Entrega 6 "Matrices dispersas y complejidad computacional"

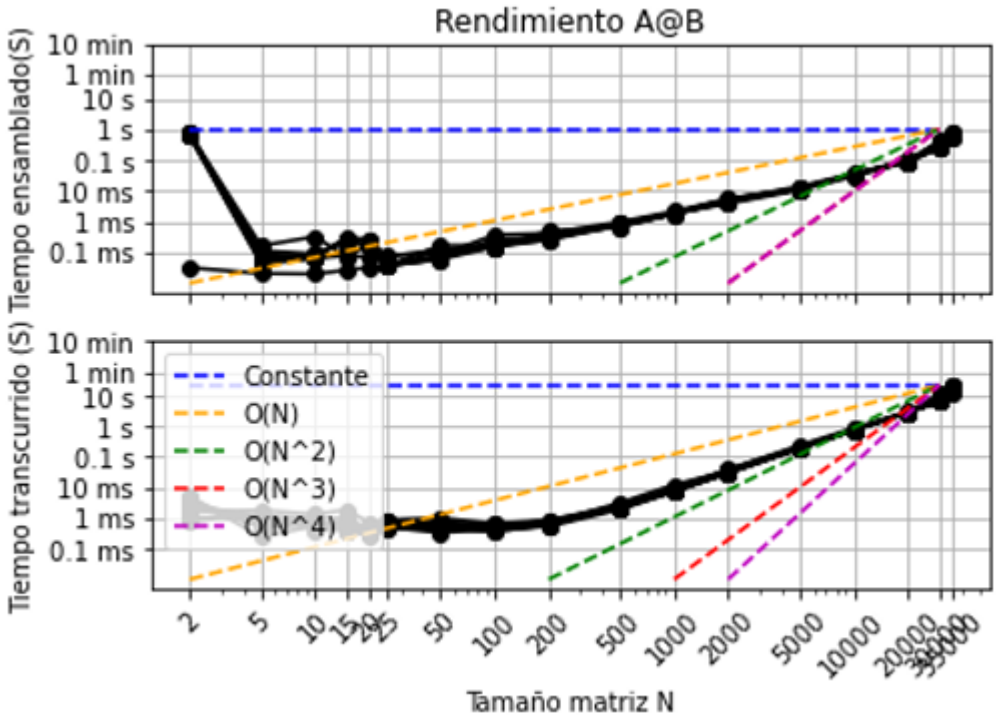
El codigo de ensamblaje que se utilizo para lograr la matriz laplaciana fue el siguiente:

def laplaciana(N,dtype): A = zeros((N,N),dtype=dtype)

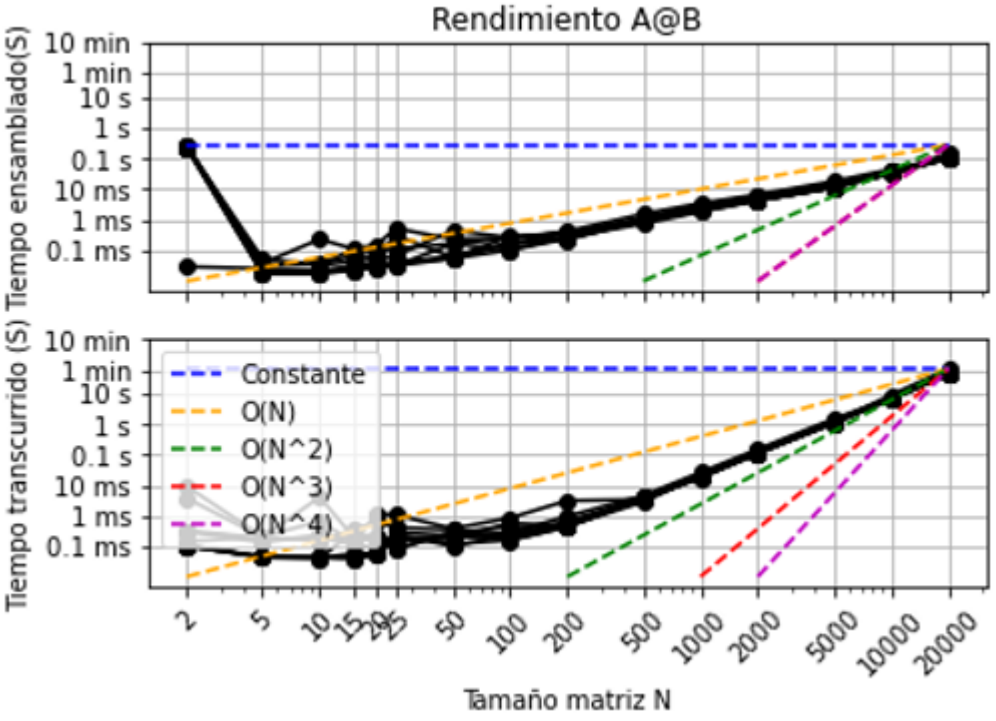
```
for i in range(N):
    A[i,i]=2

    for j in range(max(0,i-2),i):
        if abs(i-j)==1:
               A[i,i]=-1
               A[j,i]=-1
return A
```

A continuación se mostraran los graficos obtenidos para el rendimiento de solve tanto para matrices dispersas como llenas

Matriz dispersa

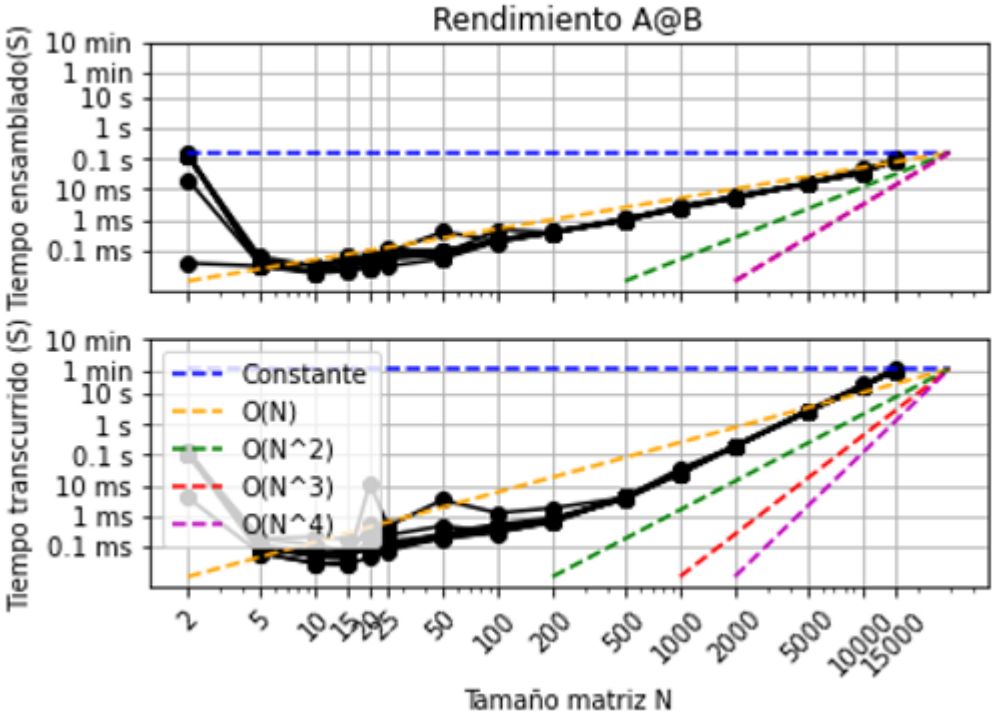


Matriz llena

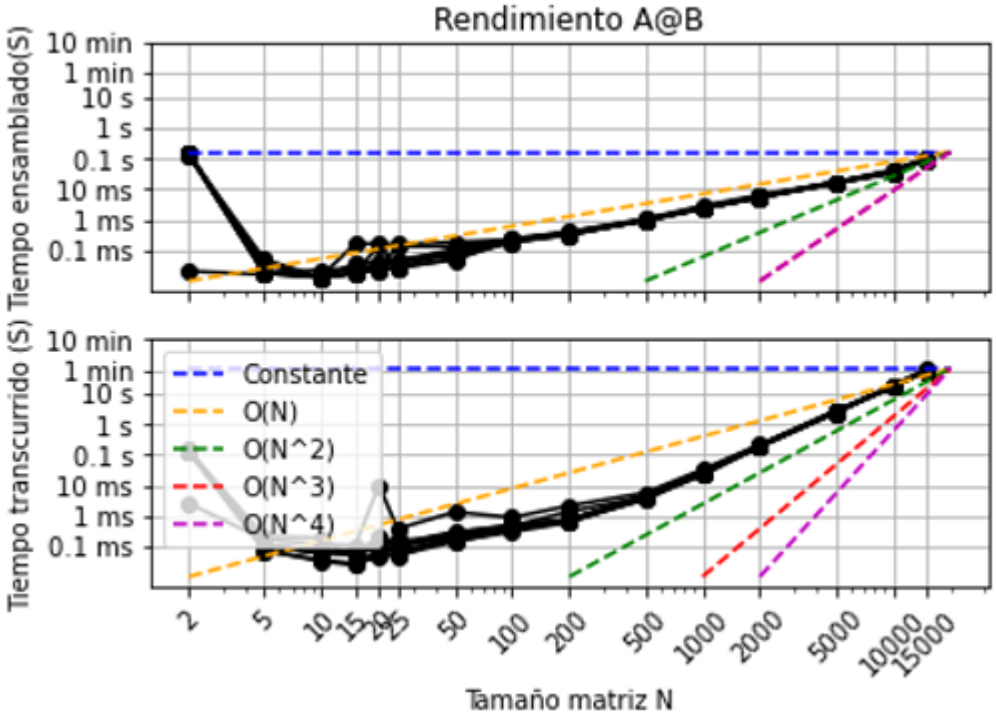


A continuación se mostraran los graficos obtenidos para el rendimiento de inv tanto para matrices dispersas como llenas

Matriz dispersa



Matriz llena



Para el caso de solve se puede apreciar que para matrices dispersas el rendimiento de estas es bastante mejor que la de matrices llenas debido a que se demora menos tiempo en llegar a la solucion y lo mas importante es que del metedo de matrices dispersas es posible llegar a un tamaño mucho mayor del tamaño N de la matriz logrando un rendimiento de un N 10000 veces mayor del cual llega las matriz llena.

