Facultad de Matemática, Astronomía, Física y Computación (FaMAF-UNC)

Curso de Física Computacional 2022-Guía N°02

Problema 1

Ecuación de Calor

Resolvemos, mediante diferentes algoritmos, la ecuación del calor

$$\frac{\partial T}{\partial t} = \frac{\kappa_T}{C\rho}\nabla^2[T(x,t)] \tag{1}$$

para el caso particular de una barra de aluminio de longitud $L = 1[m]$ y diámetro ω, alineada a lo largo del eje x; que está aislada en sus lados, pero no en sus extremos. Inicialmente la barra se encuentra a $100[°C]$, y se colocan sus extremos a $0[°C]$. El calor fluye sólo por los extremos. La conductividad térmica (κ_T), el calor específico (C) y la densidad (ρ) del aluminio son

$$\kappa_T = 237\left[\frac{W}{mK}\right];\ C = 900\left[\frac{J}{kgK}\right];\ \rho = 2700\left[\frac{kg}{m^3}\right] \tag{2}$$

Para ello:

a) Adimensionalizar la ecuación para llevarla a la forma

$$\frac{\partial[T(\tilde{x},\tilde{t})]}{\partial\tilde{t}} = \frac{\partial^2[T(\tilde{x},\tilde{t})]}{\partial^2\tilde{x}} \tag{3}$$

b) Escriba el código que resuelva la ecuación utilizando el método (de diferencias finitas) explícito de Euler "hacia adelante" (forward Euler). Utilice 100 divisiones en el eje x, y prevea hacer miles de pasos en t (es buena costumbre estimar tamaños de archivos antes de calcular (discuta en informe)! (Almacene sólo los valores de t que necesita, no todos). use $\Delta t = 0.3[s]$ (adimensionalizar y verificar que cumpla la condición de estabilidad). Escriba cada aproximadamente 300 pasos temporales los valores de la temperatura en la barra, y haga un gráfico de superficie mostrando $T(x,t)$ versus (x,t).

c) Controle que el programa dé una distribución de temperatura que varía suavemente a lo largo de la barra, y que está de acuerdo con las condiciones de contorno.

d) Verifique que el programa dé una distribución de temperatura que varía suavemente a lo largo de la barra, y que está de acuerdo con las condiciones de contorno.

e) Compare la solución analítica:

$$T(x,t) = \sum_{n=1,3,\ldots}^{\infty} \frac{4T_0}{n\pi}\sin(k_n x)\exp\left[\frac{-(k_n)^2\kappa_T t}{C\rho}\right] \tag{4}$$

con la solución numérica. Para ello, grafique $T(x,t)$, para $t_1 = 180[s]$ y $t_2 = 1800[s]$. Utilice ahora 10 divisiones en el eje x (y elija Δt de acuerdo a la condición de estabilidad) y compare los resultados obtenidos para t_1 y t_2.

f) Haga un gráfico de superficie mostrando $T(x,t)$ versus (x,t), y un gráfico de contornos mostrando las isotermas.

g) Repita los incisos (b)-(f) para los algoritmos: implícito y Crank-Nicolson. Presente figuras donde se comparen, y discutan: si las diferencias son apreciables al ojo, sus tiempos de CPU, sus errores, complejidad de métodos, etc.

h) Compare los tiempos de CPU para llegar a $t_2 = 1800[s]$ con cada método y con una precisión de 10^{-4} (error relativo).

