Multi-run script and Automated fits

1 Abstract

The exercise begins by reading matrix dimension from a file in order to define square matrices and then changing this number moving from N_{min} value to N_{max} value.

During python codes execution the programs write into three different files: on first column,matrix dimension and on second one time consuming related to one specific method implemented,that is ***by rows,by columns,by matmul function***.

Then,using these files and GNUPLOT software, i create a graph with these three different columns : one for method used.

In order to discover how interpolate plotted points i tried to implement the biggest possible difference between N_{min} and N_{max} .
Finally ,using an another python code,i realized automated fits like(but not only) previous one.

2 Theory

Matrix-matrix multiplication is defined as :

"Be K a ring . Be A and B matrices with m and n rows and n and l columns respectively . Be a_{ij} and b_{ij} A's elements and B's elements respectively that laid in K.
It will be defined C=AB with m rows and l columns as matrix product if named c_{ij} C's elements it will have to :

$$c_{ij} = \sum_{r=1}^{n} a_{ir} b_{rj} \tag{1}$$

for each row i and column j."

3 Code development

From the beginning my python and fortran codes are thought to optimize as much as possible time consuming processes.
In order to achieve this aim ,firstly, my programs write matrix dimension into a file and then they read this file, storing data into an list, and finally use this list to update new matrix dimension.

Using python codes ,the widest interval corresponds to $N_{min} = 100$ and $N_{max} = 1200$ because exceed this limit the python programs spend too much time to finish.

This also the reason why i chose to not define two functions in order to implement methods but i divided code in parts and used two Colab accounts and my personal pc (only for matmul function) and i launched them simultaneously.

Farther,opening three files simultaneously and writing one single line for each loop iteration or calling functions for each loop iteration,it means that time execution programs went on for more than 8 hours!

Finally they store matrix dimension and time consuming by methods in three different files and plots everything in one graph using GNUPLOT software.

```
gnuplot> f(x)=3*log(x)-13.8
gnuplot> g(x)=3*log(x)-23.3
gnuplot> h(x)=3*log(x)-14.2
gnuplot> plot "by_rows_(1).txt" using ($1):(log($2)) t "by rows" with lp,"by_columns.txt" using ($1):(log($
2)) t "by columns" w lp, "by_matmul.txt" using ($1):(log($2)) t "using matmul " w lp,f(x) t "3*log(x)-13.8"
,g(x) t"3*log(x)-23.3",h(x) t "3*log(x)-14.2"
gnuplot> set xlabel "matrix dimension"
gnuplot> set ylabel "log time consuming"


gnuplot> replot
gnuplot> set xrange[:1300]
gnuplot> set key top left



gnuplot> replot
gnuplot> set xtics 100
```

Otherwise using the same fortran code shown the week before,I enlarged the difference between N_{min} and N_{max}.
Now the limit reached is 4200!.

Farther,in order to estimate if interpolating functions match with results,i wrote an another python code that plots three ***residual**** graphs one after another.

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

print('Carico file txt')

m_dim=[]
t1=[]
t2=[]
t3=[]
value= 0.0
with open('residuals.txt', encoding='utf-8') as a_file:
    for a_line in a_file:
        #print( a_line.rstrip().split()    )
        for i in range(0,4) :
            value=a_line.rstrip().split()[i]
            if i==0:
                m_dim.append(int (value) )
            if i==1:
                t1.append(float(value) )
            if i==2:
                t2.append(float(value) )
            if i==3:
                t3.append( float(value)  )

title=['by rows','by columns','by matmul']
t=[t1,t2,t3]

