

 josefa-tramon / **MCOC2021-P0**  
forked from jaabell/MCOC2021-P0

Notifications

Star 0


Fork 31



<> Code Pull requests Actions Projects Wiki Security Insights

 main ▾ **MCOC2021-P0** / README.md

Go to file

...

 josefa-tramon Update README.md Latest commit 95738f1 13 seconds ago History

2 contributors  

216 lines (129 sloc) | 13.1 KB Raw Blame

# MCOC2021-P0

P01:

## Mi computador principal

- Marca/modelo: DESKTOP-RV7HP2N
- Tipo: Notebook
- Año adquisición: 2018
- Procesador:
  - Marca/Modelo: Intel Core i5-7200U
  - Velocidad Base: 2,71 GHz
  - Velocidad Máxima: 2,71 GHz
  - Numero de núcleos: 2
  - Humero de hilos: 2
  - Arquitectura: AMD64
  - Set de instrucciones:
- Tamaño de las cachés del procesador
  - L1: 128 KB
  - L2: 512 KB
  - L3: 3,0 Mb
- Memoria
  - Total: 12 GB
  - Tipo memoria: DDR4
  - Velocidad 1867 MHz
  - Numero de (SO)DIMM: 2
- Tarjeta Gráfica
  - Marca / Modelo: AMD Radeon (TM) R5 M330
  - Memoria dedicada: 2048 MB
  - Resolución: 1366x768
- Disco 1:
  - Marca: ST1000LM035
  - Tipo: Duro
  - Tamaño: 1TB
  - Particiones: 4
  - Sistema de archivos: EXT4
- Disco 2:
  - Marca: KINGSTON
  - Tipo: SSD
  - Tamaño: 223 GB

- Particiones: 3
- Sistema de archivos: EXT4
- Dirección MAC de la tarjeta wifi: 42-9F-38-C2-0D-\*\* (por privacidad prefiero no dar completa mi dirección, por si las moscas jaja)
- Dirección IP (Interna, del router): 192.168.1.32
- Dirección IP (Externa, del ISP): 190.196.168.111
- Proveedor internet: GTD Manquehue.

P02:

¿Cómo difiere del gráfico del profesor/ayudante? Difieren en el tiempo transcurrido al inicio de la ejecución del programa, siendo más lento en mi computador que en el del profesor/ayudante.

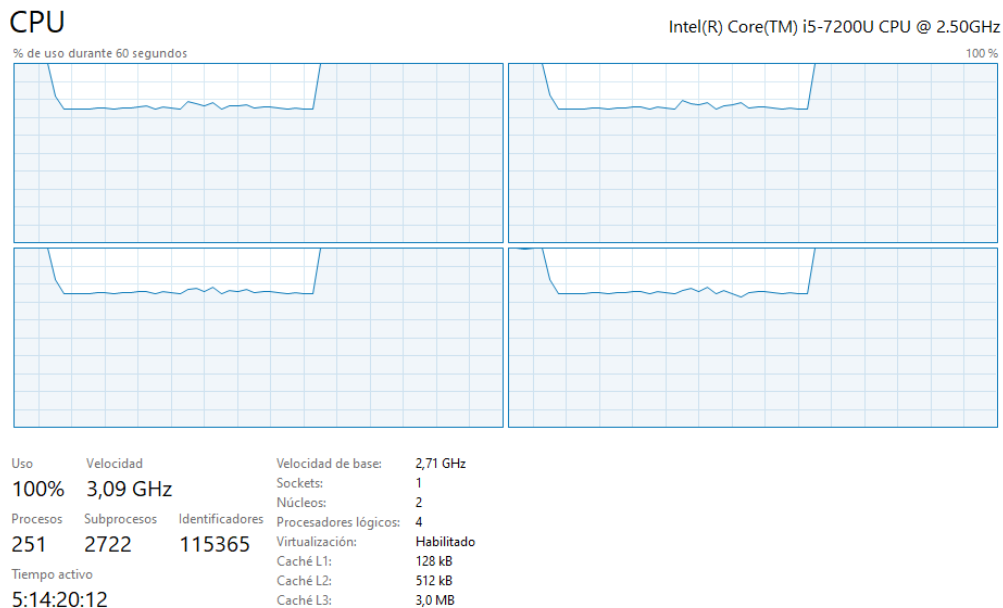
¿A qué se pueden deber las diferencias en cada corrida? Cada corrida es diferente ya que, aunque el equipo es el mismo, esta realizando diferentes tareas de trasfondo, las cuales según la prioridad que tienen de realización, afectan el rendimiento del programa.

El gráfico de uso de memoria es lineal con el tamaño de matriz, pero el de tiempo transcurrido no lo es ¿porqué puede ser? Esto se puede deber a que el espacio que ocupa una matriz es constante, por lo que si crece, crece linealmente su memoria utilizada, en cambio el tiempo de ejecución de la ponderación de las matrices varía de forma exponencial, al crecer las matrices, la dificultad del problema no varía linealmente, ya que no es constante como el crecimiento de las matrices por separado.

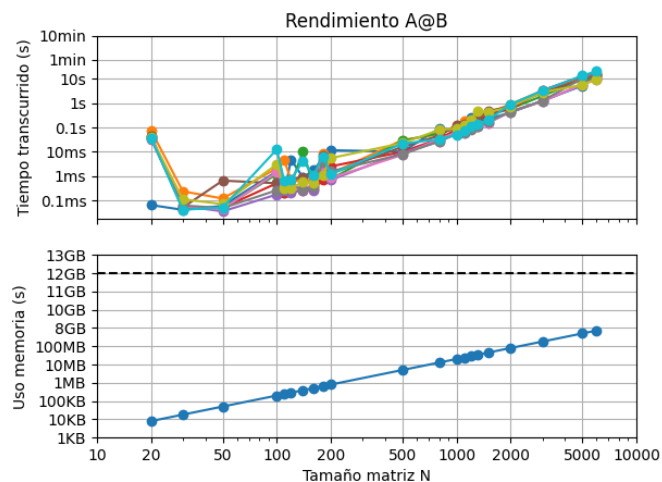
¿Qué versión de python está usando? Python 3.8

¿Qué versión de numpy está usando? Numpy 1.20.3

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, se utilizan los 2 núcleos físicos, y 4 hilos, en un 100% cada uno como se puede ver en la fotografía:



El desempeño MATMUL en mi computador es el siguiente:



Se puede apreciar como los procesadores lógicos, que serían los 4 hilos mencionados previamente, están mayoritariamente trabajando al 100% de su capacidad, cada uno representado en los gráficos. La baja en los gráficos y luego subida se debe a la variación de matrices que se están trabajando en el programa, siendo las de menor tamaño primero, y luego las de mayor tamaño, produciendo el uso completo del CPU.

