FYS-4096 Computational Physics: Exercise 1

Ilkka Kylänpää

Calendar week 2

Problem 1: Linux (Mac or Windows), GitLab, Python
Set up the environment for the course, so that you are able to complete the required tasks. Below are options and requirements that need to be considered.

1. (a) If you have a Windows laptop, you can install Ubuntu Linux 18.04 (or higher) on a virtual machine on your computer.
   - This is perhaps the best and the most convenient way for completing the course.

1. (b) Or you can use Windows / Mac as long as you can get the programs etc. at part 3 of this problem working.

1. (c) Alternatively, you can try managing the course using, e.g., the University's Linux server (linux-ssh.tuni.fi or linux-desktop.tuni.fi).
   - Instructions at https://intra.tuni.fi/en/handbook?page=2640
   - Requires for example Xming and Putty on Windows machines.

2. Create a GitLab account at gitlab.com and email ilkka.kylanpaa@tuni.fi your Gitlab username and corresponding email address (at least one of these will be needed for providing you help files later in the course). Setting up SSH keys for GitLab following https://docs.gitlab.com/ce/ssh/README.html is also encouraged.

3. In general you need to have access to (at least):
   - Python 3, Numpy, Scipy, Matplotlib, Git, OpenSSH client, xcrysden, ...
   - text editor, e.g., Emacs, Vim, Nano, Geany, ...
   - you might also be interested in numba, pandas, and Mayavi for Python 3

4. For this problem, create a file `problem1.txt` in which you very briefly describe the environment you are going to use in the course, e.g., "I'm going to use Linux", "I'm going to use Mac", "I'm going to use Windows with Linux bash shell", etc.
   - The file `problem1.txt` need be added into the exercise1 folder, once you have created it in the next problem.

Problem 2: Creating a Git repository for the course and the first exercise

- Open up a gnome terminal.
- Create a folder for the course to your preferred location (create folder: `mkdir CompPhys`, go to the folder: `cd CompPhys`, leave the folder: `cd ..`).
- Go to CompPhys folder and create in it an empty folder called exercise1 (`cd CompPhys`, `mkdir exercise1`).
- At CompPhys folder initialize an empty Git repository from the command line with `git init`
- Make a README.md file in your repository, for example, `emacs README.md &`. Write a short description of the contents and purpose of the directory (e.g., CompPhys home directory).

- Make a README.md file also into the exercise1 folder (e.g., Ex 1 README file, setting up environments, and simple initial python coding).
- Create a GitLab project (or repository) at gitlab.com and name it CompPhys (or something similar). Push the local CompPhys repository from your computer to GitLab following the guide on your newly generated project page (gitlab.com/CompPhys). It should be something like below, but with your details:

```
git init
git remote add origin https://gitlab.com/{your_username}/CompPhys.git
git add .
git commit -m "Initial commit for the course"
git push -u origin master
```

Problem 3: Programming exercise

(a) Write a Python function that numerically estimates the first derivative of a single argument Python function. The derivative formula should have $\mathcal{O}(h^2)$ error term. Write this in a file called num_calculus.py and use the following structure in the file

```
""" comments for the file here """
# import needed packages, e.g., import numpy as np

def first_derivative( function, x, dx ):
    # necessary commenting and code here

def test_first_derivative():
    # necessary commenting and code here

def main():
    # necessary commenting and testing here

if __name__=="__main__":
    main()
```

For performing the tests you should write on the command line `python3 num_calculus.py`. Test function should determine whether the first_derivative function is working, and thus, you need to compare the value obtained from the function with a known answer (that you can determine analytically).

Moreover, you would like to be able to use your function more generally in other python scripts, for example, as

```
from num_calculus import first_derivative

def fun(x): return 3*x**2

der = first_derivative(fun, 0.8, 0.001)
```

For this Python needs to know where the file num_calculus.py is located, and thus, run command `pwd` on command line to find out the location. Select and copy the location. Then on command line write

```
export PYTHONPATH=$PYTHONPATH:copied_location
```

This will temporarily add the specific location to PYTHONPATH. For more permanent use the export command should be added to `.bashrc` file.

Notice: If the file is located in the same folder as the file you want to use it, then it will be found automatically. So, specifying the location is not necessary in this exercise, but knowledge of this could be useful later.

(b) Add function for the second derivative called `second_derivative` with appropriate testing.