for i in range(3):
    fig,ax = plt.subplots(figsize=(10,8) )
    ax.set_xlabel('matrix_dimension', fontsize = 15)
    ax.set_ylabel('residuals',fontsize=15)
    ax.set_title(title[i],fontsize=15)
    ax.set_xlim([600,4100])
    ax.set_ylim([0,1.2])
    plt.yticks(np.arange(0, 1.2, step=0.2))
    plt.xticks(np.arange(600, 4200, step=200))
    plt.plot(m_dim,t[i], label=str(title[i]) )
    plt.legend()
    plt.show()
```

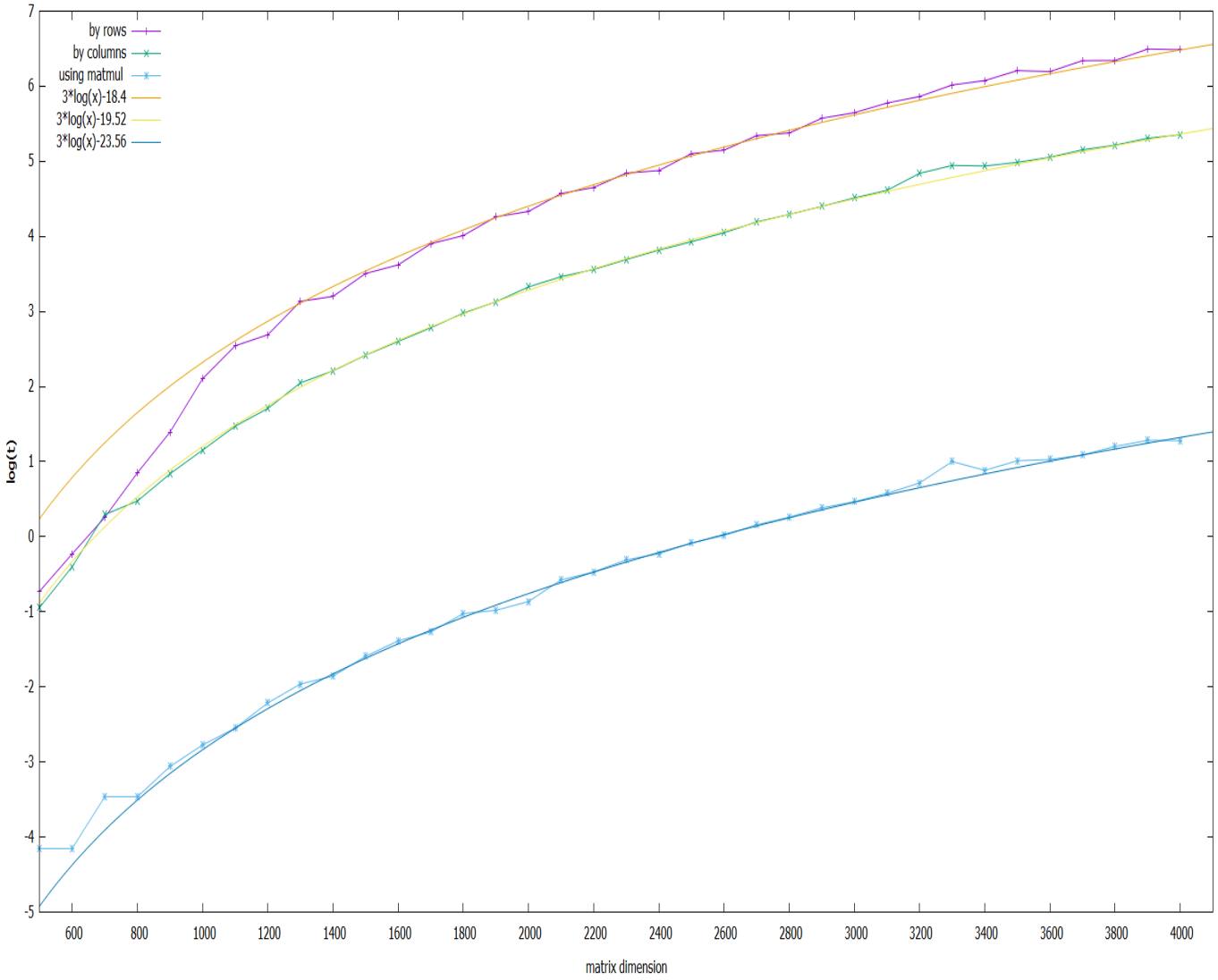
*For *residual* i mean :

$$|y_{th} - y_{ex}| \tag{1}$$

in this case y_{th} will be a log functions

4 Results