P03:

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

- El método de numpy utiliza el algoritmo de Gauss - Jordan, en el cual se encuentra la inversa de una matriz a través de ecuaciones, donde el número de ecuaciones será el mismo al número de incógnitas, las cuales pertenecerán a la matriz inversa de una matriz A conocida. Esto se encuentra al multiplicar la matriz A conocida con la matriz identidad I, y despejando la solución correspondiente.
- El método de Scipy utiliza el algoritmo de Gauss - Jordan, igual que el método de numpy.

Para el caso 2: `overwrite_a = True`, se sobrescribe el resultado obtenido en la misma matriz A, dentro del sistema, lo cual es "más beneficioso" ya que se reutiliza espacio previamente ocupado por la matriz A original. a veces se cumple y otras no, todo dependerá de cuán específico son los resultados esperados.

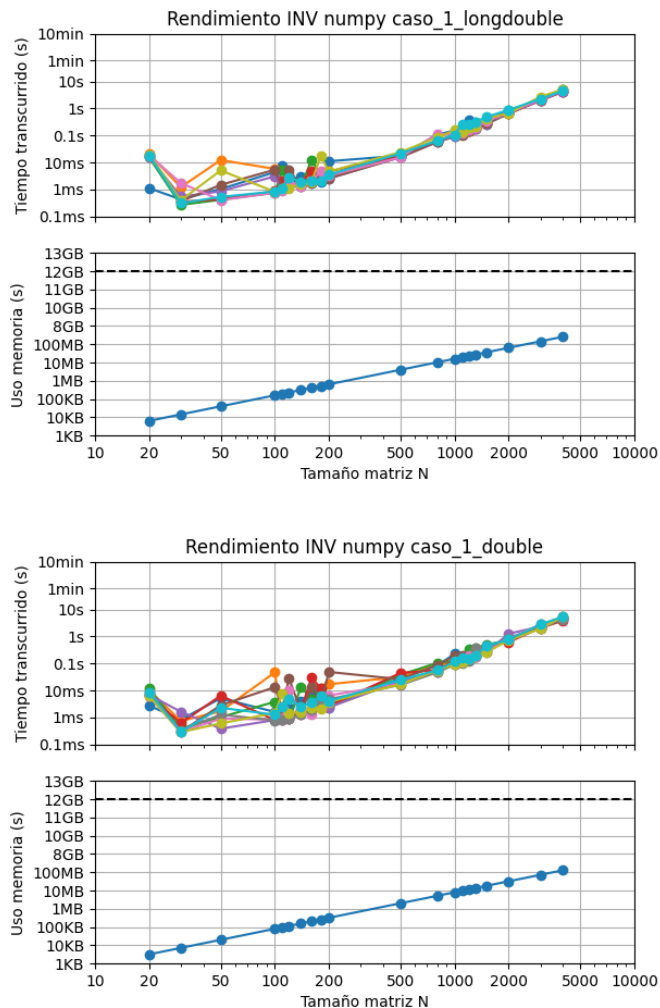
Para el caso 3: `overwrite_a = False`, No se sobrescribe el resultado obtenido, sino que se ocupará nuevo espacio para escribir el resultado (matriz inversa), en la memoria del sistema, por lo que debería ser menos eficiente.

¿Como 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.

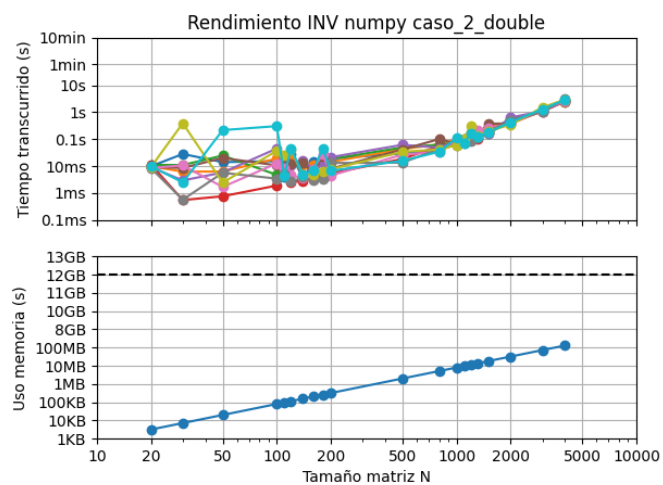
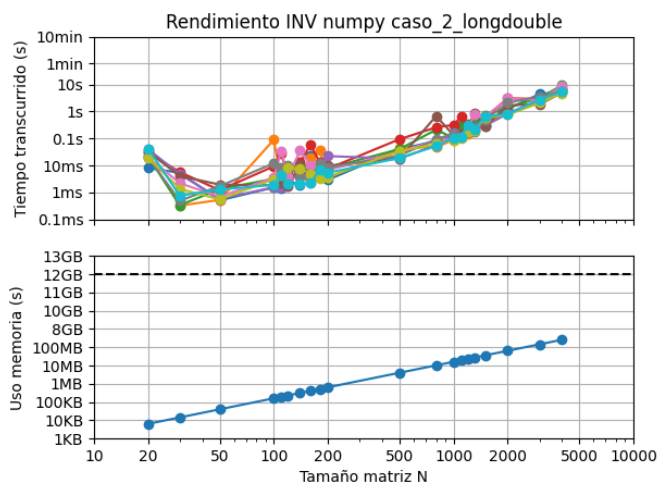
- Cuando el equipo resuelve varios trabajos simultáneamente, y por lo mismo, genera cierta "jerarquización" sobre las tareas más importantes a las de menor relevancia, para así ejecutar lo que considera más relevante terminar primero.

