

 main ▾

...

MCOC2021-P0 / README.md

 RobertoVergaraC Ready P0E6

 History

 2 contributors  

 343 lines (246 sloc) | 41.4 KB

...

MCOC2021-P0

Mi computador principal

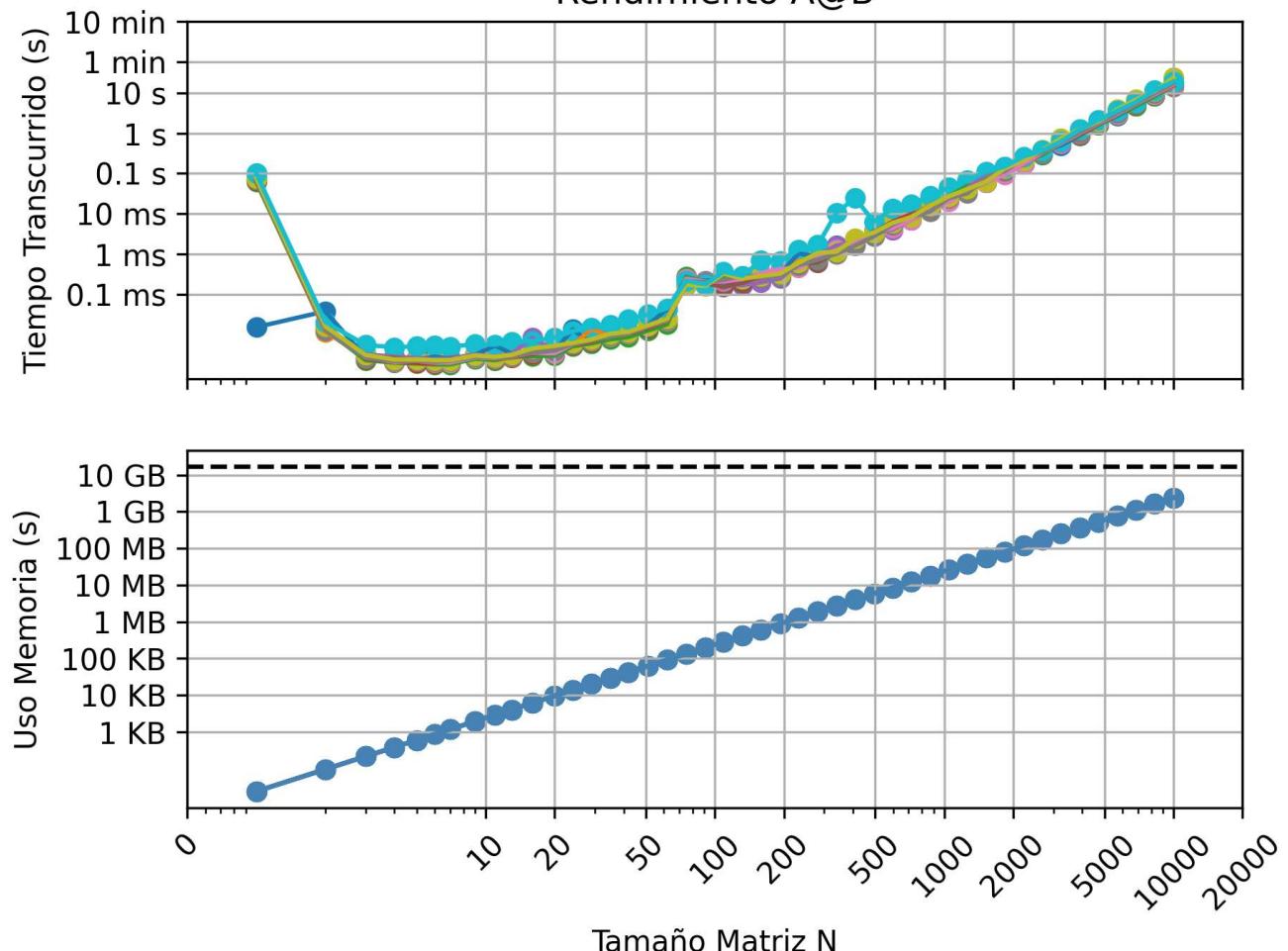
- Marca/modelo: Asus VivoBook S15 X530UF
- Tipo: Notebook
- Año adquisición: 2019
- Procesador:
 - Marca/Modelo: Intel Core i7-8550U
 - Velocidad Base: 1.80 GHz
 - Velocidad Máxima: 4.00 GHz
 - Numero de núcleos: 4
 - Numero de hilos (Procesadores lógicos): 8
 - Arquitectura: x86_64
 - Set de instrucciones: Intel® SSE4.1, Intel® SSE4.2, Intel® AVX2
- Tamaño de las cachés del procesador
 - L1: 256KB
 - L1: 1.0MB
 - L2: 8.0MB
- Memoria
 - Total: 16 GB
 - Tipo memoria: DDR4
 - Velocidad 2400 MHz
 - Numero de (SO)DIMM: 2

- Tarjeta Gráfica
 - Marca / Modelo: Nvidia GeForce MX130
 - Memoria dedicada: 2048 MB GDDR5
 - Resolución: 1920 x 1080
- Disco 1:
 - Marca: SanDisk
 - Tipo: SSD
 - Tamaño: 238GB (en teoría tiene 256GB)
 - Particiones: 3 (OS, SYSTEM, RECOVERY)
 - Sistema de archivos: NTFS (además tiene FAT32 en la partición SYSTEM)
- Disco 2:
 - Marca: Toshiba
 - Tipo: HDD
 - Tamaño: 931GB (en teoría tiene 1TB)
 - Particiones: 1
 - Sistema de archivos: NTFS
- Dirección MAC de la tarjeta wifi: 20:16:B9:44:BD:A3
- Dirección IP (Interna, del router): 192.168.100.2
- Dirección IP (Externa, del ISP): 186.67.234.139
- Proveedor internet: Entel Chile S.A. (fibra óptica)

Desempeño MATMUL

En primer lugar cabe señalar que rendimiento.txt corresponde a un archivo que por cada línea es una lista de listas, en donde cada una de estas es considerada una nueva corrida y está ordenada de la siguiente manera : [[Ns],[dts],[mems]]

Rendimiento A@B



Preguntas

- ¿Cómo difiere del gráfico del profesor/ayudante?

En primer lugar se puede notar como tengo una partida similar en tiempo a las del profesor/ayudante, luego las 9 siguientes tienen un tiempo de partida mayor. Luego es interesante señalar como en matrices con tamaños entre 50 y 60 de evidencia un salto en el tiempo de memoria en el cual en mi notebook es menor que el del profesor/ayudante. Por último el comportamiento final es prácticamente igual, terminando por cada partida un poco inferior a un minuto al igual que el profesor/ayudante. Otra diferencia significativa podría ser debido a la cantidad de N en el eje x que el profesor/ayudante seleccionó vs los que yo establecí (45).
 En estricto rigor se puede evidenciar como el pc del profesor/ayudante tiene un mejor rendimiento en un inicio y en el final, mientras que mi PC tiene mejor rendimiento en los valores de N intermedios.
 Lo anteriormente señalado ocurre debido a las distintas características de caché, RAM, y disco entre el pc del profesor/ayudante y el mio. Pero en general, los gráficos son muy similares.

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

Esto ocurre, ya que, python para el uso de memoria siempre intenta utilizar la menor cantidad de memoria posible. Además, la memoria funciona por bloques, entonces al necesitar más memoria irá en búsqueda del siguiente módulo de memoria y es por esto que se producen los "saltos" en el tiempo que se observan en el gráfico, en estos cambios de memoria manda la jerarquía de memoria es decir, primero caché, luego RAM y por último el disco. Este proceso de "cambio de módulo de memoria" es muy variable y es por eso que se puede evidenciar diferencias en los tiempos de ejecución en cada corrida.

Otro factor que puede influir es el sistema operativo, el cual prioriza distintos procesos (hace una cosa a la vez con distintas prioridades, por lo que hacer diferentes acciones en el pc (como navegar en internet) puede influir directamente), los cuales pueden influir directamente en el tiempo de ejecución, estas pueden ser acciones que uno hace directamente, como también acciones que hace el sistema operativo realiza por detrás sin que uno se de cuenta.

Por último, otros factores que pueden influir son que el computador esté utilizando una gran cantidad de recursos que rellene el proceso (por ejemplo que la temperatura del PC aumente).

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

Esto ocurre ya que, en una multiplicación de matrices existe un número predeterminado de "acciones" por lo que la memoria utilizada en la operación siempre será la misma (es por eso que al graficar las 10 corridas sigue viéndose una única gráfica). Es decir, la memoria utilizada está establecida por la operación y el porte de las matrices y no influirá si esta se desarrolla antes o después a diferencia de lo que ocurre con el tiempo.

Con respecto a la linealidad del gráfico, la memoria utilizada es lineal ya que si una matriz es el doble de grande simplemente utilizará el doble de memoria sin importar otros factores (lo mismo ocurre con la operación MATMUL), pero en el caso del tiempo, a medida que se multiplica una matriz con valores de N más grande, los recursos necesarios van creciendo exponencialmente (es decir una matriz el doble de grande no demorará necesariamente el doble del tiempo sino más), esto debido a la complejidad de la operación MATMUL que va aumentando exponencialmente a medida que aumenta los valores de N.

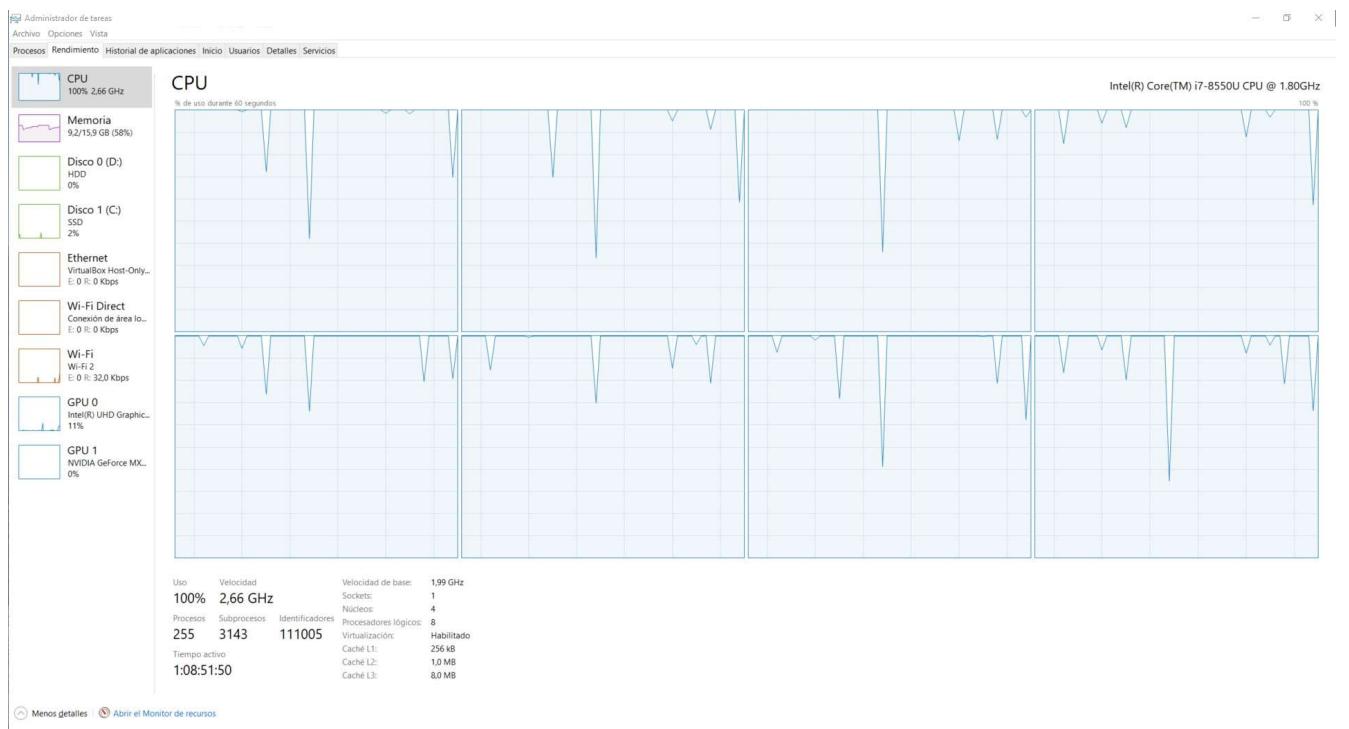
- ¿Qué versión de python está usando?

Python Version: 3.9.6

- ¿Qué versión de numpy está usando?

NumPy Version: 1.21.1

- 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.



Tal como se evidencia en la imagen, durante la ejecución del código se utilizan 8 procesadores, los cuales corresponden a todos los de mi CPU.

El hecho de que utilice los 8 procesadores (y con tanto %uso), es ya que el código necesita demasiados recursos y de esta manera utiliza demasiada CPU a tal punto de necesitar el 100% de su capacidad.

