University of Zurich - HS 2013

Computational Science I
Exercise notes: Runge-Kutta, Lorenz, Friedmann

Tobias Grubenmann

November 10, 2013

Exercise 1

The following code implements the Runge-Kutta methods of order $1, 2$ and 4. The methods excepts an autonomous inital value problem:

```
def rk4(f, y, h):

    k1 = f(y)
    k2 = f(y + 0.5*h*k1)
    k3 = f(y + 0.5*h*k2)
    k4 = f(y + h*k3)

    return y + 1/6.*h*(k1 + 2*k2 + 2*k3 + k4)

def rk1(f, y, h):

    k1 = f(y)

    return y + h*k1

def rk2(f, y, h):

    k1 = f(y)
    k2 = f(y + 0.5*h*k1)

```

```
    return y + h*k2
```

The solve the initial value problem for the Chebyshev polynomials we first need to put the equations into an autonomous problem:

Let $u_1 := (T_n, x)$, $u_1 := u_1'$ and $u_3 = x$, then the Chebyshev differential equation translates into:

$$\frac{d}{dx}\begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = \begin{pmatrix} u_2 \\ \frac{u_3 \cdot u_2 - n^2 \cdot u_1}{1 - u_3^2} \\ 1 \end{pmatrix}$$

This gives us the following solutions for the first 6 chebyshev polynomials:

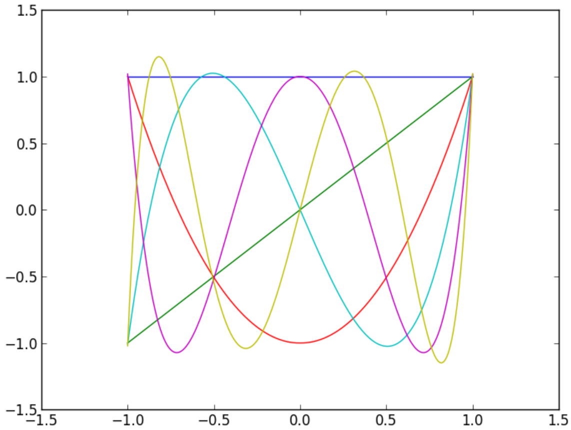

Figure 1: The first 6 Chebyshev polynomials with 1st order Runge-Kutta

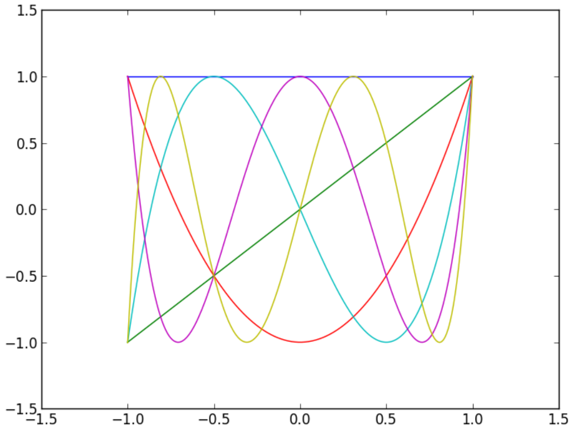

Figure 2: The first 6 Chebyshev polynomials with 2nd order Runge-Kutta

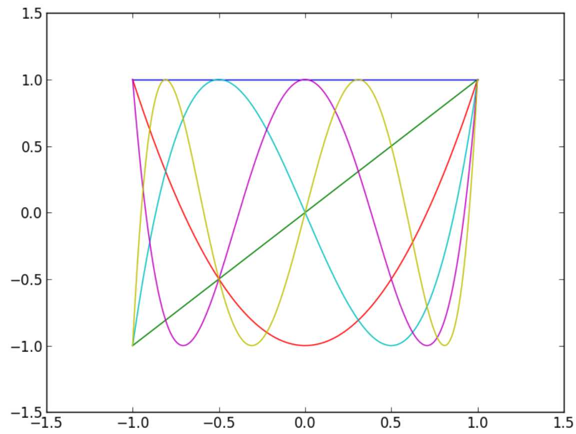

Figure 3: The first 6 Chebyshev polynomials with 4th order Runge-Kutta

Exercise 2

This codes generates a screensaver-like graph of the Lorenz attractor:

```
from Tkinter import *
from scipy.integrate import odeint
from numpy import arange

screen = Tk()
wd, ht = screen.winfo_screenwidth(), screen.
   winfo_screenheight()
screen.geometry("%dx%d+0+0"%(wd, ht))
screen.attributes("-fullscreen", 1)
canv = Canvas(screen, height = ht, width = wd, background =
    "black")
canv.pack()

# stretch factors and offsets
stretchX = 15;
stretchY = 15;
offsetX = 600;
offsetY = 370;

time = 0
timeStep = 0.04
```

```
def func(y, t):
    return [-rho*y[0]+rho*y[1], r*y[0]-y[1]-y[0]*y[2], y
       [1]*y[0]-b*y[2]]

rho = 10
r = 28
b = 8/3.

y0 = [20, 7, 27]

def frame(y0, time, timeStep):

    startOffset = 4

    t = arange(time, time + timeStep, timeStep/startOffset)
    y = odeint(func, y0, t)

    #canv.delete("all")

    for i in range(len(y)-1):
        canv.create_line(offsetX + y[i][0]*stretchX,
           offsetY +y[i][1]*stretchY,
                         offsetX + y[i+1][0]*stretchX,
                            offsetY + y[i+1][1]*stretchY,
                            fill = "#FFD700" )

    screen.after(1, frame, y[startOffset-1], time+timeStep,
        timeStep)

screen.after(1, frame, y0, time, timeStep)

canv.update()

screen.mainloop()
```

The following picture shows a screenshot of the screensaver after about 15 sec.

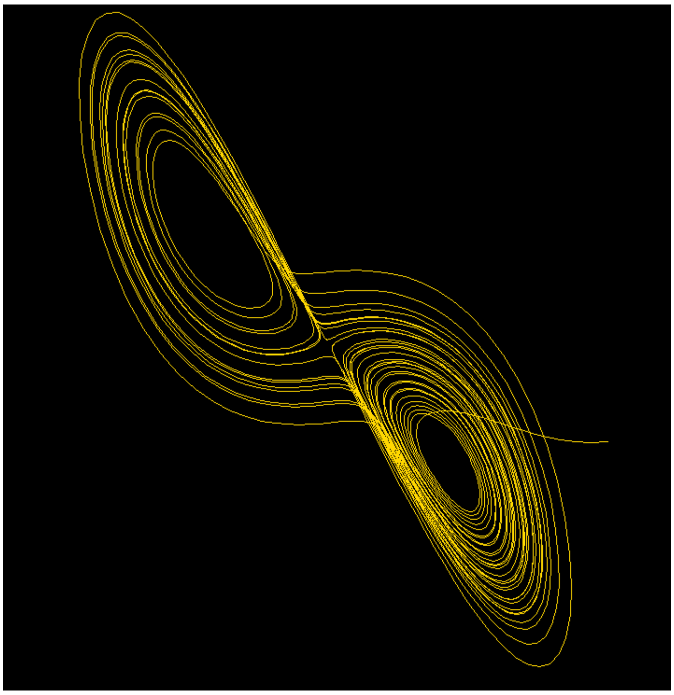

Figure 4: Screenshot of the Lorenz attractor

Exercise 3

The following script solves the Friedmann equations using the built-in ODE solver. I used initial conditions of $y(1) = (0, 0)$ and constants $H_0 = \Omega_m = \Omega_l = 1$:

```
H0 = 1
Omega_m = 1
Omega_l = 1

def H(a):
    return H0*sqrt(Omega_m/(a**3)+Omega_l)

def func(y, t):
    return [-1./(t*H(t)), -1./(t**2*H(t))]

t = arange(1, 0.3, -0.01)
y0 = [0, 0]

y = odeint(func, y0, t)

plot(t, y)
```

This gives us the following solutions for the Friedmann equations:

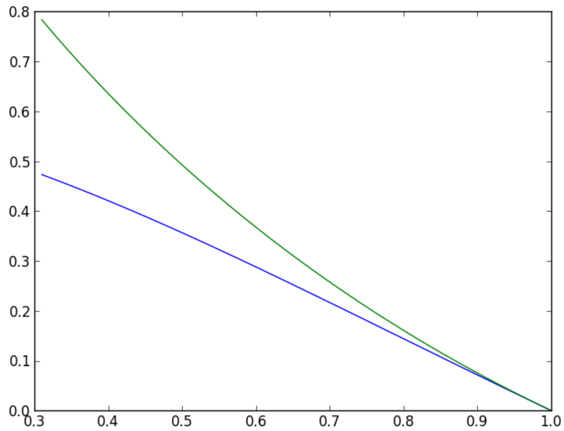

Figure 5: Solution for the Friedmann equations with $H_0 = \Omega_m = \Omega_l = 1$