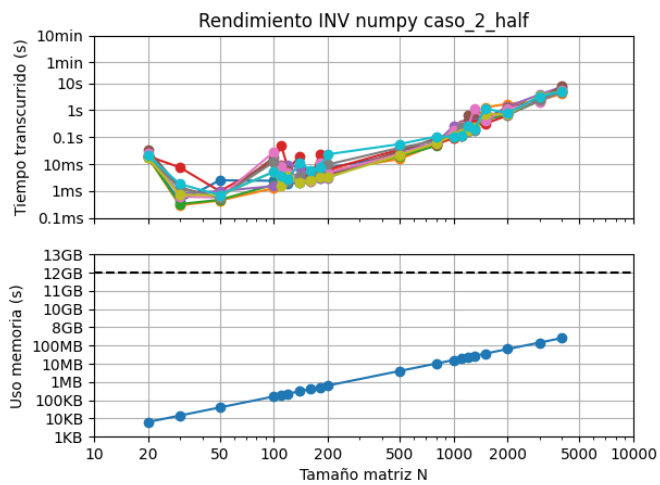
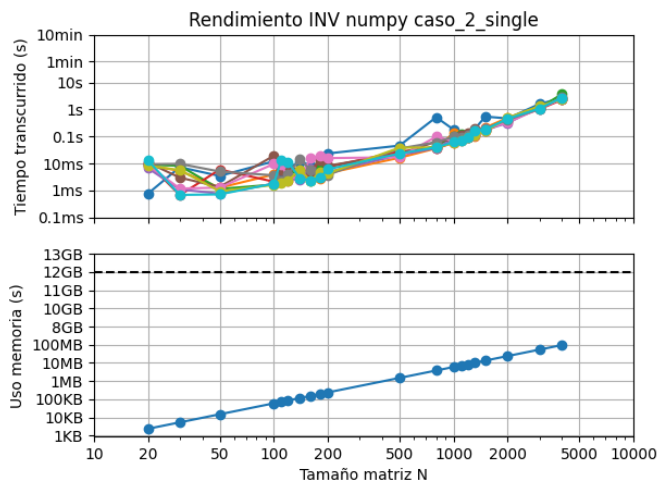
#### RESULTADOS OBTENIDOS:

\*CASO 1: Los casos 1 (numpy.inv) para half y single no fueron posibles ya que los resultados son poco exactos, y al utilizar este método, el programa primero ejecuta los cálculos con `d_type float 32`, y luego aproxima más estos resultados, no realizándolo con menos decimales ya que no realiza cálculos con tan poca especificidad.

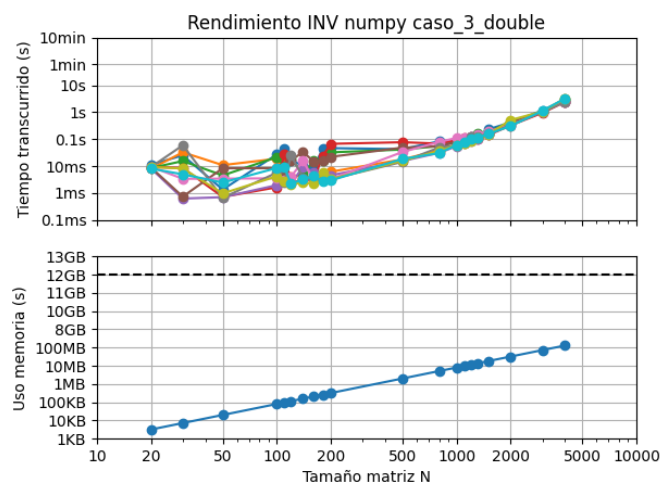
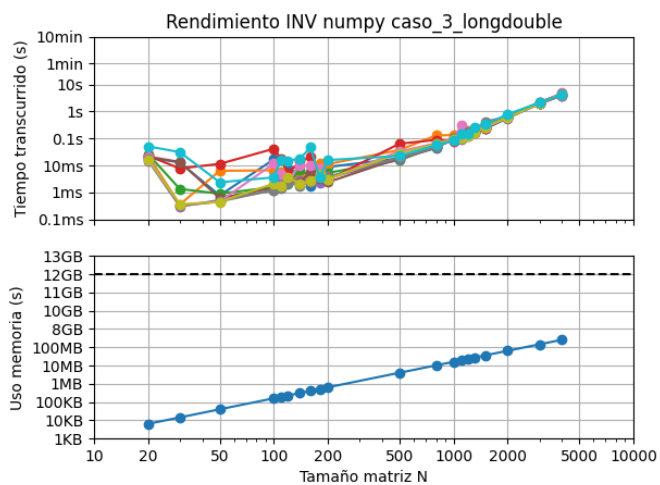


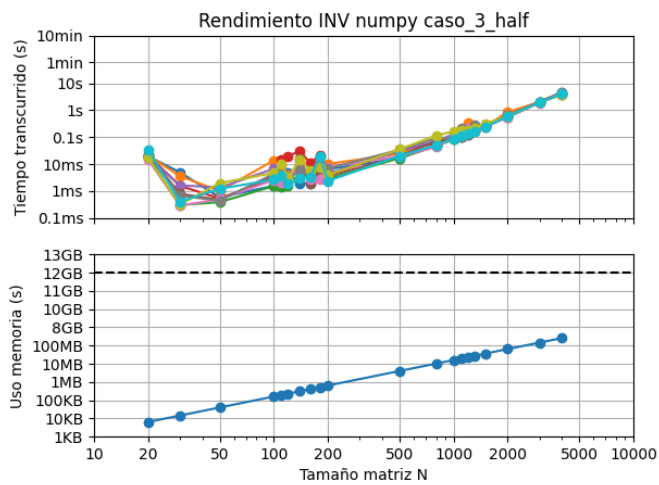
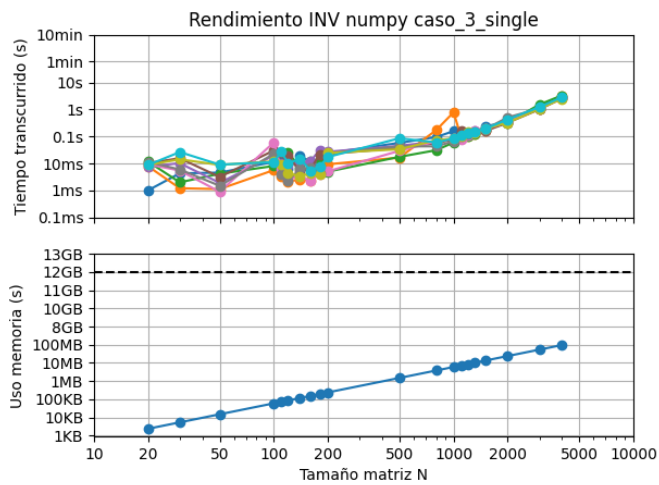
CASO 2:





CASO 3:





#### COMPORTAMIENTO EQUIPO DURANTE PROCESOS:

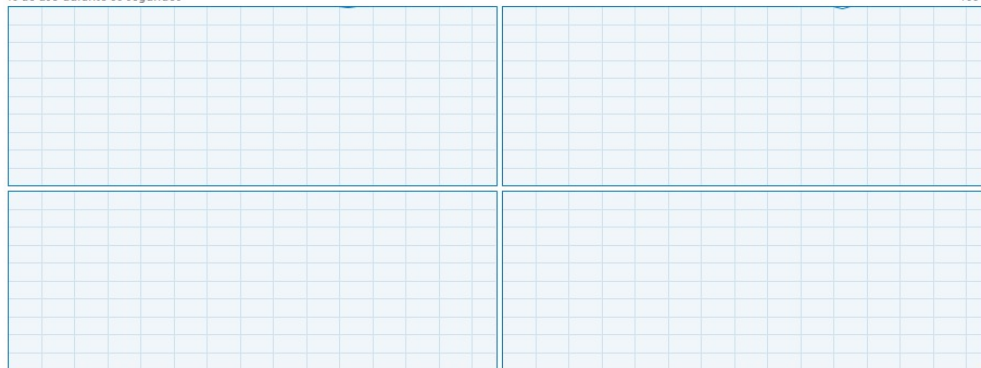
Se puede ver el uso completo de la CPU al procesar los calculos solicitados.

### CPU

Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz

% de uso durante 60 segundos

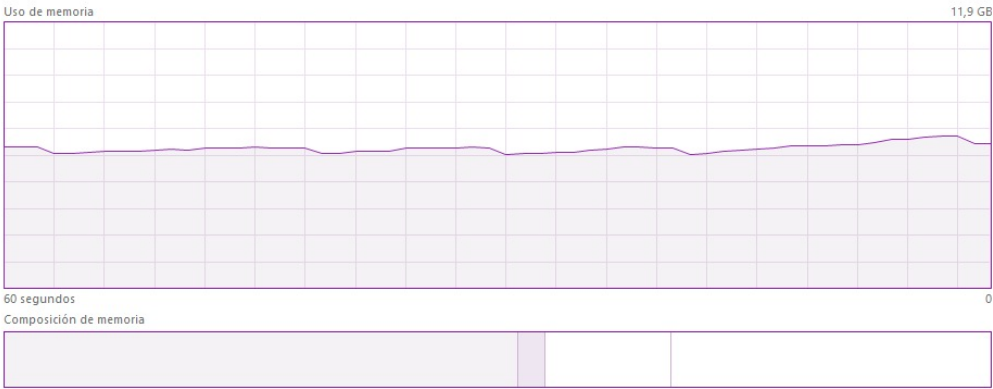
100 %



Uso	Velocidad	Velocidad de base:	2,71 GHz
100%	3,06 GHz	Sockets:	1
Procesos	Subprocesos	Núcleos:	2
265	3016	Procesadores lógicos:	4
Identificadores		Virtualización:	Habilitado
122244		Caché L1:	128 kB
		Caché L2:	512 kB
		Caché L3:	3,0 MB
Tiempo activo			
6:10:46:37			

El uso de memoria fue de un 50% aproximadamente.

# Memoria

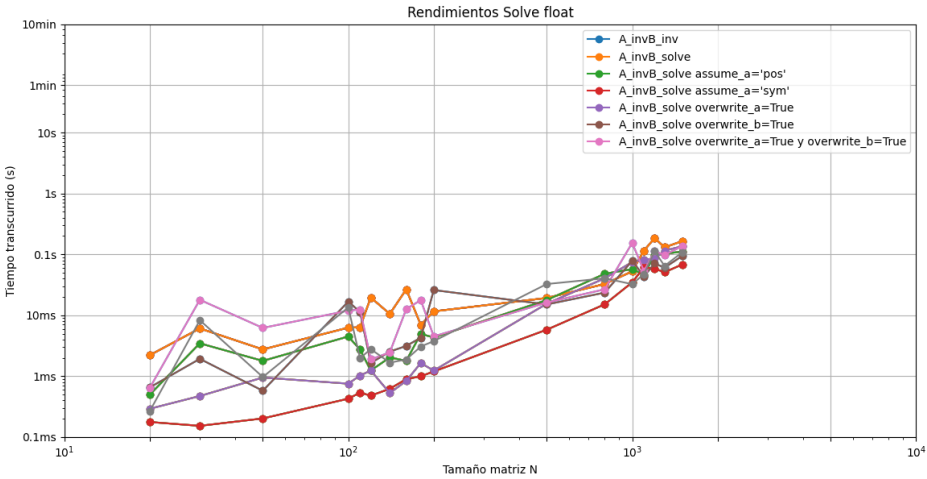


En uso (comprimido)	Disponibile	Velocidad:	2133 MHz
6,2 GB (211 MB)	5,4 GB	Ranuras usadas:	2 de 2
		Factor de forma:	SODIMM
Confirmada	En caché	Reservada para hardware:	114 MB
12,7/25,4 GB	1,8 GB		
Bloque paginado	Bloque no paginado		
518 MB	495 MB		

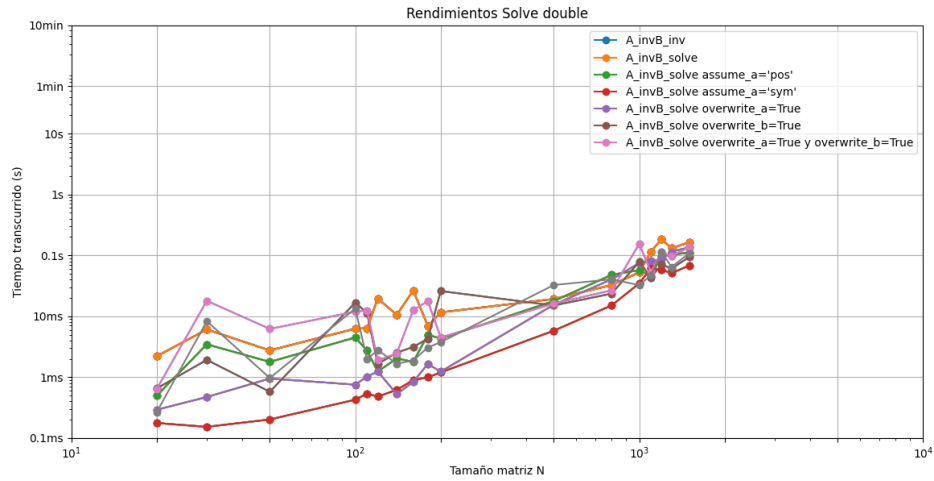
P04:

Haga un comentario completo respecto de todo lo que ve en términos de desempeño en cada problema. ¿Como es la variabilidad del tiempo de ejecucion para cada algoritmo? ¿Qué algoritmo gana (en promedio) en cada caso? ¿Depende del tamaño de la matriz? ¿A que se puede deber la superioridad de cada opción? ¿Su computador usa más de un proceso por cada corrida? ¿Que hay del uso de memoria (como crece)?