Para el caso de la funcion Inv se puede apreciar que entre las matrices llenas y dispersas no existe mayor diferencia mas que (+- 1 seg).Lo que si se puede apreciar es que con esta funcion ambas formas llegan a un tamaño N mas reducido que las demas funciones , llegando a un N max de tamaño 150000

las asintotas que mas se asemejan a cada una de las matrices se presenta a continuación:

Matriz dispersa

Solve : O(N) para ensamblado y ejecución

inv :O(N) para ensamblado y O(N^2) ejecución

Matriz llena:

Solve :O(N) para ensamblado y ejecución

inv:O(N) para ensamblado y ejecución

El tamaño de las matricez afecta al comportamiento debido a que ocupa memoria del computador , entre mas grande sea N mas recursos necesitara el computador debido a que necesita hacer cada vez mas procesos al mismo tiempo hasta que llega un punto en que el computador se queda sin memoria para poder ejecutar y se estanca todo.

Las corridas por lo menos en mi computador todas fueron en el proyecto 0 muy estables (nunca diferian en mas de 2 segundos) , esto tambien se debe quizas a que siempre que corria el programa me fijaba de que el computador no hiciera nada mas que eso cerrando cualquier navegador o aplicacion que pudiera estorbar en el proceso.

Entrega 5:

El codigo de ensamblaje que se utilizo para lograr la matriz laplaciana fue el siguiente:

def laplaciana(N,dtype): A = zeros((N,N),dtype=dtype)

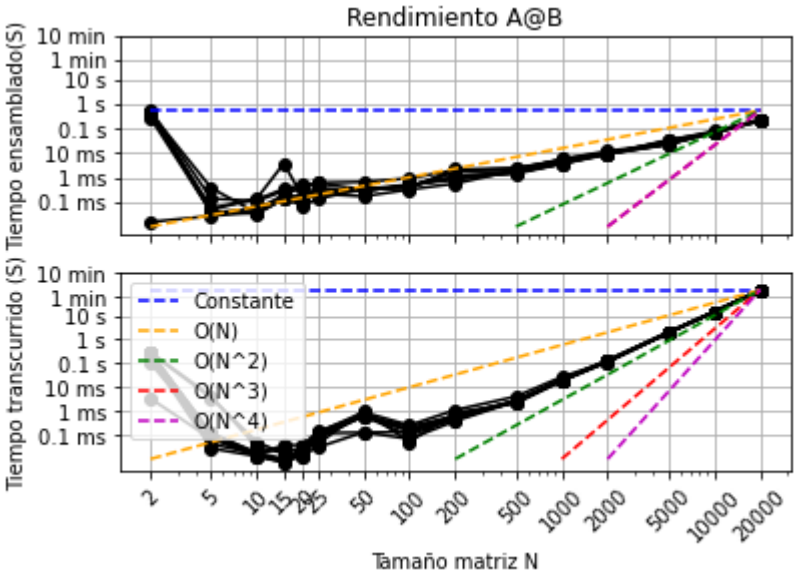
```
  for i in range(N):
      A[i,i]=2

      for j in range(max(0,i-2),i):
```

```
            if abs(i-j)==1:
                  A[i,i]=-1
                  A[j,i]=-1
    return A
```

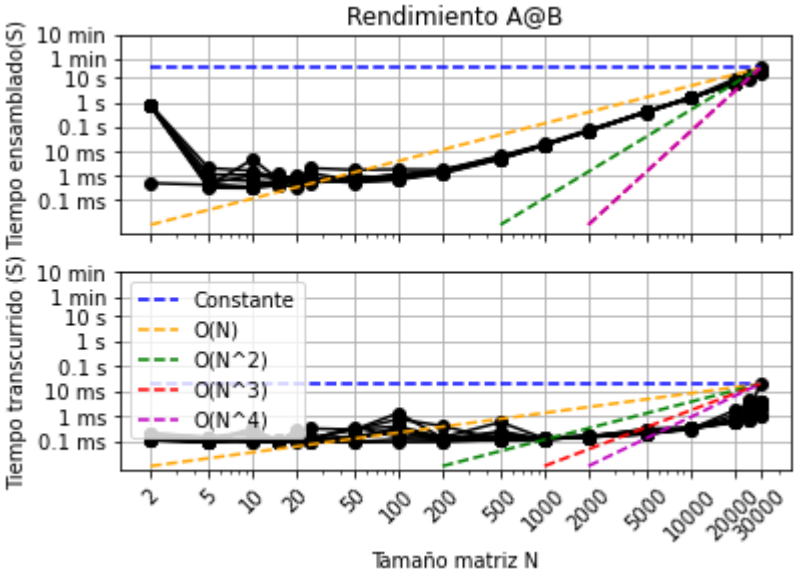
El cual fue el escogido debido a que por un motivo quizas de actualizaciones de librerias o algo que no logre entender era el unico que me funcionaba para poder lograr tanto la matriz llena como la dispersa, otros codigos me funcionaban en una matriz pero en la otra no asi que para temas practicos de comparacion utilice esta para ver las reales diferencias que existen entre estas dos matrices y asi ver el tiempo de ejecucion de cada una de estas.

A continuación se mostraran los graficos obtenidos para el rendimiento de MATMUL tanto para matrices dispersas como llenas

Matriz llena:



Matriz dispersa:



A raiz de los resultados obtenidos se puede apreciar que con las matrices dispersas ahorramos mucha eficiencia debido a que esta no considera los valores de 0 lo cual hace que el tiempo de resolucion de la multiplicacion sea mucho mas optimizada y por ende mucho mas rapida a diferencia de lo que ocurre con la matriz llena la cual considera los valores de ceros y lo cual hace que el tiempo de resolución del problema sea mucho pero mucho mas alto. Debido a lo mencionado anteriormente podemos ver que por un lado el tiempo con las matrices dispersas es considerablemente mas bajo por lo cual nos da margen para poder aumentar el Tamaño del N de las matrices que creamos lo cual fue logrado pero solamente hasta cierto punto debido a que despues de un cierto N (N=30.000 en mi caso) el problema no es con el tiempo de ejecucion sino con la memoria que llega al limite en mi computador debido a que se esta trabajando con matrices demasiado elevadas, a continuacion adjuntare una foto del error por memoria y la capacidad a la cual llega el computador.

```
  File "C:\Users\matia\OneDrive\Escritorio\Entrega5\MatrizDispersa.py", line 41, in <module>
    a=laplaciana(N,dtype=double) #dependiendo el caso lo cambio para float o double

  File "C:\Users\matia\OneDrive\Escritorio\Entrega5\MatrizDispersa.py", line 14, in
laplaciana
    A = zeros((N,N),dtype=dtype)

MemoryError: Unable to allocate 1.16 TiB for an array with shape (400000, 400000) and data
type float64


In [3]:
```

Entrega 4

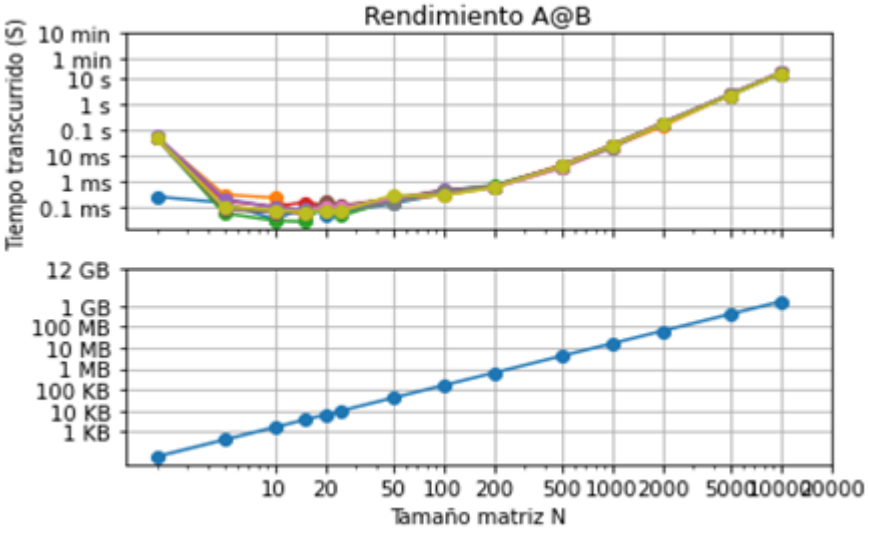
Comentario: La variabilidad del tiempo es bastante similar entre procedimientos con el mismo método de ejecución, el cual cambia considerablemente cuando pasamos de ocupar solve a o utilizar eight debido a que con este ultimo al momento de utilizar una matriz grande (N=10000) toma un tiempo considerablemente mayor al que se logra resolviéndolo con la línea de comando SOIVE.

Con respecto al algoritmo que gana en promedio se puede decir que es mucho más eficiente la línea de comando SOLVE debido a que esta para el máximo tamaño que se utilizo de matriz (N=10000) toma un tiempo promedio de ejecución que ronda entre los 20 a 30 segundos, lo cual se aleja mucho de los tiempos dados para este mismo tamaño de matriz para la línea de comando eight ya que esta misma para los mismos tamaños de matriz utilizados esta rondando en un tiempo promedio entre 105 y 120 segundos.

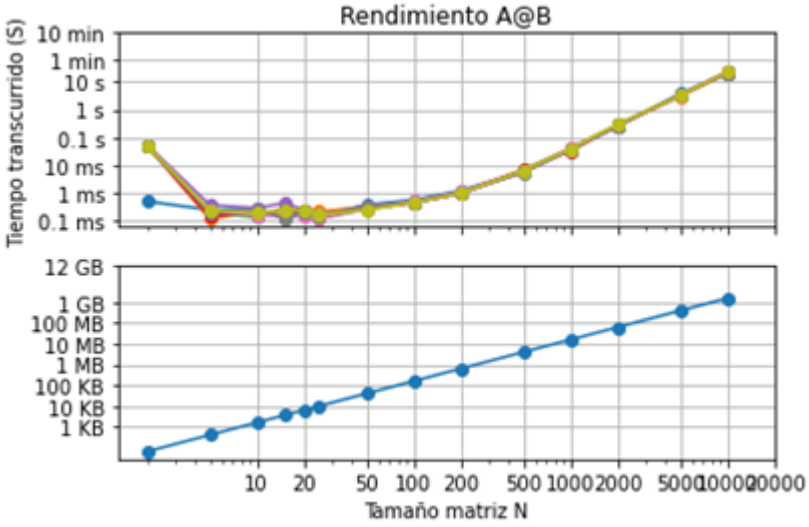
El tiempo que se demora la ejecución se puede apreciar que dependerá del tamaño de la matriz que se quiera ejecutar , entre mas grande la matriz mas tiempo se demorara y como se puede apreciar en los gráficos esta razón no es lineal y después del tamaño de matriz 5000 cualquier proceso independiente del método que se utilice se demorara mucho mas que si se calcularan matrices de tamaño 500 hacia abajo.

