Name: Pratyay Kumar
Aggie id: 800811773
Date: 09 / 26 / 2022

CS 471 – Programming #4 – Comparing interpreted and compiled codes

Problem Description: Build three program which perform Gaussian elimination with back substitution. Use system calls to measure time to capture the time it takes to run to program with different data sizes. Make sure to use an appropriate measuring tool that give you responsible time granularity. Python + NumPy can use pivoting as LU decomposition which does this by default. Other 3 implementation should not have pivoting.

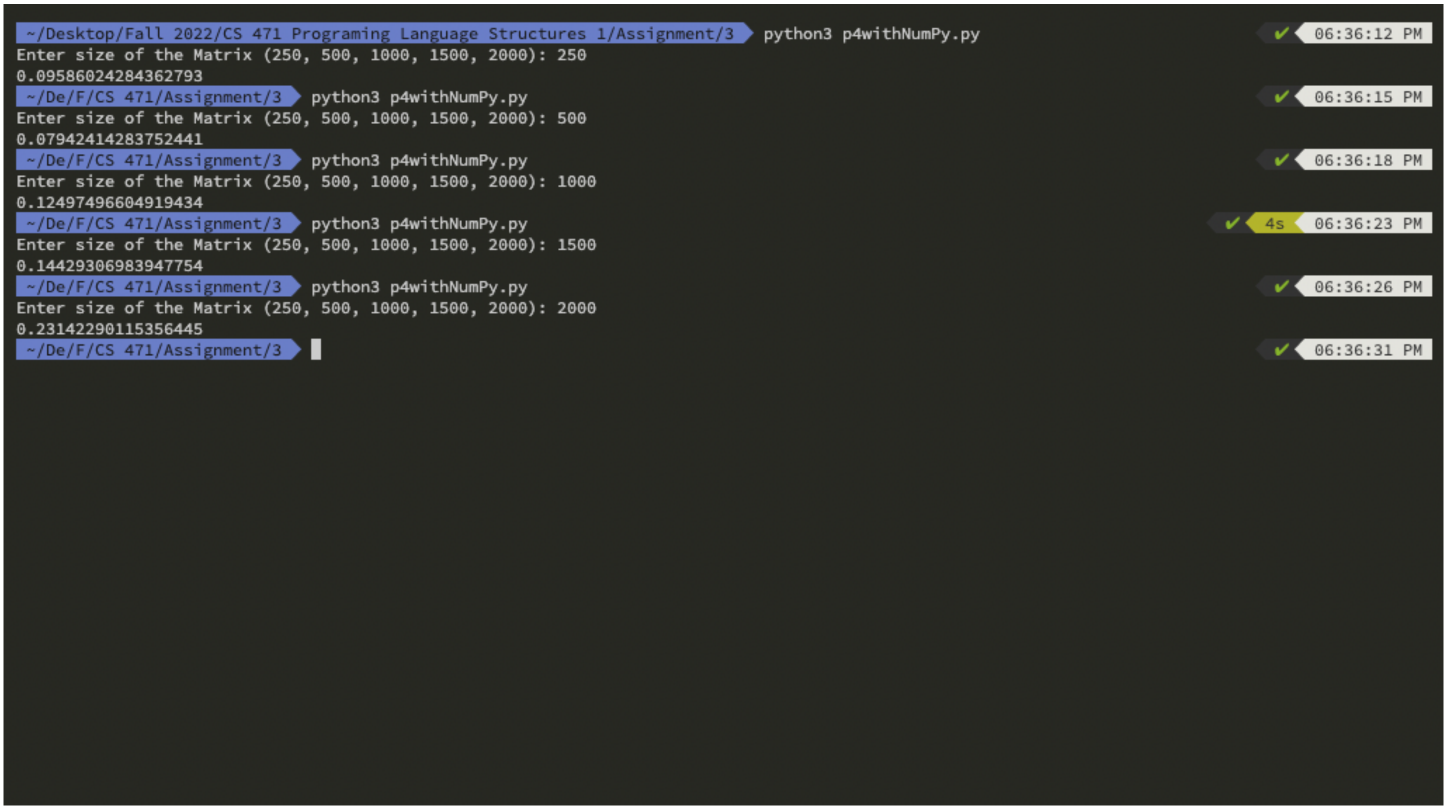
Input: Size of the matrix (250, 500, 1000, 1500, 2000), used random number generator to populate the matrix
Output: The total execution time of the program in seconds.