\*Para Solve:



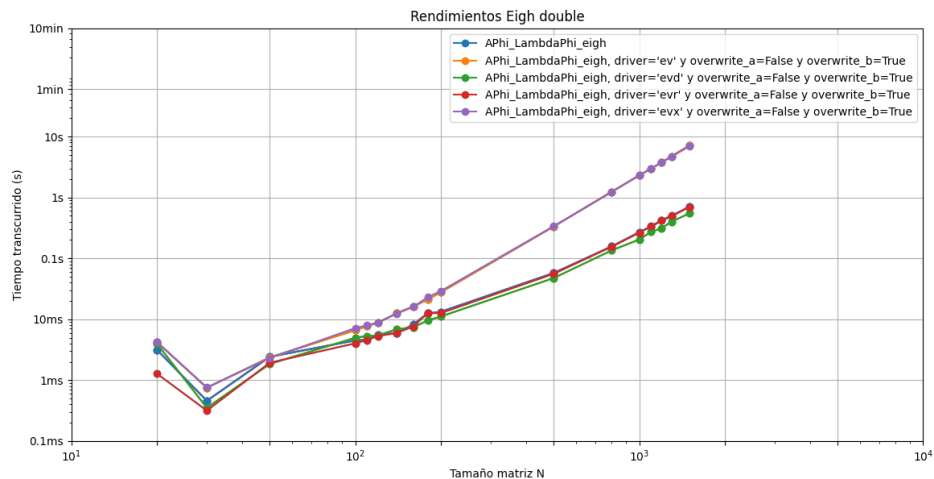
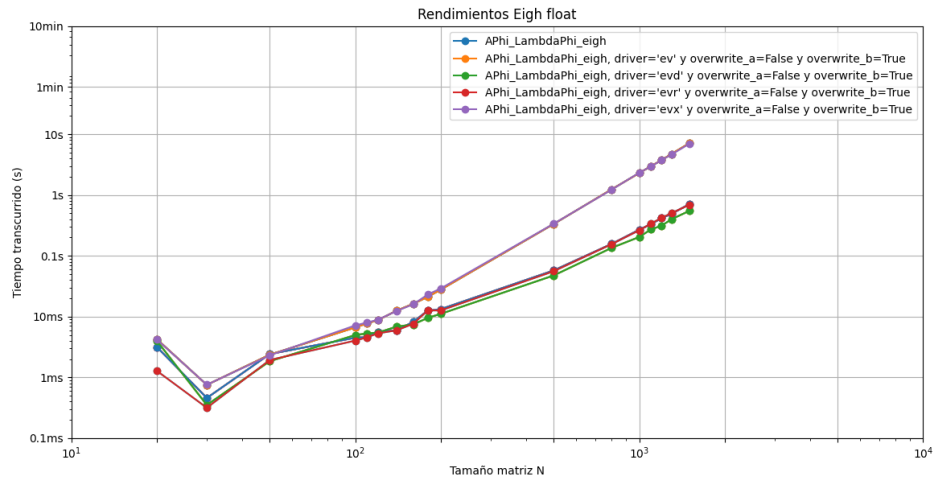




Para ambos tipos de `d_type` utilizados, float y double, se generan los mismos resultados, esto se puede deber, a que los decimales solo generan mayor exactitud de los resultados, y al ser float uno ya bastante exacto, la diferencia con double es mínima.

Si se ven las matrices en crecimiento, el caso que predomina en rendimiento es el 4to, donde Solve tiene a su parametro `assume_a = "sym"`, que trabaja la matriz como si esta fuera simétrica, lo cual se entiende ya que una matriz simétrica tiene ventajas a la hora de generarse su inversa, a diferencia de una que no lo es. Como el programa asume esa condición, le es más facil y rápido trabajar. Las menos eficientes son el caso 7 y 2, lo cual se podría explicar ya que para el primero mencionado, reescribe la matriz a y el vector b, para reutilizar memoria, sin embargo esta acción, mientras mayor es la dimensión de la matriz, más lento el proceso y por tanto, tiene menor rendimiento. Para el primer caso, asume por default, que es una matriz generica, osea no simetrica, y por tanto, su trabajo tambien es poco eficiente.

