ACM40080 Project 2

Ronan McCormack 17328461

Report with explanantions

Along with this report I've uploaded a .zip file containing the scripts for Q1, Q2 and Q3.

These were zipped within orr2 so all permissions should be preserved.

Q1

[5%] Write a linux script which will:

Use curl to download Shakespeare's play "The Tempest" from http://shakespeare.mit.edu/ (http://shakespeare.mit.edu/)

Use sed to remove all html content, and save the contents of the play as a text file called TheTempest.txt

Solution

Using vim, start a script 'script1.sh'.

#!/bin/bash

Use curl to import the file

```
curl http://shakespeare.mit.edu/tempest/full.html > TheTempest.txt
```

All HTML content is located within '<>'.

First, remove the table. which can be found between the patterns '<table' and 'table>' spannng multiple lines:

```
sed -i '/<table/,/table>/d' TheTempest.txt
```

Remove all instances of terms beginning with '<' and ending with '>' contained within a single line:

```
sed -i 's/<[^<]*>//g' TheTempest.txt
```

The html content remaining must be spanning two lines, meaning lines starting with '<' or lines ending with '>'. Hence, remove all such lines:

```
sed -i '/</d' TheTempest.txt
sed -i '/>/d' TheTempest.txt
```

After running, the file 'TheTempest.txt' will appear, free of any HTML content.

Q2

[10%] Write a linux script which will:

Use wget inside a loop to download daily data from Met Éireann for the four stations listed in stations.csv:

Data can be downloaded from the url: https://cli.fusio.net/cli/climate_data/webdata/dlySTNO.zip (https://cli.fusio.net/cli/climate_data/webdata/dlySTNO.zip) where STNO is the station number in stations.csv

Use awk to analyse the data you have downloaded, and create a single text file report which presents the following information for each station :

the overall average air temperature

the date of the highest max temperature

the date of the lowest min temperature

Solution

Copy the file 'stations.csv' into the local directory.

Using vim, begin a script 'script2.sh'.

- The important detail is the station number. Using awk, extract these numbers (the second column in stations.csv) and save in a new file "stno.txt":

```
awk -F , '{print $2}' stations.csv >stno.txt
```

- Using a for loop, cycle through each station number in stno.txt. For each number, use wget to download the data from the matching url, and save to a new file:

```
for STNO in `cat stno.txt`
do
 wget -O ${STNO}.csv https://cli.fusio.net/cli/climate_data/webdata/dly${STNO}.csv
```

- Visually inspecting the data files, the first several lines contain undesirable information. Amend the files so that only the numerical data remain. The lines with data all contain the pattern ",0,". Within the loop, using sed, delete all lines that do not contain this pattern ('!d' deletes all lines but for those containing the pattern):

```
sed -i '/,0,/!d' ${STNO}.csv
```

- **Average temperature** - Maximum temperature is stored in columnn 3 and minimum in column 5. The best estimate for the average air temperature on a given day would be the average of the min and the max (($3 + $5)/2). Get the average of these daily averages for the overall average temperature. This can be done using the following awk command:

```
awk -F , '{ total += ($3 + $5)/2; count++ } END { print total/count }' ${STNO}.csv
```

This is cycling through the rows of the data file, adding each daily average temperature to the variable 'total', and incrementing the variable 'count' by 1. It returns the overall average given by 'total/count'.

Precede this with echo and append to the report text file:

```
echo "${STNO}: $(awk -F , '{ total += ($3 + $5)/2; count++ } END { print total/count }' ${STNO}.csv)">>report.txt
```

- **Date of highest max temperature** - Max temperature is stored in column 3. For each station, use an 'if' loop in awk to cycle through each row in column 3 and pick put the max value, saving this value in the variable 'max'. At the 'END' of the loop, print the date (column 1), as well as the temperature:

```
awk -F , -v max=0 '{if($3>max){date=$1; max=$3}}END{print date, max}' ${STNO}.csv)
```

Precede this with echo and append to the temporary file 'hightemp.txt':

```
echo "${STNO}: $(awk -F , -v max=0 '{if($3>max){date=$1; max=$3}}END{print date, max}' ${STNO}.csv)">>hightemp.txt
```