Resultados y Discusiones

Inciso a)

Análisis dimensional

Adimensionalización, partimos de la ecuación unidimensional

$$\frac{\partial T}{\partial t} = D\frac{\partial^2[T(x,t)]}{\partial^2 x};\ D = \frac{\kappa_T}{C\rho} \tag{5}$$

notemos que las unidades de D (en unidades generales de tiempo (t), temperatura (T), y espacio (L) y masa (M)) las podremos averiguar conociendo las unidades de κ_T, C y ρ, entonces de (2)

$$[\kappa_T] = \left[\frac{potencia}{distancia \cdot temperatura}\right] \equiv \left[\frac{potencia}{L \cdot T}\right]$$

pero las unidades de potencia se pueden expresar de forma fundamental como

$$[potencia] = \left[\frac{trabajo}{tiempo}\right] = \left[\frac{fuerza \cdot distancia}{tiempo}\right] = \left[\frac{masa \cdot aceleración \cdot distancia}{tiempo}\right]$$

$$\Rightarrow [potencia] = \left[\frac{masa \cdot (distancia)^2}{(tiempo)^3}\right] \equiv \left[\frac{M \cdot L^2}{t^3}\right] \therefore [\kappa_T] \equiv \left[\frac{M \cdot L}{t^3 \cdot T}\right]$$

y por otro lado tendremos,

$$[C] = \left[\frac{trabajo}{masa \cdot temperatura}\right] \equiv \left[\frac{L^2}{T \cdot t^2}\right] \wedge [\rho] = \left[\frac{masa}{volumen}\right] \equiv \left[\frac{M}{L^3}\right]$$

luego

$$[D] = \frac{[\kappa_T]}{[C] \cdot [\rho]} \equiv \left[\frac{L^2}{t}\right] \equiv \left[\frac{(distancia)^2}{tiempo}\right] \tag{6}$$

y si ahora definimos las siguientes coordenadas adimensionales de espacio $(\tilde{x})$ y tiempo $(\tilde{t})$

$$\alpha\tilde{x} = x \wedge \beta\tilde{t} = t \tag{7}$$

donde α y β son parámetros característicos a definir para realizar las adimensionalidades. Si por un lado, adimensionalizamos x con la longitud característica de la barra $L \Rightarrow \alpha = L$ entonces deberá cumplirse que de (5), (7) (y usando regla de la cadena)

$$\frac{\partial}{\partial t} = \frac{\partial}{\partial \tilde{t}}\underbrace{\frac{\partial \tilde{t}}{\partial \tilde{t}}}_{=1/\beta} = \frac{1}{\beta}\frac{\partial}{\partial \tilde{t}};\ \frac{\partial^2}{\partial x^2} = \left[\frac{\partial^2}{\partial \tilde{x}^2}\underbrace{\left(\frac{\partial \tilde{x}}{\partial x}\right)^2}_{=1/\alpha} + \frac{\partial}{\partial \tilde{x}}\underbrace{\frac{\partial^2 \tilde{x}}{\partial x^2}}_{=0}\right] = \frac{1}{\alpha^2}\frac{\partial^2}{\partial \tilde{x}^2}$$

$$\Rightarrow \frac{\partial T}{\partial \tilde{t}} = \frac{\beta D}{\alpha^2}\frac{\partial^2[T(\tilde{x},\tilde{t})]}{\partial^2 \tilde{x}} \tag{8}$$

entonces, para obtener la ecuación (3) deberá cumplirse que

$$\frac{\beta D}{\alpha^2} = 1 \Rightarrow \beta = \frac{\alpha^2}{D} = \frac{L^2}{D} \tag{9}$$

es decir, encontramos el tiempo característico del sistema para adimensionalizar el la coordenada temporal y donde se verifican las unidades pues, $[\beta] = \left[L^2\right] \cdot \left[{}^{t}/_{L^2}\right] = [t] \equiv [tiempo]$.

Inciso b)

Estimación de tamaño de archivos

Se llevo a cabo una estimación de tamaño de archivos de la siguiente manera, por ejemplo, si tenemos las siguientes configuraciones en nuestro código

```
1  program estimated_size
2      use module_precision
3      implicit none
4      real(dp) :: valor1=4._dp*atan(1.0_dp),valor2=4._dp*atan(1.0_dp)
5      integer(sp) :: i,istat
6      open(10,file='estimated_size.dat',status='replace',action='write',iostat=istat)
7      write(*,*) 'Input/Output file. istat10 = ',istat
8      20 format(E9.2,x,E9.2)
9      do i=1,200; write(10,20) valor1,valor2; end do
10 end program estimated_size
11
12 ! gfortran -o estimated_size.o module_precision.f90 estimated_size.f90 && ./estimated_size.o && rm
   *.mod *.o
```

donde (dp) designa doble precisión (flotante de $8[\text{bytes}] \equiv 64[\text{bits}]$) que utiliza 26 dígitos para su representación. En primera instancia, si no imprimimos los datos con formato tamaño del archivo sería $\text{tamaño} = (\#\text{filas}) \cdot \left(\frac{\#\text{valores}}{\text{fila}}\right) \cdot \left(\frac{\text{digitos}}{\text{valor}}\right)[\text{bytes}]$, pero debemos tener en cuenta también que el sistema operativo guarda en memoria el indice de fila y columna del archivo de datos y por cada indice consume un byte (en nuestro caso tenemos 200 filas y 2 columnas y habrá que considerar 202 bytes adicionales) entonces,

$$\text{tamaño} = (200 \cdot 2 \cdot 26 + 202)[\text{bytes}] = 10602[\text{bytes}]$$

notemos que estuvimos muy cerca del tamaño real que fue de

```
-rw-rw-r-- 1 mendez mendez 10600 abr 24 09:51 estimated_size.dat
```

Ahora bien, esta es una estimación grosera pues el formato no nos imprimirá todos los dígitos de doble precisión sino que sólo utilizará 9 dígitos para guardar todo el número (contando signos, puntos, exponente, decimales, etc.), además, como en el formato tenemos un espacio en blanco por renglón el sistema operativo también gasta un byte para su representación (esto, sin embargo, no se sabe muy bien por qué razón los espacios en blanco utilizan memoria quizás se debe a los datos basura de los sistemas operativos y también podría deberse al editor de texto en si, comprimiendo o no datos para su almacenamiento, por ejemplo, en este link se discute acerca de esto y la respuesta podría estar en que si se almacenan los datos en cadenas de caracteres, cualquiera de ellos, incluso los espacios en blanco ocupan memoria) entonces debemos estimar el tamaño como

$$\text{tamaño} = (200 \cdot 2 \cdot 9 + 203 + 200)[\text{bytes}] = 4003[\text{bytes}]$$

donde el segundo término de 203 bytes se debe al gasto de almacenar indices de filas y columnas (200 filas y 3 columnas) y el tercer término de 200 bytes se debe a el gasto adicional de almacenar un espacio en blanco. El tamaño real fue de

```
-rw-rw-r-- 1 mendez mendez  4000 abr 24 10:22 estimated_size.dat
```

Dirigiéndonos ahora a las simulaciones numéricas mostramos a continuación los resultados obtenidos

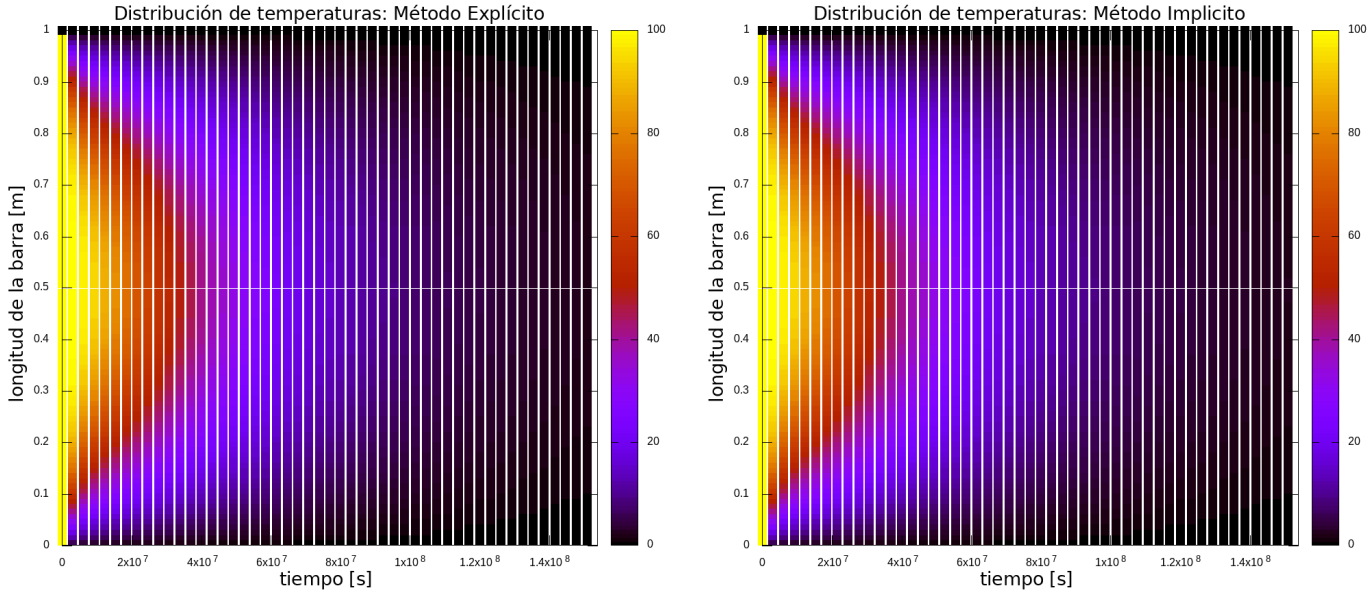


Figure 1:

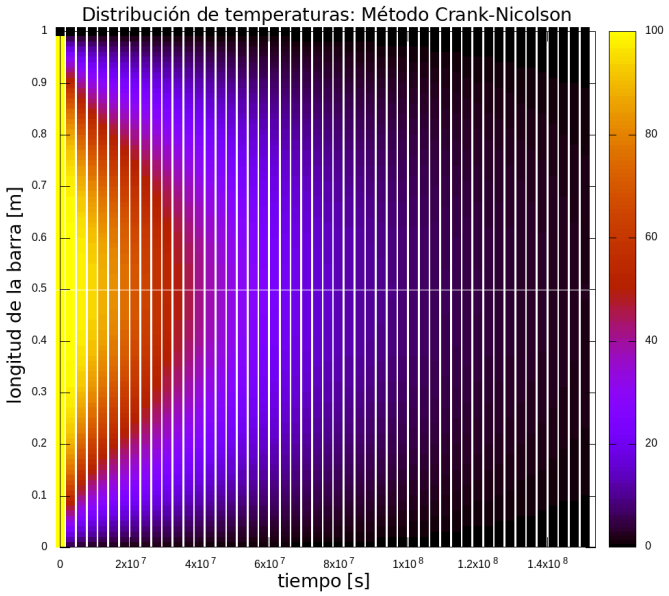


Figure 2:

Las gráficas anteriores muestran la distribución de temperatura (en colores) para cada posición de la barra (cada paso espacial simulado en el eje vertical) y cada 300 pasos de evolución temporal (eje horizontal). A simple vista no se aprecian diferencias significativas, lo que evidencia que los resultados para cada uno de los métodos son similares.

inciso e,f,g)

Control de estabilidad método explícito

Hubo mucha variación de los resultados dependiendo del parámetro de estabilidad elegido, por ejemplo, si bien teóricamente es necesario que este parámetro sea menor a $^1/_2$ para que el método explícito sea estable, si se elegía un valor cercano a $^1/_2$ la distribución de temperaturas era nula (como si la relajación fuese instantánea) lo cual sabemos que físicamente no ocurre, y se obtenía un incremento considerable del tiempo de CPU. Y, cuanto menor sea el parámetro la distribución de temperaturas será más gradual, sin embargo, tampoco debe ser muy pequeño pues no se tendrá

relajación alguna. Los resultados se muestran a continuación:

```
real(dp), parameter     :: alpha=0.45_dp (verificando estabilidad por muy poco)
mendez@mendez-notebook:~/my_repositories$ cat ../results/result_02_aprox.dat
 0.00E+00  0.00E+00
 0.11E+00  0.15-322
 0.22E+00  0.30-322
 0.33E+00  0.40-322
 0.44E+00  0.44-322
 0.56E+00  0.44-322
 0.67E+00  0.40-322
 0.78E+00  0.30-322
 0.89E+00  0.15-322
 0.10E+01  0.00E+00
```

```
real(dp), parameter     :: alpha=1.E-20_dp (muy por debajo del error de la maquina)
mendez@mendez-notebook:~/my_repositories$ cat ../results/result_02_aprox.dat
 0.00E+00  0.00E+00
 0.11E+00  0.10E+03
 0.22E+00  0.10E+03
 0.33E+00  0.10E+03
 0.44E+00  0.10E+03
 0.56E+00  0.10E+03
 0.67E+00  0.10E+03
 0.78E+00  0.10E+03
 0.89E+00  0.10E+03
 0.10E+01  0.00E+00
```

```
real(dp), parameter     :: alpha=1.E-7_dp (por encima del error de la maquina)
mendez@mendez-notebook:~/my_repositories$ cat ../results/result_02_aprox.dat
 0.00E+00  0.00E+00
 0.11E+00  0.84E+02
 0.22E+00  0.99E+02
 0.33E+00  0.10E+03
 0.44E+00  0.10E+03
 0.56E+00  0.10E+03
 0.67E+00  0.10E+03
 0.78E+00  0.99E+02
 0.89E+00  0.84E+02
 0.10E+01  0.00E+00
```

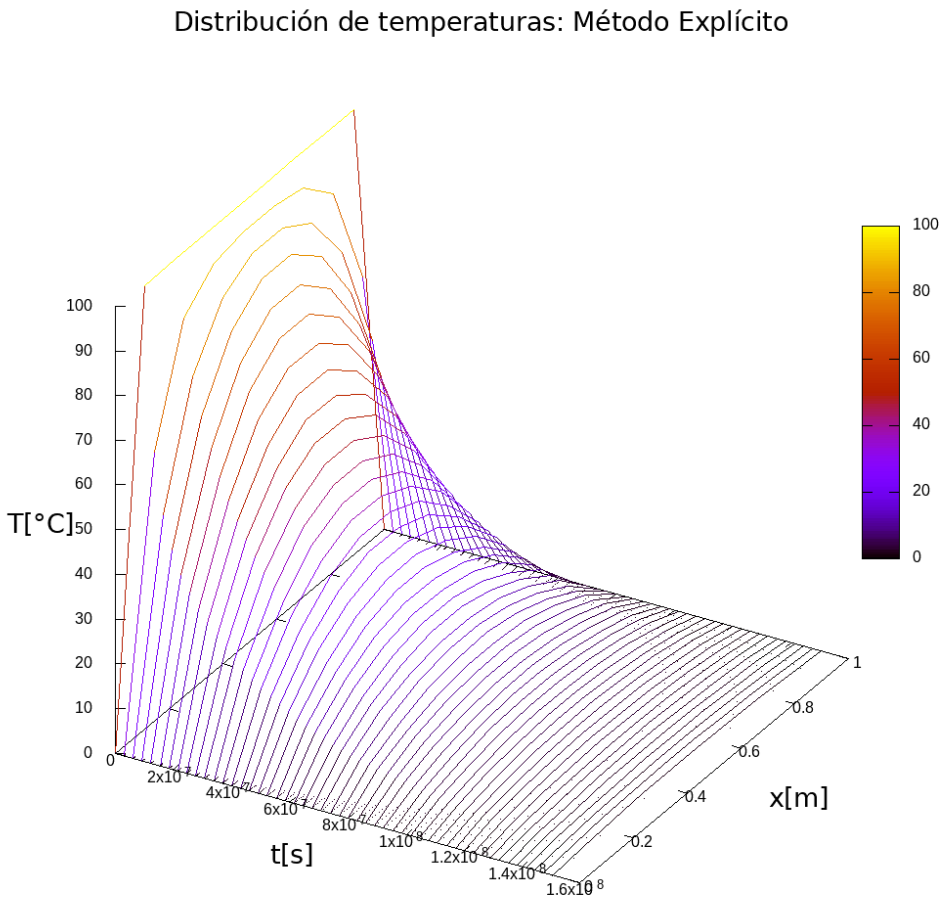
También, se probó con valores mayores a $^1/_2$ produciéndose errores numéricos (propagación de NaN) lo cual evidencia alguna división por cero o algún error aritmético similar.

```
real(dp), parameter     :: alpha=4.5_dp (no verificando estabilidad)
mendez@mendez-notebook:~/my_repositories$ cat ../results/result_02_aprox.dat
 0.00E+00  0.00E+00
 0.11E+00       NaN
 0.22E+00       NaN
 0.33E+00       NaN
 0.44E+00       NaN
 0.56E+00       NaN
 0.67E+00       NaN
 0.78E+00       NaN
 0.89E+00       NaN
 0.10E+01  0.00E+00
```

Para todos los casos el α utilizado fue de 10^{-7}. Es necesario aclarar que, el análisis anterior no es estrictamente correcto, pues los errores de los métodos dependen de los valores de α pero específicamente de los valores de Δx y Δt de forma acoplada, es decir, en los casos anteriores el

Δx siempre se mantiene fijo y el valor de Δt se ve modificado por el valor de α, esto no es un método correcto de variar α pues el error numérico puede verse modificado groseramente ya que no se acompaña una discretización espacial con una discretización temporal (estamos haciendo elementos finitos más esbeltos en la dimensión temporal en vez de hacerlos más esbeltos en ambas dimensiones y obtener elementos finitos pequeños en su totalidad).

El gráfico de superficie obtenido se muestra en la siguiente figura



La figura anterior sólo se hizo para el método explícito, teniendo en cuenta los resultados anteriores que nos permitían asegurar que los resultados no variaban apreciablemente. En la figura se observa inicialmente la temperatura con los extremos de la barra a $0[°C]$ y el resto del cuerpo de la barra a $100[°C]$, luego de un tiempo se ve la termalización de la barra, es decir, se produce la relajación de la barra a una temperatura de equilibrio de $0[°C]$.

Luego, graficando la distribución de temperaturas para los tiempos específicos de $180[s]$ y $1800[s]$, para cada uno de los métodos, tendremos:

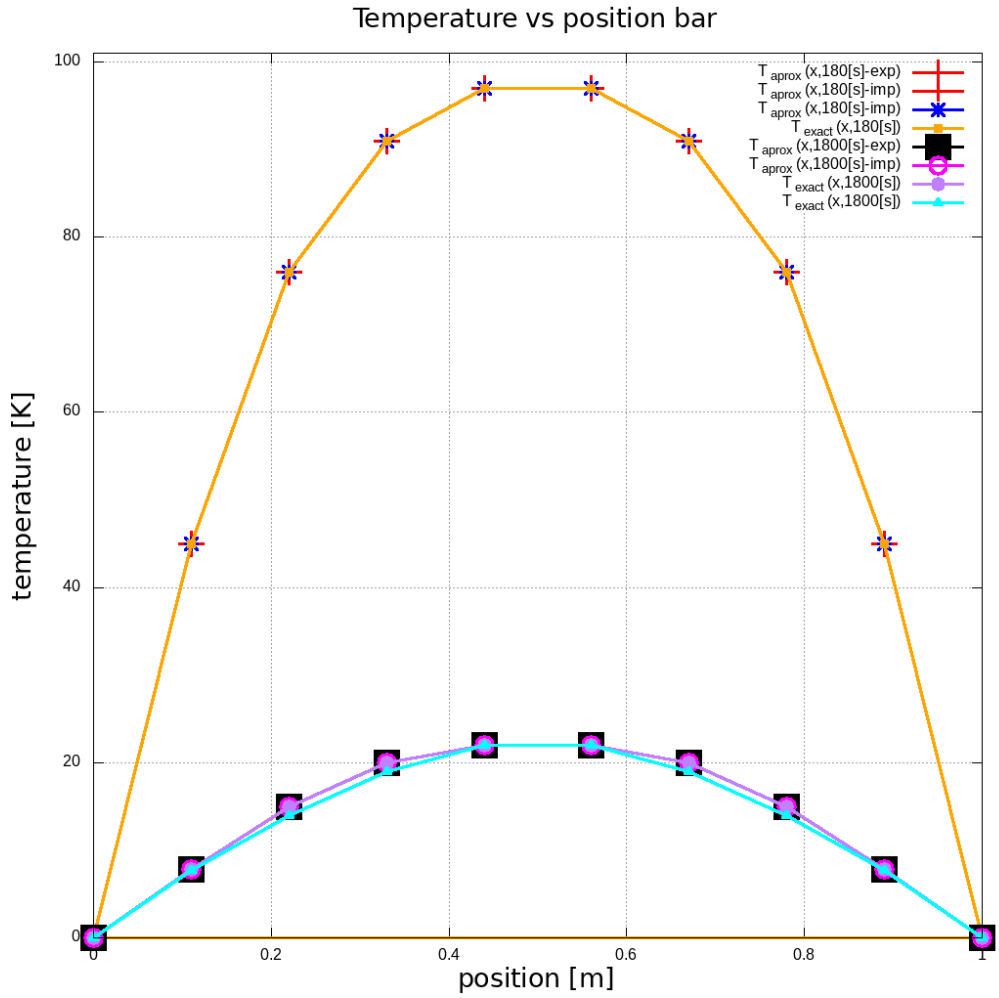


Figure 3:

En esta gráfica podemos ver que los tres métodos arrojan resultados muy próximos a la solución exacta, además, para tiempos mayores vemos que existen diferencias más apreciables respecto al valor exacto.

Inciso h)

Los tiempos fueron medidos para los distintos métodos utilizando la herramienta perf versión 5.13.19, con este comando se corrieron cada uno de los códigos una cinco veces y se pudo medir el valor medio y tolerancia del tiempo de CPU, como así también la cantidad de instrucciones por ciclo, referencias a memoria cache y cache-misses (cantidad de veces que los registros no entran completamente en memoria caché haciendo más dificultoso el procesamiento e incrementando el tiempo de cpu). Para todos los casos se relajó el sistema hasta los $1800[s]$, con 142200000 pasos temporales, y se midió el tiempo total de ejecución del programa, es decir, primero se compilaron los tres códigos (correspondientes a cada uno de los métodos), se generaron los códigos objeto y luego se aplicó el comando perf a estos códigos binarios para hacer las mediciones de performance.

Los resultados obtenidos fueron,

```
# ++++++++++++++++++++++++++++++++
# METODO EXPLÍCITO
# ++++++++++++++++++++++++++++++++
perf stat -e cycles,instructions,cache-references,cache-misses -r 5 ./heateq_comparison_01.o

 Performance counter stats for './heateq_comparison_01.o' (5 runs):

    30.997.662.307      cycles                                                ( +-  3,34% )
    95.295.666.771      instructions              #    3,07  insn per cycle           ( +-  0,00% )
         2.834.670      cache-references                                      ( +-  2,80% )
         1.012.570      cache-misses              #   35,721 % of all cache refs      ( +-  3,01% )

           11,455 +- 0,385 seconds time elapsed  ( +-  3,36% )
```

```
# ++++++++++++++++++++++++++++++++
# METODO IMPLÍCITO
# ++++++++++++++++++++++++++++++++
perf stat -e cycles,instructions,cache-references,cache-misses -r 5 ./heateq_comparison_02.o

 Performance counter stats for './heateq_comparison_02.o' (5 runs):

    91.829.376.951      cycles                                                ( +-  1,08% )
   253.610.980.072      instructions              #    2,76  insn per cycle           ( +-  0,00% )
        15.309.764      cache-references                                      ( +-  9,38% )
         6.619.732      cache-misses              #   43,239 % of all cache refs      ( +- 11,09% )

           35,429 +- 0,872 seconds time elapsed  ( +-  2,46% )
# ++++++++++++++++++++++++++++++++
# METODO CRANK-NICOLSON
# ++++++++++++++++++++++++++++++++
perf stat -e cycles,instructions,cache-references,cache-misses -r 5 ./heateq_comparison_02.o

 Performance counter stats for './heateq_comparison_03.o' (5 runs):

   204.268.612.300      cycles                                                ( +-  0,92% )
   540.648.910.045      instructions              #    2,65  insn per cycle           ( +-  0,00% )
        59.518.787      cache-references                                      ( +- 12,88% )
        17.188.257      cache-misses              #   28,879 % of all cache refs      ( +- 12,84% )

           78,652 +- 0,789 seconds time elapsed  ( +-  1,00% )
```

Claramente, el método que más tiempo de CPU obtuvo fue el de Crank-Nicolson (C-N), seguido por el método implícito y el más rápido fue el método explícito. Sin embargo, el método de(C-N obtuvo un 33% menos de cache-misses que el método implícito y un 19% menos que el método explícito. La complejidad de los códigos es, en términos generales, la misma aunque el aumento significativo de tiempo de cómputo quizás se deba a que los métodos de C-N e implícito trabajan con matrices mientras que el método explícito no, es más, el aumento de tiempo de CPU del método de C-N respecto del método implícito quizás podría deberse al agregado de operaciones de tipo multiplicación de matrices y vectores, ya que, si bien en ambos métodos se trabaja con matrices bandeadas y con una subrutina que trata con estos sistemas de ecuaciones específicos, el método de C-N realiza más operaciones matemáticas con una segunda matriz bandeada.

Códigos

Repositorio GitHub

https://github.com/mendzmartin/fiscomp2022.git

Repositorio GitHub del problema

https://github.com/mendzmartin/fiscomp2022/tree/main/lab02/prob01

Programas principales

https://github.com/mendzmartin/fiscomp2022/blob/main/lab02/prob01/code/heateq_explicit_method.f90

https://github.com/mendzmartin/fiscomp2022/blob/main/lab02/prob01/code/heateq_implicit_method.f90

https://github.com/mendzmartin/fiscomp2022/blob/main/lab02/prob01/code/heateq_crank_nicolson_method.f90

https://github.com/mendzmartin/fiscomp2022/blob/main/lab02/prob01/code/heateq_comparison_01.f90

https://github.com/mendzmartin/fiscomp2022/blob/main/lab02/prob01/code/heateq_comparison_02.f90

https://github.com/mendzmartin/fiscomp2022/blob/main/lab02/prob01/code/heateq_comparison_03.f90

Bash script para correr los códigos

https://github.com/mendzmartin/fiscomp2022/blob/main/lab02/prob01/code/script_run.sh

Referencias

Chapman, S. (2007). *Fortran 95/2003 for Scientists & Engineers*. McGraw-Hill Education. https://books.google.com/books/about/Fortran_95_2003_for_Scientists_Engineers.html?hl=&id=c8cLDQEACAAJ

Chapra, S. C., & Canale, R. P. (2007). *Métodos numéricos para ingenieros*. https://books.google.com/books/about/M%C3%A9todos_num%C3%A9ricos_para_ingenieros.html?hl=&id=hoH0MAAACAAJ

Landau, R. H., Mejía, M. J. P., PÃ¡ez, M. J., Kowallik, H., & Jansen, H. (1997). *Computational Physics*. Wiley-VCH. https://books.google.com/books/about/Computational_Physics.html?hl=&id=MJ3vAAAAMAAJ