Universidad Nacional Autónoma de México

Posgrado en Ciencia e Ingeniería de la Computación

Computo Científico con Alto Valor Agregado

Proyecto Final

Alumno: Rogelio Manuel Carrillo González

Profesor: Dr. Luis Miguel de la Cruz Salas

Semestre 2018-I

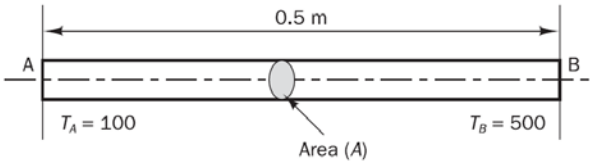
12/Junio/2018

Problema 4.1 del Libro “An Introduction to computational Fluid dynamics” de H K Versteeg and W Malalasekera

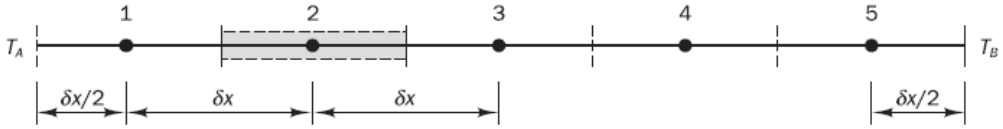
Considere el problema de la conducción de calor sin fuente en una varilla aislada cuyos extremos se mantienen a temperaturas constantes de 100 ° C y 500 ° C, respectivamente. El problema unidimensional esbozado en la Figura 4.3 se rige por:

$$\frac{\mathrm{d}}{\mathrm{d}x}\left(k\frac{\mathrm{d}T}{\mathrm{d}x}\right)=0$$

Calcule la distribución de temperatura en estado estable en la varilla. La conductividad térmica k es igual a 1000 W / m.K, el área de sección transversal A es 10 × 10-3 m2.



Podemos dividir la longitud de la varilla en 5 volúmenes de control como se muestra en la figura siguiente:



Para los nodos 2,3 y 4 la temperatura los valores al este y al oeste están disponibles como valores nodales. Por consiguiente, las ecuaciones discretas se pueden escribir fácilmente para volúmenes de control que rodean estos nodos, como:

$$\left(\frac{k_e}{\delta x_{PE}}A_e+\frac{k_w}{\delta x_{WP}}A_w\right)T_P=\left(\frac{k_w}{\delta x_{WP}}A_w\right)T_W+\left(\frac{k_e}{\delta x_{PE}}A_e\right)T_E$$

Como la conductividad térmica (k), es intervalo entre nodos (dx) y las zonas transversales (A_x) son constantes. La ecuación discreta para los puntos nodales 2, 3 y 4 es:

$$a_P T_P = a_W T_w + a_E T_E$$

Con:

$$a_P = a_W + a_E$$

$$a_W = \frac{k}{\delta x}A$$

$$a_E = \frac{k}{\delta x}A$$

Para los nodos 1 y 5 en las fronteras es necesario una atención necesaria. Para esto se ajustaron los coeficientes de los volúmenes en las fronteras. Ocupando diferencias centradas para aproximar el valor, tomando como referencia el flujo que entra por el volumen de control de la frontera. De esta manera, el proceso de discretización ha generado una ecuación para cada punto nodal (1 al 5). Los valores de cada punto se muestran en la siguiente tabla:

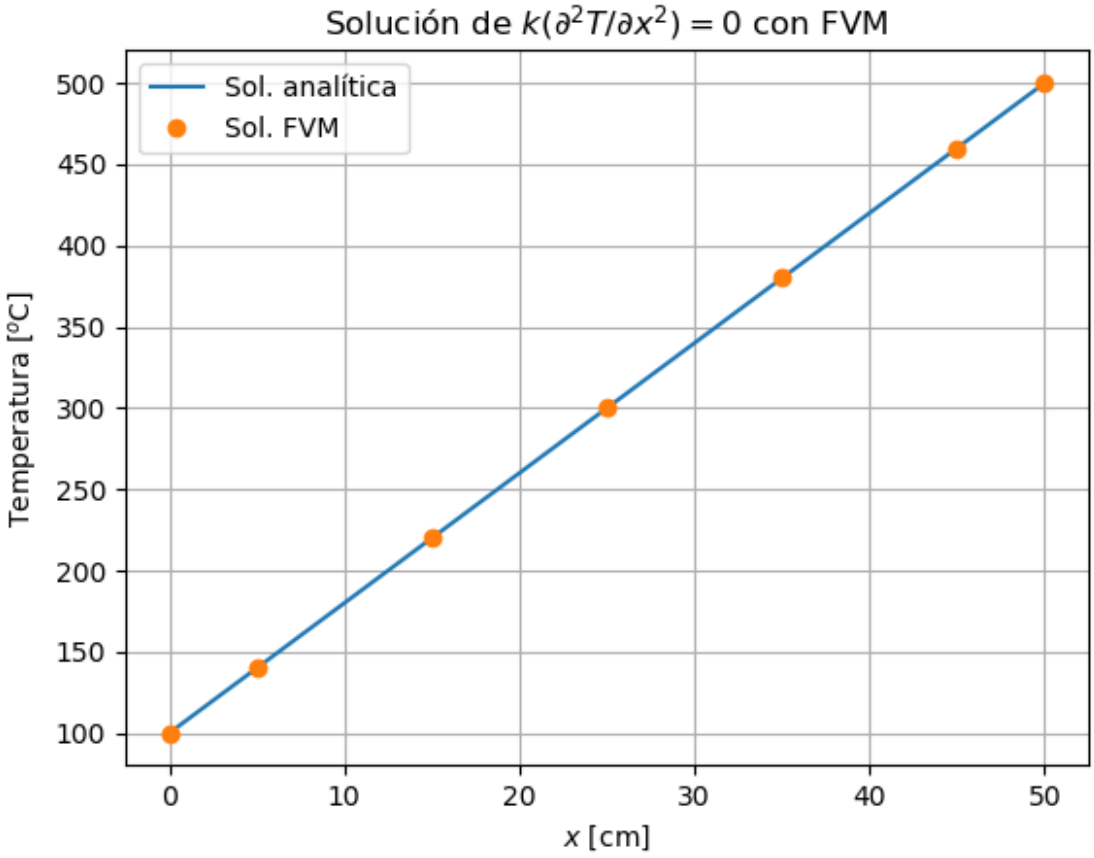
Nodo	aW	aE	Su	Sp	aP = aW + aE - Sp
1	0	100	200 TA	-200	300
2	100	100	0	0	200
3	100	100	0	0	200
4	100	100	0	0	200
5	100	0	200TB	-200	300

Obteniendo la matriz y solución siguiente:

$$\begin{bmatrix} 300 & -100 & 0 & 0 & 0 \\ -100 & 200 & -100 & 0 & 0 \\ 0 & -100 & 200 & -100 & 0 \\ 0 & 0 & -100 & 200 & -100 \\ 0 & 0 & 0 & -100 & 300 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \end{bmatrix} = \begin{bmatrix} 200T_A \\ 0 \\ 0 \\ 0 \\ 200T_B \end{bmatrix}$$

$$\begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \end{bmatrix} = \begin{bmatrix} 140 \\ 220 \\ 300 \\ 380 \\ 460 \end{bmatrix}$$

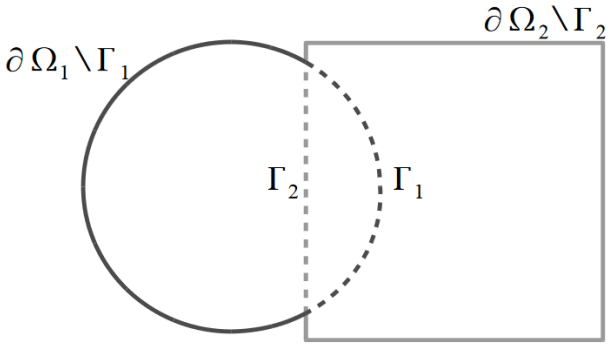
Y en la siguiente grafico observamos que la solución exacta y la solución por FVM coinciden:


Solución de $k(\partial^2 T/\partial x^2) = 0$ con FVM
Sol. analítica
Sol. FVM
Temperatura [°C]
x [cm]
100
150
200
250
300
350
400
450
500
0
10
20
30
40
50

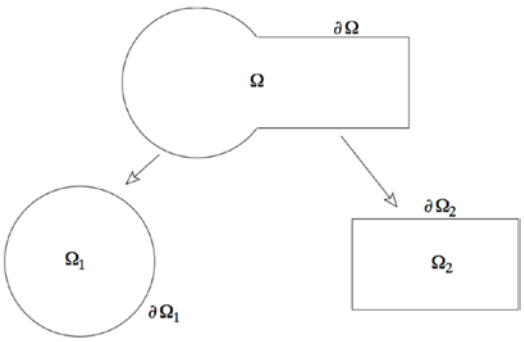
Resolviendo el problema anterior de conducción de calor en 1D implementando el algoritmo alternante de Schwarz

Para resolver el problema anterior usando el algoritmo alternante de Schwarz, fue necesario paralelizar el problema. Para esto se empleó MPI (Message Passing Interface), el cual es una definición estándar para envió de mensajes y fue diseñado para permitir el desarrollo de aplicaciones paralelas.

Primeramente, explicaremos como resolver el problema mediante el algoritmo de Schwarz. El dominio original es dividido en dos particiones Ω_1 y Ω_2 con fronteras $\delta\Omega_1$ y $\delta\Omega_2$ respectivamente:

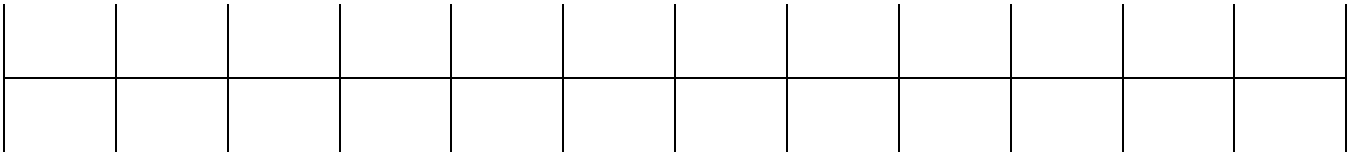


Particiones traslapadas

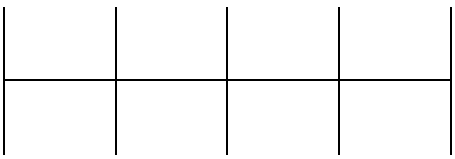


Las particiones se traslapan, así $\Omega = \Omega_1 \cup \Omega_2$. Las fronteras Γ_1 y Γ_2 son fronteras artificiales y son parte de las fronteras de Ω_1 y Ω_2 que están en el interior de Ω. Este método consiste en resolver cada partición de forma independiente, fijando condiciones tipo Dirichlet en las fronteras artificiales, primeramente, en los extremos del dominio y después fijando igualmente condiciones tipo Dirichlet en las fronteras artificiales de cada partición con los valores de la iteración previa de la partición adyacente. El método se muestra a continuación:

1. Tomando en cuenta que el problema esta en 1D, tenemos el dominio original:



2. El dominio tendrá que ser dividido entre el número deseado de subdominio. Para esto se empleó la instrucción MPI.COMM_WORLD.Create_cart



3. Dependiendo del número de procesos que el usuario desee el programa dividirá el dominio original. Esto gracias a la forma de poder ejecutar el código en la terminal de Python:

   mpiexec -n numprocs python programa.py

   Donde numprocs es equivalente al numero en que el dominio será dividido. En la sección de código:

```
grid_rows = int(1)
grid_cols = comm.size
```

   Especificamos que el numero de filas siempre será de 1, ya que será en solo una dimensión. Pero en la variable "grid_cols" se tomará el valor que ingrese el usuario al momento de ejecutar el código y se dividirá el dominio en tantos procesos se indiquen.

4. Es necesario que saber con qué dominio se está trabajando, y con ayuda de la instrucción Get_coords() podemos obtener las coordenadas del renglón y la columna en uso.

```
my_mpi_row, my_mpi_col = cartesian_communicator.Get_coords(
    cartesian_communicator.rank)
```

5. Sabiendo esto igualmente debemos saber que subdominios son los que tenemos a un lado. Y con ayuda de la instrucción. Shift() y dentro de su argumento la dirección de comunicación podremos lograr esto.

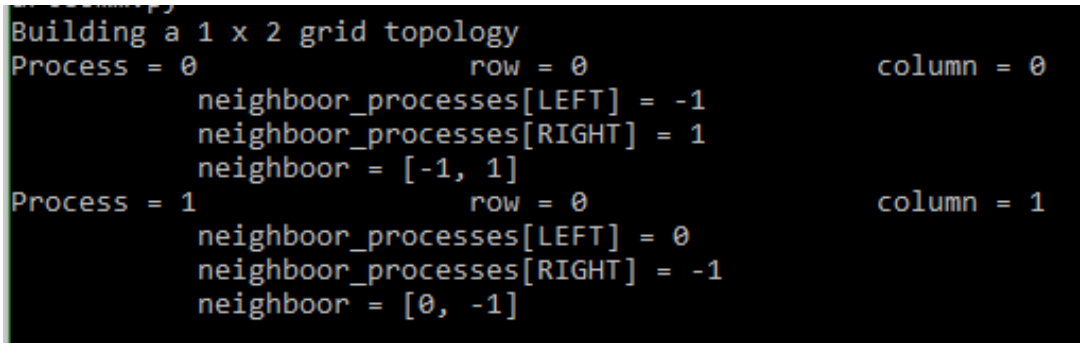
```
neighbour_processes[LEFT], neighbour_processes[RIGHT] =\
    cartesian_communicator.Shift(1, 1)
```

6. A continuación, podemos observar las diferentes salidas del programa y la comunicación entre cada subdominio

   Al ejecutar el programa con solo un proceso obtenemos el siguiente resultado:

```
Building a 1 x 1 grid topology
Process = 0                row = 0                column = 0
          neighboor_processes[LEFT] = -1
          neighboor_processes[RIGHT] = -1
          neighboor = [-1, -1]
```

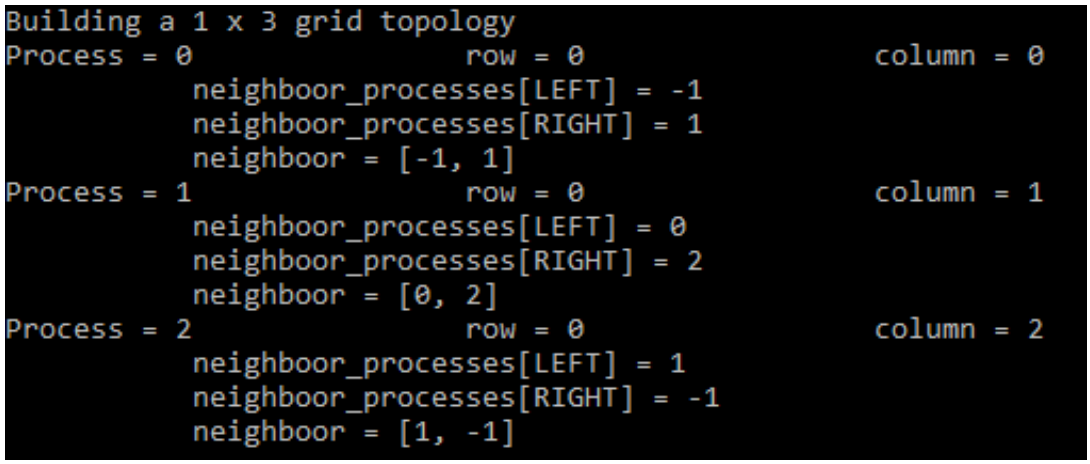
Al ejecutar el programa con dos procesos obtenemos el siguiente resultado:



Donde podemos observar que existen efectivamente dos procesos (Process 0 y Process 1). Conforme imprime cada relación de subdominio observamos las coordenadas, donde el primer proceso tiene la coordenada [0,0], esto es fila 0 y columna 0. Mientras que en el segundo proceso tenemos fila 0 y columna 1. Siempre estaremos en la fila 0 ya que como se mencionó anteriormente es un dominio de una sola dimensión.

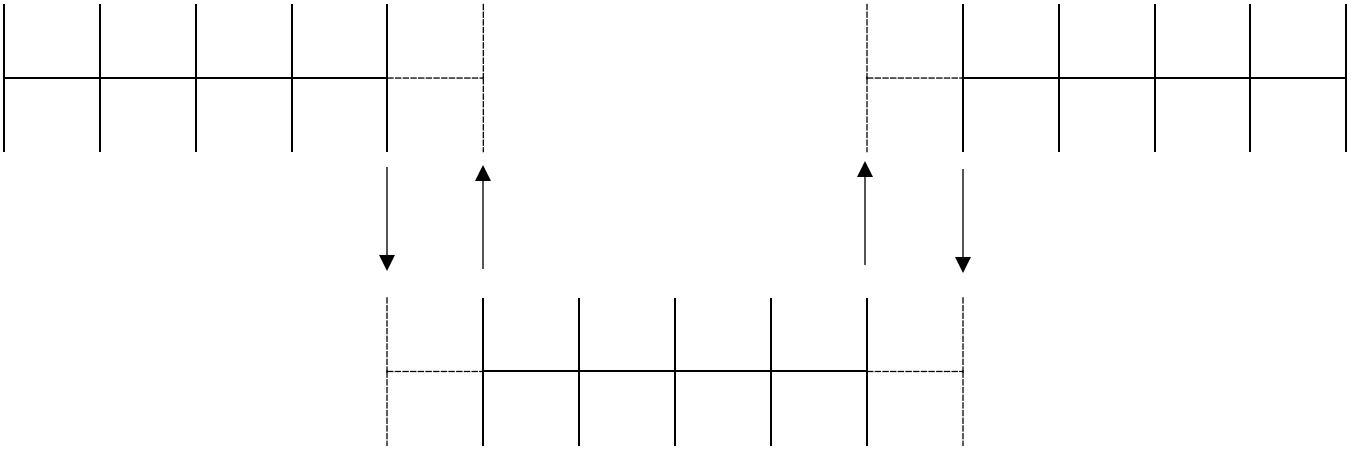
Para entender un poco más sobre la comunicación de los subdominios, basta con observar que en cada proceso se muestra con que subdominios tiene conexión cada proceso. Si se presenta un -1 en cualquier subdominio, entonces no tiene conexión con ningún otro subdominio. En este caso el subdominio en el proceso 0 tiene conexión a la derecha con el subdominio del proceso 1, mientras que a la izquierda no tiene nada. Y a su vez el subdominio en el proceso 1 tiene conexión a la izquierda con el subdominio en el proceso 0, mientras que ala derecha no tiene nada.

Para entender un poco más, a continuación, se ejemplifica con más de 2 procesos:



En este ejemplo se corrió con 3 procesos, y como se observa se está construyendo una topología de 1x3. El subdominio del proceso 0 tiene comunicación a la derecha con el subdominio del proceso 1 y este a su vez tiene comunicación con el subdominio del proceso 2 a la derecha. Mientras que el subdominio del proceso 2 tendrá comunicación con el subdominio del proceso 1 y este tendrá comunicación con el subdominio del proceso 0. De esta manera logramos saber con qué subdominios existe comunicación de cada subdominio en el que nos encontremos.

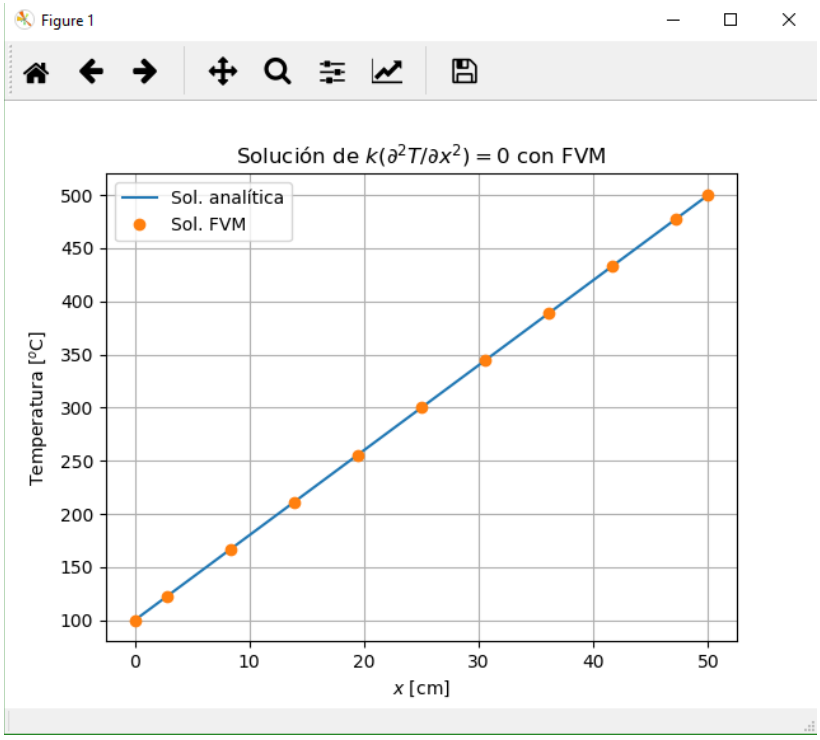
7. Al tener esto, es posible realizar cada solución independiente en cada subdominio. Para ello en cada subdominio se ajusta la malla original dependiendo del numero de subdominios. Añadiéndole fronteras fantasmas para poder asignar un valor, en este caso 0, y poder resolver el sistema en cada subdominio. Al añadir la frontera fantasma, se debe asegurar que existe un traslape para poder realizar el intercambio de información y con esto poder resolver nuevamente el sistema con los nuevos valores de frontera.



8. En un dominio 1x1 es evidente que solo existirán las fronteras originales del problema, ya que el dominio no será dividido. Al ser un dominio descompuesto de 1x2 existirán 2 fronteras fantasmas en el único traslape generado, mientras en un dominio descompuesto mayor a 1x3 existirán 2 fronteras fantasma en cada traslape que se genere.

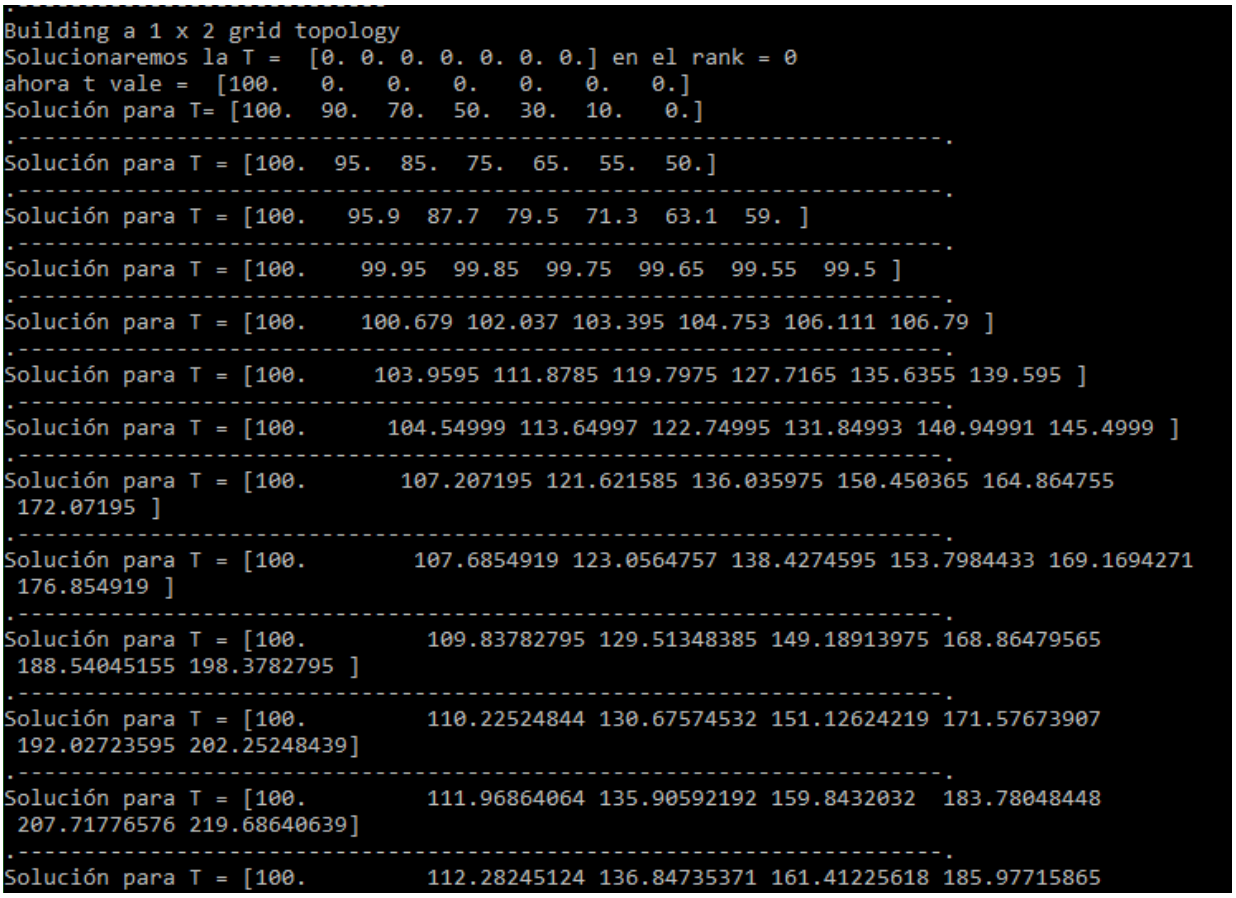
9. En la primera iteración se resuelve el problema con las fronteras dadas (extremos), se realizará el intercambio de fronteras fantasmas con el antepenúltimo valor de la solución actual y se asignara el nuevo valor a las fronteras fantasmas y se realizaran nuevamente la solución, se revisará la convergencia del problema y esto se hará n iteraciones hasta llegar al resultado donde la diferencia entre la frontera fantasma y el antepenúltimo valor de la solución actual sea mínima. Si tomamos el ejemplo 4.1 anteriormente mencionado podremos observar el proceso que sigue el programa. A continuación, se muestra:

```
Building a 1 x 1 grid topology
Solucionaremos la T =  [0. 0. 0. 0. 0. 0. 0. 0. 0. 0.] en el rank = 0
ahora t vale =  [100.   0.   0.   0.   0.   0.   0.   0.   0.   0. 500.]
Solución para T= [100.         122.22222222 166.66666667 211.11111111 255.55555556
 300.         344.44444444 388.88888889 433.33333333 477.77777778
 500.        ]
.------------------------------------------------------------------.
```



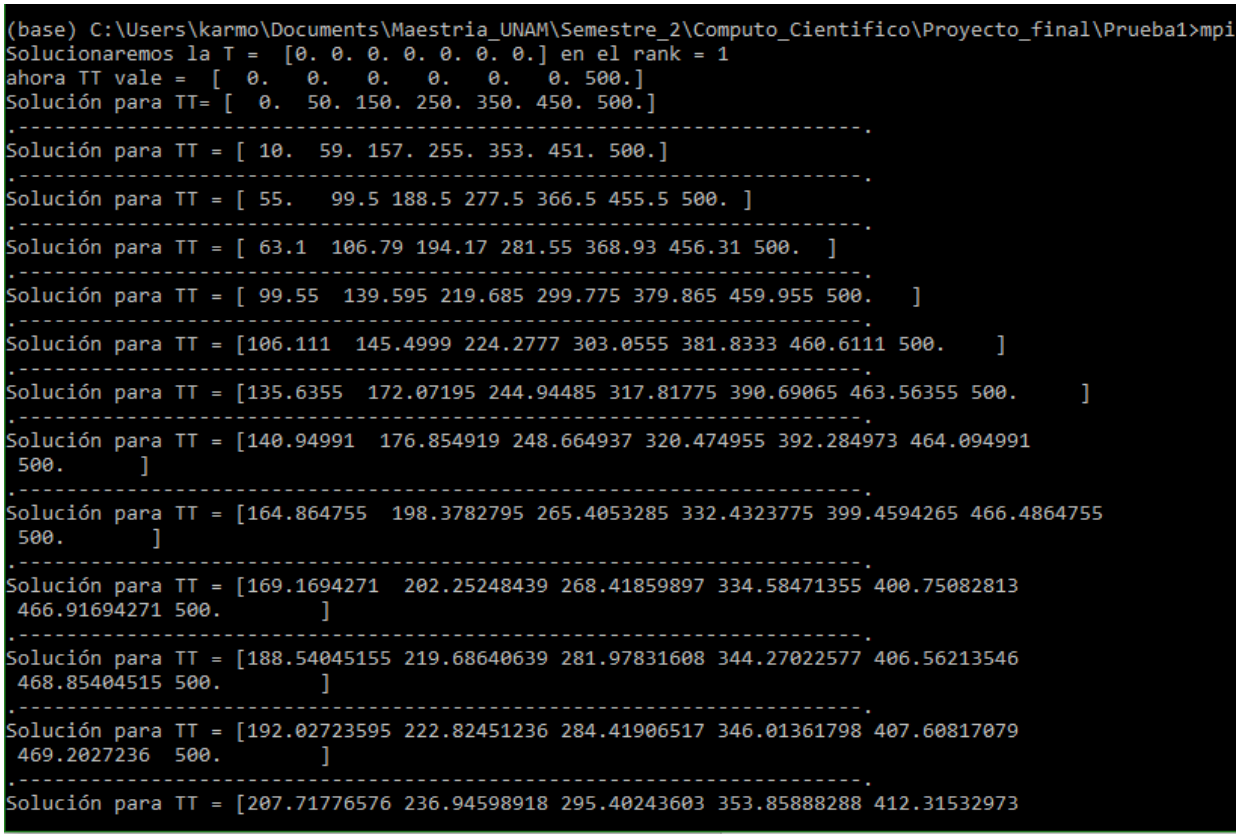
Ejemplo de topología 1x1 (no se divide el subdominio)

En esta primera imagen observamos que se asigna una topología 1 x 2 y se trabaja en el Rank = 0 (Proceso 0). Se asigna el valor de la frontera inicial de 100 (Lado izquierdo) y en la frontera fantasmas se asigna un 0, quedando T = [100, 0, 0, 0, 0, 0, 0]. Se resuelve el sistema y la primera solución para T = [100, 90, 70, 50, 30, 10, 0]. Como se puede observar en las siguientes iteraciones, el único valor que va cambiando gracias al intercambio de fronteras es la frontera derecha (frontera fantasma). Los demás valores cambian cuando el sistema se resuelve.



Topología 1x2 resolviendo en el Rank = 0

Ahora bien, en la segunda imagen se muestra la solución en Rank 1 (Proceso 1). En este caso se asigna el valor de la frontera inicial de 500 (Lado derecho) y en la frontera fantasmas se asigna un 0, quedando T = [0, 0, 0, 0, 0, 0, 500]. Se resuelve el sistema y la primera solución para T = [0, 50, 150, 250, 350, 450, 500]. Como se puede observar en las siguientes iteraciones, el único valor que va cambiando gracias al intercambio de fronteras es la frontera izquierda (frontera fantasma). Los demás valores cambian cuando el sistema se resuelve.



Topología 1x2 resolviendo en Rank = 1

Con esto anteriormente mostrado se puede observar el intercambio de valor anterior a la frontera fantasmas como nuevo valor de la frontera del subdominio contiguo. Como se muestra a continuación:

Iteración 1:

T = [100, 90, 70, 50, 30, 10, 0] TT = [0, 50, 150, 250, 350, 450, 500]

Iteración 2:

T = [100, 90, 70, 50, 30, 10, 50] TT = [10, 50, 150, 250, 350, 450, 500]

T = [100, 95, 85, 75, 65, 55, 50] TT = [10, 59, 157, 255, 353, 451, 500]

Iteración 3:

T = [100, 95, 85, 75, 65, 55, 59] TT = [55, 59, 157, 255, 353, 451, 500]

T = [100, 95.9, 87.7, 79.5, 71.3, 63.1, 59] TT = [55, 99.5, 188.5, 277.5, 366.5, 456.31, 500]

Y así sucesivamente. Dándonos como resultado:

```
Solucion =  [100.0, 121.05263157894733, 163.15789473684197, 205.26315789473665, 247.36842105263133,
```

```
310.52631578947336, 352.63157894736815, 394.73684210526295, 436.8421052631578, 478.9473684210526, 500.0]
```