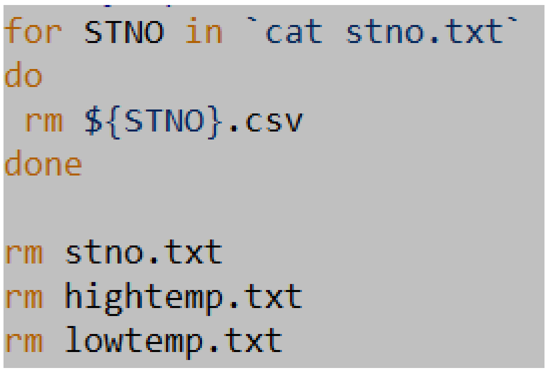
- **Date of lowest min temperature** - Same idea. Min temperature is stored in column 5. Replace '>' with '<', append to file 'lowtemp.txt':

```
echo "${STNO}: $(awk -F , -v min=0 '{if($5<min){date=$1; min=$5}}END{print date, min}' ${STNO}.csv)">>lowtemp.txt
```

- Append hightemp.txt and lowtemp.txt to the report file, with appropriate titles.

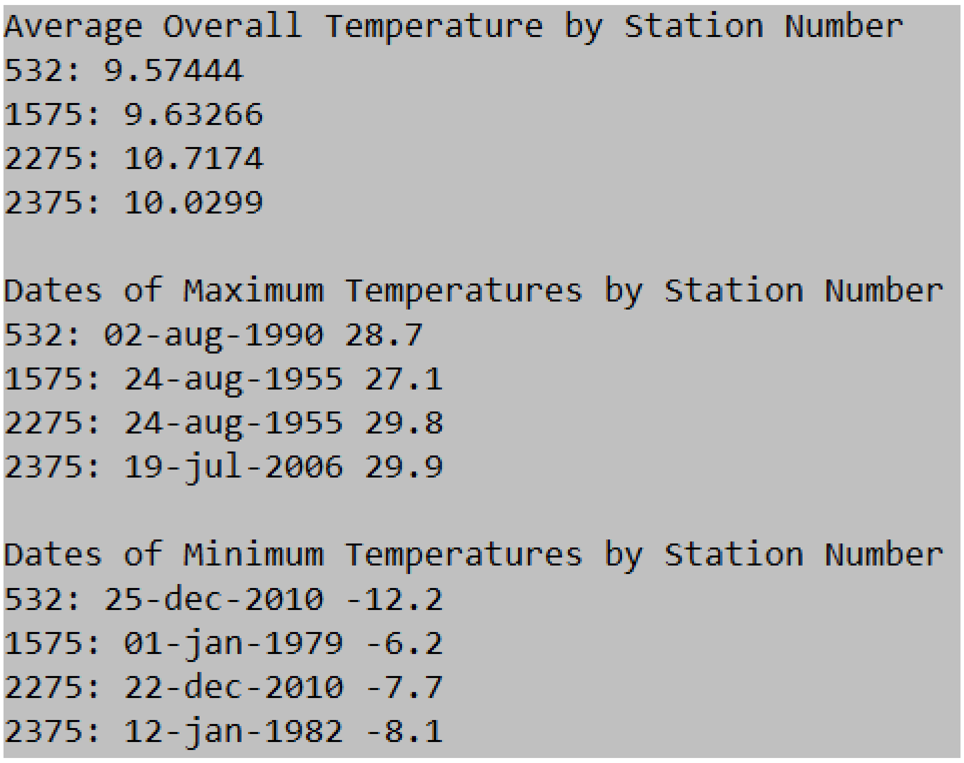
```
echo -e "\nDates of Maximum Temperatures by Station Number">>report.txt
cat hightemp.txt>>report.txt
echo -e "\nDates of Minimum Temperatures by Station Number">>report.txt
cat lowtemp.txt>>report.txt
```

- Delete the temporary files:



```
for STNO in `cat stno.txt`
do
 rm ${STNO}.csv
done

rm stno.txt
rm hightemp.txt
rm lowtemp.txt
```

Once executed, the final report is as follows:



```
Average Overall Temperature by Station Number
532: 9.57444
1575: 9.63266
2275: 10.7174
2375: 10.0299

Dates of Maximum Temperatures by Station Number
532: 02-aug-1990 28.7
1575: 24-aug-1955 27.1
2275: 24-aug-1955 29.8
2375: 19-jul-2006 29.9

Dates of Minimum Temperatures by Station Number
532: 25-dec-2010 -12.2
1575: 01-jan-1979 -6.2
2275: 22-dec-2010 -7.7
2375: 12-jan-1982 -8.1
```