Python code with NumPy:

```python
# Author: Pratyay Kumar
# Date: 24 September 2022
# Purpose: Performing Gaussian Elimination
# Assumption: Random int will be produced in-between 0 - 10
# Description: This program performs Gaussian Elimination for solving matrix equation of the form A
x = b.
#              The following code uses NumPy python library and SciPy.

# Importing required libraries
import numpy as np
from numpy import zeros
import time
import scipy

# Function: gaussianElimination
# Description: Will find the Gaussian Elimination using scipy.
# Param: A
# Return: No return value. Matrix will be modified since passing by reference.
def gaussianElimination (A):
    P, L, U = scipy.linalg.lu(A)

def main ():
    N = int(input('Enter size of the Matrix (250, 500, 1000, 1500, 2000): '))
    # Timer starts after taking in the input size of matrix
    start_time = time.time()

    # Populating the matrix
    A = np.random.randint(-100, 100, size=(N, N+1))

    # Function call to start Gaussian Elimination
    gaussianElimination (A)
    # Timer just after gaussian elimination is done
    end_time = time.time()
    print (end_time-start_time)
```

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

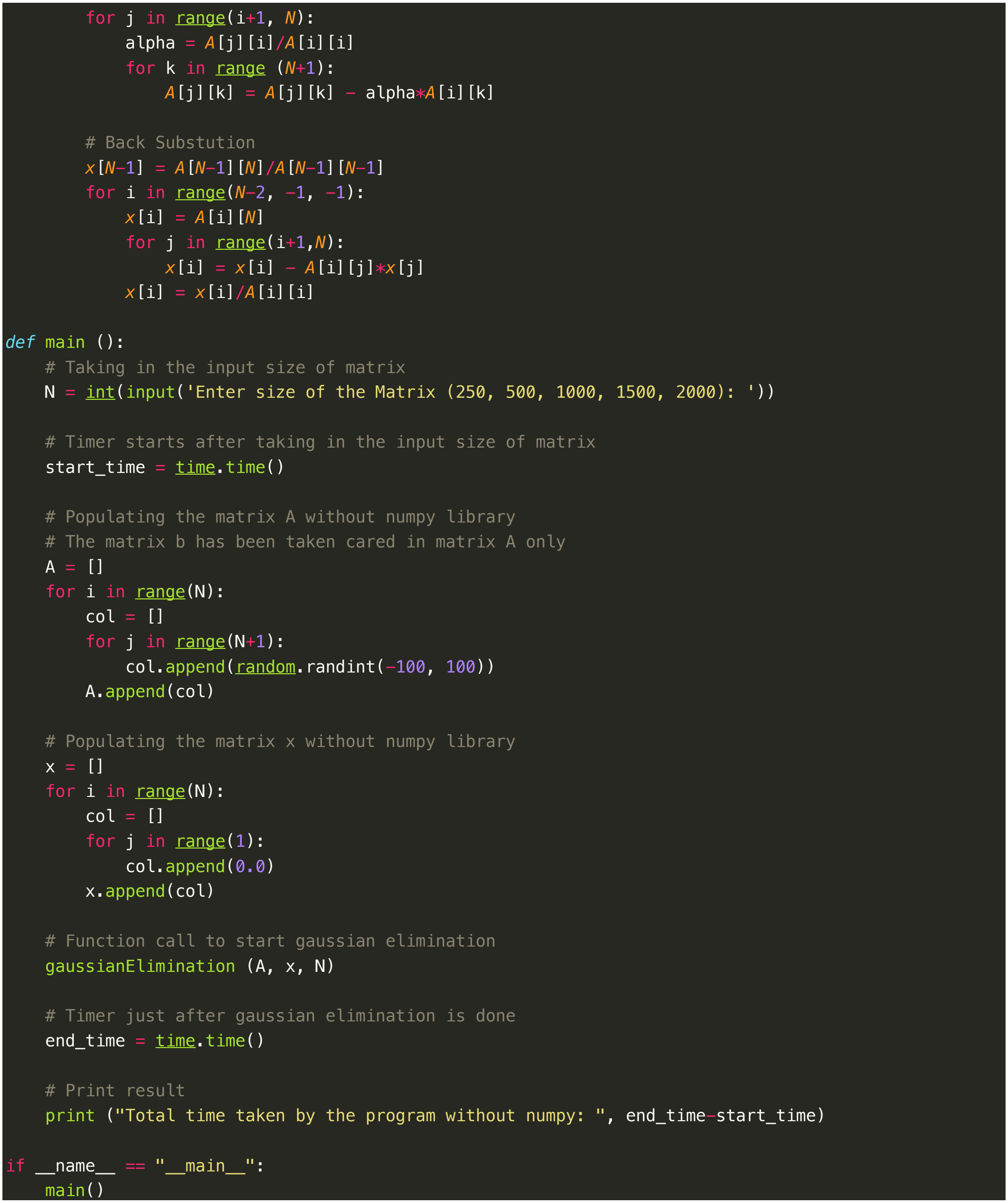
Output for the above code:



```
~/Desktop/Fall 2022/CS 471 Programing Language Structures 1/Assignment/3  python3 p4withNumPy.py      ✔  06:36:12 PM
Enter size of the Matrix (250, 500, 1000, 1500, 2000): 250
0.09586024284362793
~/De/F/CS 471/Assignment/3  python3 p4withNumPy.py                                                      ✔  06:36:15 PM
Enter size of the Matrix (250, 500, 1000, 1500, 2000): 500
0.07942414283752441
~/De/F/CS 471/Assignment/3  python3 p4withNumPy.py                                                      ✔  06:36:18 PM
Enter size of the Matrix (250, 500, 1000, 1500, 2000): 1000
0.12497496604919434
~/De/F/CS 471/Assignment/3  python3 p4withNumPy.py                                                   ✔ 4s  06:36:23 PM
Enter size of the Matrix (250, 500, 1000, 1500, 2000): 1500
0.14429306983947754
~/De/F/CS 471/Assignment/3  python3 p4withNumPy.py                                                      ✔  06:36:26 PM
Enter size of the Matrix (250, 500, 1000, 1500, 2000): 2000
0.23142290115356445
~/De/F/CS 471/Assignment/3  █                                                                           ✔  06:36:31 PM
```