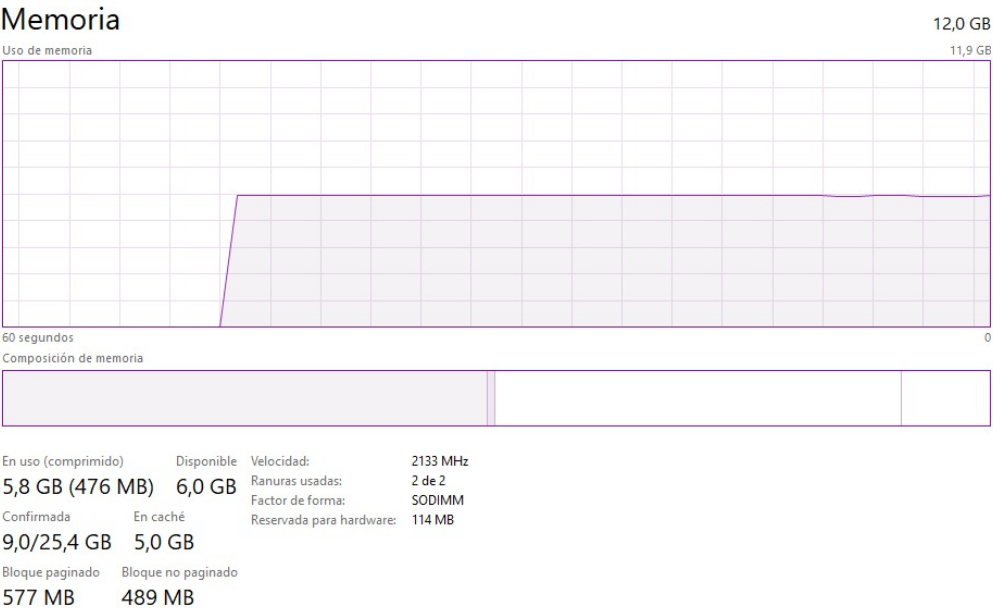
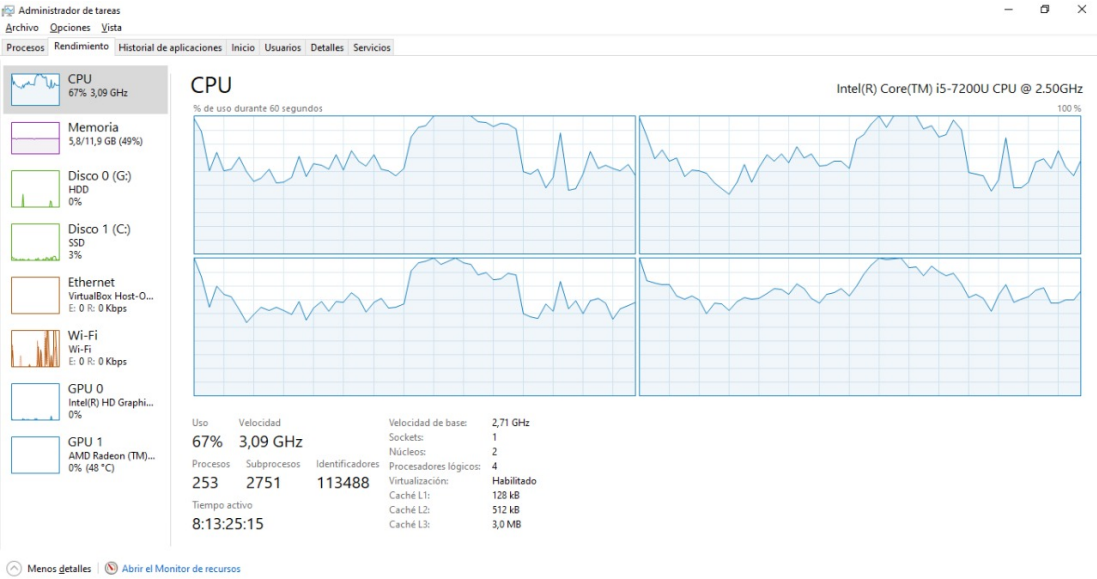
\*Para Eigh:



Al igual que en el calculo de solve, para ambos tipos de d\_type utilizados, float y double, se generan los mismos resultados, esto se puede deber, a que los decimales solo generan mayor exactitud de los resultados, y al ser float uno ya bastante exacto, la diferencia con double es mínima.

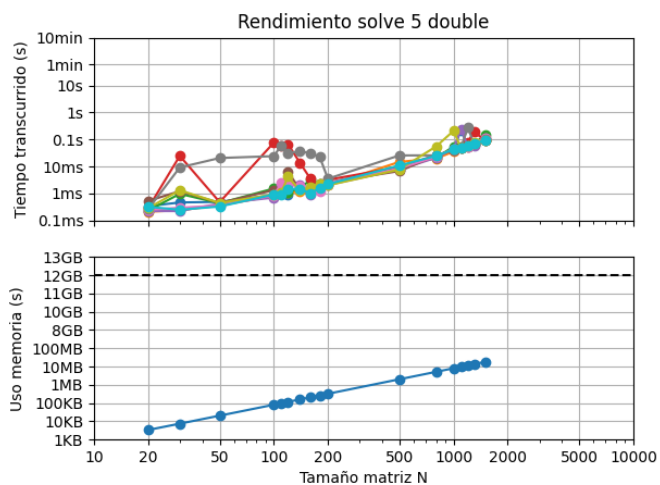
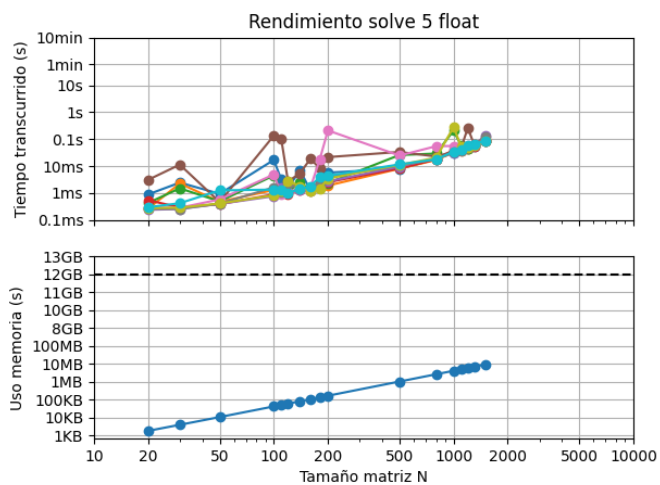
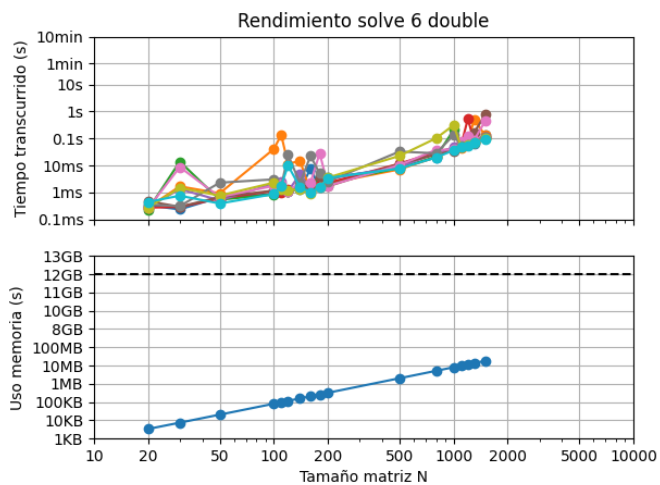
El caso de mayor eficiencia es en un inicio, con matrices de dimensión menor a 200, es el número 4, donde el parametro driver = "evr", no se sobrescribe la matriz A, pero si el vector b. Posterior a esa dimensión, tiene mayor eficiencia en tiempo el caso 3, donde la unica diferencia con el caso 4 es que el parametro driver = "evd". Esto es porque los driver routines son más o menos eficientes según el procesador, y para este caso, estos funcionan así, siendo el 3 caso, el driver routine que mejor resuelve ecuaciones lineales en el sistema.