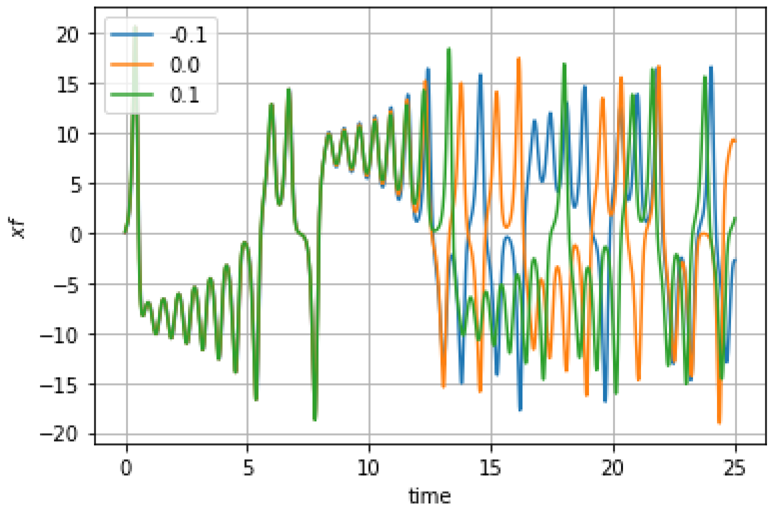
Q3

[10%] Write a linux script which will:

Use sed in a loop to create three python codes, using lorenz_x.py as a template, where each code has a different initial x value [-0.1, 0.0, 0.1].

Use task farming to run these python codes on separate cores of orr2, and save all calculated values of xf to a file.

Create a png image plot of the three different xf time series, like this:



Solution

Overview

Five scripts are written to complete this task.

- A bash script '**modlorenz.sh**' to modify the template file.
- A qsub script '**lorenz.qsub**' that generates python scripts based on the template for each inital x value, and then runs these on inividual cores, i.e. task farming.
- A python script '**plot.py**' that plots the 'xf' outputs against time 't' and saves the figure as 'plot_lorenz.png'.
- A qsub script that runs '**plot.py**' and deletes unwanted files.
- A master bash script '**master3.sh**' that dictates the running of all scripts, ensuring that the first qsub submission has been completed before submitting the second. This is the only script that needs to be run.