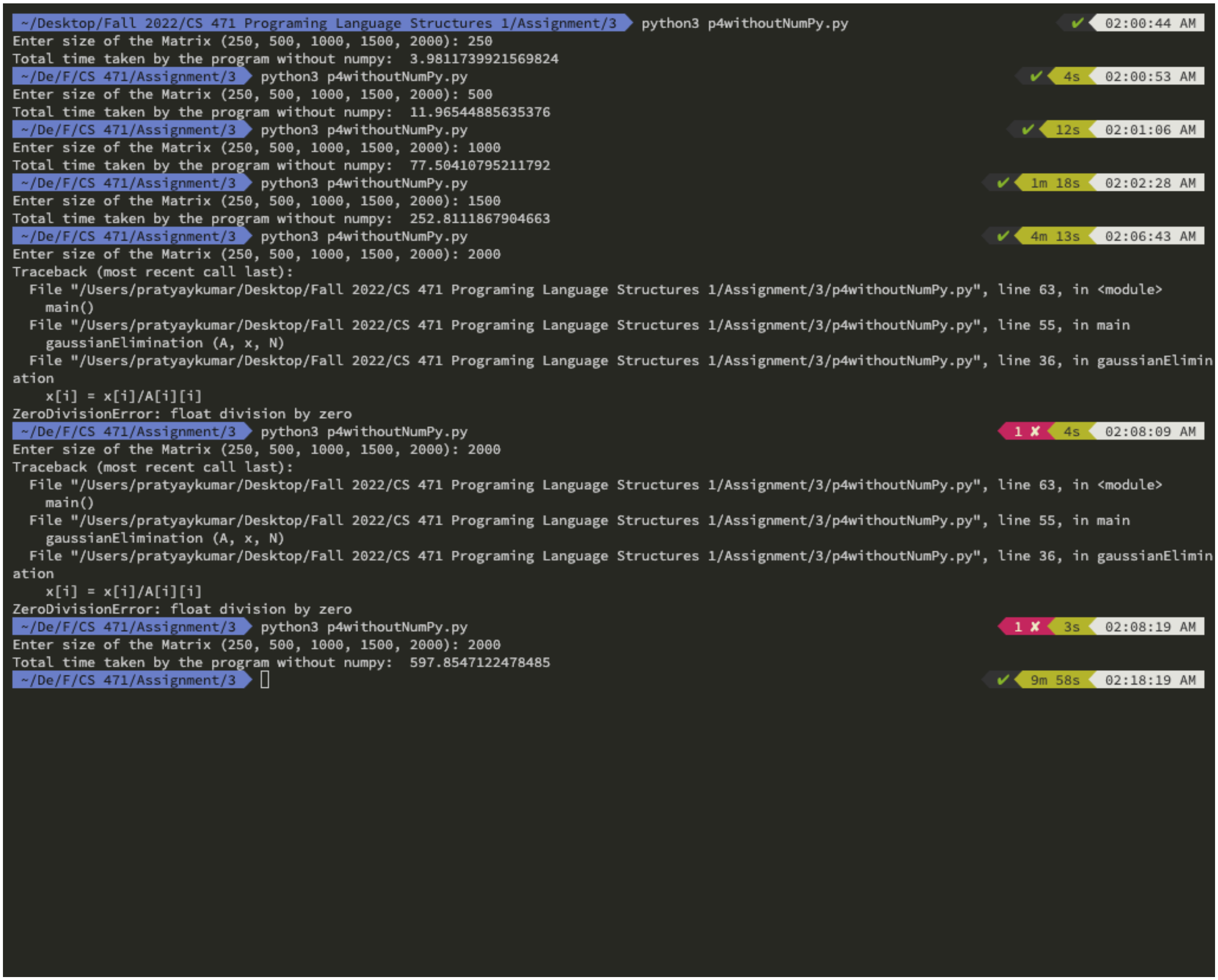
Python code without using NumPy:

```
# Author: Pratyay Kumar
# Date: 24 September 2022
# Purpose: Performing Gaussian Elimination without using NumPy.
# Assumption: The matrix has been populated from number -100 to 100.
# Description: This program performs Gaussian Elimination for solving matrix equation of the form A
x = b.
#               The following code does not use the python NumPy library.
#               NO pivoting has been performed to solve gaussian elimination.
#               Backward Substitution has been performed.

# Importing required libraries
import random
import time

def gaussianElimination (A, x, N):
    # Applying Gaussian Elimination
    for i in range(N):
        if A[i][i] == 0.0:
            print('Divide by zero detected!')
            exit()
```