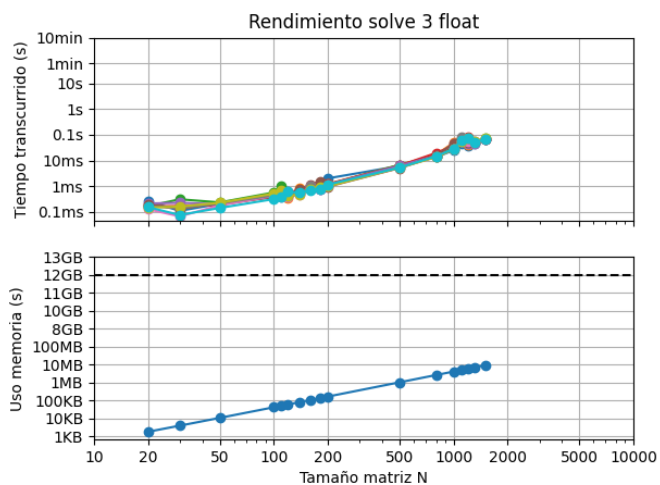
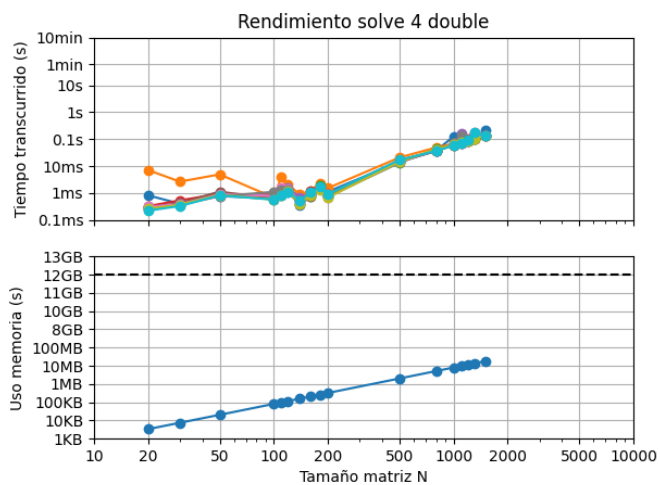
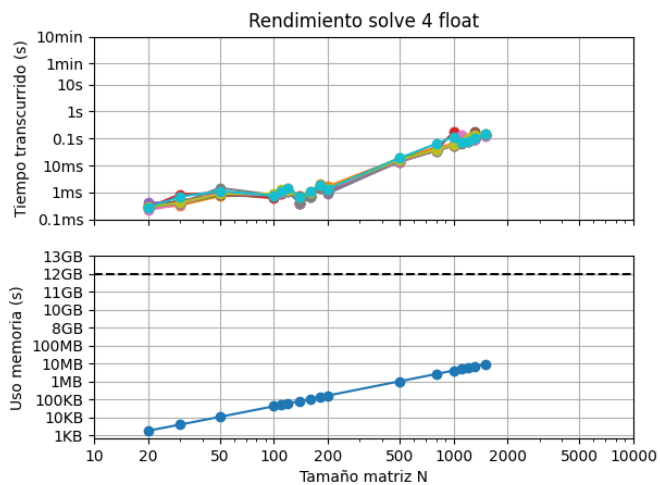
En cuanto al uso de memoria:

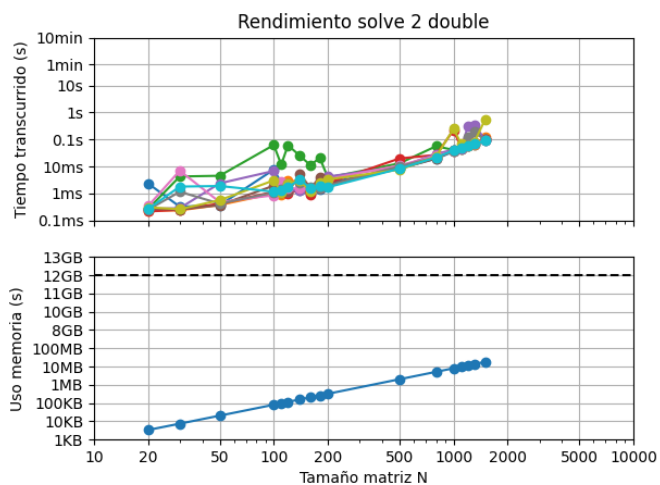
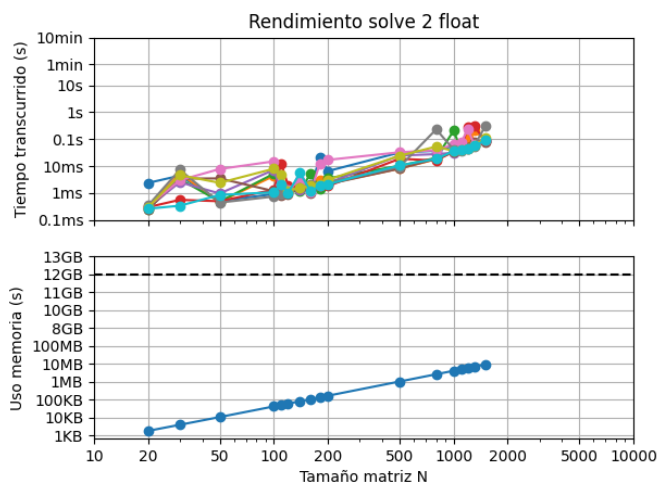
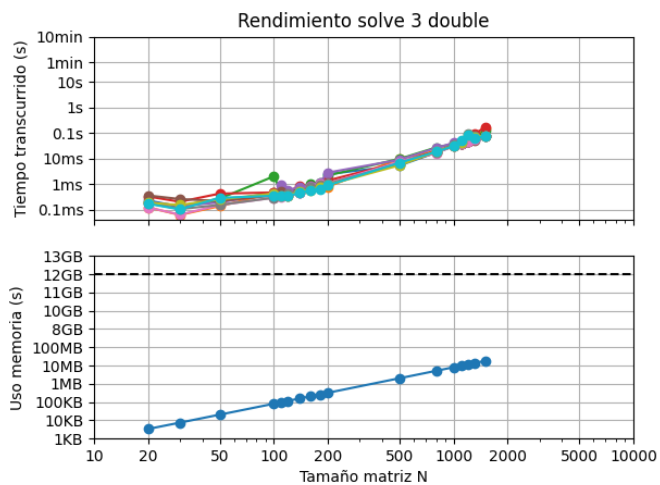