The template file 'lorenz_x.py' must be present in the working directory also

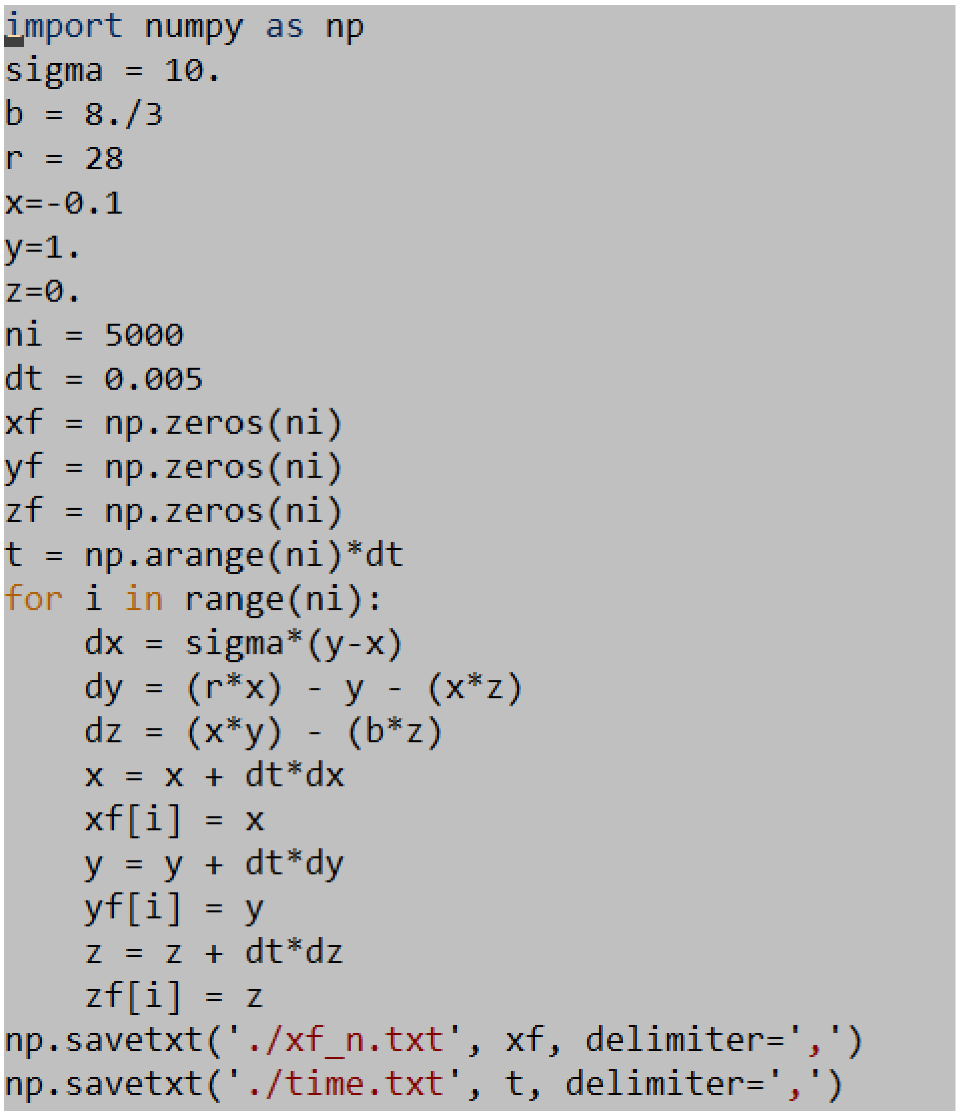
---

- **Template** - Copy in lorenz_x.py to the directory. This needs few amendments, which will be carried written in the script 'modlorenz.py':

Two lines must be appended to the end of the template - one to save the 'xf' array to a file 'xf_n.txt', the other to save the 't' array to a file 'time.txt'. The 'n' will be replaced in the 'lorenz.qsub' script.

```
echo "np.savetxt('./xf_n.txt', xf, delimiter=',')" >>lorenz_x.py
echo "np.savetxt('./time.txt', t, delimiter=',')" >>lorenz_x.py
```

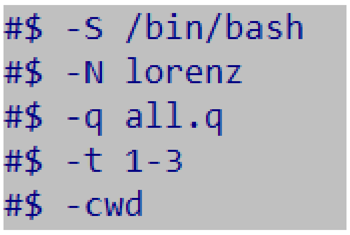
And so, the amended template 'lorenz_x.py' is as follows:



```
import numpy as np
sigma = 10.
b = 8./3
r = 28
x=-0.1
y=1.
z=0.
ni = 5000
dt = 0.005
xf = np.zeros(ni)
yf = np.zeros(ni)
zf = np.zeros(ni)
t = np.arange(ni)*dt
for i in range(ni):
    dx = sigma*(y-x)
    dy = (r*x) - y - (x*z)
    dz = (x*y) - (b*z)
    x = x + dt*dx
    xf[i] = x
    y = y + dt*dy
    yf[i] = y
    z = z + dt*dz
    zf[i] = z
np.savetxt('./xf_n.txt', xf, delimiter=',')
np.savetxt('./time.txt', t, delimiter=',')
```

- **lorenz.qsub** - This script will be submitted to the q, generating and running three python scripts to output three sets of 'xf' and time values 't'.

Start the script with the standard commands. Here, the task is named 'lorenz', it will be submitted to all q's (all.q), the the code will run on three different cores (1-3), and everything is directed to the current working directory (-cwd):



```
#$ -S /bin/bash
#$ -N lorenz
#$ -q all.q
#$ -t 1-3
#$ -cwd
```

Using echo, write the desired inital x values, with linebreaks (\n), into the file 'xvals':

```
echo -e "-0.1\n0.0\n0.1" >xvals
```

Now to generate the three python scripts based on the template lorenz_x.py. First, initialise a variable 'count'. This will be incremented and used to attach integers from 1 to 3 to the script filenames. Begin a 'for' loop to cycle through the values in 'xval'.

```
count=0
for val in $(cat xvals)
do
 let count+=1
```

Within the loop, using sed, replace all instances of 'x=' in the template file with an inital x value from xval, and save to the new file 'lorenz_n.py' (n an integer from the incrementing count variable):

```
 sed "s/x=/x=$val/" ./lorenz_x.py >./lorenz_${count}.py
```

Use another sed command to replace instances of 'xf_n' in the same script with the count number replacing 'n':

```
 sed -i "s/xf_n/xf_$count/" ./lorenz_${count}.py
```

End the loop

The three python scripts have been generated.