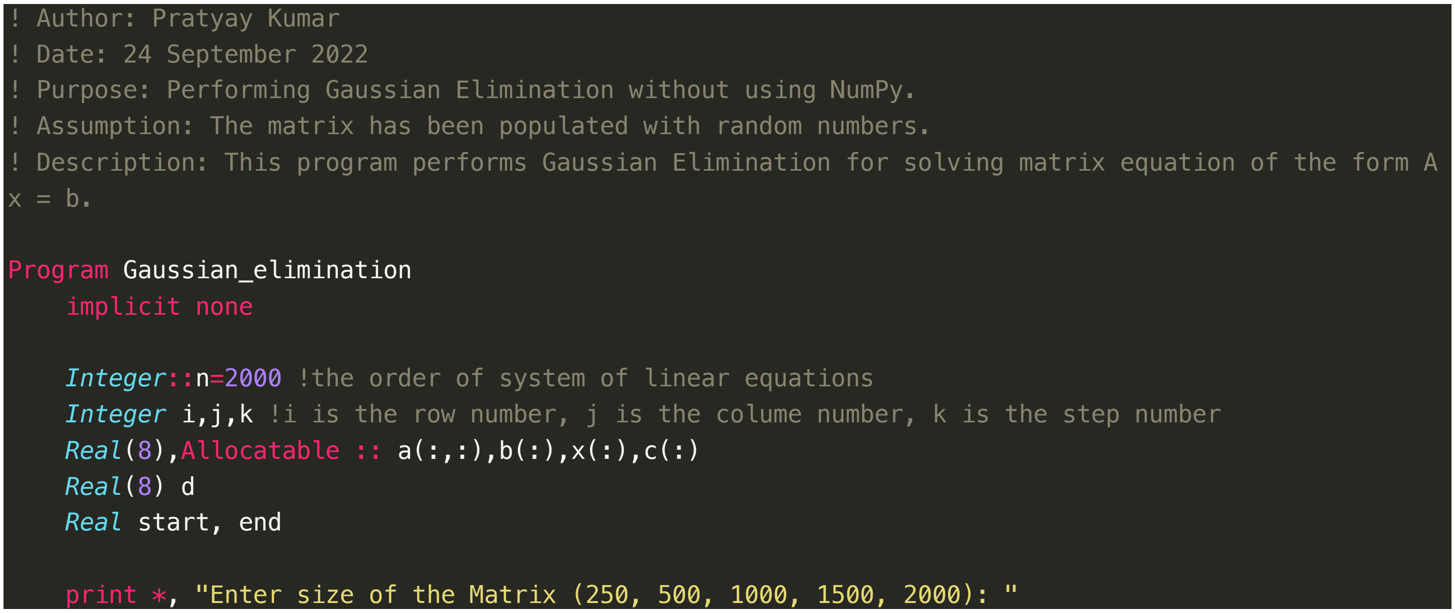
```
        for j in range(i+1, N):
            alpha = A[j][i]/A[i][i]
            for k in range (N+1):
                A[j][k] = A[j][k] - alpha*A[i][k]

        # Back Substution
        x[N-1] = A[N-1][N]/A[N-1][N-1]
        for i in range(N-2, -1, -1):
            x[i] = A[i][N]
            for j in range(i+1,N):
                x[i] = x[i] - A[i][j]*x[j]
            x[i] = x[i]/A[i][i]

def main ():
    # Taking in the input size of matrix
    N = int(input('Enter size of the Matrix (250, 500, 1000, 1500, 2000): '))

    # Timer starts after taking in the input size of matrix
    start_time = time.time()

    # Populating the matrix A without numpy library
    # The matrix b has been taken cared in matrix A only
