Program #4

Joey Troyer
09/26/22

Problem description:

The task of this programming assignment was to test and compare the speed of interpreted and compiled languages. I was tasked with creating three different programs that would perform gaussian elimination with back substitution. The programs to develop were in python, python with NumPy, and Fortran. At the start of the program, you should be able to enter a number that will determine the size of the matrix and fill that matrix with randomly generated values. We are then to time how long each program takes to perform gaussian elimination on each matrix for every size.

FORTRAN:

```
! Name: Joey Troyer
! Date: 09/26/22
! input: a number to determine the size of the matrix
! output: the time it take to preform gaussian elimination on the matrix
! problem: The task is to create a program that can generate an matrix with random entires
!          and preform gaussian elimination and backsubstitution on the matrix. The goal
!          was to time how long it would take to solve the matrix for each increase in size

PROGRAM gaussian_elimination
    IMPLICIT NONE

    !create all of our variables and arrays
    INTEGER::i,j,x,y,n
    REAL::s, rand, start, finish
    REAL,ALLOCATABLE,DIMENSION(:)::e
    REAL, ALLOCATABLE, DIMENSION(:,:)::a

    read *, n

    ALLOCATE(a(n,n))

    ! Fill the matrix with random values
    do x = 1, n
        do y = 1, n - 1
            call random_number(rand)
            a(x,y) = floor((rand * 1000) + 1)
        end do
    end do


    ! log time at start if algorithm
    CALL cpu_time(start)

    ! perform gaussian elimination
    DO j=1,n
        DO i=j+1,n
            a(i,:)=a(i,:)-a(j,:)*a(i,j)/a(j,j)
        END DO
    END DO


    ALLOCATE(e(n))

    ! perform back traceing
    DO i=n,1,-1
        s=a(i,n+1)
        DO j=i+1,n
            s=s-a(i,j)*e(j)
        END DO
        e(i)=s/a(i,i)
    END DO

    ! log time at end of algorithm
    CALL cpu_time(finish)

    ! print formatted time in seconds
    PRINT'("Time = ",f10.7," seconds.")',(finish - start)

END PROGRAM
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  JUPYTER

jtroyer@granville:~/CS471/Lab4> ./a.out
500
Time =  0.0946240 seconds.
jtroyer@granville:~/CS471/Lab4>
```

PYTHON WITH NUMPY:

```python
# Name: Joey Troyer
# Date: 09/26/22
# input: a number to determine the size of the matrix
# output: the time it take to preform gaussian elimination on the matrix
# problem: The task is to create a program that can generate an matrix with random entires
#           and preform gaussian elimination and backsubstitution on the matrix. The goal
#           was to time how long it would take to solve the matrix for each increase in size

import time
import numpy as np
import random as rand

size = int(input())

matrix = np.zeros(shape=[size,size])

#fill matrix with random float values from 10 to 10000
for x in range(0, size):
    for y in range(0, size):
        matrix[x,y] = float(rand.uniform(10.0, 10000.0))

#create the indent matrix
x = np.zeros(shape= size, dtype=float)

def gaussElim(a,b):
    n = len(b)
    #Gaussian Elimination phase
    for k in range(0,n-1):
        for i in range(k+1,n):
            if a[i,k] != 0.0:
                #if not null define λ
                lam = a [i,k]/a[k,k]
                #we calculate the new row of the matrix
                a[i,k+1:n] = a[i,k+1:n] - lam*a[k,k+1:n]
                #we update vector b
                b[i] = b[i] - lam*b[k]
    # backward substitution
    for k in range(n-1,-1,-1):
        b[k] = (b[k] - np.dot(a[k,k+1:n],b[k+1:n]))/a[k,k]

    return b


#log the time at start
start = time.time()

#call the function with out matrix
gaussElim(matrix, x)

#log time at finish
finish = time.time()

#subtract them to get the time it took
total = finish - start

print(f"{total} in seconds")
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

jtroyer@granville:~/CS471/Lab4> python3 gNumpy.py
500
0.41405272483825684 in seconds
jtroyer@granville:~/CS471/Lab4>
```

PYTHON:

```python
# Name: Joey Troyer
# Date: 09/26/22
# input: a number to determine the size of the matrix
# output: the time it take to preform gaussian elimination on the matrix
# problem: The task is to create a program that can generate an matrix with random entires
#          and preform gaussian elimination and backsubstitution on the matrix. The goal
#          was to time how long it would take to solve the matrix for each increase in size

import math
import random as rand
import time

#gets input from user and casts it as an int
size = int(input())

#creates the matrixs
matrix = [[0] * (size + 1) for i in range(0, size)]
b = [0] * size
x = [0] * size

#loop to fill the matrix with random float numbers from range 10 to 100000
for col in range(0, size):
    for row in range(0, size+1):
        matrix[col][row] = float(math.floor(rand.uniform(10.0, 10000.0)))

#function to make it easy to perform the operation
def gauss(matrix, x):

    #loops to perform gaussian elimination
    for k in range(size - 1):
        for i in range(k+1, size):
            factor = matrix[i][k] / matrix[k][k]
            for j in range(k, size):
                matrix[i][j] = matrix[i][j] - factor * matrix[k][j]
            b[i] = b[i] - factor * b[k]

    #back tracing
    for i in range(size-1, -1, -1):
        sum = matrix[i][size]
        for j in range(i+1, size):
            sum = sum - matrix[i][j] * x[j]
        x[i] = sum/matrix[i][i]

#logs time at start
start = time.time()

#call the function
gauss(matrix, x)

#logs time at end
finish = time.time()

#subtracts finish by start time to get the time it took to run the function
total = (finish - start)

print(f"{total} in seconds")
```