- **Execute python scripts in 'q'** - Having downloaded anaconda to my home directory, the command 'which python' returns '~/anaconda3/bin/python'. Hence, to execute a python script in the 'q', the script must be preceded with this.

The global variable 'SGE_TASK_ID' stores the number of the core on which a task is being run in the q. Seeing as the cores specified in the header are '1-3', the core numbers are numbered 1, 2 and 3, and so by specifying the lorenz script number using this variable one script is run on each core:

```
~/anaconda3/bin/python ./lorenz_$SGE_TASK_ID.py
```

Once these scripts have been run, files xf_1.txt, xf_2.txt and xf_3.txt containing the final x values and time.txt are output.

- **plot.py** - The data needed for a plot are now generated and have been stored in the current working directory. Create a separate python script, 'plot.py' for generating the plot.

First, import the necessary libraries:

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

The data must be loaded into local variables:

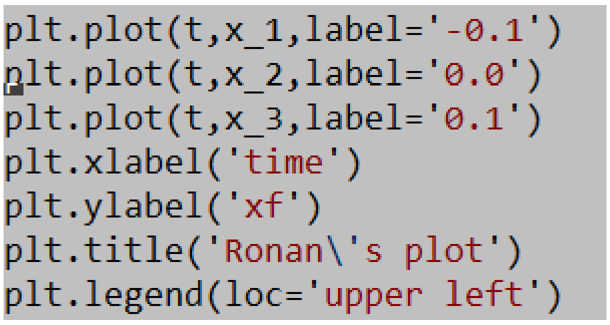
```
xf_1 = np.loadtxt('./xf_1.txt')
xf_2 = np.loadtxt('./xf_2.txt')
xf_3 = np.loadtxt('./xf_3.txt')
time = np.loadtxt('./time.txt')
```

The data are not yet in a plottable format. To remedy this, generate arrays for each via a 'for' loop:

```
x_1 = []
for line in xf_1:
    x_1.append(float(line))

x_2 = []
for line in xf_2:
    x_2.append(float(line))

x_3 = []
for line in xf_3:
    x_3.append(float(line))

t = []
for line in time:
    t.append(float(line))
```

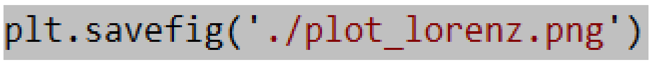
Now the data are ready for plotting.

Plot each 'x' array vs 't', labelling them by the corresponding initial x value. Label the x and y axes 'time' and 'xf' respectively. Give the plot a title and a legend located in the upper left corner:



```
plt.plot(t,x_1,label='-0.1')
plt.plot(t,x_2,label='0.0')
plt.plot(t,x_3,label='0.1')
plt.xlabel('time')
plt.ylabel('xf')
plt.title('Ronan\'s plot')
plt.legend(loc='upper left')
```

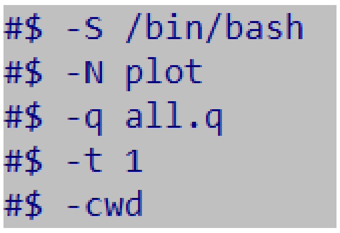
Save the plot as a png file back in the working directory:



```
plt.savefig('./plot_lorenz.png')
```

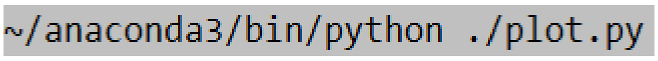
- **plot.qsub** - Second qsub script for submitting 'plot.py' to the q

Start the script with the same commands as before, though specify that the script be run on only one core this time (-t 1):



```
#$ -S /bin/bash
#$ -N plot
#$ -q all.q
#$ -t 1
#$ -cwd
```

Call upon and run the python plot script that's just been written:



```
~/anaconda3/bin/python ./plot.py
```

Now to tidy up and remove all unnecessary and used files

First, delete the three data-generating lorenz scripts, and the 'xf_n.txt' files they generated, with the help of a 'for' loop:

```
num=0
for i in $(cat xvals)
do
  let num+=1
  rm ./lorenz_${num}.py
  rm xf_${num}.txt
done
```

Now delete 'xvals' and 'time.txt', as well as the error and output files from the two qsub jobs:

```
rm xvals
rm time.txt
rm lorenz.e* lorenz.o*
rm plot.e* plot.o*
```

- **master3.sh** - The only script that needs to manually be run.