    A = []
    for i in range(N):
        col = []
        for j in range(N+1):
            col.append(random.randint(-100, 100))
        A.append(col)

    # Populating the matrix x without numpy library
    x = []
    for i in range(N):
        col = []
        for j in range(1):
            col.append(0.0)
        x.append(col)

    # Function call to start gaussian elimination
    gaussianElimination (A, x, N)

    # Timer just after gaussian elimination is done
    end_time = time.time()

    # Print result
    print ("Total time taken by the program without numpy: ", end_time-start_time)

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

Output for the above code:

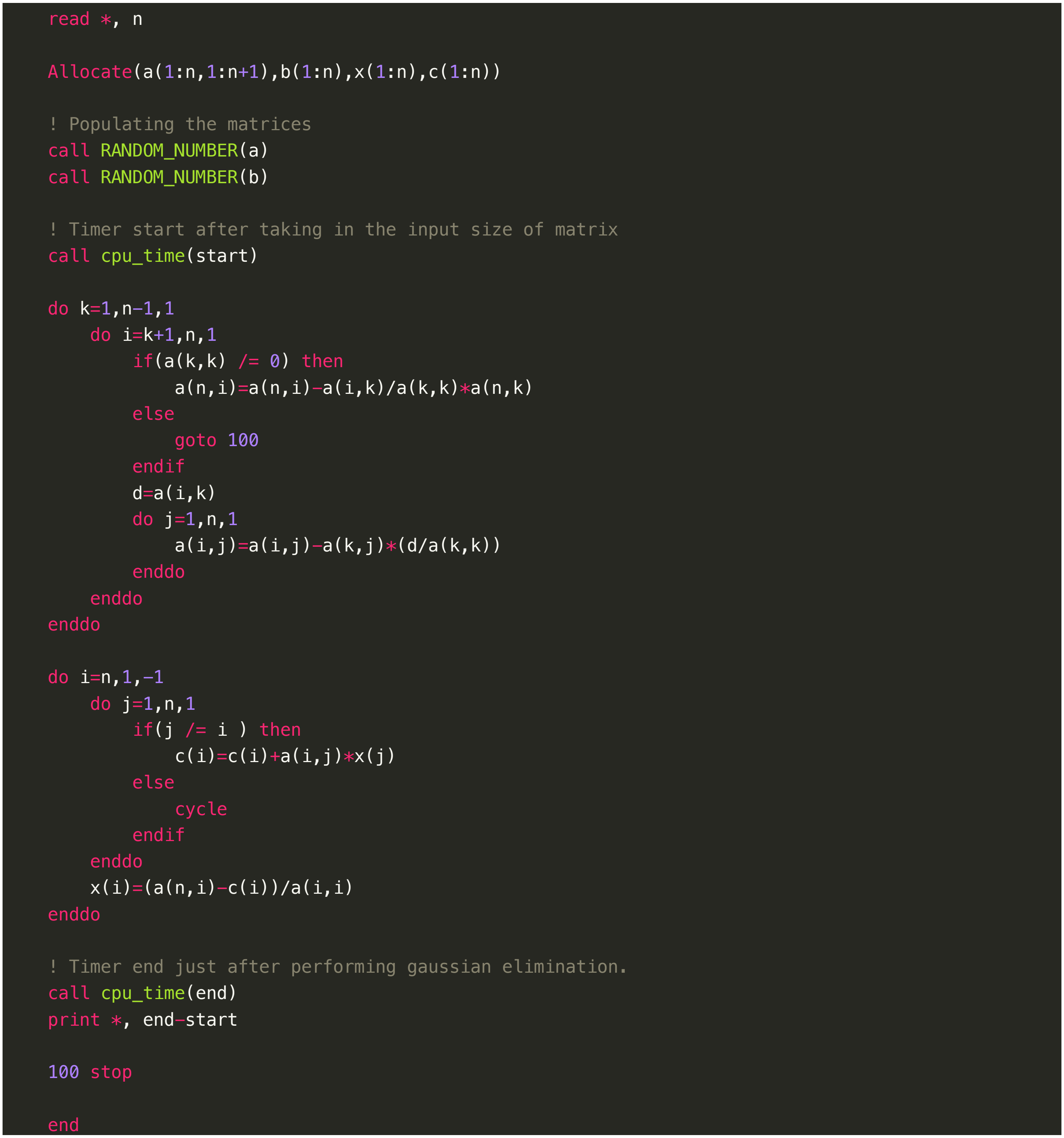