In the following i'll show my results both using python codes and GNUPLOT software.



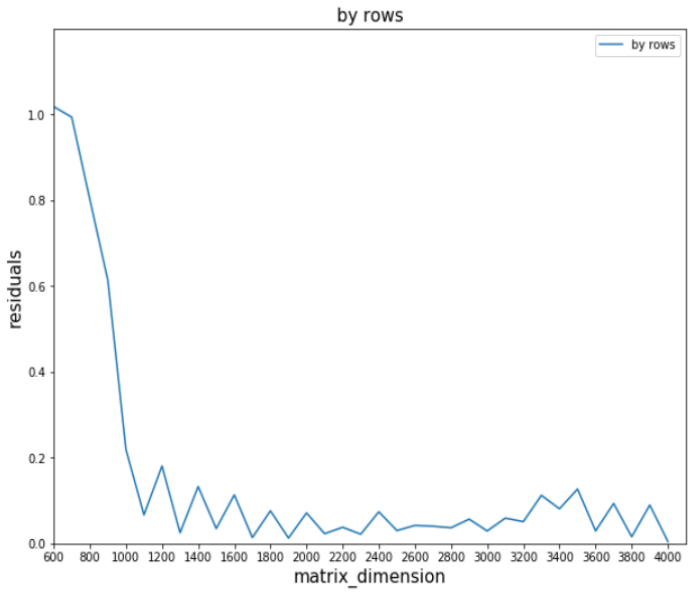
N_{min}=100 and N_{max}=4200

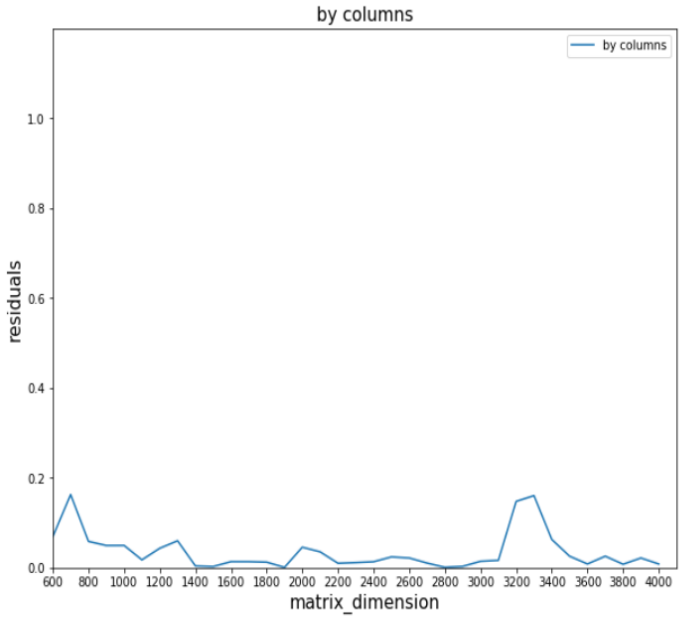
Interpolating functions are :

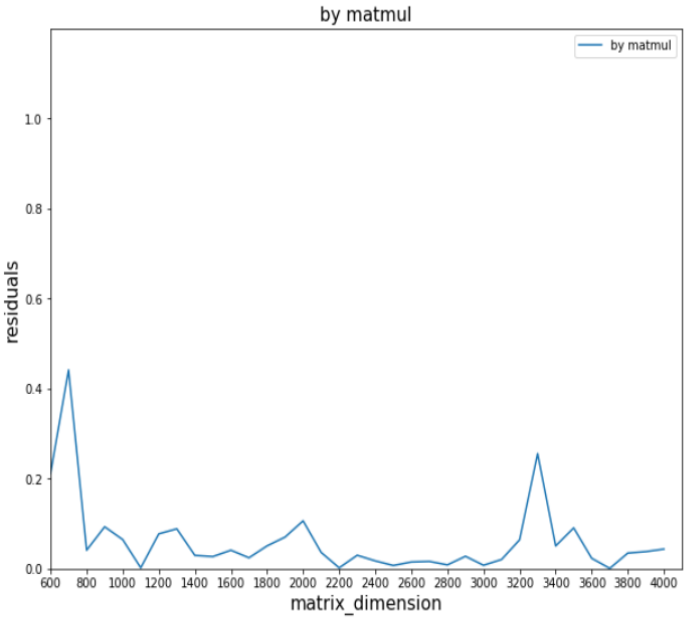
$f(x) = 3log(x) - 18.4$ used to ***by rows*** points (orange line)

$g(x) = 3log(x) - 19.52$ used to ***by columns*** points (yellow line)

$h(x) = 3log(x) - 23.56$ used to ***by matmul*** points (blue line)







5 Self-evaluation

Probably,seeing ***by row*** and ***by matmul*** residual graphs , it could find more accurate interpolating functions.