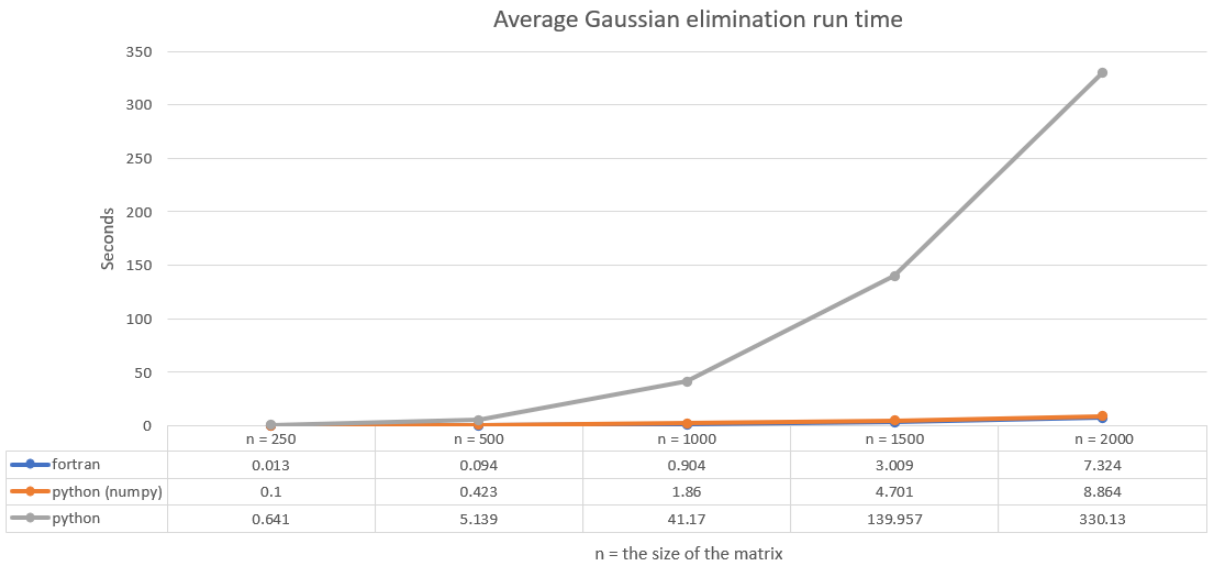
```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  JUPYTER

jtroyer@granville:~/CS471/Lab4> python3 gaussian.py
500
5.0515217781066895 in seconds
jtroyer@granville:~/CS471/Lab4>
```

FORTRAN In seconds	250	500	1000	1500	2000
TEST 1	0.013	0.095	0.905	3.071	7.296
TEST 2	0.013	0.093	0.903	3.045	7.263
TEST 3	0.013	0.096	0.906	2.839	7.254
TEST 4	0.013	0.093	0.910	3.057	7.259
TEST 5	0.013	0.094	0.897	3.032	7.549
AVERAGE	0.013	0.094	0.904	3.009	7.324
STD	0.000	0.001	0.004	0.096	0.126

Python (Numpy) in seconds	250	500	1000	1500	2000
TEST 1	0.100	0.423	1.870	4.686	9.017
TEST 2	0.099	0.421	1.858	4.712	8.850
TEST 3	0.100	0.425	1.846	4.697	8.819
TEST 4	0.099	0.425	1.852	4.707	8.808
TEST 5	0.103	0.422	1.873	4.704	8.827
AVERAGE	0.1	0.423	1.86	4.701	8.864
STD	0.001	0.001	0.011	0.010	0.086

Python in seconds	250	500	1000	1500	2000
TEST 1	0.668	5.136	40.957	138.679	327.127
TEST 2	0.634	5.124	41.016	138.855	328.394
TEST 3	0.635	5.135	41.036	139.840	329.918
TEST 4	0.635	5.134	41.333	140.576	332.361
TEST 5	0.635	5.166	41.509	141.837	332.850
AVERAGE	0.641	5.139	41.17	139.957	330.13
STD	0.014	0.015	0.239	1.301	2.472

Graph shows the average time of each program



	n = 250	n = 500	n = 1000	n = 1500	n = 2000
fortran	0.013	0.094	0.904	3.009	7.324
python (numpy)	0.1	0.423	1.86	4.701	8.864
python	0.641	5.139	41.17	139.957	330.13

Python without NumPy performed the slowest by far. It was relatively close in speed compared to python, with NumPy and Fortran up to 500. After 500, it took a wildly longer time than the other two. Rendering it as almost entirely impractical for this use case. Python with NumPy had impressively fast speed, cutting its time down substantially. Fortran performed the best of the three, with each of its times fastest. In conclusion these results are to be expected because python is an interpreted language known to be slower. However, I was pleasantly surprises by how much NumPy improved the runtime. Of course, the language with the fastest time is Fortran because it is a compiled language.

Sources:

https://labmathdu.wordpress.com/gaussian-elimination-without-pivoting/

https://youtu.be/r89yH82OAFw

https://techgoggler.com/computer-engineering/linear-equations-python-gauss-elimination-method/