```
~/Desktop/Fall 2022/CS 471 Programing Language Structures 1/Assignment/3  python3 p4withoutNumPy.py          ✔  02:00:44 AM
Enter size of the Matrix (250, 500, 1000, 1500, 2000): 250
Total time taken by the program without numpy:  3.9811739921569824
~/De/F/CS 471/Assignment/3  python3 p4withoutNumPy.py                                         ✔  4s  02:00:53 AM
Enter size of the Matrix (250, 500, 1000, 1500, 2000): 500
Total time taken by the program without numpy:  11.96544885635376
~/De/F/CS 471/Assignment/3  python3 p4withoutNumPy.py                                        ✔  12s  02:01:06 AM
Enter size of the Matrix (250, 500, 1000, 1500, 2000): 1000
Total time taken by the program without numpy:  77.50410795211792
~/De/F/CS 471/Assignment/3  python3 p4withoutNumPy.py                                     ✔  1m 18s  02:02:28 AM
Enter size of the Matrix (250, 500, 1000, 1500, 2000): 1500
Total time taken by the program without numpy:  252.8111867904663
~/De/F/CS 471/Assignment/3  python3 p4withoutNumPy.py                                     ✔  4m 13s  02:06:43 AM
Enter size of the Matrix (250, 500, 1000, 1500, 2000): 2000
Traceback (most recent call last):
  File "/Users/pratyaykumar/Desktop/Fall 2022/CS 471 Programing Language Structures 1/Assignment/3/p4withoutNumPy.py", line 63, in <module>
    main()
  File "/Users/pratyaykumar/Desktop/Fall 2022/CS 471 Programing Language Structures 1/Assignment/3/p4withoutNumPy.py", line 55, in main
    gaussianElimination (A, x, N)
  File "/Users/pratyaykumar/Desktop/Fall 2022/CS 471 Programing Language Structures 1/Assignment/3/p4withoutNumPy.py", line 36, in gaussianElimin
ation
    x[i] = x[i]/A[i][i]
ZeroDivisionError: float division by zero
~/De/F/CS 471/Assignment/3  python3 p4withoutNumPy.py                                     1 ✘  4s  02:08:09 AM
Enter size of the Matrix (250, 500, 1000, 1500, 2000): 2000
Traceback (most recent call last):
  File "/Users/pratyaykumar/Desktop/Fall 2022/CS 471 Programing Language Structures 1/Assignment/3/p4withoutNumPy.py", line 63, in <module>
    main()
  File "/Users/pratyaykumar/Desktop/Fall 2022/CS 471 Programing Language Structures 1/Assignment/3/p4withoutNumPy.py", line 55, in main
    gaussianElimination (A, x, N)
  File "/Users/pratyaykumar/Desktop/Fall 2022/CS 471 Programing Language Structures 1/Assignment/3/p4withoutNumPy.py", line 36, in gaussianElimin
ation
    x[i] = x[i]/A[i][i]
ZeroDivisionError: float division by zero
~/De/F/CS 471/Assignment/3  python3 p4withoutNumPy.py                                     1 ✘  3s  02:08:19 AM
Enter size of the Matrix (250, 500, 1000, 1500, 2000): 2000
Total time taken by the program without numpy:  597.8547122478485
~/De/F/CS 471/Assignment/3  ▯                                                              ✔  9m 58s  02:18:19 AM
```

Fortran Code:

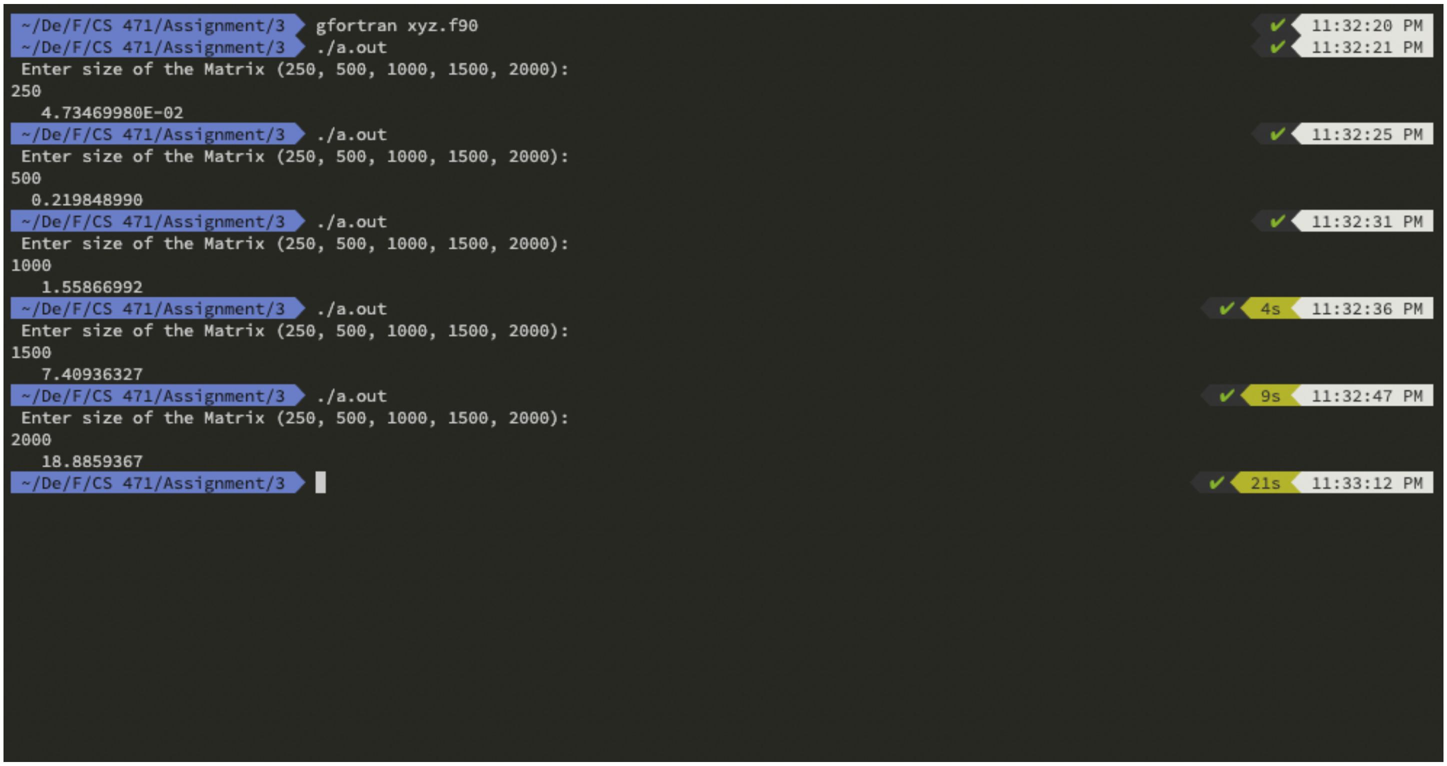