Con respecto a esta superioridad se debe a que el computador utiliza mejor el proceso de la línea de comando solve a eight pudiendo utilizar todos los procesadores de una forma mas optima y eficiente lo cual se refleja en los tiempos de ejecución. gráficos de rendimiento para cada caso: Casos float : A.I) image

gráficos de rendimiento para cada caso: Casos float :



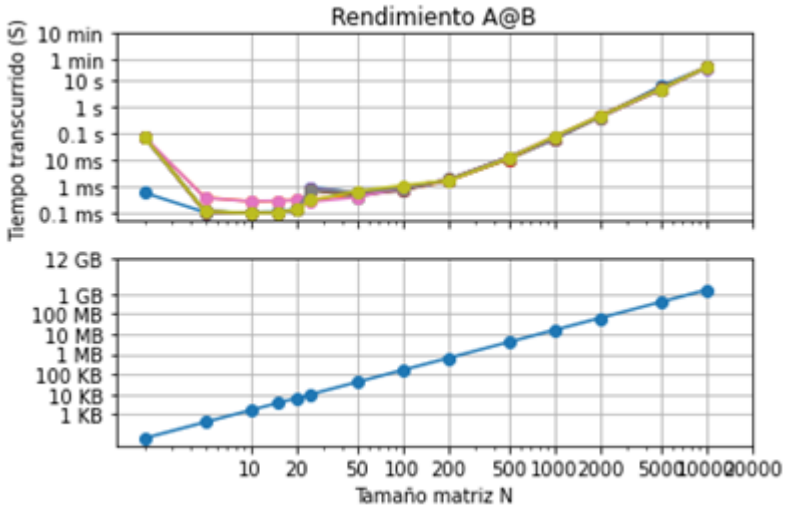
A.I)



A.II)

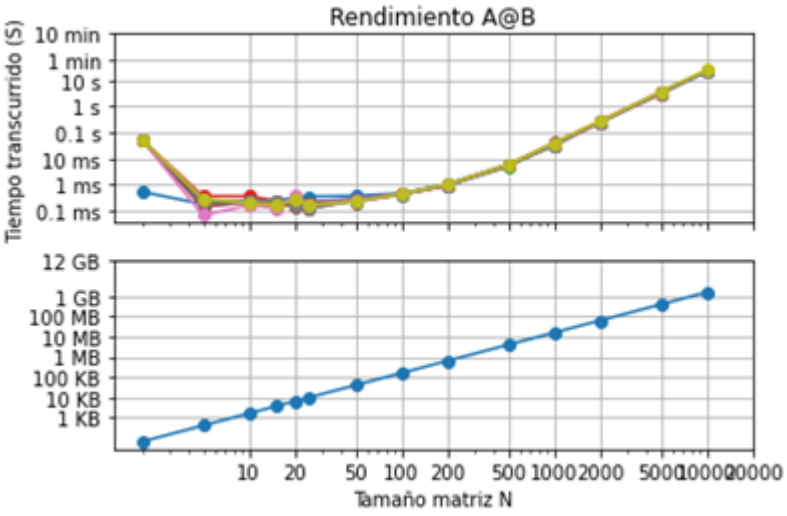
A.III)

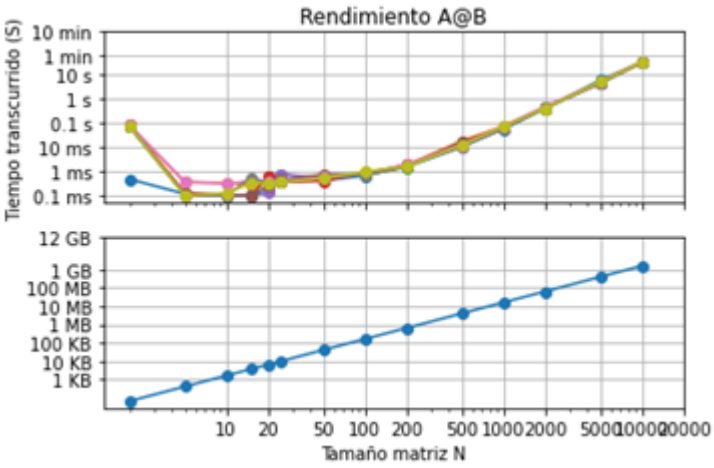
```
Users/matia/OneDrive/Escritorio/Tarea4/Caso1')

In [30]: runfile('C:/Users/matia/OneDrive/Escritorio/Tarea4/Caso1/grafico.py', wdir='C:/
Users/matia/OneDrive/Escritorio/Tarea4/Caso1')
Traceback (most recent call last):

  File "C:\Users\matia\OneDrive\Escritorio\Tarea4\Caso1\grafico.py", line 73, in <module>
    plt.loglog(x,y[1],marker="o")

IndexError: list index out of range


In [31]:
```



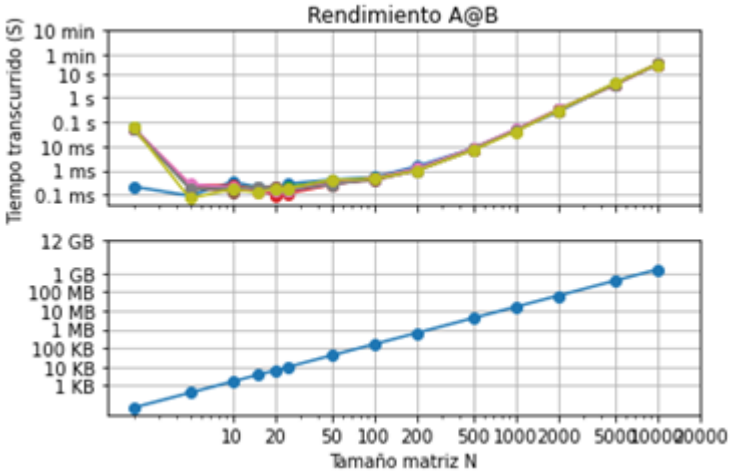
A.IV)

A.V)

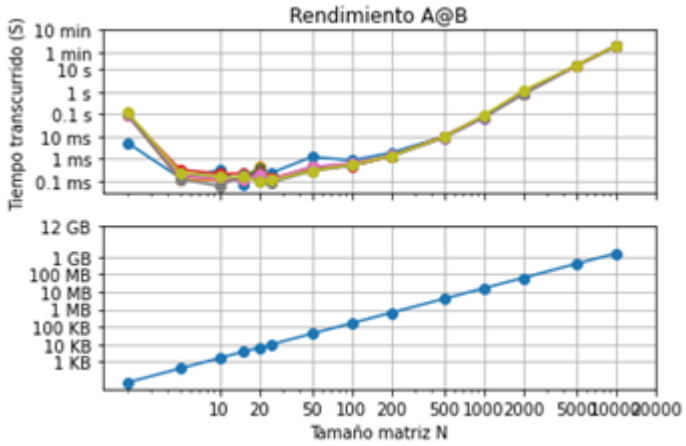

Rendimiento A@B
Tiempo transcurrido (S)
10 min
1 min
10 s
1 s
0.1 s
10 ms
1 ms
0.1 ms
12 GB
1 GB
100 MB
10 MB
1 MB
100 KB
10 KB
1 KB
10 20 50 100 200 500 1000 2000 5000 10000 20000
Tamaño matriz N


Rendimiento A@B
Tiempo transcurrido (S)
10 min
1 min
10 s
1 s
0.1 s
10 ms
1 ms
0.1 ms
12 GB
1 GB
100 MB
10 MB
1 MB
100 KB
10 KB
1 KB
10 20 50 100 200 500 1000 2000 5000 10000 20000
Tamaño matriz N

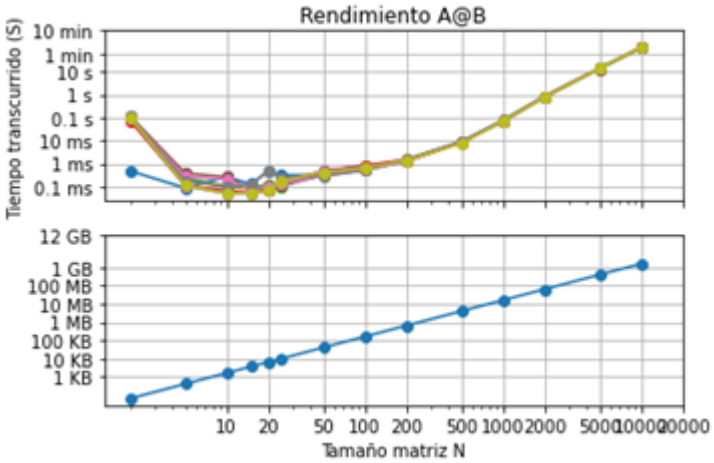
A.VI)


Rendimiento A@B
Tiempo transcurrido (S)
10 min
1 min
10 s
1 s
0.1 s
10 ms
1 ms
0.1 ms
12 GB
1 GB
100 MB
10 MB
1 MB
100 KB
10 KB
1 KB
10 20 50 100 200 500 1000 2000 5000 10000 20000
Tamaño matriz N

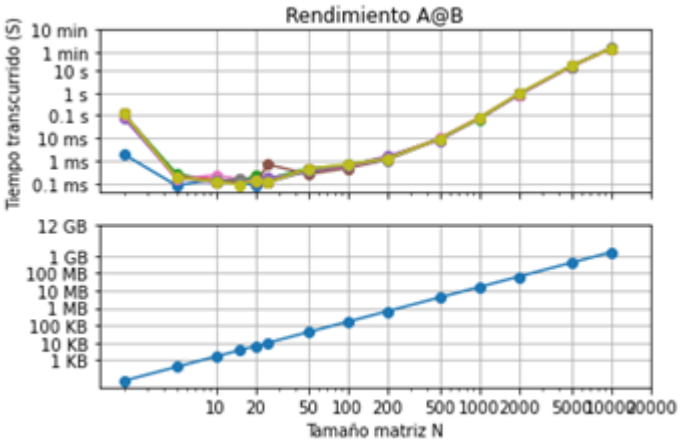
A.Vii


Rendimiento A@B
Tiempo transcurrido (S)
10 min
1 min
10 s
1 s
0.1 s
10 ms
1 ms
0.1 ms
12 GB
1 GB
100 MB
10 MB
1 MB
100 KB
10 KB
1 KB
10 20 50 100 200 500 1000 2000 5000 10000 20000
Tamaño matriz N