Firstly, the 'modlorenz.sh' script is run:

```
./modlorenz.sh
```

Then, submit the first of the two qsub scripts, 'lorenz.qsub', to the q:

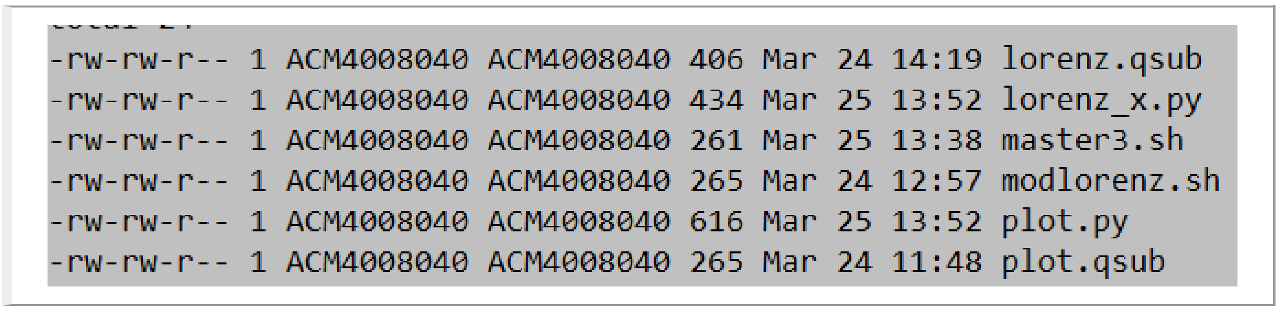
```
qsub lorenz.qsub
```

Finally, submit the second qsub script, 'plot.qsub', to the q. The option here instructs the q to wait until the first job 'lorenz' has been completed before starting the 'plot' job:

```
qsub -hold_jid lorenz ./plot.qsub
```

The Outcome

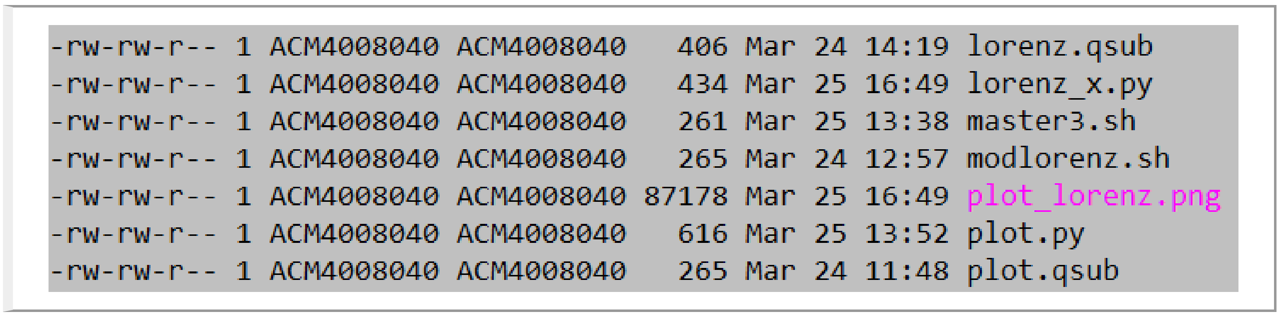
Before running, the working directory should resemble the following:



```
-rw-rw-r-- 1 ACM4008040 ACM4008040 406 Mar 24 14:19 lorenz.qsub
-rw-rw-r-- 1 ACM4008040 ACM4008040 434 Mar 25 13:52 lorenz_x.py
-rw-rw-r-- 1 ACM4008040 ACM4008040 261 Mar 25 13:38 master3.sh
-rw-rw-r-- 1 ACM4008040 ACM4008040 265 Mar 24 12:57 modlorenz.sh
-rw-rw-r-- 1 ACM4008040 ACM4008040 616 Mar 25 13:52 plot.py
-rw-rw-r-- 1 ACM4008040 ACM4008040 265 Mar 24 11:48 plot.qsub
```