```fortran
! Author: Pratyay Kumar
! Date: 24 September 2022
! Purpose: Performing Gaussian Elimination without using NumPy.
! Assumption: The matrix has been populated with random numbers.
! Description: This program performs Gaussian Elimination for solving matrix equation of the form A
x = b.

Program Gaussian_elimination
    implicit none

    Integer::n=2000 !the order of system of linear equations
    Integer i,j,k !i is the row number, j is the colume number, k is the step number
    Real(8),Allocatable :: a(:,:),b(:),x(:),c(:)
    Real(8) d
    Real start, end

    print *, "Enter size of the Matrix (250, 500, 1000, 1500, 2000): "
```



```
    read *, n

    Allocate(a(1:n,1:n+1),b(1:n),x(1:n),c(1:n))

    ! Populating the matrices
    call RANDOM_NUMBER(a)
    call RANDOM_NUMBER(b)

    ! Timer start after taking in the input size of matrix
    call cpu_time(start)

    do k=1,n-1,1
        do i=k+1,n,1
            if(a(k,k) /= 0) then
                a(n,i)=a(n,i)-a(i,k)/a(k,k)*a(n,k)
            else
                goto 100
            endif
            d=a(i,k)
            do j=1,n,1
                a(i,j)=a(i,j)-a(k,j)*(d/a(k,k))
            enddo
        enddo
    enddo

    do i=n,1,-1
        do j=1,n,1
            if(j /= i ) then
                c(i)=c(i)+a(i,j)*x(j)
            else
                cycle
            endif
        enddo
        x(i)=(a(n,i)-c(i))/a(i,i)
    enddo

    ! Timer end just after performing gaussian elimination.
    call cpu_time(end)
    print *, end-start

