Calculate the trajectories for the time interval $0 \leq t \leq 10$ s and the initial values

$$x_o = 0 \qquad y_o = 0 \qquad z_o = 0$$

$$v_{xo} = 0 \qquad v_{yo} = 1.5 \qquad v_{zo} = 2\pi$$

(a) for the magnetic field

$$f_1(\mathbf{r}, t) = \frac{1}{\sqrt{2}}\,\mathrm{e}^{\gamma t} \qquad f_2(\mathbf{r}, t) = \frac{1}{\sqrt{2}}\,\mathrm{e}^{\gamma t} \qquad f_3(\mathbf{r}, t) = 0$$

with

$$\gamma = 0.5\,,$$

(b) for the magnetic field

$$f_1(\mathbf{r}, t) = \sqrt{2}\,\mathrm{e}^{\gamma t} \qquad f_2(\mathbf{r}, t) = \sqrt{2}\,\mathrm{e}^{\gamma t} \qquad f_3(\mathbf{r}, t) = 0$$

with

$$\gamma = -0.25\,.$$

Example: "Tilted" magnetic field:

Calculate the trajectories for the time interval $0 \le t \le 20$ s and the initial values

$$x_o = 0 \qquad y_o = 0 \qquad z_o = 0$$

$$v_{xo} = 0 \qquad v_{yo} = 2\pi \qquad v_{zo} = 0.25$$

In the time interval $0 \le t \le t_a$ the magnetic field is considered constant and to point in z-direction

$$\mathbf{f}(\mathbf{r},t) = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} .$$

In the time interval $t_a \le t \le t_b$ the vector $\mathbf{f}$ rotates *linearly in time* towards the x-direction (see fig. 4).

For times $t > t_b$ the magnetic field is constant and points in x-direction:

$$\mathbf{f}(\mathbf{r},t) = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} .$$

Time parameter: (a) $t_a = 8.5$ s $t_b = 9.5$ s
(b) $t_a = 9.0$ s $t_b = 10.0$ s

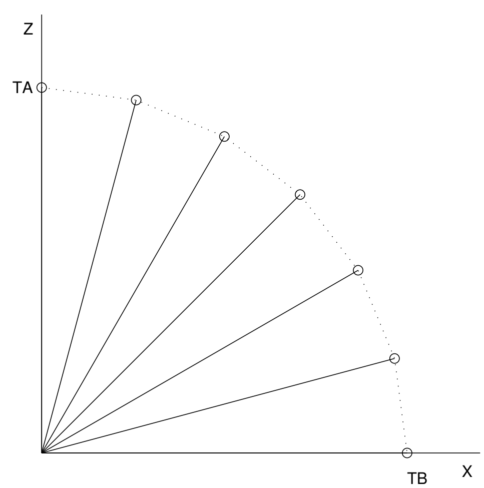

Figure 4: Tilting the magnetic field

3. Exercise: van der Pol oscillator

Theoretical background

We want to study the properties of the so-called *van der Pol* oscillator. This kind of oscillators appears for example in an electromagnetic transmitter, whose losses through the resistance are compensated by backcoupling to a control grid of a triode.
The differential equation describing this case can in principle be *only solved numerically*. There are, however, analytic approximations for the *attractor curve*, as will be described later.

The van der Pol oscillator is described by the following equation of motion

$$m\ddot{x} - 2m\gamma(x)\dot{x} + m\omega^2 x = 0 \qquad (12)$$

with $x(t=0) = x_0$ and $\dot{x}(t=0) = v_0$. The reason for only being able to solve this numerically lies in the *non-linearity* of $\gamma(x)$

$$\gamma(x) := \gamma_0 \left[1 - \frac{x(t)^2}{x_{kr}^2}\right] \qquad (13)$$

appearing in the damping term (with constants $\gamma_0 > 0$ and x_{kr}). x_{kr} is a *critical displacement*, above which oscillations are damped. For displacements $x < x_{kr}$ we have $\gamma(x) < 0$, in which case the oscillations are amplified.

- Solve the initial value problem (12), (13) by employing a *Runge-Kutta algorithm* thrice with following parameters (γ_0, ω are given in units of s^{-1}, x_{kr} and x_0 in cm and v_0 in cm/s):

  1. $\omega = \pi$ $\quad\gamma_0 = 0.025$ $\quad x_{kr} = 0.5$
     $x_0 = 5.0$ $\quad v_0 = 0.0$
     $0 \le t \le 61\,s$ $\quad$ EPS $= 0.00001$
  2. $\omega = \pi$ $\quad\gamma_0 = 1.25$ $\quad x_{kr} = 0.5$
     $x_0 = 5.0$ $\quad v_0 = 0.0$
     $0 \le t \le 31\,s$ $\quad$ EPS $= 0.00001$
  3. $\omega = \pi$ $\quad\gamma_0 = 2.5$ $\quad x_{kr} = 1.0$
     $x_0 = 0.1$ $\quad v_0 = 0.0$
     $0 \le t \le 31\,s$ $\quad$ EPS $= 0.0001$

- Plot the function $x(t)$ for all of these cases.

- Show a *phase diagram* of your results, i.e. depict the displacements $x(t)$ as functions of the velocities $\dot{x}(t)$. You will see that in all cases the oscillations heads towards a limiting curve, which is a periodic solution of the system itself. This solution, called *attractor curve*, can be approximately stated in an analytic form

$$x(t) = 2x_{kr}\cos(\omega t) - \gamma_0 \frac{x_{kr}}{2\omega}\sin(3\omega t) \ ,$$

and its respective time derivative. Plot these approximations in the same figures with your phase diagrams.[1]

- Discuss and interpret the three cases. What type of oscillations are they?

[1] More information on the van-der-Pol oscillator can be found example in: F. Scheck, *Mechanik: von den Newtonschen Gesetzen zum deterministischen Chaos*, Springer-Lehrbuch 1992, S.293f;
Ch. Gerthsen, Physik, 18. Auflage, neubearbeitet von H. Vogel, Springer, Berlin, 1995, S. 976f.