Exercise 01

Submit your files before next Friday at 8.00 am to luca.donati@fu-berlin.de

1.1 Series limit (10 Points)

In this exercise, you can practice the use of for and while loops. Consider the convergent series

$$\log(x+1) = \sum_{n=0}^{\infty} (-1)^n \frac{x^{n+1}}{n+1} \qquad x \in (-1, 1]$$

(a) Write a program that numerically estimates the value of the series using a for-loop. Pick a real number x and test the convergence of the series.
Plot the estimate as a function of N up to$N = 20$.
Try to vectorize the script.

(b) Write a program which evaluates the series until the absolute value of the increment is less than a pre-defined threshold. How many terms are needed until the increment is less than 10^{-2}, 10^{-4}, 10^{-6}, 10^{-8}?

(c) Often one does not know in advance how long it will take until a certain threshold is met. To prevent a program from running too long, one stops the calculation if either the threshold or a maximum number of iterations is reached. Modify the program from (b) such that it stops if one of the following conditions is met:

- the increment is less than a predefined threshold,
- a maximum number of terms have been evaluated.

1.2 Wave packet dynamics (15 Points)

The Schrödinger Equation of the harmonic oscillator is:

$$-\frac{\hbar^2}{2m}\frac{\mathrm{d}^2}{\mathrm{d}x^2}|\phi\rangle + V(x)|\phi\rangle = E|\phi\rangle$$

with

$$V(x) = \frac{m\omega^2}{2}x^2$$

The solutions are the functions:

$$\phi_n(x) = N_n \cdot H_n(\sqrt{\alpha}x) \cdot \exp\left(-\frac{1}{2}\alpha x^2\right)$$

with

$$\alpha = \frac{m\omega}{\hbar}$$

$$N_n = \frac{1}{\sqrt{2^n n!}}\left(\frac{\alpha}{\pi}\right)^{1/4}$$

and $H_n(\sqrt{\alpha}x)$ are the Hermite polynomials. The eigenvalues of the Schrödinger Equation are:

$$E_n = \hbar\omega\left(n + \frac{1}{2}\right)$$

The time-dependent superposition of the eigenstates is the function:

$$\psi(x,t) = \frac{1}{N}\sum_{n=0}^{\infty} c_n \cdot \phi_n(x) \cdot \exp\left(-\frac{iE_n t}{\hbar}\right)$$

with

$$N = \sqrt{\sum_{n=0}^{\infty} |c_n|^2}$$

You are going to write a a program that plots the time-dependent superposition of the harmonic oscillator eigenstates. The program is made by two files, wavepacket.py is the main program, userfunctions.py contains the necessary functions.
Let's start with the file userfunctions.py: In the first lines, we import the necessary modules:

```
import numpy as np                                  #numpy
from numpy.polynomial.hermite import hermval        #Hermite polynomials
from scipy.misc import factorial                    #factorial
```

Immediately after, we define the parameters of the oscillator. Defining these variables before the functions, makes them "global". In this way, it is not necessary to pass them to all the functions every time we need.

```
h_bar = 1.
omega = 1.
m = 1.
alpha = m*omega / h_bar
L = 20. #length of the x-axis
numStates = 4   #Superposition of the first 4 states
c = np.empty(numStates)

c[0] = 1 #coefficients for the superposition
c[1] = 0 #try different values
c[2] = 1
c[3] = 0
```

Now we are going to define two functions, one to compute the eigenstates and one to compute the superposition.

```
def eigenstates(numStates,x):    #This function accepts the number of states
                                 #and an array x with the position
    phi = np.empty((numStates,len(x)))  #eigenstates
    Nn = np.empty(numStates)     #normalizing constants
    E = np.empty(numStates)      #eigenvalues

    for n in range(numStates):   #we compute Eigenvalues, Normalization
                                 #constants and eigenstates for each state
        E[n] = ...
        Nn[n] = ...

        h=np.zeros(n+1) #necessary for hermval()
        h[n]=1          #If you are interested look for
                        #hermval()@Hermite Polynomials

        phi[n,:] = ... * hermval(x, h) * ...

    return phi, E       #the function returns both the eigenstates
                        #and the eigenvalues

def superposition(numStates,t,phi,E):    #we pass to the superposition function:
                                         #number of states, time, eigenstates
                                         #and eigenvalues
    psi = 0
    N = 0       #Normalizing constant
    for n in range(numStates):
        N = N + ...
        psi = ...

    N = ...
```

```
    return ...
```

Now we are going to write the main program wavepacket.py. As usual we start importing the modules and the functions of the file userfunctions.py

```
import numpy as np
import matplotlib.pyplot as plt
from userfunctions import *
```

We proceed setting the other parameters.

```
Nsteps = 100
dt = 0.1 #time resolution

x = np.linspace(-L/2,L/2,10000) #to create a x-axis with length=L
                                #centered at 0
delta_x = x[1]-x[0]

V = ... #potential V(x)
```

Now we compute the eigenstates and the eigenvalues:

```
phi, E = ...
```

To compute the superposition of eigenstates, we start a for block.

```
f, ax = plt.subplots(2, sharex=True) #subplot creates more graphs
                                     #in the same window

for step in range(Nsteps):
    t = step*dt #real time

    psi = ...

    ax[0].plot(x,np.real(psi)) #plot the real part
    ax[0].plot(x,np.imag(psi),color='r') #plot the imaginary part
    ax[0]...    #plot the potential V
    ax[0]...    #add graphical details (labels, legend,...)

    ax[1].plot(x,...) #plot the prob. distribution |psi|^2
    ax[1]...    #plot the potential V
    ax[1]...    #add graphical details (labels, legend,...)

    plt.pause(0.01) #for the animation
    ax[0].cla()
    ax[1].cla()

    print "Time: ", t
```