(c) Implement trapezoid or Simpson integration function for the case where the integrand function values are given on a uniform grid. Remember test function(s). You should be able to use it, for example, as

```
from num_calculus import simpson_int
import numpy as np

x = np.linspace(0,np.pi/2,100)
f = np.sin(x)

I = simpson_int(f,x)
```

(d) Implement a function for Monte Carlo integration using the code below. Add comments and tests.

```
import numpy as np

def monte_carlo_integration(fun,xmin,xmax,blocks=10,iters=100):
    block_values=np.zeros((blocks,))
    L=xmax-xmin
    for block in range(blocks):
        for i in range(iters):
            x = xmin+np.random.rand()*L
            block_values[block]+=fun(x)
        block_values[block]/=iters
    I = L*np.mean(block_values)
    dI = L*np.std(block_values)/np.sqrt(blocks)
    return I,dI

def func(x):
    return np.sin(x)

def main():
    I,dI=monte_carlo_integration(func,0.,np.pi/2,10,100)
    print('Integrated value: {0:0.5f} +/- {1:0.5f}'.format(I,2*dI))

if __name__=="__main__":
    main()
```

Problem 4: Numerics and plotting

- Within the same git repository (exercise1), create a Python script called convergence_check.py. This script should illustrate the convergence properties of the functions you implemented in Problem 3.
- Select non-trivial test functions for which you can obtain analytic values for the derivatives and the definite integrals, e.g., a polynomial, $\sin(x)$, $\cos^2(x)$, $\exp(-ax)\cos(x)$, ... It's OK to use, e.g., $\sin(x)$ in all cases.
- Plot the absolute error of the numerically obtained estimate as a function of grid spacing dx for the first derivative and for one of the integration methods (Simpson or trapezoid). Plot both curves into the same figure and use legend and different line style.
- Consider how the error scales and if it is according to the expected scaling for your implementation.
- For plotting you can modify the code below. You might like to consider making your plotting script a callable function, e.g., `my_plot(x,f,...)`.
- For more information on plotting you might like to visit matplotlib related pages such as pyplot intro, sample plots, customizing pyplot, and figure environment.

```
import numpy as np
import matplotlib.pyplot as plt

plt.rcParams['legend.handlelength'] = 2
plt.rcParams['legend.numpoints'] = 1
plt.rcParams['text.usetex'] = True
plt.rcParams['font.size'] = 12

fig = plt.figure()
# - or, e.g., fig = plt.figure(figsize=(width, height))
# - so-called golden ratio: width=height*(np.sqrt(5.0)+1.0)/2.0
# - width and height are given in inches (1 inch is roughly 2.54 cm)

ax = fig.add_subplot(111)

x = np.linspace(0,np.pi/2,100)
f = np.sin(x)
g = np.exp(-x)

# plot and add label if legend desired
ax.plot(x,f,label=r'$f(x)=\sin(x)$')
ax.plot(x,g,'--',label=r'$f(x)=\exp(-x)$')

# include legend (with best location, i.e., loc=0)
ax.legend(loc=0)

# set axes labels and limits
ax.set_xlabel(r'$x$')
ax.set_ylabel(r'$f(x)$')
ax.set_xlim(x.min(), x.max())

fig.tight_layout(pad=1)
```

```
# save figure as pdf with 200dpi resolution
fig.savefig('testfile.pdf',dpi=200)
plt.show()
```

Problem 5 (Extra problem): Setting up a Python program

- This is additional, but highly encouraged problem, since it will be beneficial for developers using Python code and possibly publishing their Python codes.
- Code the `first_derivative` function (of Problem 3a) within a multi-file Python project structured according to

  `http://python-packaging.readthedocs.io/en/latest/minimal.html`

- Name your Python module as num_calculus and the file name for the derivative should be differentiation.py. Then your function can be taken into use as

  `from num_calculus.differentiation import first_derivative`

- Unit testing should be possible as `python3 setup.py test`

Returning your exercise

1. Create a file "solvedProblems.txt" in your exercise1 folder. Inside it, write a comma separated list of problems you have solved, e.g., 1,2,3.
2. Notice that the "solvedProblems.txt" needs to be found in the folder.
3. If you have only partially finished a problem you should indicate that as well. For example, in the list use notation: 3a and b, or 4x, where x indicates that you have partially done problem 4, but it is unfinished or not working as expected. You could also use 3ax if needed. Below the comma separated list you can give more detailed descriptions if you like.
4. Make sure all your source files are under version control and push them to GitLab (e.g., at CompPhys folder write `git add .` , then commit with message `git commit -m "Message here"` and `git push -u origin master`).
5. Notice: Do not include the python environment files into your exercise or CompPhys course repository.
6. If you want, you may tag your final solution with "final" tag: `git tag final`
7. Push your commits (and the final tag) to GitLab **before noon on Monday January 18**:
   `git push --all && git push --tags`
8. Notice: No late submissions!! Unless discussed with the teacher well in advance.
9. Remember to share your GitLab project with the teacher before or right after the final git push. Be careful in providing proper permissions (Reporter, Developer or Maintainer), so that the teacher can also access the data. (Teacher's GitLab account: kylanpaait, ilkka.kylanpaa@tuni.fi).