    100 stop

    end
```

<u>Output for the above code:</u>



```
~/De/F/CS 471/Assignment/3  gfortran xyz.f90                                    ✔  11:32:20 PM
~/De/F/CS 471/Assignment/3  ./a.out                                             ✔  11:32:21 PM
 Enter size of the Matrix (250, 500, 1000, 1500, 2000):
250
   4.73469980E-02
~/De/F/CS 471/Assignment/3  ./a.out                                             ✔  11:32:25 PM
 Enter size of the Matrix (250, 500, 1000, 1500, 2000):
500
  0.219848990
~/De/F/CS 471/Assignment/3  ./a.out                                             ✔  11:32:31 PM
 Enter size of the Matrix (250, 500, 1000, 1500, 2000):
1000
   1.55866992
~/De/F/CS 471/Assignment/3  ./a.out                                        ✔  4s  11:32:36 PM
 Enter size of the Matrix (250, 500, 1000, 1500, 2000):
1500
   7.40936327
~/De/F/CS 471/Assignment/3  ./a.out                                        ✔  9s  11:32:47 PM
 Enter size of the Matrix (250, 500, 1000, 1500, 2000):
2000
   18.8859367
~/De/F/CS 471/Assignment/3                                                ✔  21s  11:33:12 PM
```

Plot Values (time in seconds): The following times has been calculated on my personal laptop with 8GB RAM, and apple silicon M1 chipset.

Python code with NumPy

N	Time 1	Time 2	Time 3	Time 4	Time 5	Average	Stdev
250	0.1414	0.0964	0.1081	0.1293	0.0935	0.1137	0.0209
500	0.2375	0.2849	0.2079	0.1944	0.1918	0.2233	0.0389
1000	0.6047	0.5616	0.5692	0.5837	0.5738	0.5786	0.0166
1500	1.2479	1.3289	1.2121	1.1934	1.2070	1.2378	0.0547
2000	2.1413	2.1056	2.1156	2.0638	2.1531	2.1158	0.0348

Python code without NumPy

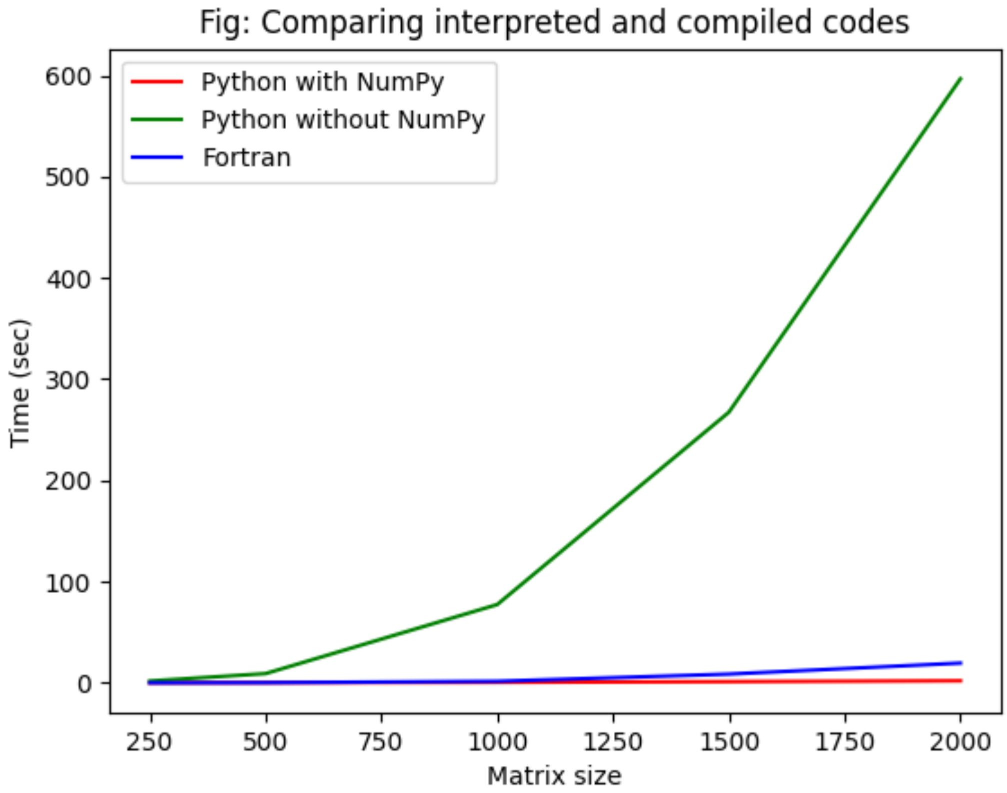
N	Time 1	Time 2	Time 3	Time 4	Time 5	Average	Stdev
250	1.1341	1.1543	3.0612	3.0111	1.1423	1.9006	1.0367
500	9.1638	9.0594	9.1340	9.1570	9.4663	9.1961	0.1566
1000	77.1295	76.5679	76.3524	81.7269	75.1471	77.3847	2.5327
1500	259.3287	268.2499	276.6778	262.4955	269.6956	267.2895	6.740
2000	597.0153	596.2550	593.7847	596.3493	599.9265	596.6661	2.1979

Fortran code

N	Time 1	Time 2	Time 3	Time 4	Time 5	Average	Stdev
250	0.0510	0.0495	0.0438	0.0477	0.0386	0.0461	0.0049
500	0.2176	0.2062	0.2213	0.2073	0.2122	0.2129	0.0065
1000	1.6804	1.5932	1.5811	1.5732	1.5764	1.6008	0.0451
1500	8.7704	8.5278	8.8966	8.4195	8.9280	8.7084	0.2255
2000	19.6005	19.4062	19.6542	19.4220	19.7249	19.5615	0.1417

Graph:

Used MatPlot python library to generate the graph taking above data points.



Explanation of graph:

For better comparison, all the values are plotted in one graph. X-axis is the matrix size and Y-axis is the time in seconds.

Python with NumPy plot is marked in red color.

Python without NumPy plot is marked in green color.

Fortran plot is marked in blue color.

So, from this experiment we can conclude that Interpreted language (python) generally takes more time than compiled language (Fortran).