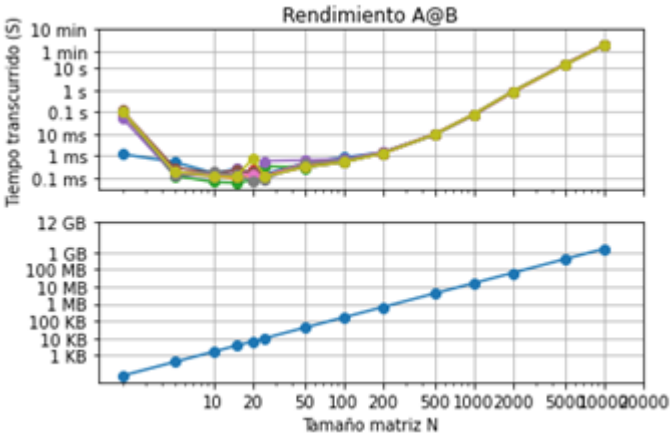
B.I)


Rendimiento A@B
Tiempo transcurrido (S)
10 min
1 min
10 s
1 s
0.1 s
10 ms
1 ms
0.1 ms
12 GB
1 GB
100 MB
10 MB
1 MB
100 KB
10 KB
1 KB
10 20 50 100 200 500 1000 2000 5000 10000 20000
Tamaño matriz N

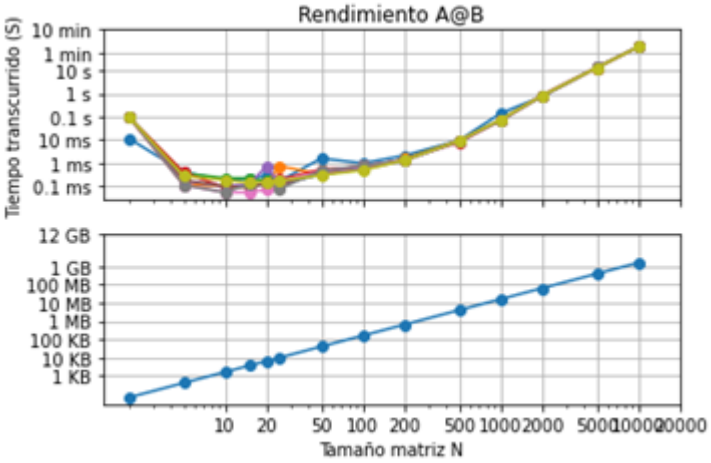
B.II)


Rendimiento A@B
Tiempo transcurrido (S)
10 min
1 min
10 s
1 s
0.1 s
10 ms
1 ms
0.1 ms
12 GB
1 GB
100 MB
10 MB
1 MB
100 KB
10 KB
1 KB
10 20 50 100 200 500 1000 2000 5000 10000 20000
Tamaño matriz N

B.III)

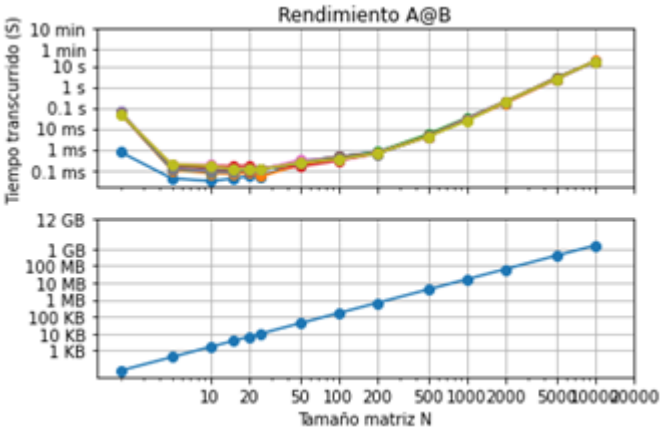


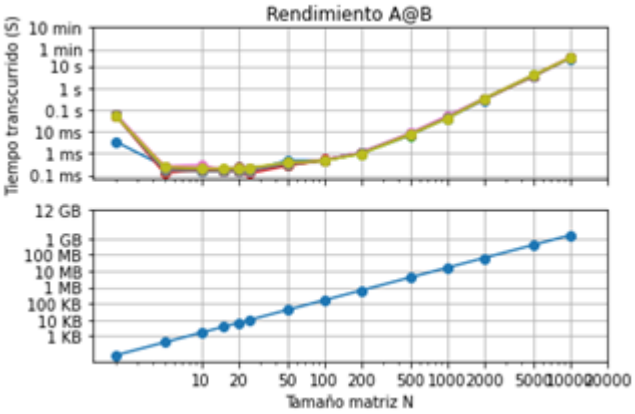
B.IV)



B.V)

Casos double : A.I)



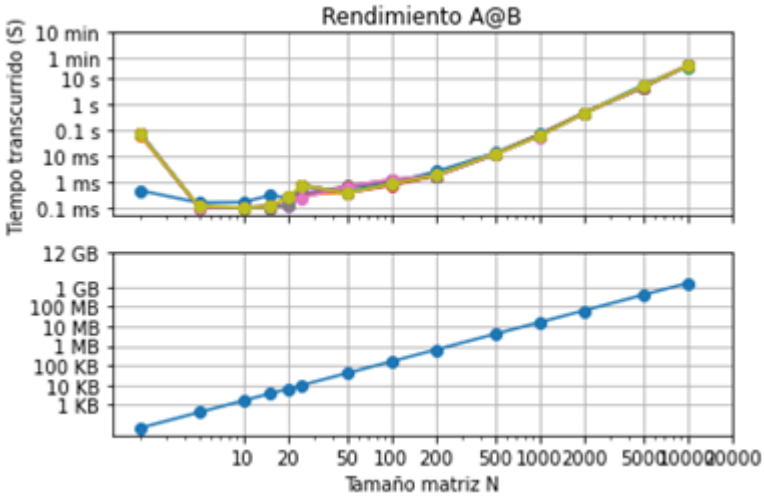


A.II)

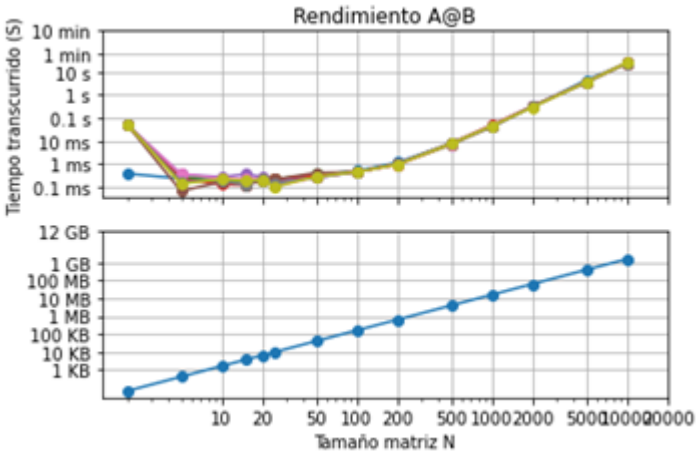
A.III)

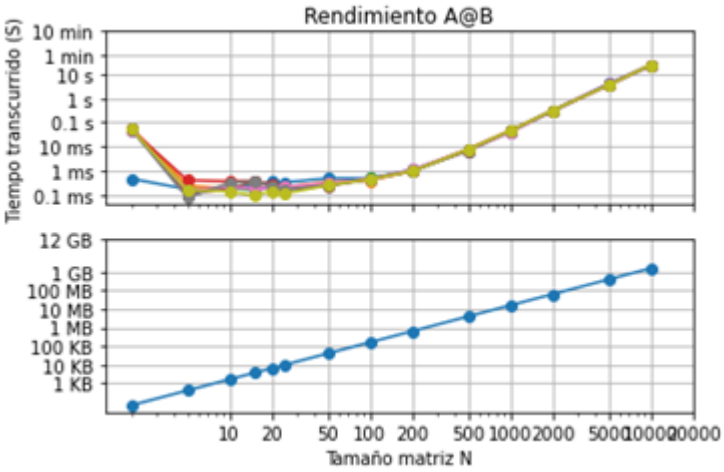
```
Users/matia/OneDrive/Escritorio/Tarea4/Caso1')

In [30]: runfile('C:/Users/matia/OneDrive/Escritorio/Tarea4/Caso1/grafico.py', wdir='C:/
Users/matia/OneDrive/Escritorio/Tarea4/Caso1')
Traceback (most recent call last):

  File "C:\Users\matia\OneDrive\Escritorio\Tarea4\Caso1\grafico.py", line 73, in <module>
    plt.loglog(x,y[1],marker="o")

IndexError: list index out of range


In [31]:
```

A.IV)

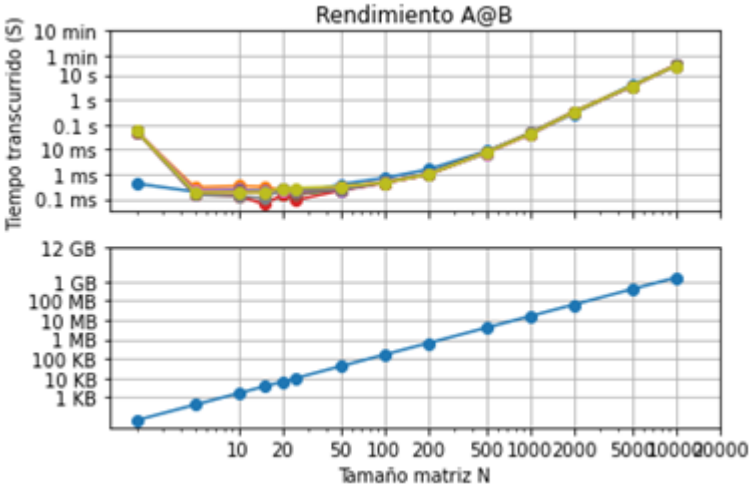


A.V)

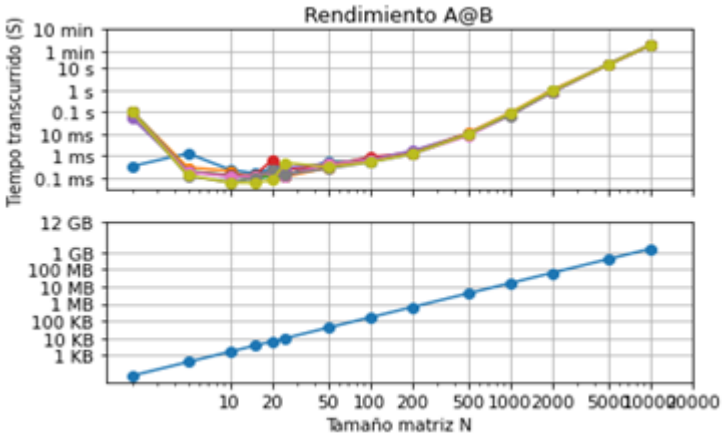

Rendimiento A@B
Tiempo transcurrido (S)
10 min
1 min
10 s
1 s
0.1 s
10 ms
1 ms
0.1 ms
12 GB
1 GB
100 MB
10 MB
1 MB
100 KB
10 KB
1 KB
10 20 50 100 200 500 1000 2000 5000 10000 20000
Tamaño matriz N


Rendimiento A@B
Tiempo transcurrido (S)
10 min
1 min
10 s
1 s
0.1 s
10 ms
1 ms
0.1 ms
12 GB
1 GB
100 MB
10 MB
1 MB
100 KB
10 KB
1 KB
10 20 50 100 200 500 1000 2000 5000 10000 20000
Tamaño matriz N

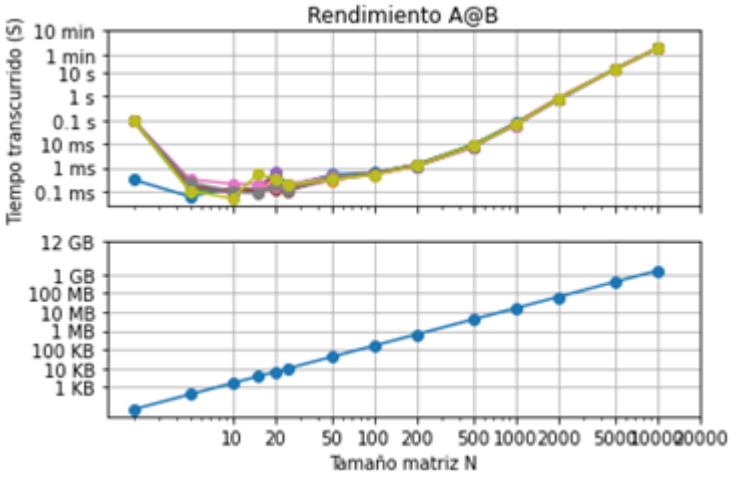
A.VI)


Rendimiento A@B
Tiempo transcurrido (S)
10 min
1 min
10 s
1 s
0.1 s
10 ms
1 ms
0.1 ms
12 GB
1 GB
100 MB
10 MB
1 MB
100 KB
10 KB
1 KB
10 20 50 100 200 500 1000 2000 5000 10000 20000
Tamaño matriz N

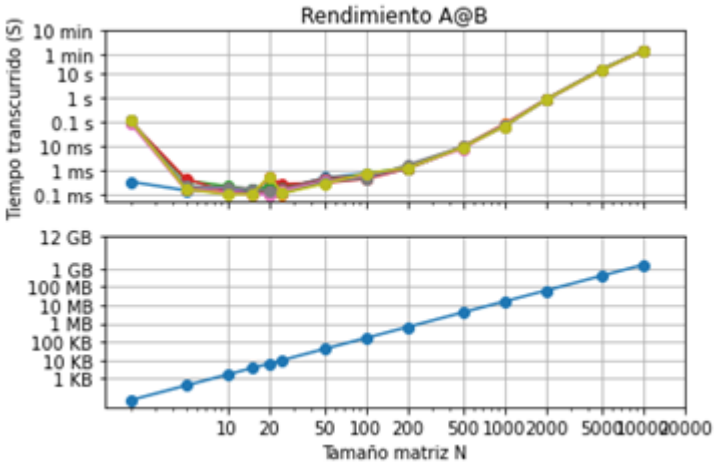
A.VII)


Rendimiento A@B
Tiempo transcurrido (S)
10 min
1 min
10 s
1 s
0.1 s
10 ms
1 ms
0.1 ms
12 GB
1 GB
100 MB
10 MB
1 MB
100 KB
10 KB
1 KB
10 20 50 100 200 500 1000 2000 5000 10000 20000
Tamaño matriz N

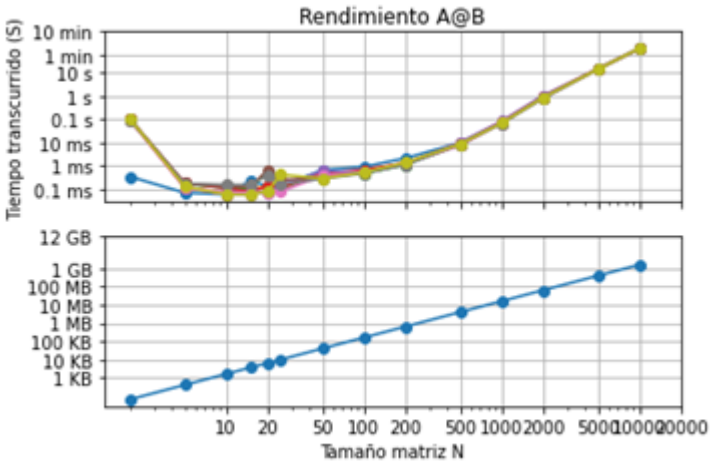
B.I)


Rendimiento A@B
Tiempo transcurrido (S)
10 min
1 min
10 s
1 s
0.1 s
10 ms
1 ms
0.1 ms
12 GB
1 GB
100 MB
10 MB
1 MB
100 KB
10 KB
1 KB
10 20 50 100 200 500 1000 2000 5000 10000 20000
Tamaño matriz N

B.II)

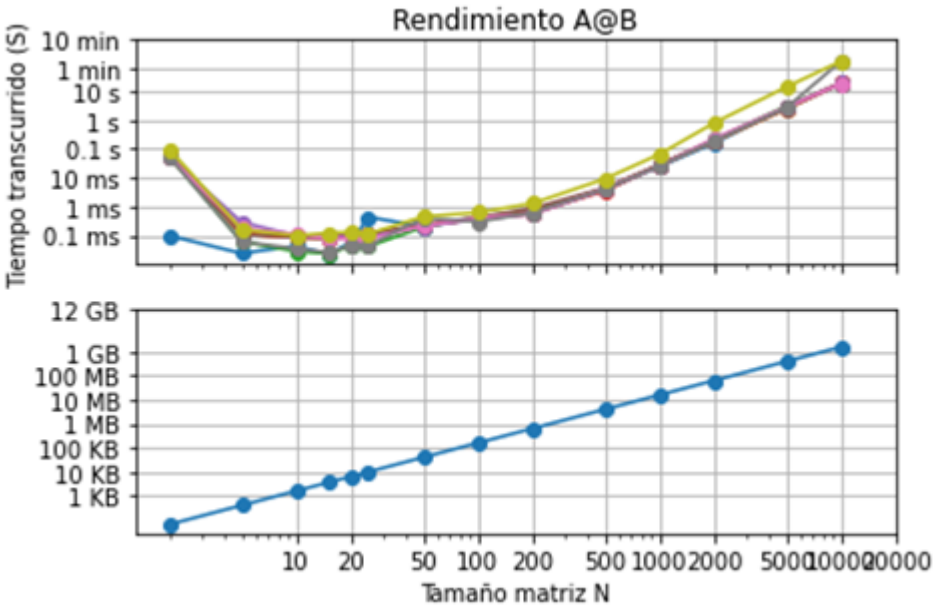

Rendimiento A@B
Tiempo transcurrido (S)
10 min
1 min
10 s
1 s
0.1 s
10 ms
1 ms
0.1 ms
12 GB
1 GB
100 MB
10 MB
1 MB
100 KB
10 KB
1 KB
10 20 50 100 200 500 1000 2000 5000 10000 20000
Tamaño matriz N

B.III)



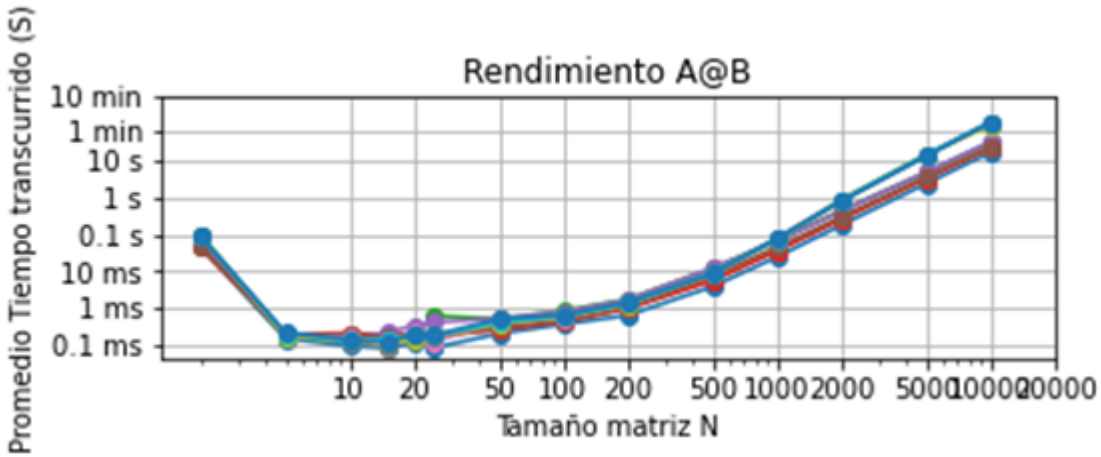
B.IV)

B.V)

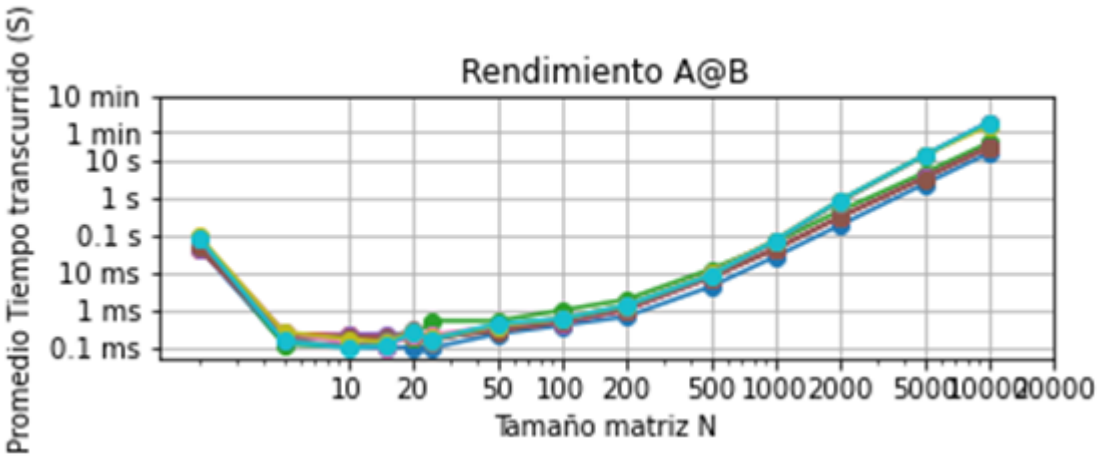


2)Comparación promedio de 10 corridas para cada caso (float-double)