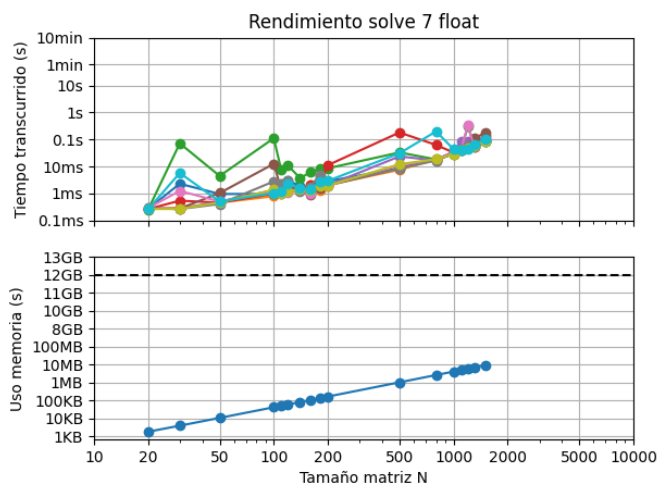
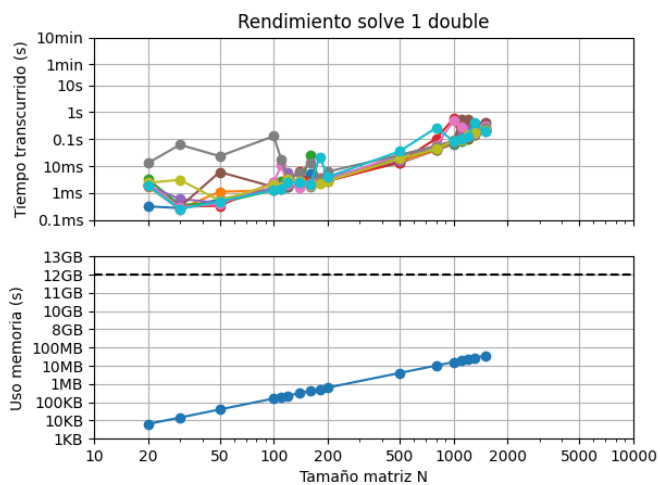
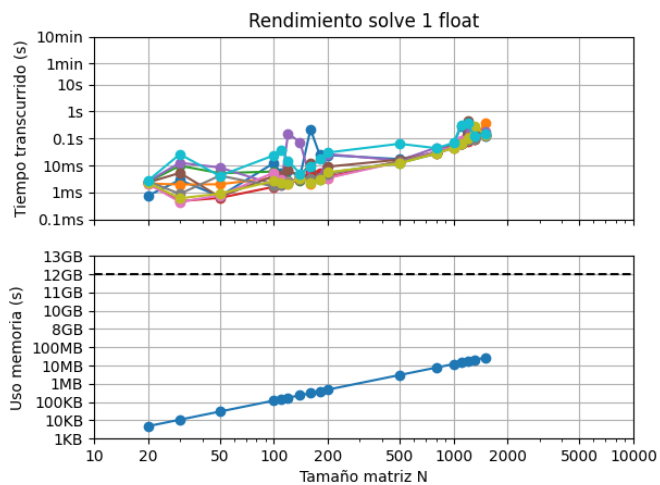
Para ambos casos, se utilizan los 4 hilos, y la memoria es utilizada en un 50%. Por ultimo, para las matrices, con el uso de Eig, su memoria crece linealmente, en cambio en solve, este cálculo ocupa memoria variable, ya que como las dimensiones no crecen linealmente sino que como exponentes, su cambio en el uso de memoria tambien se comporta así.

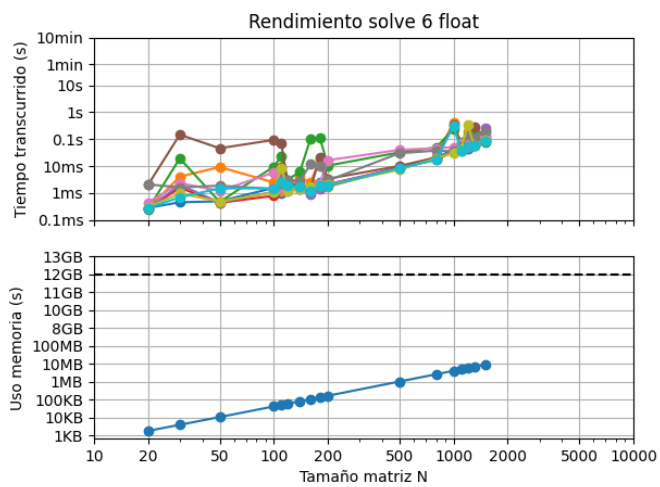
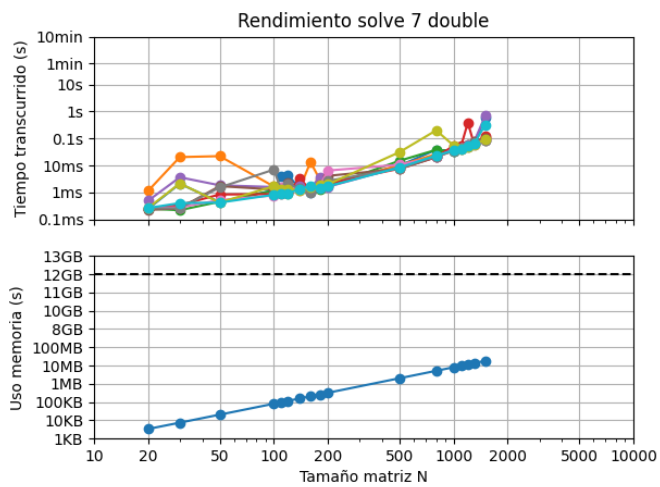
RESULTADOS CASOS SOLVER:











RESULTADOS CASOS EIGH:

