**Sistemas Complejos en Máquinas Paralelas**

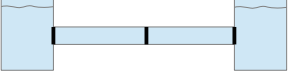
**2do cuatrimestre 2014**

**1era entrega**

Alumno: Ezequiel Darío Gambaccini

LU: 715/13

Se posee un tubo conectado a dos recipientes, como se ve en la figura:



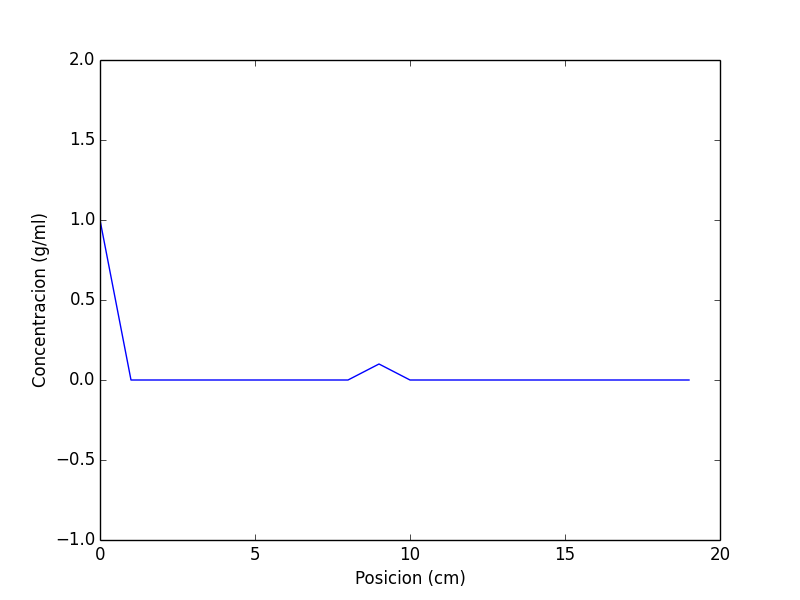
Cada extremo del tubo está conectado a un recipiente mediante una membrana semipermeable. Se modelará el transporte de la concentración de la especie química i en el tubo sólo por difusión. El recipiente de la izquierda posee una concentración de 1 g/ml de dicha especie química, que mantiene de forma constante. Dentro de la membrana del centro se produce de forma inmediata una

reacción química tal que su concentración es igual a 0,1 g/ml más el promedio de la concentración de la especie i entre ambos lados de la membrana. Además, en la membrana de la derecha se produce otra reacción química, debida a un reactivo, que elimina la concentración de la especie i, también de forma inmediata. El largo del tubo es 20 cm y el coeficiente de difusión de la especie i es 0,1 cm2/s. La concentración inicial de dicha especie en el tubo es 0.

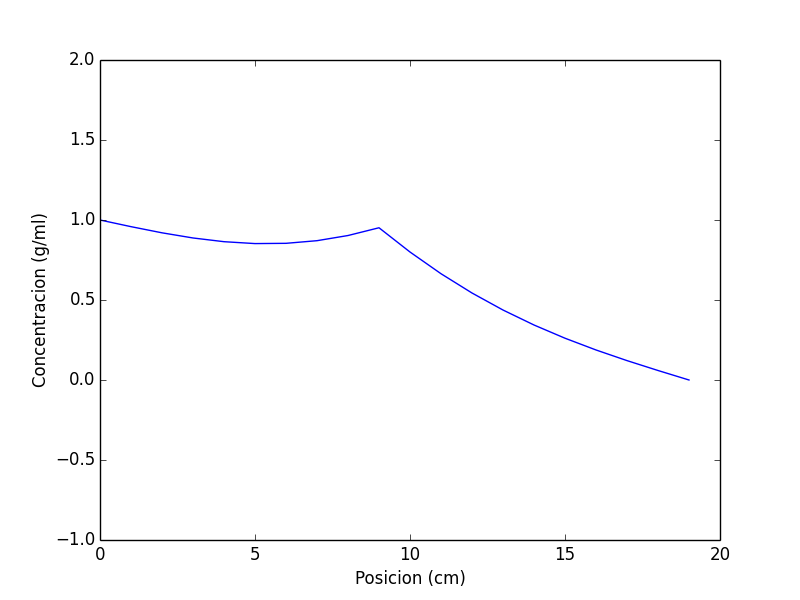
Se usará para la simulación un método explícito por diferencias finitas. Modele la membrana central como una condición de borde que cambia con la concentración.

1.a) Grafique los resultados de los tiempos 0s, 50s, y 100s.

0s:

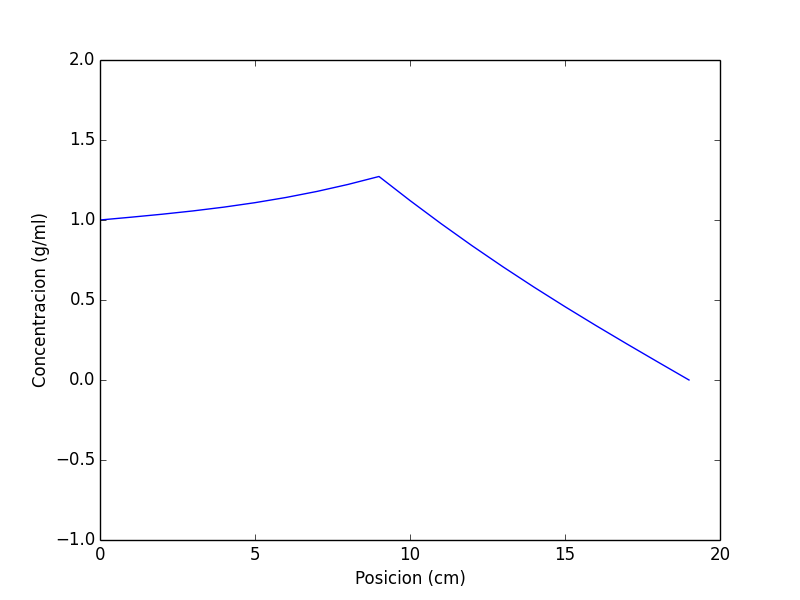


50s:



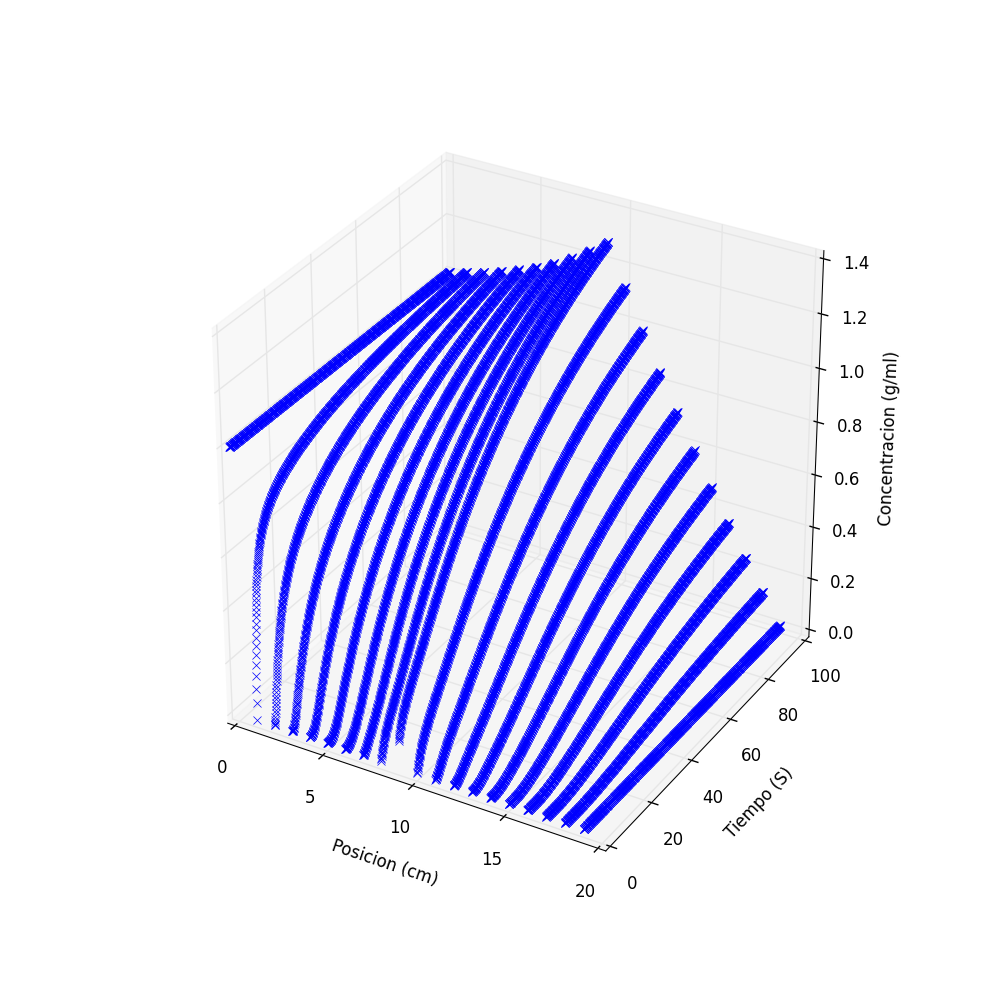
Concentración en x = 10cm: 0.950473 g/ml

100s:

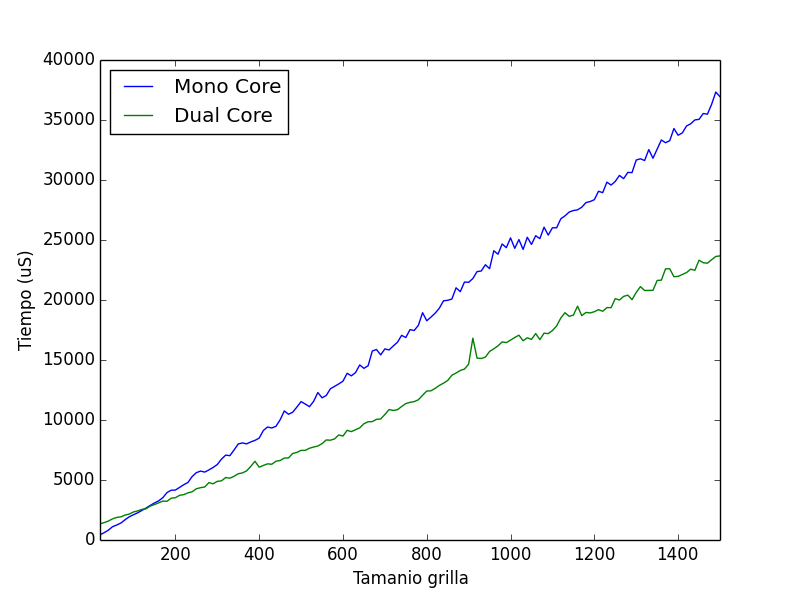


Concentración en x = 10cm : 1.272326 g/ml

Evolución a través del tiempo:

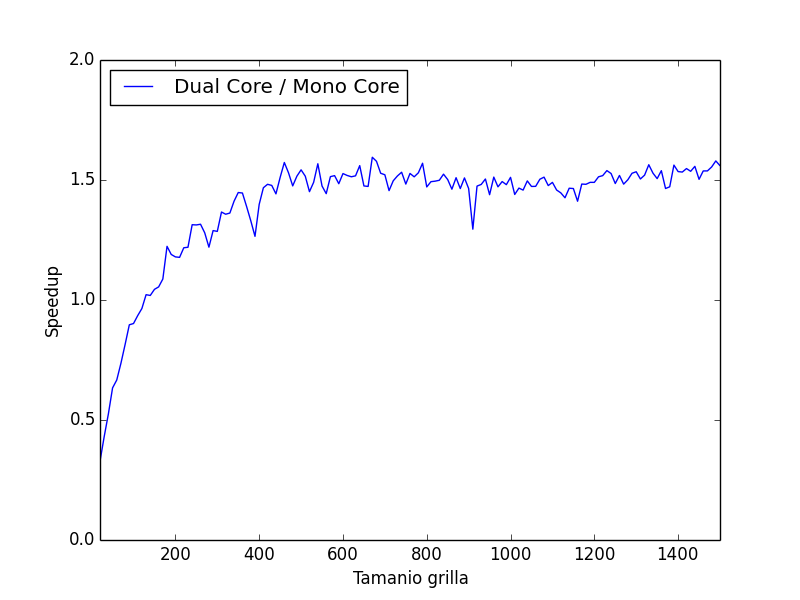


c) Si se deja correr el sistema más de 100 segundos, el valor en x = 10cm tiende a infinito. Esto se debe a que no se puede modelar completamente un fenómeno físico. En la realidad, la solución se saturaría y el soluto decantaría al fondo del tubo, en vez de que su concentración tienda a infinito.

2) A continuación se ve la diferencia en tiempo de ejecución del modelo a medida que se incrementa el tamaño de la grilla entre 1 core y 2 cores.

Para conseguir estos valores, se generó un promedio de 1000 iteraciones para cada tamaño de grilla probado. El intervalo de tiempo de cada simulación fue de 100s y el paso de tiempo fue de 0,1s.

A continuación se muestra la mejora de velocidad entre 1 core y 2 cores.



Como se puede ver, a medida que aumenta el tamaño de la grilla, la mejora de velocidad también lo hace, hasta estabilizarse aproximadamente en 1,6 veces más rápido, o en otras palabras, un 60% más rápido. Esto tiene sentido respecto a lo que dice la ley de Ahmdal:

(n = cantidad de threads, B = parte serial del algoritmo)

Siendo n = 2:

Cuando la grilla aumenta de tamaño, la parte serial del algoritmo disminuye, hasta llegar a 0,25, donde la mejora en velocidad termina tendiendo hasta 1,6 veces.

En el desarrollo de este problema, lo más complejo fue traducir de manera correcta la ecuación diferencial de difusión a su forma de diferencias finitas para luego usar en el programa. En mi caso particular, como ya tenía experiencia programando en C y con programación en paralelo, no tuve mayores incovenientes con el desarrollo del modelo.

**Glosario de archivos adjuntos y sus funciones:s**

La salida del código mpi con las concentraciones en pasos de a 0,1s se encuentran en el archivo merge\_out.txt. La salida del código serial se encuentra en serial\_out.txt.

Programas adjuntos:

data\_plotter.py:

Generador de gráficos 3D para mostrar la evolución en el tiempo de la solución.

Pasándole el 3er parámetro en true, se genera un gráfico interactivo donde se puede ver la evolución de la solución.

Se usa de la siguiente manera:

python data\_plotter.py salida.txt true

Donde salida.txt es alguno de merge\_out.txt o serial\_out.txt.

Si no se le pasa el parámetro true, se genera un archivo .png con el gráfico.

plot\_steps.py:

Generador de gráficos 2D de concentración/posición.

Dado un archivo de salida, genera gráficos de concentración/posición por cada paso del archivo. Por ejemplo, si serial\_out.txt tiene 1000 pasos, se generaran 1000 archivos .png

serial\_run.sh, mpi\_run.sh:

Scripts para correr 1000 iteraciones del programa en mono core y dual core, respectivamente, incrementando la malla desde 20 hasta 1500 en pasos de a 10.

Para correr estas pruebas se deshabilito la escritura a archivo de los programas así solamente se medía el tiempo que tardaban en calcular la difusión.

plot\_cmp\_speedup.py:

Script para graficar la salida de las pruebas anteriores.

Si se le pasan 2 archivos con la salida anterior, en el orden mono core, dual core, el script primero graficará los tiempos de las 2 ejecuciones a medida que crece el tamaño de la malla, y luego graficara la mejora en velocidad de dual core versus mono core.