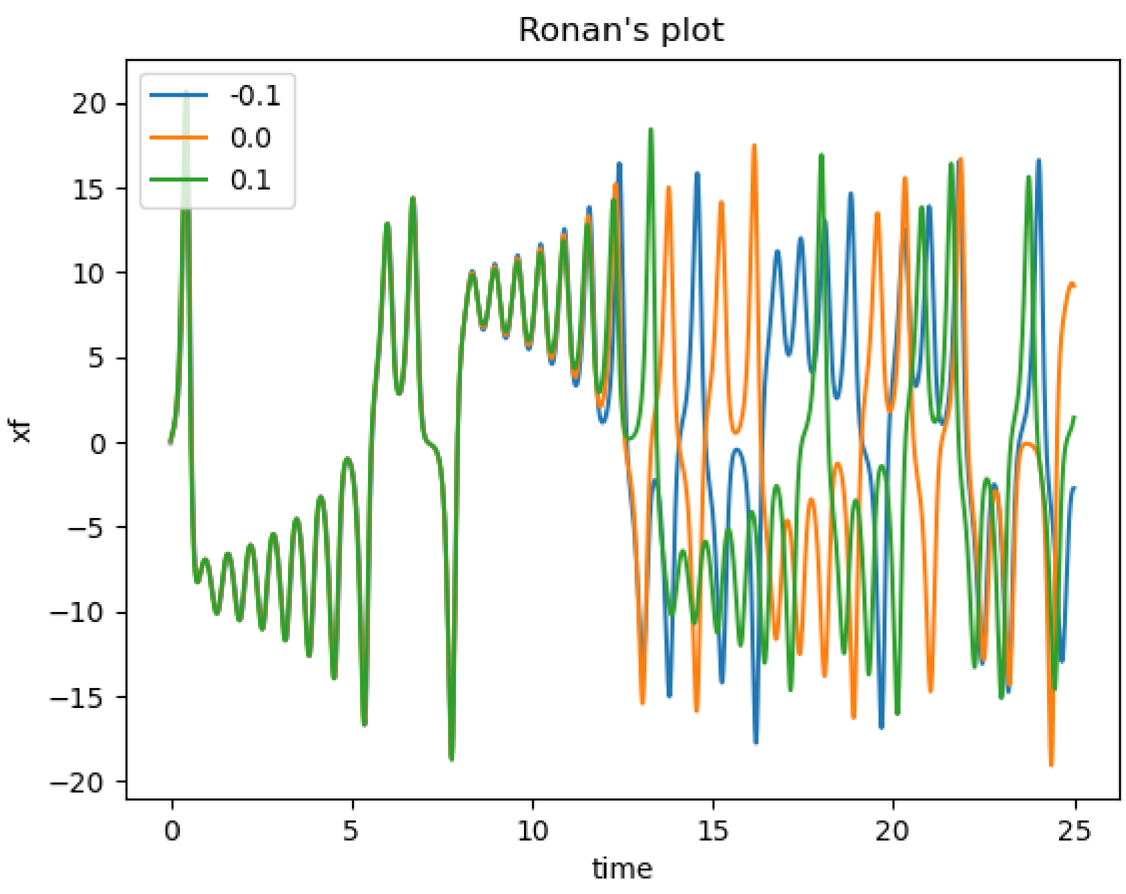
Shortly after executing the command './master3.sh',

The file 'lorenz_plot.png' should appear in the directory:



```
-rw-rw-r-- 1 ACM4008040 ACM4008040   406 Mar 24 14:19 lorenz.qsub
-rw-rw-r-- 1 ACM4008040 ACM4008040   434 Mar 25 16:49 lorenz_x.py
-rw-rw-r-- 1 ACM4008040 ACM4008040   261 Mar 25 13:38 master3.sh
-rw-rw-r-- 1 ACM4008040 ACM4008040   265 Mar 24 12:57 modlorenz.sh
-rw-rw-r-- 1 ACM4008040 ACM4008040 87178 Mar 25 16:49 plot_lorenz.png
-rw-rw-r-- 1 ACM4008040 ACM4008040   616 Mar 25 13:52 plot.py
-rw-rw-r-- 1 ACM4008040 ACM4008040   265 Mar 24 11:48 plot.qsub
```

Using scp to copy the image onto the local PC and viewing, this is the resulting plot:



That concludes this report.