Pontificia Universidad Católica de Chile
Instituto de Ingeniería Matemática
IMT2112

Tarea 2

2 de octubre de 2020

Fernando De Diego Ávila

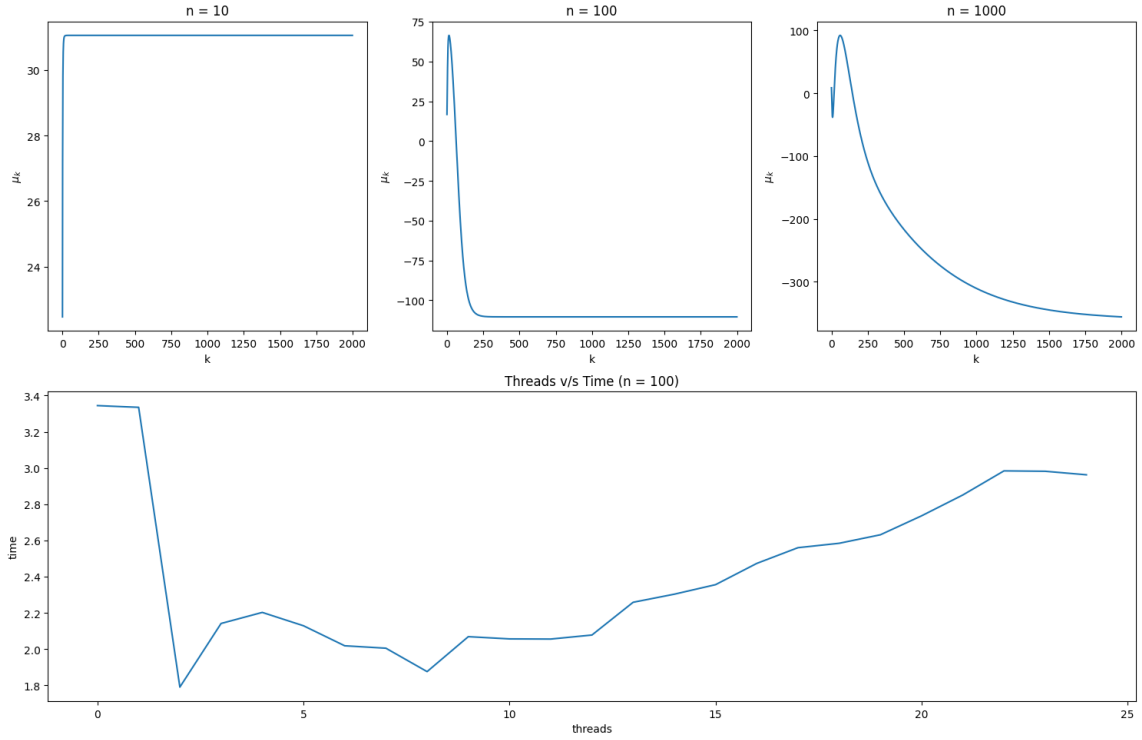
Resumen

Mi tarea consta de tres archivos principales:

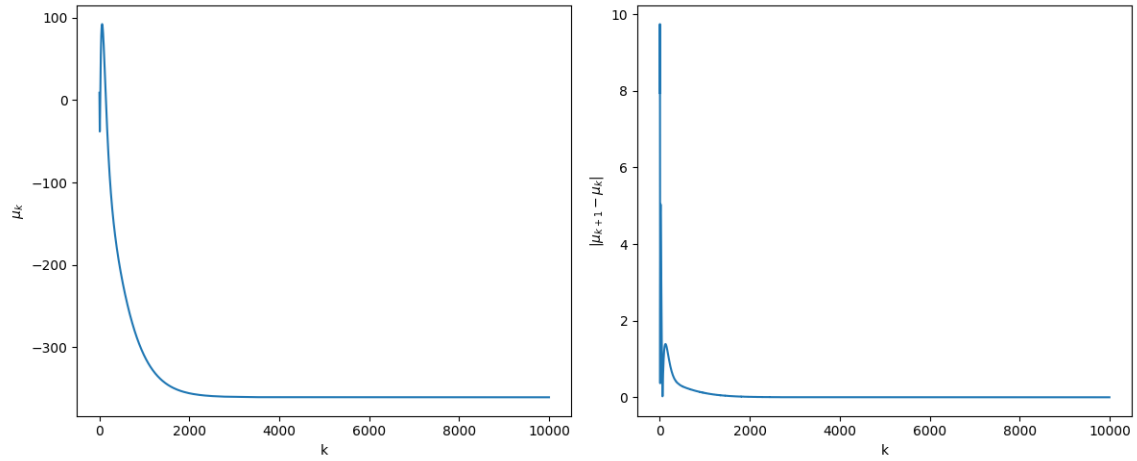
- `functions.cpp`: Se definen todas las funciones que se utilizan en la ejecución. Incluye tanto versiones lineales, como paralelizables. Es importante mencionar que para este trabajo se utilizaron matrices aleatorias simétricas, ya que son diagonalizables.
- `main.cpp`: Corresponde al archivo principal de ejecución, en el cual se pueden ir variando los distintos parámetros para ver cómo se comportan los resultados. Al final del archivo, hay una recomendación de cómo compilar y correr el programa, de manera tal de que se ejecute un *script* con nombre `plot_main.py`, el cual grafica la convergencia de los valores propios.
- `statistics.cpp`: Genera un compilado de resultados interesantes. Primero, se grafica la convergencia (en 2000 iteraciones y 2 *threads*) de tres ejemplos, con dimensiones $10, 100$ y 1000. Luego, se grafican los tiempos de ejecución para un problema con dimensión 1000, 1000 iteraciones y un número variable de *threads*, que va entre 0 (versión lineal, sin OpenMP) y 24. Al igual que en el archivo anterior, al final se encuentra una recomendación de cómo compilar y ejecutar el programa, para graficar los datos con la ayuda de un *script* llamado `plot_statistics.py`.

Resultados

A continuación, se muestra una recopilación de los resultados obtenidos al correr los archivos `statistics.cpp` y `plot_statistics.py`, en donde fue utilizada la *seed* aleatoria 0 para generar la matriz y la *seed* 1 para el vector (estos valores se pueden cambiar en el archivo `functions.cpp`):



Como podemos ver en los tres primeros gráficos, el algoritmo parece converger, con tiempos 0,014s, 0,064s, 4,976s (aproximadamente), para los casos $n = 10, 100$ y 1000, respectivamente. Por otra parte, el tiempo de ejecución en función de los *threads* disminuye abruptamente al utilizar dos hilos, pero incrementa a medida que aumentamos el número de *threads*, probablemente debido al *overhead* asociado a generar y comunicar los distintos hilos. A continuación, se muestra más detalladamente la convergencia del algoritmo en caso $n = 1000$, pero aumentando a 10000 las iteraciones:



En el gráfico de la izquierda, podemos apreciar que μ_k converge a 360,707, aproximadamente; mientras que la variación $|\mu_{k+1} - \mu_k|$ es prácticamente nula luego de 10000 iteraciones (gráfico

derecho).

Conclusión

El algoritmo converge en todos los casos probados. Por otra parte, el tiempo de ejecución en función del número de hilos usados disminuye en un comienzo, mejorando el rendimiento del algoritmo. Sin embargo, a medida que incrementamos los *threads*, comienza a aumentar. Estos cambios en el tiempo de ejecución son causados, probablemente, por el *overhead* generado por iniciar los hilos y comunicarlos, además de por las características de los procesadores de mi sistema. Por otra parte, es probable que, como paralelizamos bucles en función de la dimensión (n), si aumentamos aún más su valor, entonces se obtengan resultados más notorios.