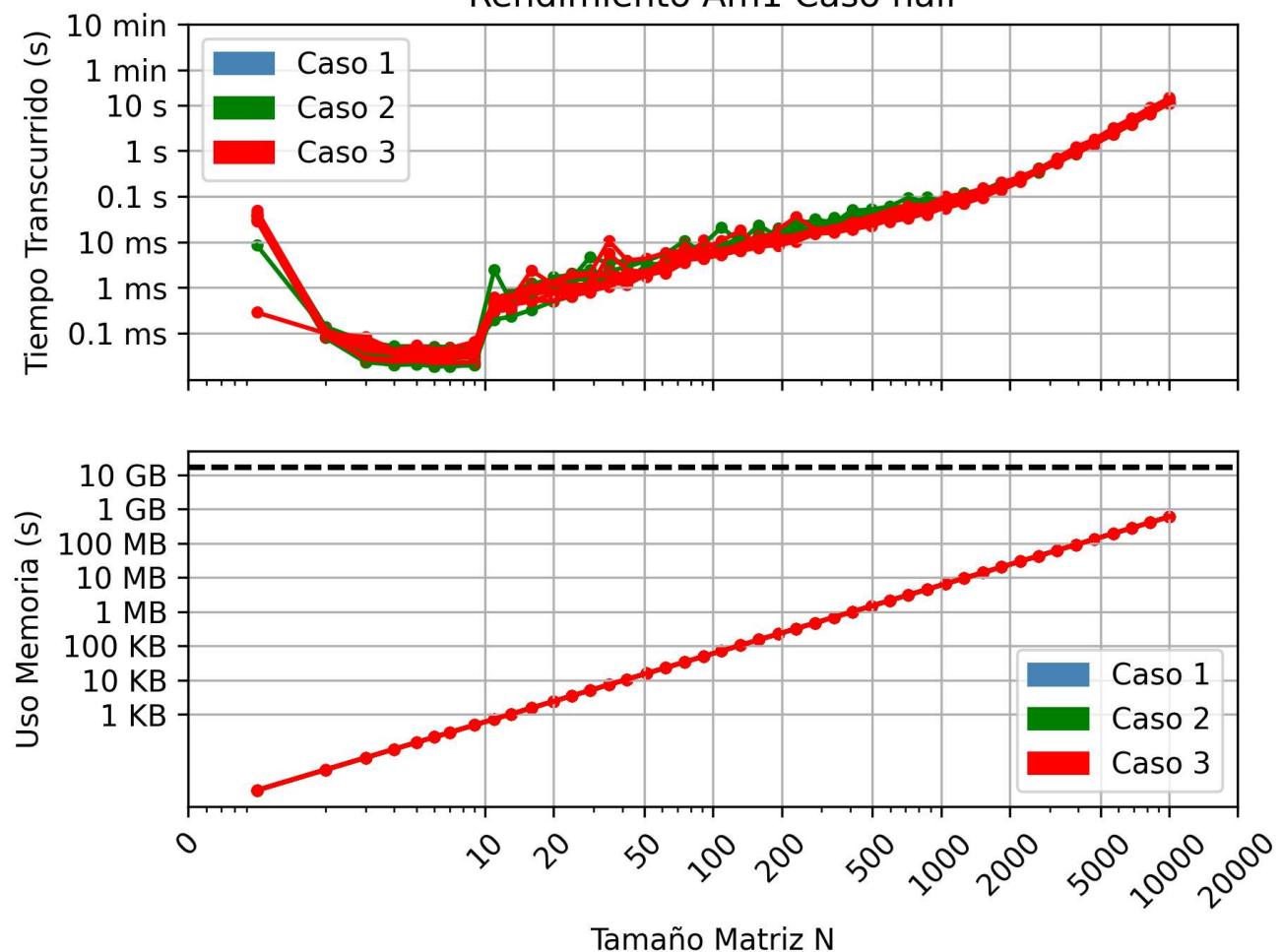
Desempeño de INV

Se realizan 12 archivos .txt, uno por cada tipo en cada uno de los tres casos. Es importante señalar que tanto el tipo `half` y `longdouble` en el caso 1 (usando librería numpy) no se logran realizar, ya que son tipos que `linalg` de numpy no soportan.

A continuación se muestran las comparaciones de los casos por cada tipo de dato:

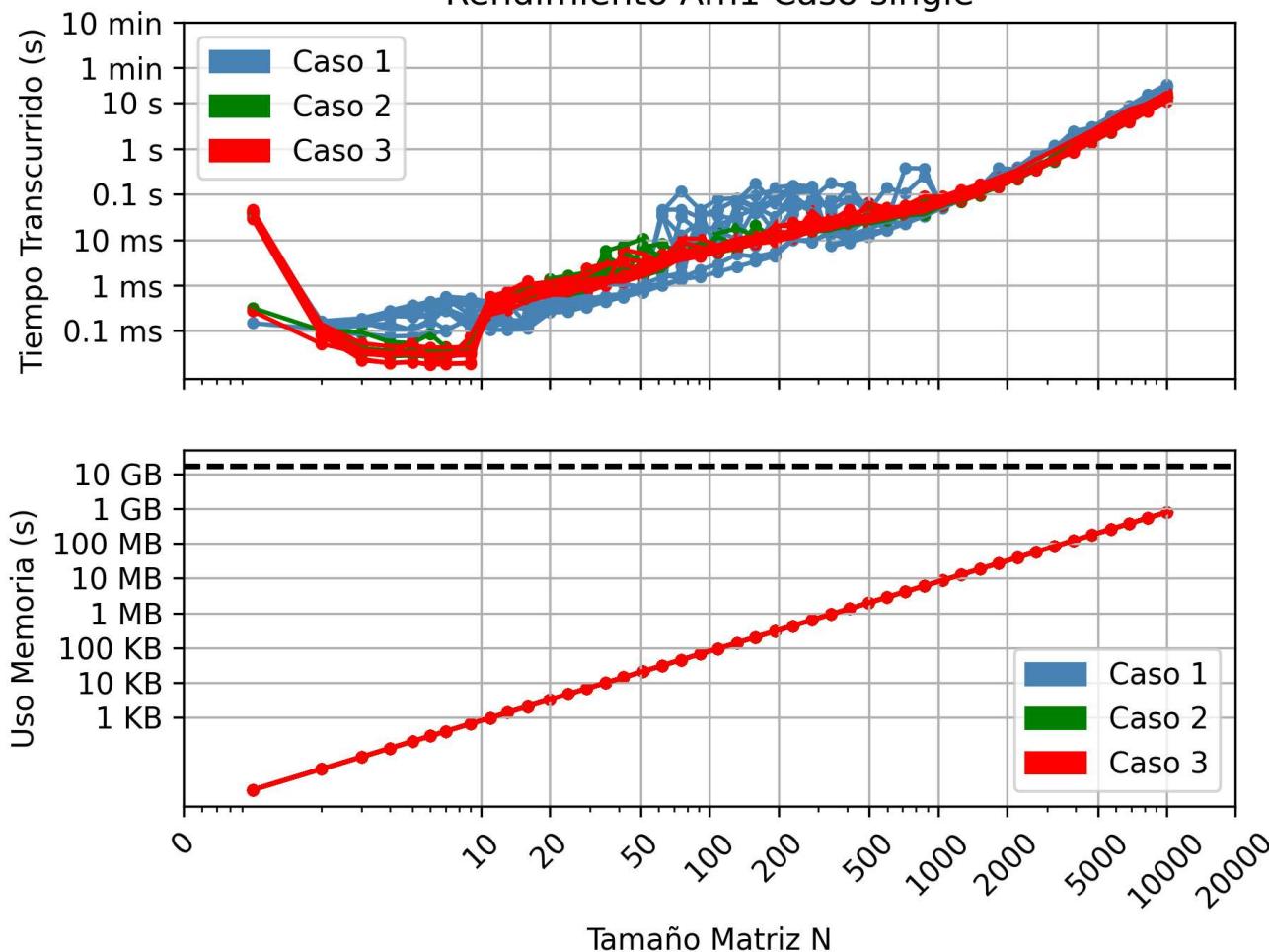
DTYPE: HALF

Rendimiento Am1 Caso half



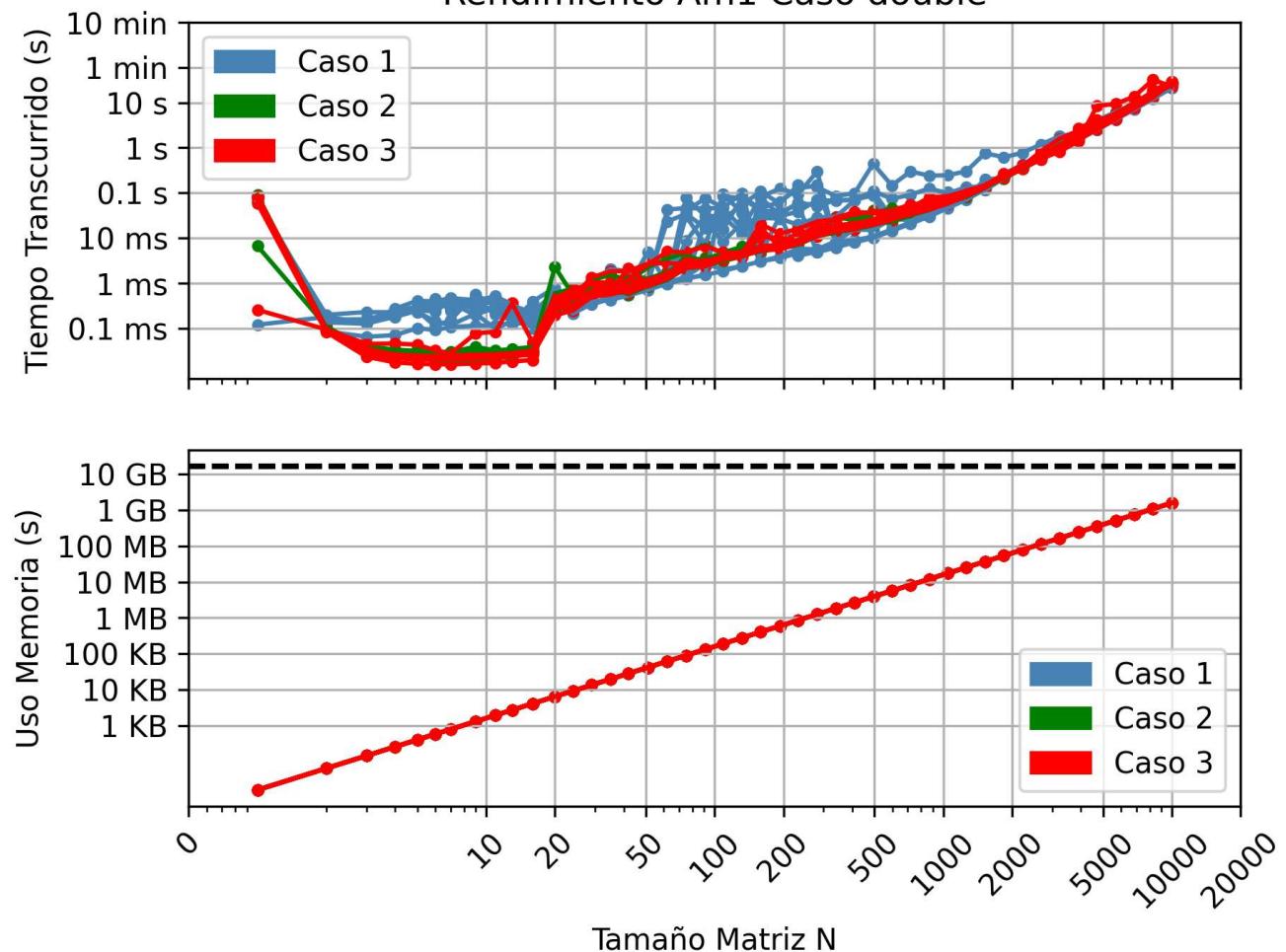
DTYPE: SINGLE

Rendimiento Am1 Caso single



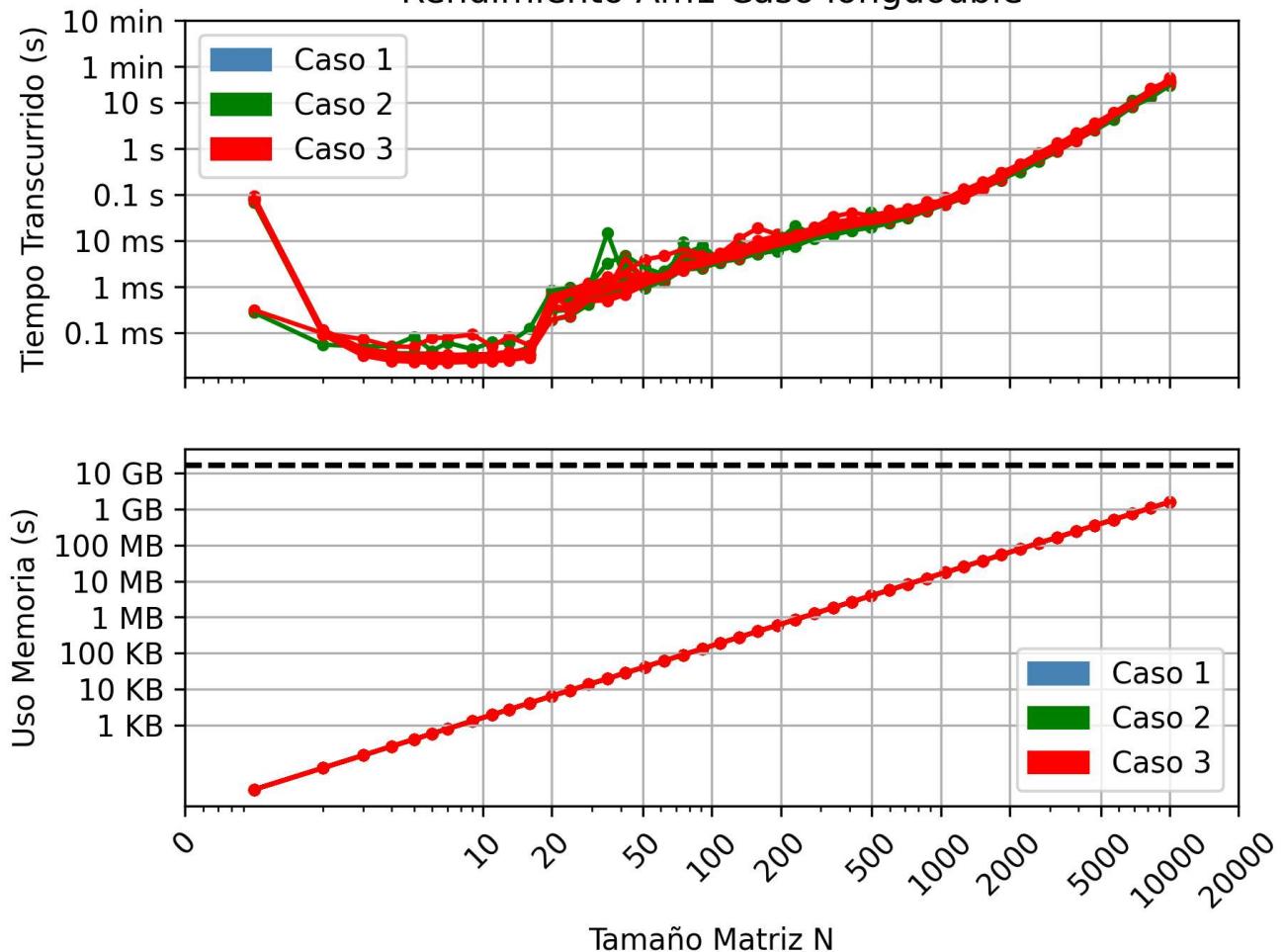
DTYPE: DOUBLE

Rendimiento Am1 Caso double



DTYPE: LONGDOUBLE

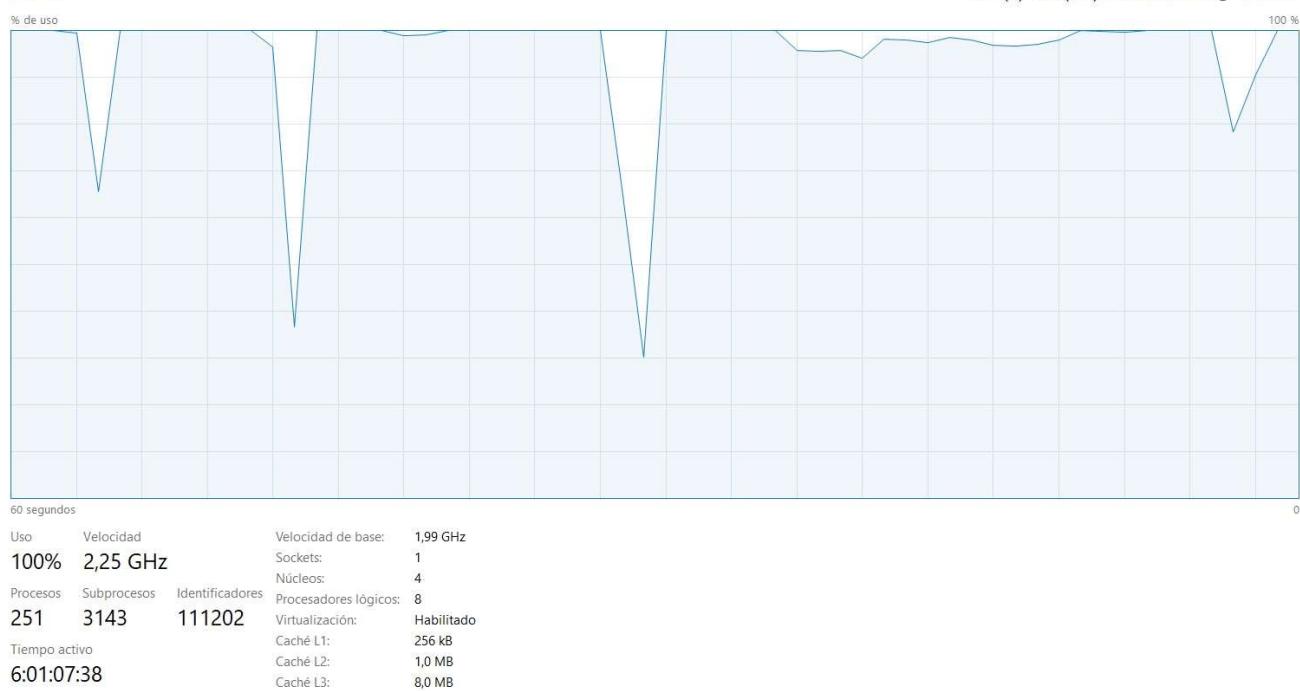
Rendimiento Am1 Caso longdouble



Además tanto el desempeño del procesador como de la memoria en todos los casos que los códigos funcionaron los resultado fueron prácticamente idénticos, es decir el procesador usando casi todos sus recursos (todos incluso), mientras que la memoria prácticamente no cambiaba y si lo hacía era muy leve. Si hubo alguna diferencia fue prácticamente imperceptible.

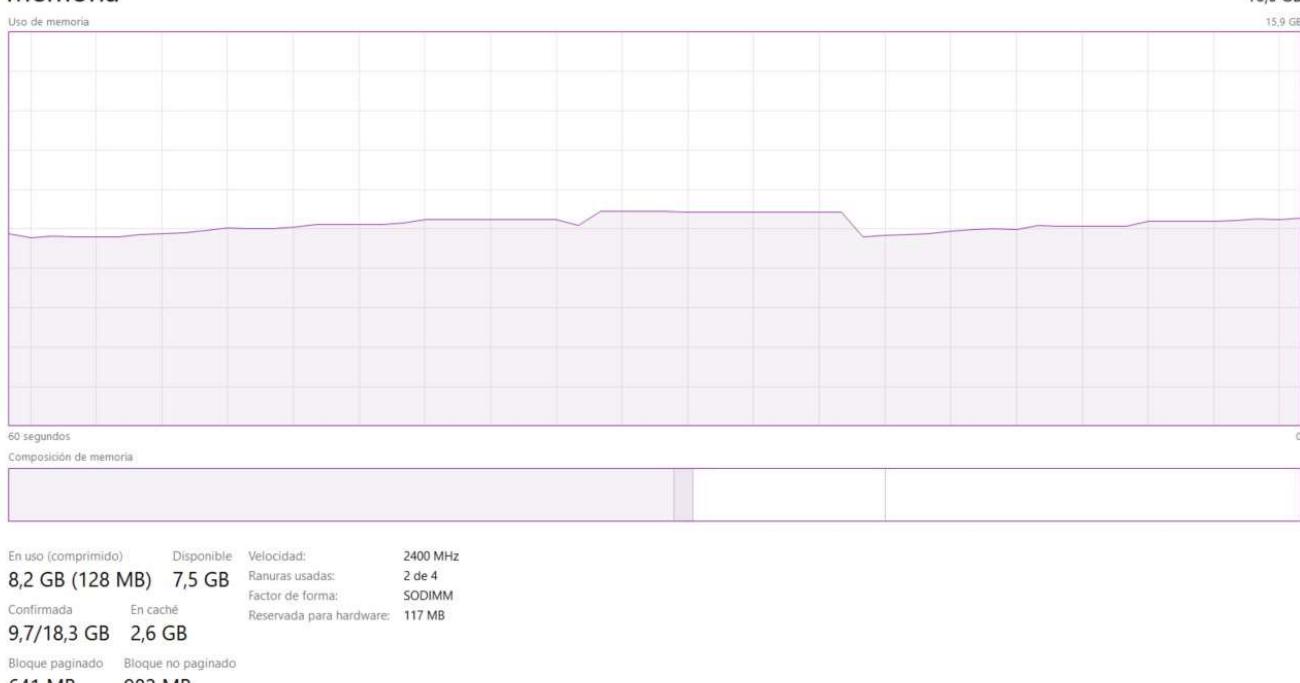
Procesador durante corridas:

CPU



Memoria durante corridas:

Memoria



Preguntas

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

NumPy: numpy.linalg.inv(A) lo que hace es llamar a la función numpy.linalg.solve(A,I) (esta función realiza $Ax = I$, en donde x correspondería en este caso a la matriz inversa de A), en donde I corresponde a la matriz identidad, y luego mediante lapack's LU factorization (paquete de Descomposición LU) resuelve el sistema de solve. Esto significa que finalmente ocupa eliminación Gaussiana en donde la ortogonalidad no se detecta por default. Cabe destacar que este método aumenta el error si el array dado no es cuadrado o la inversión falla.

SciPy: Muy parecido a NumPy, lo que hace Scipy (scipy.linalg.inv(A)) es llamar directamente a los paquetes LAPACK (get_lapack_funcs("función que se quiere")) y desde ahí mediante descomposición LU realiza la inversión de la matriz. Es importante señalar que la opción overwrite_a lo que realiza es reemplazar los valores de la matriz (los va descartando), es por eso que esta opción en True podría incrementar el rendimiento.

Importante decir que se puede notar que scipy realiza en general el proceso más rápido ya que llama directamente a los paquetes para realizar la descomposición Lu, en cambio, numpy.linalg.inv() llama a numpy.linalg.solve(), y es esta función la que llama al mismo paquete, es decir "utiliza un paso más"

- ¿Cómo incide el paralelismo y la estructura de caché de su procesador en el desempeño en cada caso? Justifique su comentario en base al uso de procesadores y memoria observado durante las corridas.

La lógica del paralelismo es realizar varias tareas al mismo tiempo, para esto el computador divide el procesador en "mini procesadores" que cada uno realiza diferentes acciones. Sabiendo esto, al correr el programa con datos como **half** vs **longdouble** el procesador con **half** ("menos información"), trabaja de mejor manera con estos "mini procesadores", ya que esa tarea puede distribuirla mejor entre ellos sin tener que colapsar los caché, en cambio con **longdouble** el procesador necesita usar mayores recursos destinando más cantidad de "mini procesadores" y así dificulta y alarga más el tiempo de ejecución, ya que estos estarán entregando mayor rendimiento con un tamaño de caché al máximo.

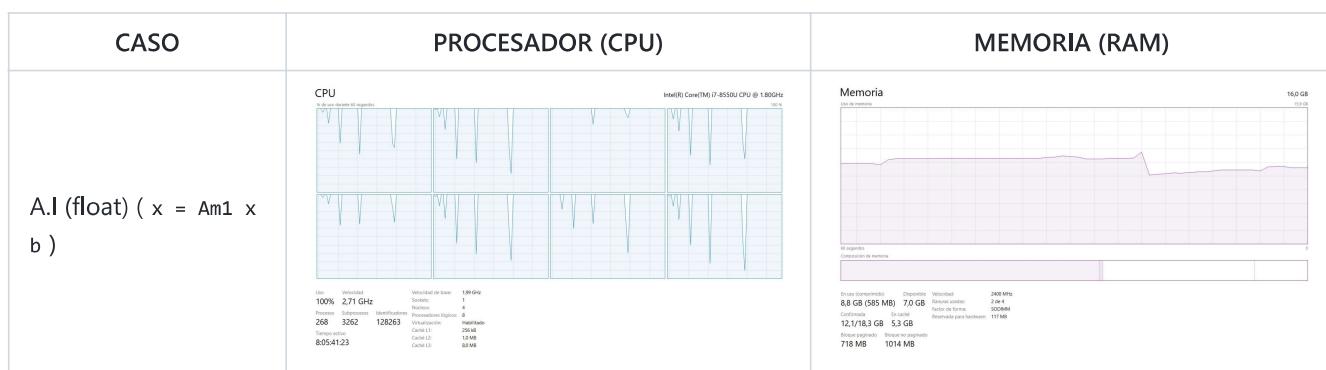
Es importante señalar que en mi caso, para todos los tipos de datos el tiempo de corrida fue similar, ligeramente los datos más pequeños fueron un poco más rápido, pero es de manera prácticamente imperceptible. Esto quiere decir que mi pc, para poder correr datos de **half** utiliza varios de estos "mini procesadores" y con la capacidad de los caché si no es al máximo, muy cercano, en el caso de un dato más grande como **longdouble**, probablemente este usando todo los recursos.

Por último, para el caso de la memoria, al tener una gran cantidad y con mucho desuso no significó un gran problema para todos los tipos de datos, sin embargo con datos pequeños esta trabajo leve e imperceptiblemente menos que con datos más grandes.

Desempeño de SOLVE y EIGH

Se realizaron 4 archivos .py (uno para todos los puntos del caso A, otro para todos los puntos del caso B y esto por cada tipo de dato), además de estos se obtuvieron 4 archivos .txt los cuales por línea contienen los subcasos de A y B, pero solo los valores promedios de las 10 corridas por subcaso (son del tipo [[Ns],[dts]]).

A continuación se muestra una tabla con los desempeños del procesador y la memoria por cada caso (subcaso) realizado.



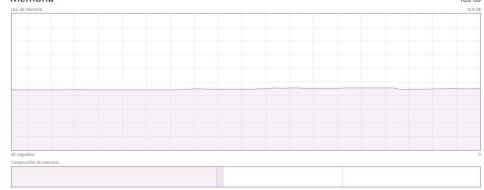
CASO	PROCESADOR (CPU)	MEMORIA (RAM)
A.II (float) (<code>scipy.linalg.solve</code> by default)	<p>CPU</p> <p>Use: Utilidad 100%: 2.35 GHz Procesos: Subprocesos: Identificación: 77% 3012 125757 Velocidad: Velocidad de base: 1.99 GHz Sockets: 1 Nodos: 4 Procesadores lógicos: 8 Vibraciones: Habilitado Tiempo activo: 259 s Cache L1: 13.0 MB Cache L2: 8.0 MB Tiempos actuales: 8:05:55:10</p>	<p>Memoria</p> <p>Uso de memoria: 7.5 GB (551 MB) 8.2 GB Velocidad: 2400 MHz Capacidad: 11.5/18.3 GB 5.3 GB Reservado para hardware: 718 MB Reservado para el sistema: 1012 MB</p>
A.III (float) (using <code>assume_a='pos'</code>)	<p>CPU</p> <p>Use: Utilidad 100%: 2.25 GHz Procesos: Subprocesos: Identificación: 77% 3107 135454 Velocidad: Velocidad de base: 1.99 GHz Sockets: 1 Nodos: 4 Procesadores lógicos: 8 Vibraciones: Habilitado Tiempo activo: 259 s Cache L1: 13.0 MB Cache L2: 8.0 MB Tiempos actuales: 8:05:59:34</p>	<p>Memoria</p> <p>Uso de memoria: 7.5 GB (587 MB) 8.0 GB Velocidad: 2400 MHz Capacidad: 11.5/18.3 GB 5.3 GB Reservado para hardware: 718 MB Reservado para el sistema: 1016 MB</p>
A.IV (float) (using <code>assume_a='sym'</code>)	<p>CPU</p> <p>Use: Utilidad 100%: 2.67 GHz Procesos: Subprocesos: Identificación: 77% 3207 127918 Velocidad: Velocidad de base: 1.99 GHz Sockets: 1 Nodos: 4 Procesadores lógicos: 8 Vibraciones: Habilitado Tiempo activo: 259 s Cache L1: 13.0 MB Cache L2: 8.0 MB Tiempos actuales: 8:06:04:52</p>	<p>Memoria</p> <p>Uso de memoria: 7.0 GB (429 MB) 8.2 GB Velocidad: 2400 MHz Capacidad: 10.5/18.3 GB 5.1 GB Reservado para hardware: 718 MB Reservado para el sistema: 1016 MB</p>
A.V (float) (using <code>overwrite_a=True</code>)	<p>CPU</p> <p>Use: Utilidad 74%: 2.32 GHz Procesos: Subprocesos: Identificación: 74% 3166 127098 Velocidad: Velocidad de base: 1.99 GHz Sockets: 1 Nodos: 4 Procesadores lógicos: 8 Vibraciones: Habilitado Tiempo activo: 259 s Cache L1: 13.0 MB Cache L2: 8.0 MB Tiempos actuales: 8:06:09:37</p>	<p>Memoria</p> <p>Uso de memoria: 7.5 GB (488 MB) 8.7 GB Velocidad: 2400 MHz Capacidad: 10.5/18.3 GB 5.3 GB Reservado para hardware: 719 MB Reservado para el sistema: 1016 MB</p>
A.VI (float) (using <code>overwrite_b=True</code>)	<p>CPU</p> <p>Use: Utilidad 49%: 2.63 GHz Procesos: Subprocesos: Identificación: 49% 3199 127122 Velocidad: Velocidad de base: 1.99 GHz Sockets: 1 Nodos: 4 Procesadores lógicos: 8 Vibraciones: Habilitado Tiempo activo: 259 s Cache L1: 13.0 MB Cache L2: 8.0 MB Tiempos actuales: 8:06:14:20</p>	<p>Memoria</p> <p>Uso de memoria: 7.5 GB (621 MB) 8.2 GB Velocidad: 2400 MHz Capacidad: 10.5/18.3 GB 5.4 GB Reservado para hardware: 719 MB Reservado para el sistema: 1017 MB</p>
A.VII (float) (using <code>overwrite_a=True</code> and <code>overwrite_b=True</code>)	<p>CPU</p> <p>Use: Utilidad 82%: 2.22 GHz Procesos: Subprocesos: Identificación: 82% 3264 127908 Velocidad: Velocidad de base: 1.99 GHz Sockets: 1 Nodos: 4 Procesadores lógicos: 8 Vibraciones: Habilitado Tiempo activo: 259 s Cache L1: 13.0 MB Cache L2: 8.0 MB Tiempos actuales: 8:06:19:37</p>	<p>Memoria</p> <p>Uso de memoria: 8.4 GB (502 MB) 7.3 GB Velocidad: 2400 MHz Capacidad: 11.5/18.3 GB 5.3 GB Reservado para hardware: 720 MB Reservado para el sistema: 1014 MB</p>

CASO	PROCESADOR (CPU)	MEMORIA (RAM)
A.I (double) ($x = \text{A}m1$ $x \cdot b$)	<p>CPU</p> <p>Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz</p> <p>Velocidad: 100% 2.63 GHz Velocidad de base: 1.99 GHz Sobrecarga: 4 Nucleos: 4 Procesadores lógicos: 8 Habilitado: 1 Cores L1: 256 kB Cores L2: 12.9 MB Cores L3: 8.0 MB</p> <p>Tiempo actual: 8:07:00:53</p> <p>Memoria</p> <p>16.0 GB</p> <p>Velocidad: 100% 2.63 GHz Velocidad de base: 1.99 GHz Sobrecarga: 4 Nucleos: 4 Procesadores lógicos: 8 Habilitado: 1 Cores L1: 256 kB Cores L2: 12.9 MB Cores L3: 8.0 MB</p> <p>En uso (comprimido): 9.1 GB (548 MB) Disponible: 6.7 GB Velocidad: 2400 MHz Ranuras: 4 x 16GB Factor de forma: SO-DIMM Reservada para hardware: 117 MB</p> <p>13.0/18.3 GB 5.5 GB Bloque paginado: Bloque no paginado 724 MB 1020 MB</p>	<p>Memoria</p> <p>16.0 GB</p> <p>Velocidad: 100% 2.63 GHz Velocidad de base: 1.99 GHz Sobrecarga: 4 Nucleos: 4 Procesadores lógicos: 8 Habilitado: 1 Cores L1: 256 kB Cores L2: 12.9 MB Cores L3: 8.0 MB</p> <p>En uso (comprimido): 9.1 GB (548 MB) Disponible: 6.7 GB Velocidad: 2400 MHz Ranuras: 4 x 16GB Factor de forma: SO-DIMM Reservada para hardware: 117 MB</p> <p>13.0/18.3 GB 5.5 GB Bloque paginado: Bloque no paginado 724 MB 1020 MB</p>
A.II (double) (<code>scipy.linalg.solve</code> by default)	<p>CPU</p> <p>Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz</p> <p>Velocidad: 100% 2.71 GHz Velocidad de base: 1.99 GHz Sobrecarga: 4 Nucleos: 4 Procesadores lógicos: 8 Habilitado: 1 Cores L1: 256 kB Cores L2: 12.9 MB Cores L3: 8.0 MB</p> <p>Tiempo actual: 8:07:14:03</p> <p>Memoria</p> <p>16.0 GB</p> <p>Velocidad: 100% 2.71 GHz Velocidad de base: 1.99 GHz Sobrecarga: 4 Nucleos: 4 Procesadores lógicos: 8 Habilitado: 1 Cores L1: 256 kB Cores L2: 12.9 MB Cores L3: 8.0 MB</p> <p>En uso (comprimido): 9.0 GB (607 MB) Disponible: 5.6 GB Velocidad: 2400 MHz Ranuras: 4 x 16GB Factor de forma: SO-DIMM Reservada para hardware: 117 MB</p> <p>13.2/18.3 GB 5.6 GB Bloque paginado: Bloque no paginado 725 MB 1020 MB</p>	<p>Memoria</p> <p>16.0 GB</p> <p>Velocidad: 100% 2.71 GHz Velocidad de base: 1.99 GHz Sobrecarga: 4 Nucleos: 4 Procesadores lógicos: 8 Habilitado: 1 Cores L1: 256 kB Cores L2: 12.9 MB Cores L3: 8.0 MB</p> <p>En uso (comprimido): 9.0 GB (607 MB) Disponible: 5.6 GB Velocidad: 2400 MHz Ranuras: 4 x 16GB Factor de forma: SO-DIMM Reservada para hardware: 117 MB</p> <p>13.2/18.3 GB 5.6 GB Bloque paginado: Bloque no paginado 725 MB 1020 MB</p>
A.III (double) (using <code>assume_a='pos'</code>)	<p>CPU</p> <p>Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz</p> <p>Velocidad: 82% 2.87 GHz Velocidad de base: 1.99 GHz Sobrecarga: 4 Nucleos: 4 Procesadores lógicos: 8 Habilitado: 1 Cores L1: 256 kB Cores L2: 12.9 MB Cores L3: 8.0 MB</p> <p>Tiempo actual: 8:07:18:28</p> <p>Memoria</p> <p>16.0 GB</p> <p>Velocidad: 82% 2.87 GHz Velocidad de base: 1.99 GHz Sobrecarga: 4 Nucleos: 4 Procesadores lógicos: 8 Habilitado: 1 Cores L1: 256 kB Cores L2: 12.9 MB Cores L3: 8.0 MB</p> <p>En uso (comprimido): 7.2 GB (492 MB) Disponible: 8.5 GB Velocidad: 2400 MHz Ranuras: 4 x 16GB Factor de forma: SO-DIMM Reservada para hardware: 117 MB</p> <p>11.1/18.3 GB 5.4 GB Bloque paginado: Bloque no paginado 724 MB 1017 MB</p>	<p>Memoria</p> <p>16.0 GB</p> <p>Velocidad: 82% 2.87 GHz Velocidad de base: 1.99 GHz Sobrecarga: 4 Nucleos: 4 Procesadores lógicos: 8 Habilitado: 1 Cores L1: 256 kB Cores L2: 12.9 MB Cores L3: 8.0 MB</p> <p>En uso (comprimido): 7.2 GB (492 MB) Disponible: 8.5 GB Velocidad: 2400 MHz Ranuras: 4 x 16GB Factor de forma: SO-DIMM Reservada para hardware: 117 MB</p> <p>11.1/18.3 GB 5.4 GB Bloque paginado: Bloque no paginado 724 MB 1017 MB</p>
A.IV (double) (using <code>assume_a='sym'</code>)	<p>CPU</p> <p>Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz</p> <p>Velocidad: 80% 2.77 GHz Velocidad de base: 1.99 GHz Sobrecarga: 4 Nucleos: 4 Procesadores lógicos: 8 Habilitado: 1 Cores L1: 256 kB Cores L2: 12.9 MB Cores L3: 8.0 MB</p> <p>Tiempo actual: 8:07:22:27</p> <p>Memoria</p> <p>16.0 GB</p> <p>Velocidad: 80% 2.77 GHz Velocidad de base: 1.99 GHz Sobrecarga: 4 Nucleos: 4 Procesadores lógicos: 8 Habilitado: 1 Cores L1: 256 kB Cores L2: 12.9 MB Cores L3: 8.0 MB</p> <p>En uso (comprimido): 8.4 GB (611 MB) Disponible: 7.3 GB Velocidad: 2400 MHz Ranuras: 4 x 16GB Factor de forma: SO-DIMM Reservada para hardware: 117 MB</p> <p>12.4/18.3 GB 5.4 GB Bloque paginado: Bloque no paginado 724 MB 1017 MB</p>	<p>Memoria</p> <p>16.0 GB</p> <p>Velocidad: 80% 2.77 GHz Velocidad de base: 1.99 GHz Sobrecarga: 4 Nucleos: 4 Procesadores lógicos: 8 Habilitado: 1 Cores L1: 256 kB Cores L2: 12.9 MB Cores L3: 8.0 MB</p> <p>En uso (comprimido): 8.4 GB (611 MB) Disponible: 7.3 GB Velocidad: 2400 MHz Ranuras: 4 x 16GB Factor de forma: SO-DIMM Reservada para hardware: 117 MB</p> <p>12.4/18.3 GB 5.4 GB Bloque paginado: Bloque no paginado 724 MB 1017 MB</p>
A.V (double) (using <code>overwrite_a=True</code>)	<p>CPU</p> <p>Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz</p> <p>Velocidad: 65% 3.17 GHz Velocidad de base: 1.99 GHz Sobrecarga: 4 Nucleos: 4 Procesadores lógicos: 8 Habilitado: 1 Cores L1: 256 kB Cores L2: 12.9 MB Cores L3: 8.0 MB</p> <p>Tiempo actual: 8:07:30:00</p> <p>Memoria</p> <p>16.0 GB</p> <p>Velocidad: 65% 3.17 GHz Velocidad de base: 1.99 GHz Sobrecarga: 4 Nucleos: 4 Procesadores lógicos: 8 Habilitado: 1 Cores L1: 256 kB Cores L2: 12.9 MB Cores L3: 8.0 MB</p> <p>En uso (comprimido): 7.3 GB (510 MB) Disponible: 8.5 GB Velocidad: 2400 MHz Ranuras: 4 x 16GB Factor de forma: SO-DIMM Reservada para hardware: 117 MB</p> <p>11.2/18.3 GB 5.4 GB Bloque paginado: Bloque no paginado 724 MB 1018 MB</p>	<p>Memoria</p> <p>16.0 GB</p> <p>Velocidad: 65% 3.17 GHz Velocidad de base: 1.99 GHz Sobrecarga: 4 Nucleos: 4 Procesadores lógicos: 8 Habilitado: 1 Cores L1: 256 kB Cores L2: 12.9 MB Cores L3: 8.0 MB</p> <p>En uso (comprimido): 7.3 GB (510 MB) Disponible: 8.5 GB Velocidad: 2400 MHz Ranuras: 4 x 16GB Factor de forma: SO-DIMM Reservada para hardware: 117 MB</p> <p>11.2/18.3 GB 5.4 GB Bloque paginado: Bloque no paginado 724 MB 1018 MB</p>
A.VI (double) (using <code>overwrite_b=True</code>)	<p>CPU</p> <p>Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz</p> <p>Velocidad: 31% 3.34 GHz Velocidad de base: 1.99 GHz Sobrecarga: 4 Nucleos: 4 Procesadores lógicos: 8 Habilitado: 1 Cores L1: 256 kB Cores L2: 12.9 MB Cores L3: 8.0 MB</p> <p>Tiempo actual: 8:07:32:26</p> <p>Memoria</p> <p>16.0 GB</p> <p>Velocidad: 31% 3.34 GHz Velocidad de base: 1.99 GHz Sobrecarga: 4 Nucleos: 4 Procesadores lógicos: 8 Habilitado: 1 Cores L1: 256 kB Cores L2: 12.9 MB Cores L3: 8.0 MB</p> <p>En uso (comprimido): 7.7 GB (505 MB) Disponible: 8.1 GB Velocidad: 2400 MHz Ranuras: 4 x 16GB Factor de forma: SO-DIMM Reservada para hardware: 117 MB</p> <p>11.3/18.3 GB 5.4 GB Bloque paginado: Bloque no paginado 723 MB 1019 MB</p>	<p>Memoria</p> <p>16.0 GB</p> <p>Velocidad: 31% 3.34 GHz Velocidad de base: 1.99 GHz Sobrecarga: 4 Nucleos: 4 Procesadores lógicos: 8 Habilitado: 1 Cores L1: 256 kB Cores L2: 12.9 MB Cores L3: 8.0 MB</p> <p>En uso (comprimido): 7.7 GB (505 MB) Disponible: 8.1 GB Velocidad: 2400 MHz Ranuras: 4 x 16GB Factor de forma: SO-DIMM Reservada para hardware: 117 MB</p> <p>11.3/18.3 GB 5.4 GB Bloque paginado: Bloque no paginado 723 MB 1019 MB</p>

CASO	PROCESADOR (CPU)	MEMORIA (RAM)
A.VII (double) (using overwrite_a=True and overwrite_b=True)	<p>CPU</p> <p>Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz</p> <p>94% 2.53 GHz Utilidad</p> <p>Velocidad de base: 1.99 GHz</p> <p>Procesador: Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz</p> <p>Núcleos: 4</p> <p>Operaciones lógicas: 4</p> <p>Habilitado: Sí</p> <p>Ventilación: 254 MB</p> <p>Cooling: 11.98</p> <p>Cache L1: 12.0 MB</p> <p>Cache L2: 6.0 MB</p> <p>Tiempo activo: 8:07:36:59</p> <p>Memoria</p> <p>16.0 GB</p> <p>8.0 GB Disponible</p> <p>Velocidad: 2400 MHz</p> <p>Factor de forma: 2x4</p> <p>Reservada para hardware: 117 MB</p> <p>En uso (comprimido): 7.7 GB (490 MB)</p> <p>Continuado: 11.9/18.3 GB 5.4 GB</p> <p>Bloque paginado: 725 MB</p> <p>Bloque no paginado: 1019 MB</p>	<p>Memoria</p> <p>16.0 GB</p> <p>8.0 GB Disponible</p> <p>Velocidad: 2400 MHz</p> <p>Factor de forma: 2x4</p> <p>Reservada para hardware: 117 MB</p> <p>En uso (comprimido): 7.7 GB (490 MB)</p> <p>Continuado: 11.9/18.3 GB 5.4 GB</p> <p>Bloque paginado: 725 MB</p> <p>Bloque no paginado: 1019 MB</p>
B.I (float) (scipy.linalg.eigh by default)	<p>CPU</p> <p>87% 2.57 GHz Utilidad</p> <p>Velocidad de base: 1.99 GHz</p> <p>Procesador: Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz</p> <p>Núcleos: 4</p> <p>Operaciones lógicas: 4</p> <p>Habilitado: Sí</p> <p>Ventilación: 254 MB</p> <p>Cooling: 11.98</p> <p>Cache L1: 12.0 MB</p> <p>Cache L2: 6.0 MB</p> <p>Tiempo activo: 8:13:26:33</p> <p>Memoria</p> <p>16.0 GB</p> <p>7.7 GB Disponible</p> <p>Velocidad: 2400 MHz</p> <p>Factor de forma: 2x4</p> <p>Reservada para hardware: 117 MB</p> <p>En uso (comprimido): 6.0 GB (472 MB)</p> <p>Continuado: 12.0/18.3 GB 6.0 GB</p> <p>Bloque paginado: 913 MB</p> <p>Bloque no paginado: 1.1 GB</p>	<p>Memoria</p> <p>16.0 GB</p> <p>7.7 GB Disponible</p> <p>Velocidad: 2400 MHz</p> <p>Factor de forma: 2x4</p> <p>Reservada para hardware: 117 MB</p> <p>En uso (comprimido): 6.0 GB (472 MB)</p> <p>Continuado: 12.0/18.3 GB 6.0 GB</p> <p>Bloque paginado: 913 MB</p> <p>Bloque no paginado: 1.1 GB</p>
B.II.1 (float) (driver='ev' and overwrite_a=False)	<p>CPU</p> <p>34% 3.71 GHz Utilidad</p> <p>Velocidad de base: 1.99 GHz</p> <p>Procesador: Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz</p> <p>Núcleos: 4</p> <p>Operaciones lógicas: 4</p> <p>Habilitado: Sí</p> <p>Ventilación: 254 MB</p> <p>Cooling: 11.98</p> <p>Cache L1: 12.0 MB</p> <p>Cache L2: 6.0 MB</p> <p>Tiempo activo: 8:13:37:49</p> <p>Memoria</p> <p>16.0 GB</p> <p>7.9 GB Disponible</p> <p>Velocidad: 2400 MHz</p> <p>Factor de forma: 2x4</p> <p>Reservada para hardware: 117 MB</p> <p>En uso (comprimido): 7.6 GB (477 MB)</p> <p>Continuado: 12.2/18.3 GB 6.4 GB</p> <p>Bloque paginado: 919 MB</p> <p>Bloque no paginado: 1.1 GB</p>	<p>Memoria</p> <p>16.0 GB</p> <p>7.9 GB Disponible</p> <p>Velocidad: 2400 MHz</p> <p>Factor de forma: 2x4</p> <p>Reservada para hardware: 117 MB</p> <p>En uso (comprimido): 7.6 GB (477 MB)</p> <p>Continuado: 12.2/18.3 GB 6.4 GB</p> <p>Bloque paginado: 919 MB</p> <p>Bloque no paginado: 1.1 GB</p>
B.II.2 (float) (driver='ev' and overwrite_a=True)	<p>CPU</p> <p>42% 3.56 GHz Utilidad</p> <p>Velocidad de base: 1.99 GHz</p> <p>Procesador: Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz</p> <p>Núcleos: 4</p> <p>Operaciones lógicas: 4</p> <p>Habilitado: Sí</p> <p>Ventilación: 254 MB</p> <p>Cooling: 11.98</p> <p>Cache L1: 12.0 MB</p> <p>Cache L2: 6.0 MB</p> <p>Tiempo activo: 8:13:58:08</p> <p>Memoria</p> <p>16.0 GB</p> <p>6.5 GB Disponible</p> <p>Velocidad: 2400 MHz</p> <p>Factor de forma: 2x4</p> <p>Reservada para hardware: 117 MB</p> <p>En uso (comprimido): 7.2 GB (469 MB)</p> <p>Continuado: 11.3/18.3 GB 6.5 GB</p> <p>Bloque paginado: 918 MB</p> <p>Bloque no paginado: 1.1 GB</p>	<p>Memoria</p> <p>16.0 GB</p> <p>6.5 GB Disponible</p> <p>Velocidad: 2400 MHz</p> <p>Factor de forma: 2x4</p> <p>Reservada para hardware: 117 MB</p> <p>En uso (comprimido): 7.2 GB (469 MB)</p> <p>Continuado: 11.3/18.3 GB 6.5 GB</p> <p>Bloque paginado: 918 MB</p> <p>Bloque no paginado: 1.1 GB</p>
B.III.1 (float) (driver='evd' and overwrite_a=False)	<p>CPU</p> <p>100% 2.73 GHz Utilidad</p> <p>Velocidad de base: 1.99 GHz</p> <p>Procesador: Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz</p> <p>Núcleos: 4</p> <p>Operaciones lógicas: 4</p> <p>Habilitado: Sí</p> <p>Ventilación: 254 MB</p> <p>Cooling: 11.98</p> <p>Cache L1: 12.0 MB</p> <p>Cache L2: 6.0 MB</p> <p>Tiempo activo: 8:14:22:05</p> <p>Memoria</p> <p>16.0 GB</p> <p>9.1 GB Disponible</p> <p>Velocidad: 2400 MHz</p> <p>Factor de forma: 2x4</p> <p>Reservada para hardware: 117 MB</p> <p>En uso (comprimido): 6.8 GB (325 MB)</p> <p>Continuado: 11.2/18.3 GB 6.8 GB</p> <p>Bloque paginado: 918 MB</p> <p>Bloque no paginado: 1.1 GB</p>	<p>Memoria</p> <p>16.0 GB</p> <p>9.1 GB Disponible</p> <p>Velocidad: 2400 MHz</p> <p>Factor de forma: 2x4</p> <p>Reservada para hardware: 117 MB</p> <p>En uso (comprimido): 6.8 GB (325 MB)</p> <p>Continuado: 11.2/18.3 GB 6.8 GB</p> <p>Bloque paginado: 918 MB</p> <p>Bloque no paginado: 1.1 GB</p>
B.III.2 (float) (driver='evd' and overwrite_a=True)	<p>CPU</p> <p>92% 2.68 GHz Utilidad</p> <p>Velocidad de base: 1.99 GHz</p> <p>Procesador: Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz</p> <p>Núcleos: 4</p> <p>Operaciones lógicas: 4</p> <p>Habilitado: Sí</p> <p>Ventilación: 254 MB</p> <p>Cooling: 11.98</p> <p>Cache L1: 12.0 MB</p> <p>Cache L2: 6.0 MB</p> <p>Tiempo activo: 8:14:25:02</p> <p>Memoria</p> <p>16.0 GB</p> <p>8.9 GB Disponible</p> <p>Velocidad: 2400 MHz</p> <p>Factor de forma: 2x4</p> <p>Reservada para hardware: 117 MB</p> <p>En uso (comprimido): 7.0 GB (353 MB)</p> <p>Continuado: 11.5/18.3 GB 6.8 GB</p> <p>Bloque paginado: 919 MB</p> <p>Bloque no paginado: 1.1 GB</p>	<p>Memoria</p> <p>16.0 GB</p> <p>8.9 GB Disponible</p> <p>Velocidad: 2400 MHz</p> <p>Factor de forma: 2x4</p> <p>Reservada para hardware: 117 MB</p> <p>En uso (comprimido): 7.0 GB (353 MB)</p> <p>Continuado: 11.5/18.3 GB 6.8 GB</p> <p>Bloque paginado: 919 MB</p> <p>Bloque no paginado: 1.1 GB</p>

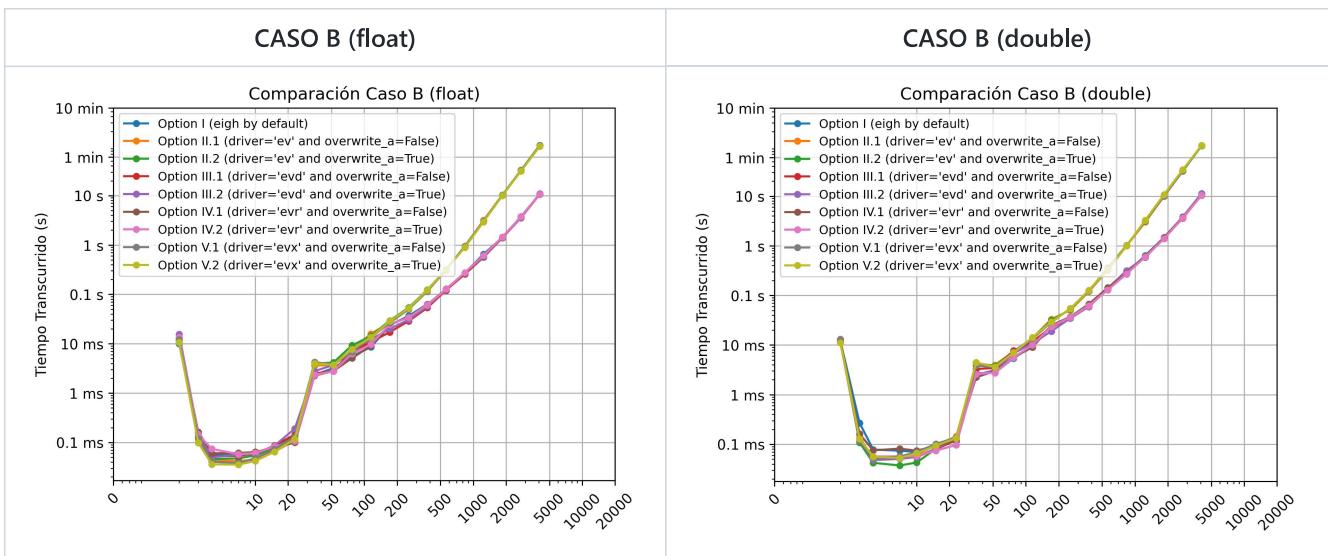
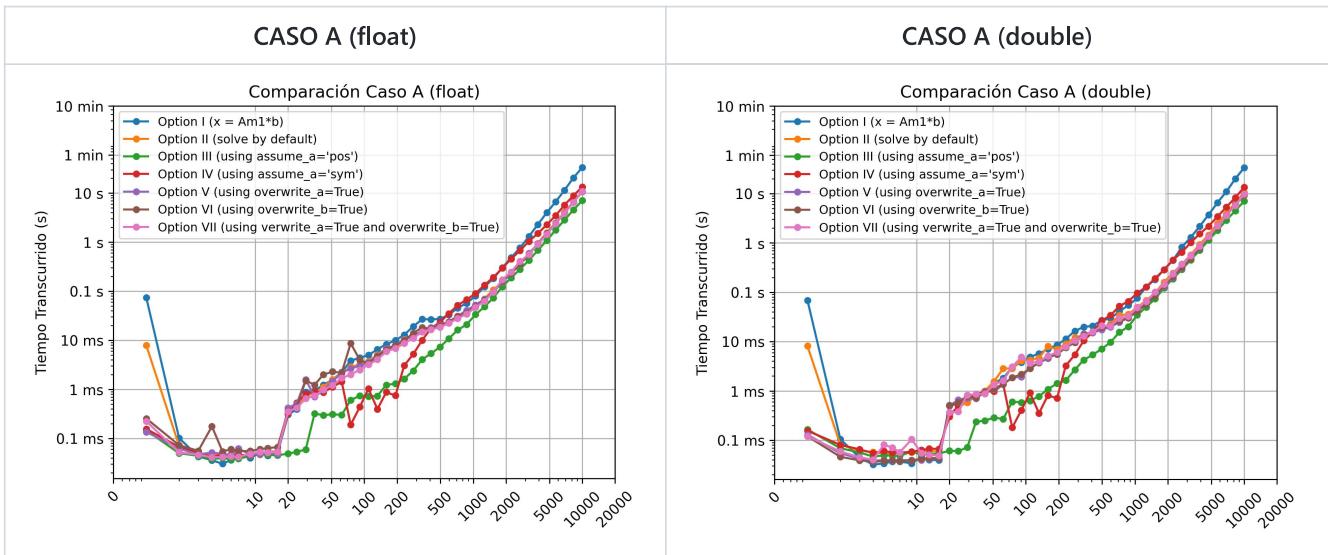
CASO	PROCESADOR (CPU)	MEMORIA (RAM)
B.VI.1 (float) (driver='evr' and overwrite_a=False)	<p>CPU</p> <p>Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz</p> <p>99% 2.69 GHz Utilidad</p> <p>Procesos Subprocesos Identificación</p> <p>265 3509 135224</p> <p>Tiempo activo: 8:14:27.22</p> <p>Velocidad de base: 1.99 GHz</p> <p>Sobres. 1 Nucleos 4</p> <p>Procesadores lógicos 4</p> <p>Virtuales 4</p> <p>Habilitado</p> <p>Cache L1: 256 MB</p> <p>Cache L2: 11.9 MB</p> <p>Cache L3: 8.0 MB</p> <p>Memoria</p> <p>16.0 GB Util de memoria</p> <p>Al registrado: 6.9 GB (635 MB) Disponible: 8.9 GB Velocidad: 2400 MHz Factor de forma: 2 de 4 Reservada para hardware: 117 MB</p> <p>Continuado: 11.2/18.3 GB 6.8 GB Bloque paginado: 920 MB Bloque no paginado: 1.1 GB</p>	<p>Memoria</p> <p>16.0 GB Util de memoria</p> <p>Al registrado: 6.9 GB (635 MB) Disponible: 8.9 GB Velocidad: 2400 MHz Factor de forma: 2 de 4 Reservada para hardware: 117 MB</p> <p>Continuado: 11.2/18.3 GB 6.8 GB Bloque paginado: 920 MB Bloque no paginado: 1.1 GB</p>
B.VI.2 (float) (driver='evr' and overwrite_a=True)	<p>CPU</p> <p>Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz</p> <p>100% 2.65 GHz Utilidad</p> <p>Procesos Subprocesos Identificación</p> <p>264 3482 134249</p> <p>Tiempo activo: 8:14:30.02</p> <p>Velocidad de base: 1.99 GHz</p> <p>Sobres. 1 Nucleos 4</p> <p>Procesadores lógicos 4</p> <p>Virtuales 4</p> <p>Habilitado</p> <p>Cache L1: 254 MB</p> <p>Cache L2: 10.9 MB</p> <p>Cache L3: 8.0 MB</p> <p>Memoria</p> <p>16.0 GB Util de memoria</p> <p>Al registrado: 7.1 GB (522 MB) Disponible: 8.7 GB Velocidad: 2400 MHz Factor de forma: 2 de 4 Reservada para hardware: 117 MB</p> <p>Continuado: 11.4/18.3 GB 6.8 GB Bloque paginado: 920 MB Bloque no paginado: 1.1 GB</p>	<p>Memoria</p> <p>16.0 GB Util de memoria</p> <p>Al registrado: 7.1 GB (522 MB) Disponible: 8.7 GB Velocidad: 2400 MHz Factor de forma: 2 de 4 Reservada para hardware: 117 MB</p> <p>Continuado: 11.4/18.3 GB 6.8 GB Bloque paginado: 920 MB Bloque no paginado: 1.1 GB</p>
B.V.1 (float) (driver='evx' and overwrite_a=False)	<p>CPU</p> <p>Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz</p> <p>99% 3.57 GHz Utilidad</p> <p>Procesos Subprocesos Identificación</p> <p>270 3571 136432</p> <p>Tiempo activo: 8:14:36.00</p> <p>Velocidad de base: 1.99 GHz</p> <p>Sobres. 1 Nucleos 4</p> <p>Procesadores lógicos 4</p> <p>Virtuales 4</p> <p>Habilitado</p> <p>Cache L1: 254 MB</p> <p>Cache L2: 12.9 MB</p> <p>Cache L3: 8.0 MB</p> <p>Memoria</p> <p>16.0 GB Util de memoria</p> <p>Al registrado: 7.2 GB (522 MB) Disponible: 8.6 GB Velocidad: 2400 MHz Factor de forma: 2 de 4 Reservada para hardware: 117 MB</p> <p>Continuado: 11.5/18.3 GB 6.7 GB Bloque paginado: 920 MB Bloque no paginado: 1.1 GB</p>	<p>Memoria</p> <p>16.0 GB Util de memoria</p> <p>Al registrado: 7.2 GB (522 MB) Disponible: 8.6 GB Velocidad: 2400 MHz Factor de forma: 2 de 4 Reservada para hardware: 117 MB</p> <p>Continuado: 11.5/18.3 GB 6.7 GB Bloque paginado: 920 MB Bloque no paginado: 1.1 GB</p>
B.V.2 (float) (driver='evx' and overwrite_a=True)	<p>CPU</p> <p>Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz</p> <p>99% 3.47 GHz Utilidad</p> <p>Procesos Subprocesos Identificación</p> <p>261 3393 132311</p> <p>Tiempo activo: 8:14:59.51</p> <p>Velocidad de base: 1.99 GHz</p> <p>Sobres. 1 Nucleos 4</p> <p>Procesadores lógicos 4</p> <p>Virtuales 4</p> <p>Habilitado</p> <p>Cache L1: 254 MB</p> <p>Cache L2: 10.9 MB</p> <p>Cache L3: 8.0 MB</p> <p>Memoria</p> <p>16.0 GB Util de memoria</p> <p>Al registrado: 7.0 GB (505 MB) Disponible: 8.8 GB Velocidad: 2400 MHz Factor de forma: 2 de 4 Reservada para hardware: 117 MB</p> <p>Continuado: 11.6/18.3 GB 6.7 GB Bloque paginado: 921 MB Bloque no paginado: 1.1 GB</p>	<p>Memoria</p> <p>16.0 GB Util de memoria</p> <p>Al registrado: 7.0 GB (505 MB) Disponible: 8.8 GB Velocidad: 2400 MHz Factor de forma: 2 de 4 Reservada para hardware: 117 MB</p> <p>Continuado: 11.6/18.3 GB 6.7 GB Bloque paginado: 921 MB Bloque no paginado: 1.1 GB</p>
B.I (double) (scipy.linalg.eigh by default)	<p>CPU</p> <p>Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz</p> <p>99% 2.85 GHz Utilidad</p> <p>Procesos Subprocesos Identificación</p> <p>269 4351 131044</p> <p>Tiempo activo: 8:21:19.58</p> <p>Velocidad de base: 1.99 GHz</p> <p>Sobres. 1 Nucleos 4</p> <p>Procesadores lógicos 4</p> <p>Virtuales 4</p> <p>Habilitado</p> <p>Cache L1: 254 MB</p> <p>Cache L2: 11.9 MB</p> <p>Cache L3: 8.0 MB</p> <p>Memoria</p> <p>16.0 GB Util de memoria</p> <p>Al registrado: 7.9 GB (104 MB) Disponible: 7.7 GB Velocidad: 2400 MHz Factor de forma: 2 de 4 Reservada para hardware: 117 MB</p> <p>Continuado: 9.8/18.3 GB 3.2 GB Bloque paginado: 677 MB Bloque no paginado: 1016 MB</p>	<p>Memoria</p> <p>16.0 GB Util de memoria</p> <p>Al registrado: 7.9 GB (104 MB) Disponible: 7.7 GB Velocidad: 2400 MHz Factor de forma: 2 de 4 Reservada para hardware: 117 MB</p> <p>Continuado: 9.8/18.3 GB 3.2 GB Bloque paginado: 677 MB Bloque no paginado: 1016 MB</p>
B.II.1 (double) (driver='ev' and overwrite_a=False)	<p>CPU</p> <p>Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz</p> <p>99% 3.54 GHz Utilidad</p> <p>Procesos Subprocesos Identificación</p> <p>264 4205 129741</p> <p>Tiempo activo: 8:21:23.31</p> <p>Velocidad de base: 1.99 GHz</p> <p>Sobres. 1 Nucleos 4</p> <p>Procesadores lógicos 4</p> <p>Virtuales 4</p> <p>Habilitado</p> <p>Cache L1: 254 MB</p> <p>Cache L2: 12.9 MB</p> <p>Cache L3: 8.0 MB</p> <p>Memoria</p> <p>16.0 GB Util de memoria</p> <p>Al registrado: 7.5 GB (104 MB) Disponible: 8.0 GB Velocidad: 2400 MHz Factor de forma: 2 de 4 Reservada para hardware: 117 MB</p> <p>Continuado: 9.5/18.3 GB 3.4 GB Bloque paginado: 678 MB Bloque no paginado: 1019 MB</p>	<p>Memoria</p> <p>16.0 GB Util de memoria</p> <p>Al registrado: 7.5 GB (104 MB) Disponible: 8.0 GB Velocidad: 2400 MHz Factor de forma: 2 de 4 Reservada para hardware: 117 MB</p> <p>Continuado: 9.5/18.3 GB 3.4 GB Bloque paginado: 678 MB Bloque no paginado: 1019 MB</p>

CASO	PROCESADOR (CPU)	MEMORIA (RAM)
B.II.2 (double) (driver='ev' and overwrite_a=True)	<p>CPU</p> <p>Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz</p> <p>Usa: 42% Velocidad: 3.12 GHz Submisiones: 258 Identificaciones: 2967 Tiempo en el: 82:49:23</p> <p>Velocidad de base: 1.98 GHz Socorro: 1 Núcleos: 4 Virtualización: Habilitado Procesamiento lógico: 8 Hiperthread: Sí Cache L1: 256 kB Cache L2: 1.0 MB Cache L3: 6.0 MB</p>	<p>Memoria</p> <p>16.0 GB</p> <p>0 segundos Compresión de memoria</p> <p>En uso (comprimido): 7.3 GB (178 MB) Disponible (no paginado): 8.4 GB Velocidad: 240 MHz Factor de formateo: 2000MM Reservada para hardware: 117 MB</p> <p>9.4/18.3 GB 3.8 GB</p> <p>Bloque asignado: Bloque no paginado Bloque no paginado: 679 MB Factor de formateo: 1.0 GB</p>
B.III.1 (double) (driver='evd' and overwrite_a=False)	<p>CPU</p> <p>Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz</p> <p>Usa: 100% Velocidad: 2.67 GHz Submisiones: 256 Identificaciones: 3055 Tiempo en el: 82:12:52</p> <p>Velocidad de base: 1.98 GHz Socorro: 1 Núcleos: 4 Virtualización: Habilitado Procesamiento lógico: 8 Hiperthread: Sí Cache L1: 256 kB Cache L2: 1.0 MB Cache L3: 6.0 MB</p>	<p>Memoria</p> <p>16.0 GB</p> <p>0 segundos Compresión de memoria</p> <p>En uso (comprimido): 7.4 GB (161 MB) Disponible (no paginado): 8.4 GB Velocidad: 240 MHz Factor de formateo: 2000MM Reservada para hardware: 117 MB</p> <p>9.9/18.3 GB 4.0 GB</p> <p>Bloque asignado: Bloque no paginado Bloque no paginado: 685 MB Factor de formateo: 1.0 GB</p>
B.III.2 (double) (driver='evd' and overwrite_a=True)	<p>CPU</p> <p>Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz</p> <p>Usa: 100% Velocidad: 2.71 GHz Submisiones: 257 Identificaciones: 2908 Tiempo en el: 82:15:20</p> <p>Velocidad de base: 1.99 GHz Socorro: 1 Núcleos: 4 Virtualización: Habilitado Procesamiento lógico: 8 Hiperthread: Sí Cache L1: 256 kB Cache L2: 1.0 MB Cache L3: 6.0 MB</p>	<p>Memoria</p> <p>16.0 GB</p> <p>0 segundos Compresión de memoria</p> <p>En uso (comprimido): 7.4 GB (161 MB) Disponible (no paginado): 8.4 GB Velocidad: 240 MHz Factor de formateo: 2000MM Reservada para hardware: 117 MB</p> <p>9.8/18.3 GB 4.0 GB</p> <p>Bloque asignado: Bloque no paginado Bloque no paginado: 690 MB Factor de formateo: 1.0 GB</p>
B.IV.1 (double) (driver='evr' and overwrite_a=False)	<p>CPU</p> <p>Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz</p> <p>Usa: 100% Velocidad: 2.64 GHz Submisiones: 257 Identificaciones: 2913 Tiempo en el: 82:18:21</p> <p>Velocidad de base: 1.99 GHz Socorro: 1 Núcleos: 4 Virtualización: Habilitado Procesamiento lógico: 8 Hiperthread: Sí Cache L1: 256 kB Cache L2: 1.0 MB Cache L3: 6.0 MB</p>	<p>Memoria</p> <p>16.0 GB</p> <p>0 segundos Compresión de memoria</p> <p>En uso (comprimido): 7.4 GB (158 MB) Disponible (no paginado): 8.3 GB Velocidad: 240 MHz Factor de formateo: 2000MM Reservada para hardware: 117 MB</p> <p>9.5/18.3 GB 4.0 GB</p> <p>Bloque asignado: Bloque no paginado Bloque no paginado: 690 MB Factor de formateo: 1.0 GB</p>
B.IV.2 (double) (driver='evr' and overwrite_a=True)	<p>CPU</p> <p>Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz</p> <p>Usa: 100% Velocidad: 2.84 GHz Submisiones: 253 Identificaciones: 2851 Tiempo en el: 82:21:12</p> <p>Velocidad de base: 1.98 GHz Socorro: 1 Núcleos: 4 Virtualización: Habilitado Procesamiento lógico: 8 Hiperthread: Sí Cache L1: 256 kB Cache L2: 1.0 MB Cache L3: 6.0 MB</p>	<p>Memoria</p> <p>16.0 GB</p> <p>0 segundos Compresión de memoria</p> <p>En uso (comprimido): 7.4 GB (157 MB) Disponible (no paginado): 8.2 GB Velocidad: 240 MHz Factor de formateo: 2000MM Reservada para hardware: 117 MB</p> <p>9.6/18.3 GB 4.1 GB</p> <p>Bloque asignado: Bloque no paginado Bloque no paginado: 688 MB Factor de formateo: 1.0 GB</p>
B.V.1 (double) (driver='evx' and overwrite_a=False)	<p>CPU</p> <p>Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz</p> <p>Usa: 42% Velocidad: 3.43 GHz Submisiones: 248 Identificaciones: 2850 Tiempo en el: 82:22:56</p> <p>Velocidad de base: 1.98 GHz Socorro: 1 Núcleos: 4 Virtualización: Habilitado Procesamiento lógico: 8 Hiperthread: Sí Cache L1: 256 kB Cache L2: 1.0 MB Cache L3: 6.0 MB</p>	<p>Memoria</p> <p>16.0 GB</p> <p>0 segundos Compresión de memoria</p> <p>En uso (comprimido): 6.9 GB (154 MB) Disponible (no paginado): 8.7 GB Velocidad: 240 MHz Factor de formateo: 2000MM Reservada para hardware: 117 MB</p> <p>9.1/18.3 GB 4.8 GB</p> <p>Bloque asignado: Bloque no paginado Bloque no paginado: 689 MB Factor de formateo: 1.0 GB</p>

CASO	PROCESADOR (CPU)	MEMORIA (RAM)
B.V.2 (double) (driver='evx' and overwrite_a=True)	 <p>44% 3.53 GHz 258 3029 124666 8:22:5327</p>	 <p>6.9 GB (171 MB) 2400 MHz 9.6/18.3 GB 4.3 GB 500MHz 697 MB 0.08</p>

Se puede evidenciar que en general en las corridas el procesador alcanza su máximo en los puntos con mayores valores de N, y es en esos casos que la memoria ram tiende a subir un poco, para ayudar a procesar la información. Otro aspecto interesante a señalar, que para el caso B opción II y V el procesador no usaba en promedio ni el 50% y la memoria nunca se vio afectada, lo curioso es que estos casos fueron los que más tiempo demoraron en ejecutarse (esto explica mejor su tiempo de duración ya que al utilizar menos recursos su tiempo de realización es más lento, quizás esto puede deberse a que la operación no tenía una dificultad tan grande pero si un proceso extenso).

A continuación se muestran los gráficos encontrados:



De los gráficos para el caso A, se puede apreciar que son muy similares entre ellos, casi no hay diferencias, y si las hay no tienen un claro patrón hasta el final (es decir hasta los valores más grandes de N), queda claro que la opción de calcular la inversa y luego multiplicar esta por la matriz b es la manera más lenta de realizar la operación, mientras que la más rápida es utilizando `scipy.linalg.solve` asumiendo una matriz A positiva (`assume_a='pos'`). Es importante señalar que los resultados entre el caso A usando datos de tipo float vs usando datos de tipo double son prácticamente idénticos.

Por otro lado, observando lo ocurrido en el caso B, se puede apreciar que si existen mayores diferencias entre las diferentes opciones, en donde las opciones II.1, II.2, V.1 y V.2 (`driver='ev` y `driver=evx`) son evidentemente más lentas, mientras que todas las demás tienen tiempos prácticamente iguales obteniendo un mejor rendimiento. Al igual que el caso anterior, la diferencia entre el tipo de dato double y float es prácticamente imperceptible.

Preguntas

- Haga un comentario completo respecto de todo lo que ve en términos de desempeño en cada problema.

Tal como se menciona antes se puede apreciar como en el caso A son todos los casos muy parecidos, siendo el método de utilizar la inversa el más lento, mientras que el más rápido es asumiendo que la matriz A es positiva.

Para el caso B, usar `driver='ev` y `driver=evx` relentizan considerablemente el proceso, mientras que los demás trabajan de manera más eficiente.

(Más comentarios debajo de los gráficos...).

- ¿Como es la variabilidad del tiempo de ejecucion para cada algoritmo?

En el caso A se puede evidenciar que no hay tanta variabilidad de tiempo, pero que para valores de N entre aproximadamente 60 y 220 el caso más optimo será asumir una matriz A simétrica, mientras que para los todos los demás casos de N (menores y mayores), el tiempo de ejecución de usar `scipy.linalg.solve` asumiendo una matriz positiva es mejor.

Para el caso B utilizando `scipy.linalg.eigh`, claramente hay una diferencia notable de tiempo entre los subcasos con `driver='ev'` y `driver=evx`, los cuales serán siempre más lentos, mientras que todos los demás casos tienen un comportamiento casi identico con un tiempo de ejecución considerablemente mejor.

Cabe agregar que el método `scipy.linalg.solve` es evidentemente más rápido que `scipy.linalg.eigh`.

- ¿Qué algoritmo gana (en promedio) en cada caso?

En el caso A en promedio el algoritmo más rápido es `scipy.linalg.solve(assume_a='pos')`.

En el caso B en promedio el algoritmo más rápido es `scipy.linalg.eigh(driver="evd", overwrite_a=True)` --> Es levemente más rápido, prácticamente imperceptible.

- ¿Depende del tamaño de la matriz?

Para el caso A, si es influyente el tamaño de la matriz, porque como ya mencioné antes el subcaso IV tiene mejor rendimiento que cualquiera con matrices de tamaño entre $60 < N < 220$.

Para el caso B, el porte de la matriz no es un factor influyente, siempre existe el mismo comportamiento (si hay diferencias es prácticamente imperceptible).

- ¿A que se puede deber la superioridad de cada opción?

La superioridad de cada opción tanto en el caso A como en el caso B, se debe plenamente a los paquetes y el proceso que reliza "por detrás" los diferentes métodos con sus diferentes parámetros. Por ejemplo, si se utiliza un argumento `overwrite_a=True`, se utilizarían menos recursos, ya que no se está creando una nueva matriz, sino que se está reescribiendo la misma, otros ejemplos podrían ser que al asumirla de un tipo (positiva o simétrica), en donde el cálculo es más fácil, y el paquete al estar informado, realiza "trucos" de resolución más óptimos.

- ¿Su computador usa más de un proceso por cada corrida?

Tal como se evidencia en las imágenes del procesador, este utiliza los 8 procesadores lógicos para la mayoría de los casos, hay un par de excepciones en donde utiliza 2 al máximo y los demás en mitad del rendimiento, pero en estricto rigor siempre esta utilizando cerca del 100% de la memoria de la CPU en los casos con N altos. Que el computador utilice más de un proceso significa que esta realizando muchas acciones diferentes simultáneamente y esto viene definido en parte por el computador (por que decide usar), pero también por las librerías y paquetes que se utilicen, los cuales pueden requerir de hacer varias acciones en simultáneo para hacerlas de manera más eficiente.

- ¿Qué hay del uso de memoria (como crece)?

En mi caso, la memoria se vio muy poco afectada, ya que mi procesador es bastante potente, solo cambiaba levemente cuando el procesador sobrepasaba el 100% de su rendimiento, y cuando lo hacía de todas maneras lo que cambiaba la memoria era muy poco. Probablemente esto haya ocurrido para los valores de N que eran más grandes. Todo lo anteriormente señalado, ocurre por la jerarquización de la memoria en los computadores.

Matrices dispersas y complejidad computacional

Se realizó un archivo .py, el cual desarrollaba la operación MATMUL tanto con matrices llenas como con dispersas, se puede notar que tanto el tiempo de ensamblaje de matrices laplácianas, tanto como resolver la operación, era infinitamente más rápido usando matrices dispersas, es más, para poder notar el gráfico, para las matrices dispersas se tuvo que utilizar Nmax = 1.000.000.

Ensamblaje

Para realizar el ensamblaje tanto de las matrices dispersas como de las llenas, se definieron las siguientes funciones para poder desarrollarlas:

Función Laplaciana Matrices Llenas

```
def matriz_laplaciana_llena(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,j] = -1  
                A[j,i] = -1  
    return A
```

Función Laplaciana Matrices Dispersas

```
def matriz_laplaciana_dispersa(N, dtype):  
    return 2*sparse.eye(N, dtype = dtype) - sparse.eye(N, N, 1, dtype = dtype) - sparse.eye(N, N, -1, dtype
```

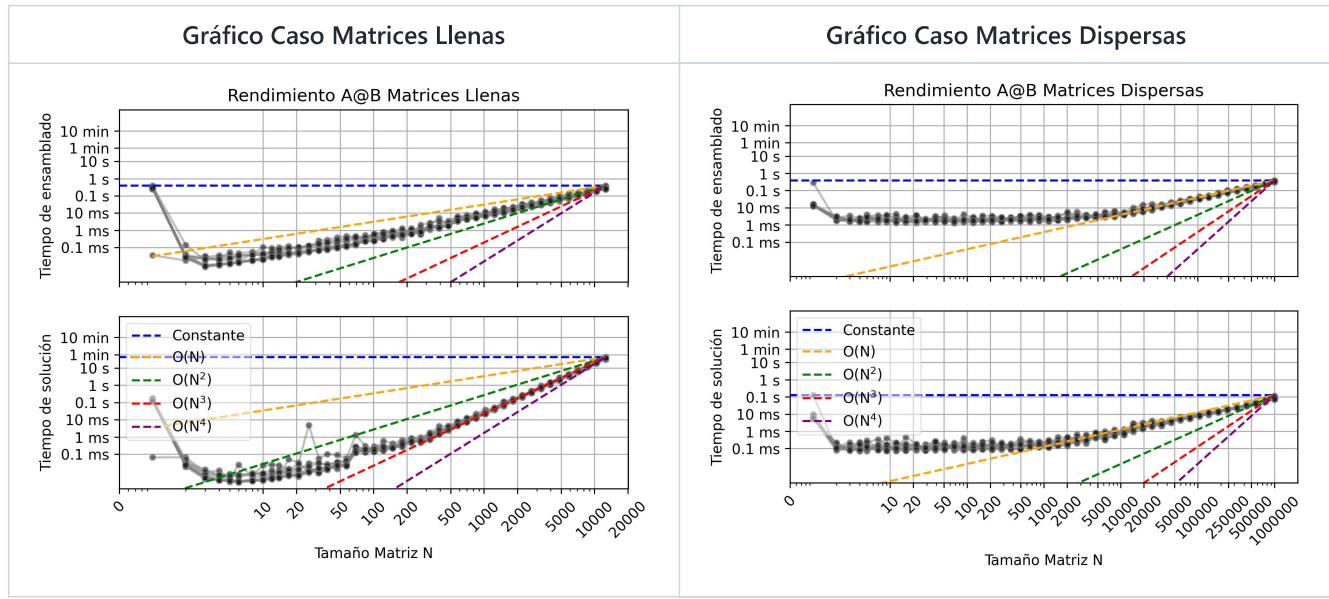
Para las matrices llenas no se pudo utilizar el método eye, ya que para el tipo de dato double no lo soportaba, mientras que para el caso de matrices dispersas si.

Solución

Para la solución simplemente se uso el método matmul (@), pero es importante para el caso de las matrices dispersas antes de realizar la operación, tanto las matrices A como B se transformaron a una matriz de tipo CSR.

Gráficos

A continuación se muestran los gráficos encontrados:



Es importante destacar que para poder desarrollar los gráficos representativos de complejidad computacional, se tuvo que trabajar pensando que se tenía un gráfico doblemente logarítmico, es decir del tipo $\log(y) = k \cdot \log(x) + \log(a)$ que otra manera de representarlo sería $y = a \cdot x^k$. De esta manera de despejan los distintos niveles de complejidad, ya que se conoce un punto (el máximo) y su pendiente, quedando de la siguiente manera: $\frac{y_{max}}{N_{max}^k} \cdot N_{values}^k$.

Luego se puede interpretar como para el caso de las matrices llenas para el caso del ensamblaje tiene un nivel de complejidad máxima de $O(N^2)$ y para la solución de MATMUL en la parte final casi de $O(N^4)$.

Para el caso de las matrices dispersas tuvieron que agregarse más valores ya que osino no se podía ver que hubiera una complejidad dada. Finalmente se puede notar que para el ensamblaje existe una complejidad máxima de $O(N^2)$ y para el caso de la solución de $O(N^2)$.

Matrices dispersas y complejidad computacional (parte 2)

Para esta entrega se realizaron 4 archivos .py, los cuales corresponden a los dos métodos a utilizar para matrices llenas y para dispersas. Para las matrices llenas se optó por usar las versiones más eficientes utilizando así el método de la inversa (caso con `overwrite_a=True`) y de solve (`assume_a='pos'`), para así compararlo con las matrices dispersas, que en todo caso, siempre lograron una mejor eficiencia.

Solución

Inversa: Para la solución simplemente se uso el método `inv` de `scipy.linalg`, pero es importante que para el caso de las matrices dispersas antes de realizar la operación, la matriz 'A' es necesaria transformarla al tipo CSC, ya que de esta manera el método consigue la mejor eficiencia.

Solve: Al igual que en el caso pasado, para el caso de matrices llenas se utilizó el método `solve` de `scipy.linalg`, mientras que para las matrices dispersas se utilizó el método `spsolve` de `scipy.sparse.linalg`, en donde la matriz 'A' (laplaciana) es transformada al tipo CSR, mientras que la matriz 'b', al ser una matriz compuesta de solo unos, no es necesario realizar ningún tipo de transformación.

Gráficos

A continuación se muestran los gráficos encontrados:

CASO INVERSA

Gráfico Caso Matrices Llenas (`overwrite_a=True`)

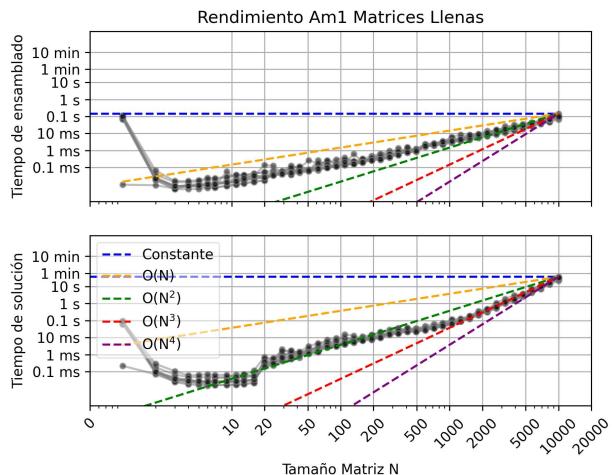
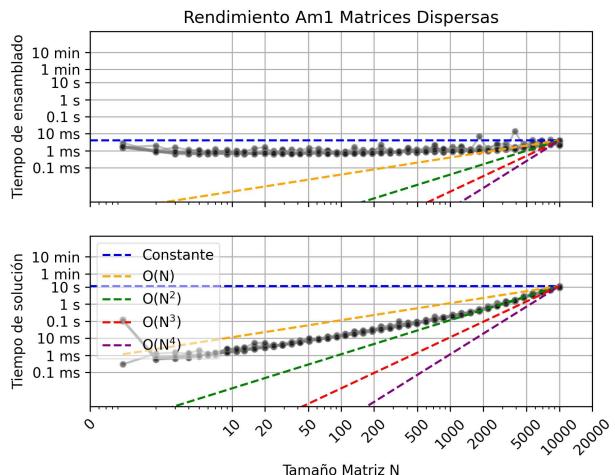


Gráfico Caso Matrices Dispersas



CASO SOLVE

Gráfico Caso Matrices Llenas (`assume_a='pos'`)

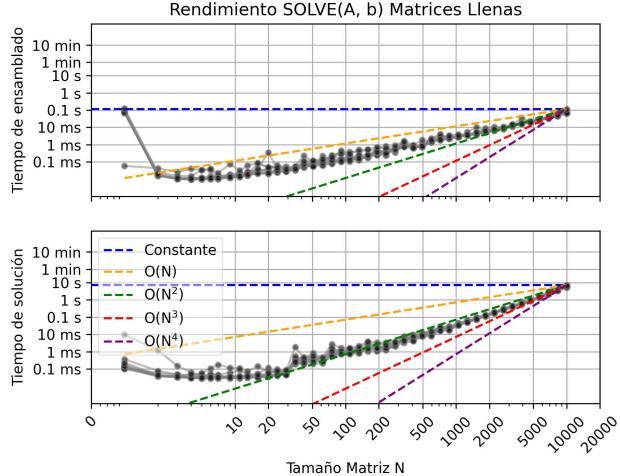
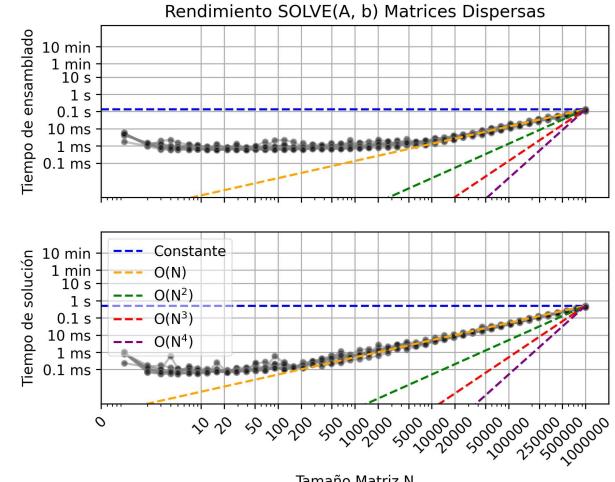


Gráfico Caso Matrices Dispersas



Ensamblaje Laplaciana

Función Laplaciana Matrices Llenas

```
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,j] = -1
                A[j,i] = -1
    return A
```

Función Laplaciana Matrices Dispersas

```
def laplaciana(N, dtype):
    return 2*sp.eye(N, dtype = dtype) - sp.eye(N, N, 1, dtype = dtype) - sp.eye(N, N, -1, dtype = dtype)
```

- Comente cómo esta elección se ve reflejada en el desempeño y complejidad algorítmica mostrada.

En primer lugar, se puede notar como en el caso de la laplaciana de matrices dispersas el rendimiento es mucho mejor, esto puede deberse principalmente a dos factores:

1.- El uso de la función 'eye' es mucho más eficiente que reemplazar los elementos de una lista (caso laplaciana dispersas vs caso laplaciana llenas).

2.- Utilizar el método `scipy.sparse` procesa los datos mucho más rápido.

Debido a todo lo aprendido se puede deducir que en general la opción 2 será más significante, ya que matrices dispersas son mucho más eficientes que las llenas.

Cabe señalar que la manera de almacenar los datos de las matrices llenas, tal como indica el nombre, es guardando toda la información y por ende al momento de realizar operaciones, muchos procesos que realiza son reiterativos e innecesarios, al contrario de las matrices dispersas que solo guarda los valores importantes y los demás los da conocidos por teoría (guarda los numeros y los 0 no, por lo que por ejemplo multiplicaciones por 0 no los toma en consideración). Por lo anteriormente señalado, lógicamente, la manera que trabaja `scipy.sparse` vs `scipy` será mucho más eficiente obteniendo un rendimiento mucho mejor (sobretodo en una matriz laplaciana la cual contiene muchos '0').

Por último, podemos argumentar que la complejidad algorítmica mostrada es directamente proporcional al valor de N, pero utilizando un mismo valor de N, las matrices llenas tienen una complejidad mayor a las de las dispersas.

Para obtener la misma complejidad entre estos dos tipos de matrices se necesita un valor de N muy superior en el caso de las matrices dispersas para poder igualar la complejidad de las matrices llenas.

Preguntas

Antes de contestar las preguntas, es importante señalar que el comportamiento de el método `solve` y del método `inv` tienen rendimientos muy parecidos, por lo que en la mayoría de los casos, las respuestas serán considerando ambas opciones, a no ser que se hable específicamente de cada uno.

- Comente las diferencias que ve en el comportamiento de los algoritmos en el caso de matrices llenas y dispersas.

La primera diferencia clara es el tiempo de ensamblado y de solución de las matrices llenas y dispersas, el cual en el caso de matrices dispersas es mucho menor.

Otra diferencia que se puede notar es que el comportamiento de las matrices llenas es del tipo exponencial, es decir, a medida que aumenta el valor de N el tiempo va incrementando de manera exponencial su tiempo de ejecución, pero en el caso de las matrices dispersas, al tener una eficiencia muy alta se puede observar que tiene un comportamiento lineal e incluso constante hablando de el tiempo de ensamblaje. Lo ocurrido con las matrices dispersas puede ocurrir ya que no se están utilizando valores de N lo suficientemente altos, pero ya se está utilizando un número muy alto, por lo que para casos más grandes no debieran necesitarse.

- ¿Cuál parece la complejidad asintótica (para $N \rightarrow \infty$) para el ensamblado y solución en ambos casos y porqué?

Matrices Llenas: Para este caso, podemos observar como en el caso de inv, el método pasa por todas las complejidades hasta llegar a un $O(N^3)$, tanto en el ensamblaje como en la solución, esto significa que el proceso que se utiliza para desarrollarlo, tal como se puede ver en el tiempo invertido, tiene una complejidad de un grado bastante considerable, es decir, "le cuesta" al sistema realizar la inversa con números de N muy grandes.

Probablemente si los valores siguen creciendo, debido al comportamiento, este caso termine siendo asintótico a una complejidad superior ($O(N^4)$).

Para el caso solve, podemos ver que los gráficos llegan a una complejidad del tipo $O(N^2)$, es decir, el método solve no es tan complejo para resolver para el computador como el caso de inv. Es decir, es un proceso más sencillo en términos computacionales. Si se aumentará el número de N tendiendo al infinito podría observarse una complejidad asintótica de hasta $O(N^3)$.

Matrices Dispersas: Para el caso de las matrices dispersas para el caso de inv, se puede observar que el tiempo de ensamblaje tiene un tiempo prácticamente constante, por lo que tiene una complejidad prácticamente nula. En el caso de la solución tiene una complejidad asintótica de $O(N^2)$, el cual con un N tendiendo al infinito termine siendo $O(N^3)$ con una complejidad más óptima que el caso de matrices llenas, lo que quiere decir que es un proceso que el computador desarrolla de mejor manera.

Para el caso de solve, el ensamblaje tiene una complejidad de $O(N)$ y una solución de $O(N)$ que probablemente con N mayores termine siendo $O(N^2)$, los cuales corresponden a complejidades que utilizarían muy pocos recursos.

- ¿Cómo afecta el tamaño de las matrices al comportamiento aparente?

Claramente en todos los casos se puede observar que con mayores valores de N, el tiempo de ensamblaje y de solución aumenta, es interesante señalar que llegado un valor de N bastante grande, las curvas comienzan a tener un comportamiento tipo por lo cual se puede predecir su comportamiento con N mayores. Además es interesante señalar, como en el caso de matrices dispersas, a pesar de tener matrices muy grandes, el tiempo de ejecución sigue siendo bastante bajo, con lo que se lleva a decir que su comportamiento entre N termina siendo muy similar, tendiendo a rectas constantes y lineales.

- ¿Qué tan estables son las corridas (se parecen todas entre sí siempre, nunca, en un rango)?

Se puede observar que en general las corridas tienen comportamientos similares, pero siempre dentro de un rango, se puede observar como hay saltos "inexplicables" en algunas corridas que se pueden explicar debido a las diferentes actividades que el computador haya estado realizando en ese momento y al nivel de importancia que le haya otorgado a la realización de esa iteración en específico, pero tal como dije antes, dentro de un rango, las corridas en todos los casos son bastante similares.

Lo anteriormente señalado se explica en entregas pasadas, en donde el computador realiza múltiples procesos que muchas veces desconocemos y prioriza algunas actividades sobre otras, pero al haber muchas actividades "invisibles", es decir aplicaciones que trabajan por detrás sin que uno lo sepa (recursos de windows por ejemplo, o incluso el antivirus), muchas veces la iteración se ve afectada.