I)Comparación promedio 10 corridas para float



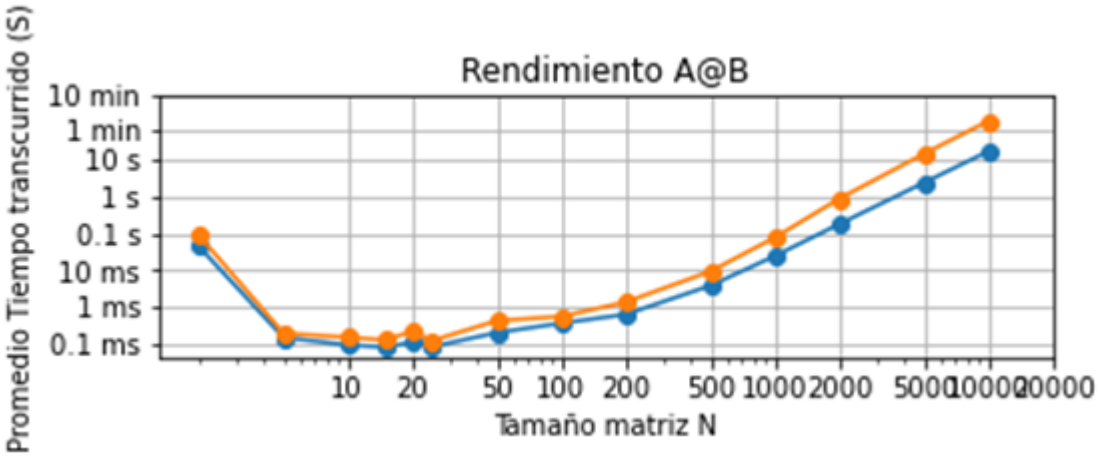
II)Comparación promedio 10 corridas para double



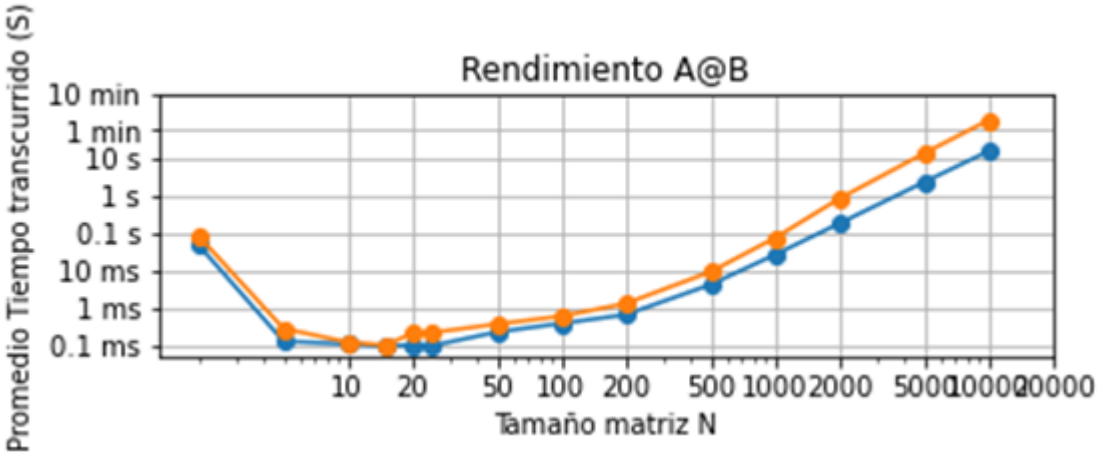
3.

Comparación de rendimientos promedio de desempeño de solve v/s eigh (para caso float y double)

I)Desempeño solve v/s eigh (float)



II)Desempeño solve v/s eigh (double)



#Entrega 3

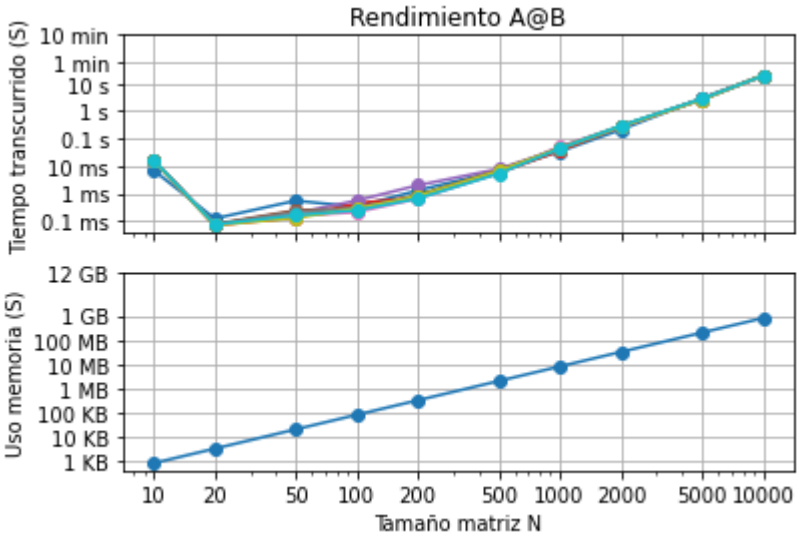
Memeoria y eficiencia cada caso :

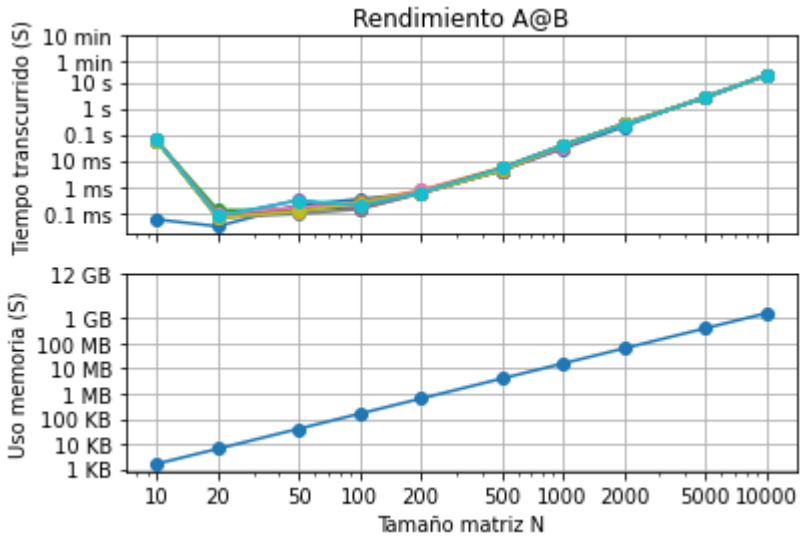
Procedimiento con numpy.linalg

caso 1 (half) :

```
Console 1/A
In [25]: runfile('C:/Users/matia/OneDrive/Escritorio/Clase 2/CasoNumpy/
timing_inv_caso_Numpy_half.py', wdir='C:/Users/matia/OneDrive/Escritorio/Clase 2/CasoNumpy')
Traceback (most recent call last):

  File "C:\Users\matia\OneDrive\Escritorio\Clase 2\CasoNumpy\timing_inv_caso_Numpy_half.py", line 71,
in <module>
    plt.loglog(x,y[0],marker="o")

IndexError: list index out of range


In [26]:
```

caso 2 (single) :



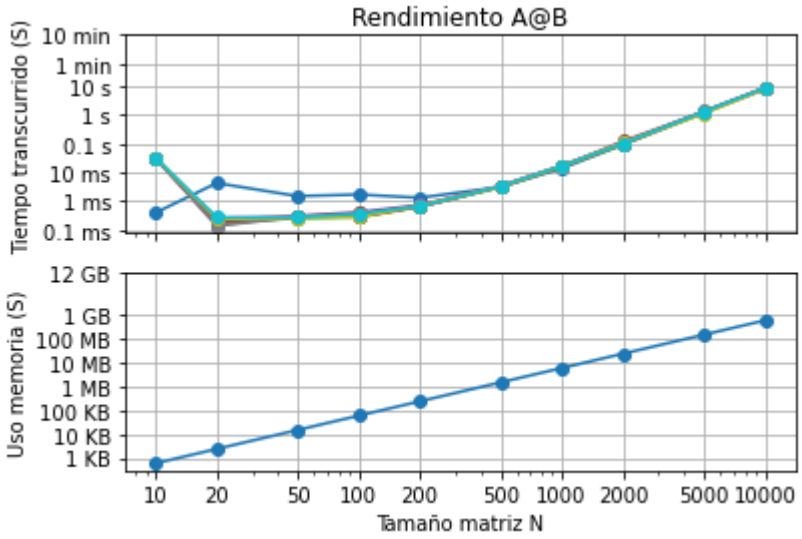


caso 3 (double) :

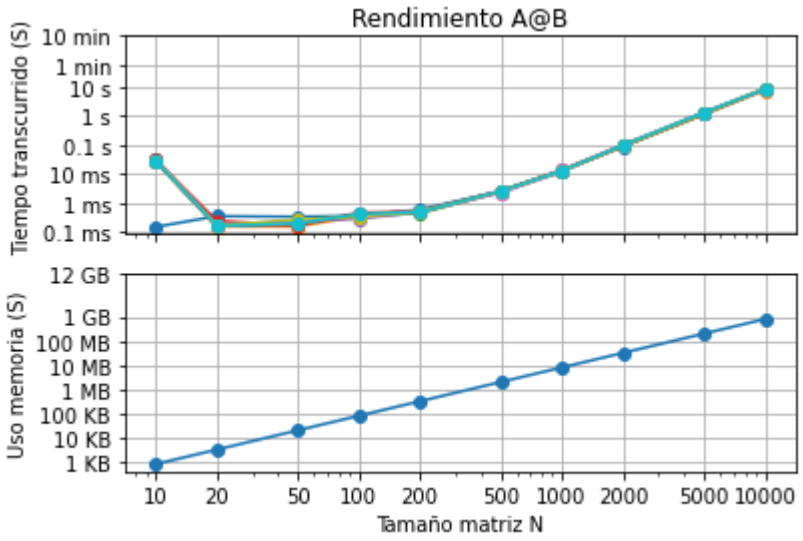
caso 4 (longdouble) :

```
Console 1/A

In [28]: runfile('C:/Users/matia/OneDrive/Escritorio/Clase 2/CasoNumpy/
timing_inv_caso_Numpy_longdouble.py', wdir='C:/Users/matia/OneDrive/Escritorio/Clase 2/CasoNumpy')
Traceback (most recent call last):

  File "C:\Users\matia\OneDrive\Escritorio\Clase 2\CasoNumpy\timing_inv_caso_Numpy_longdouble.py",
line 72, in <module>
    plt.loglog(x,y[0],marker="o")

IndexError: list index out of range


In [29]:
```

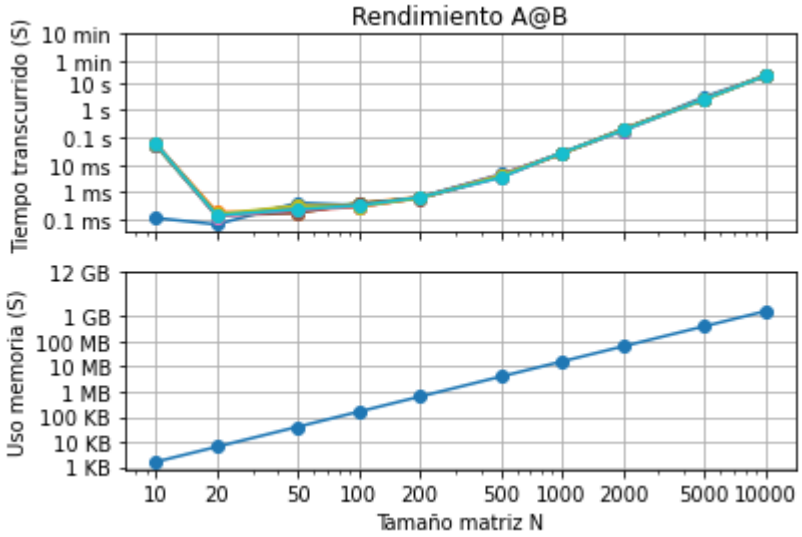
Procedimiento con scipy.linalg (overwrite_a=False)



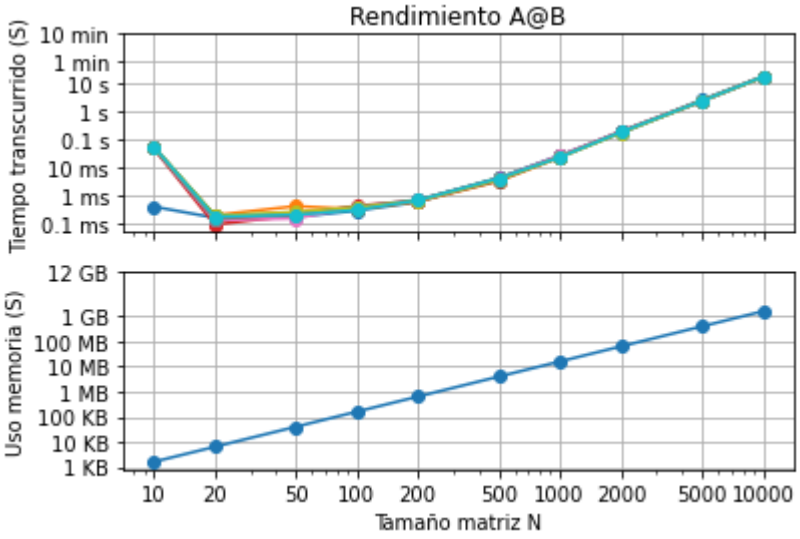
caso 5 (half) :



caso 6 (single) :

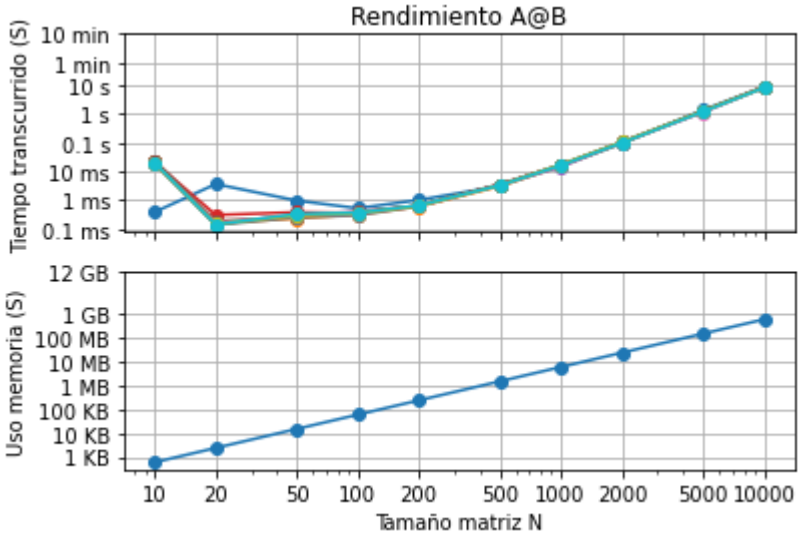


caso 7 (double) :

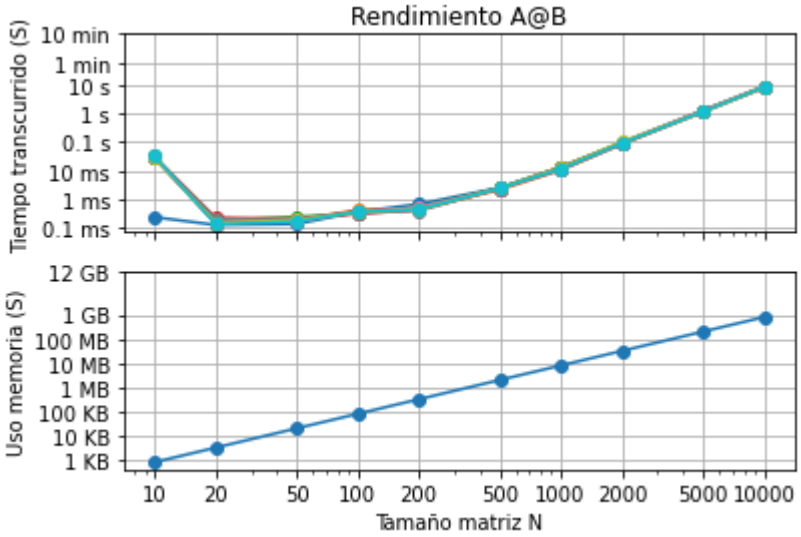


caso 8 (longdouble):

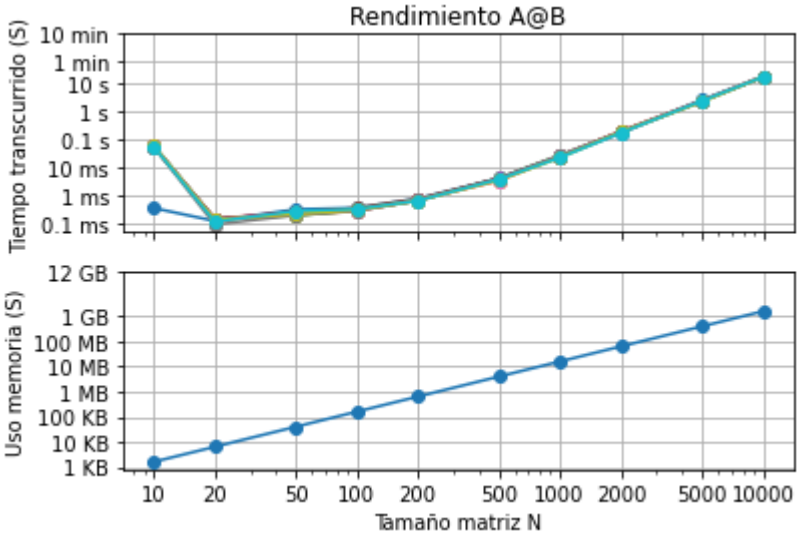
Procedimiento con scipy.linalg (overwrite_a=True)



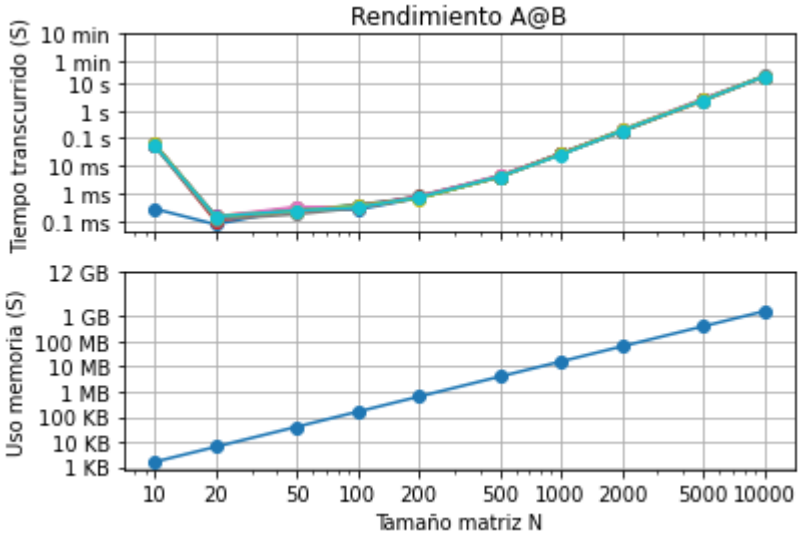
caso 9(half) :



caso 10 (single) :



caso 11 (double) :



caso 12 (longdouble):

PREGUNTAS:

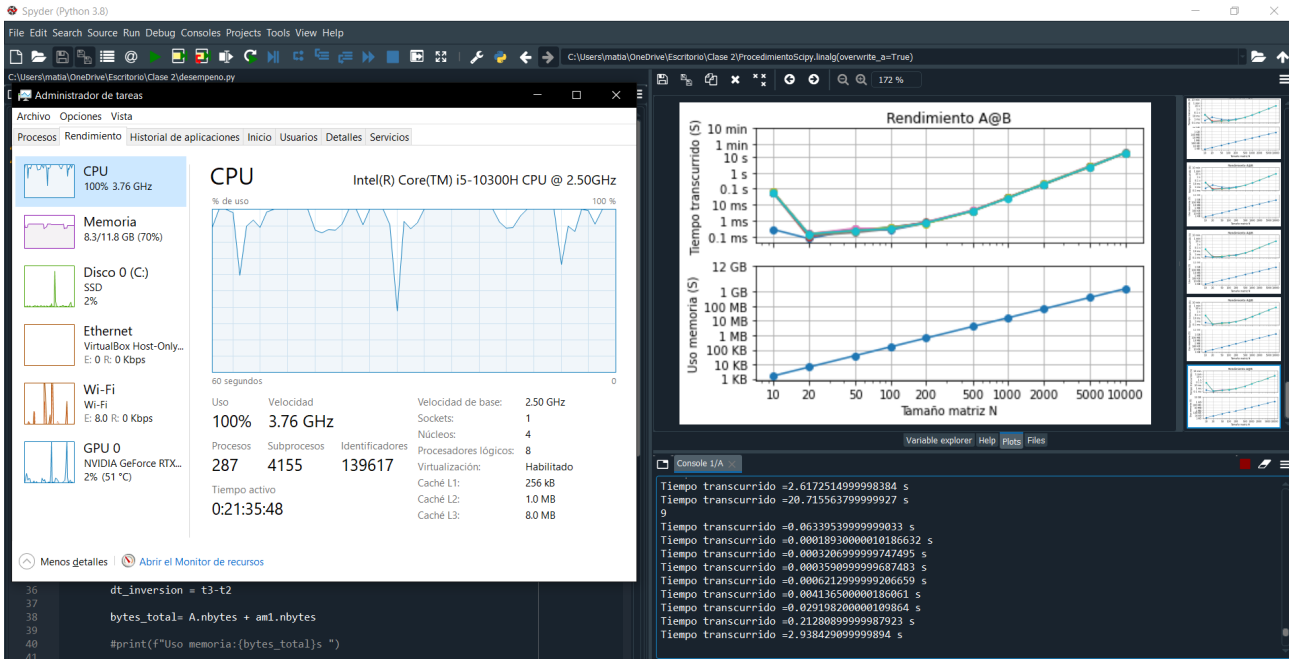
¿Qué algoritmo de inversión cree que utiliza cada método (ver wiki)? Justifique claramente su respuesta.

#half 1 byte #single 2 bytes #double 4 bytes #long double 8 bytes

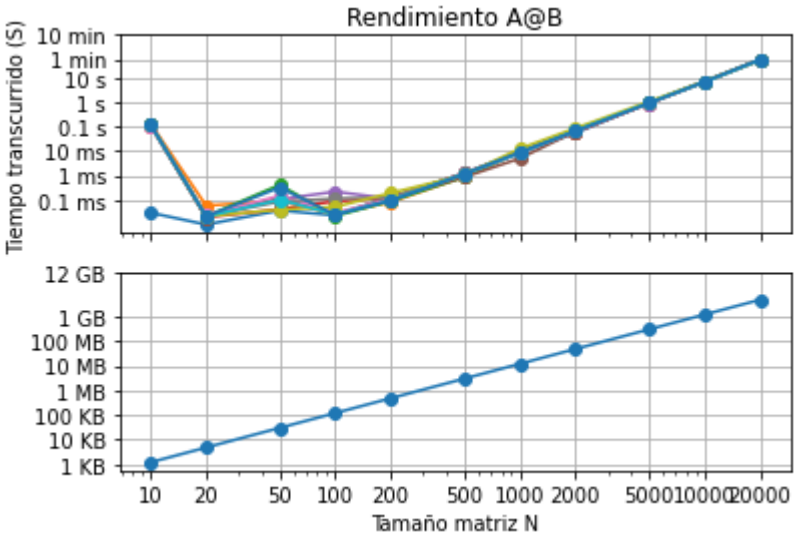
¿Como incide el paralelismo y la estructura de caché de su procesador en el desempeño en cada caso?

incide en el aspecto en que se ejecutara el programa, este implica que los procesadores funcionen simultaneamente al mismo tiempo lo que hace que resuelva varios calculos grandes de manera simultanea para que se acelere el proceso.

A continuacion se puede apreciar que al funcionar el programa con las 10 corridas seguidas el procesador esta funcionando al maximo de su capacidad y como se puede apreciar en la imagen esta realizando mas de 270 procesos los cuales los esta realizando mediante el paralelismo y capacidad del computador al mismo tiempo.



#Entrega 2 (preguntas):



¿Cómo difiere del gráfico del profesor/ayudante?

Difiere con respecto al inicio del procedimiento de python lo cual se puede dar debido a que mi procesador puede que sea de una velocidad mas baja por lo cual tiene menos procesadores para dedicarle a este proceso

¿A qué se pueden deber las diferencias en cada corrida?

a los distintos procesos que en si el computador esta haciendo entre si , pueden haber mil factores , desde el apretar una tecla , ocupar otras funciones o hasta incluso un correo o algun mensaje de alguna aplicación, depende de la cantidad de procesos que este ejecutando el computador en el instante en que ejecuta el programa.

El gráfico de uso de memoria es lineal con el tamaño de matriz, pero el de tiempo transcurrido no lo es ¿porqué puede ser?

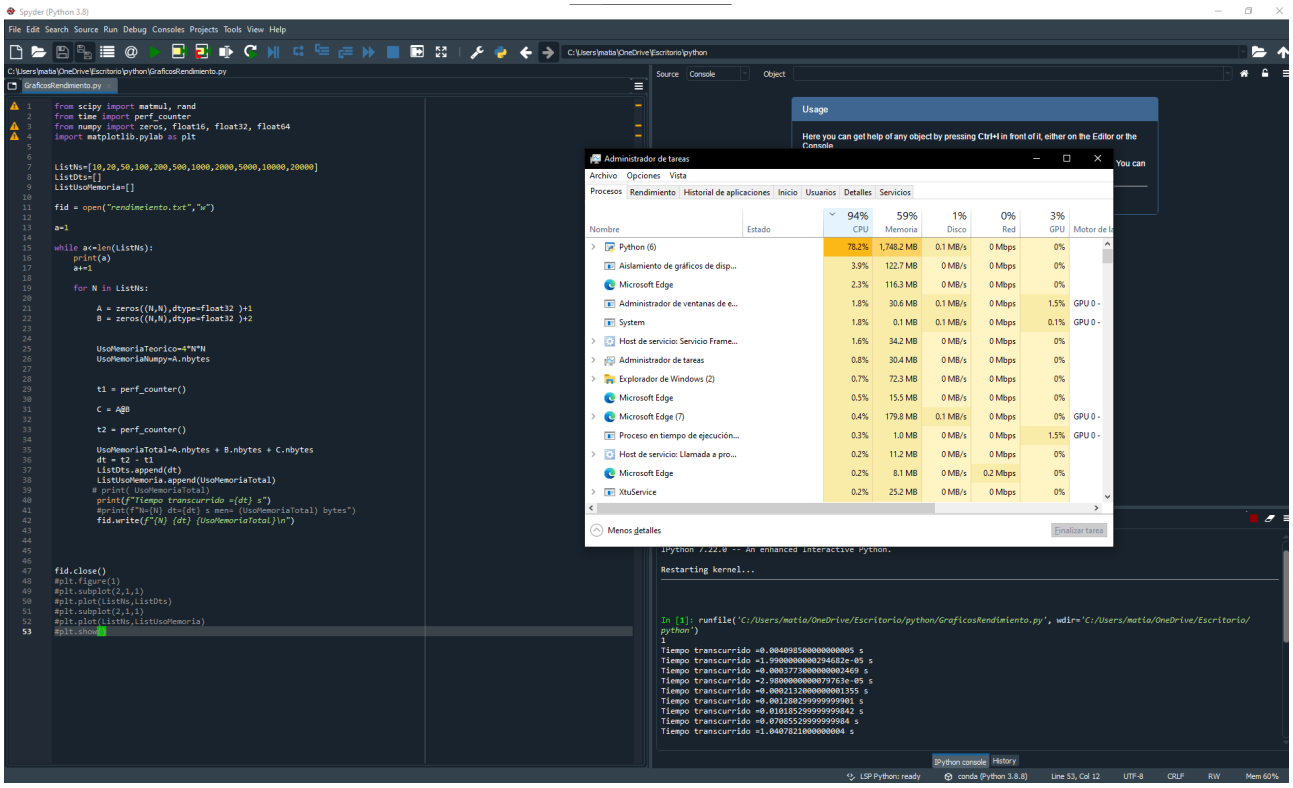
debido a que el proceso que hace siempre tiene un punto de inicio y uno de fin , por ende siempre llegara donde mismo gastando la misma capacidad de memoria siempre , el tiempo varia dependiendo del camino que se tome y de los procesos que esta ejecutando la maquina.Se demorara mucho mas tiempo si se esta jugando por ejemplo a que si el compputador esta solo ejecutando la accion especifica.

¿Qué versión de python está usando? 3.8.8 (default, Apr 13 2021, 15:08:03) [MSC v.1916 64 bit (AMD64)]

¿Qué versión de numpy está usando?

1.20.1

Durante la ejecución de su código ¿se utiliza más de un procesador? Muestre una imagen (screenshot) de su uso de procesador durante alguna corrida para confirmar. si:



MCOC2021-P0

Mi computador principal

- Tipo: Notebook

- Año adquisición: 2020

System Information

```
  Time of this report: 8/5/2021, 20:43:13
         Machine name: LAPTOP-DKSFM6BM
           Machine Id: {4508A5A5-DD57-4464-9501-3D16CCCC1B7B}
     Operating System: Windows 10 Home 64-bit (10.0, Build 19043)
(19041.vb_release.191206-1406)
             Language: Spanish (Regional Setting: Spanish)
  System Manufacturer: HP
         System Model: OMEN Laptop 15-ek0xxx
                 BIOS: F.05 (type: UEFI)
            Processor: Intel(R) Core(TM) i5-10300H CPU @ 2.50GHz (8 CPUs),
~2.5GHz
               Memory: 12288MB RAM
  Available OS Memory: 12128MB RAM
            Page File: 15005MB used, 12929MB available
          Windows Dir: C:\windows
      DirectX Version: DirectX 12
  DX Setup Parameters: Not found
     User DPI Setting: 120 DPI (125 percent)
   System DPI Setting: 96 DPI (100 percent)
      DWM DPI Scaling: Disabled
             Miracast: Available, with HDCP
```

Microsoft Graphics Hybrid: Not Supported DirectX Database Version: 1.2.4 DxDiag Version: 10.00.19041.0928 64bit Unicode