Trabajo Práctico I

"No creo que a él le gustara eso"

Métodos Numéricos
Segundo Cuatrimestre de 2014

Integrante	LU	Correo electrónico
Fosco, Martin Esteban	449/13	mfosco2005@yahoo.com.ar
Minces Müller, Javier Nicolás	231/13	javijavi1994@gmail.com
Chibana, Christian Ezequiel	586/13	christian.chiba93@gmail.com

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Ciudad Universitaria - (Pabellón I/Planta Baja)
Intendente Güiraldes 2160 - C1428EGA
Ciudad Autónoma de Buenos Aires - Rep. Argentina
Tel/Fax: (54 11) 4576-3359
http://www.fcen.uba.ar

Resumen

En este trabajo se implementa una solución al problema de encontrar una manera de determinar qué sanguijuelas pegadas a un parabrisas son peligrosas y eliminarlas. Para esto, se plantea un sistema de ecuaciones que permita hallar la temperatura del parabrisas en cada punto, con una precisión determinada por la granularidad, tomando como incógnita a cada punto del parabrisas.
Para resolver dicho sistema de ecuaciones se recurre a técnicas matemáticas basadas en métodos de resolución de sistemas, y se divide el programa en distintos módulos con funciones específicas al parabrisas y a la matriz del sistema de ecuaciones para facilitar la comprensión de la implementación hecha.

Índice

1. Introducción Teórica

El problema a resolver por este trabajo consiste en modelar la distribución del calor en el parabrisas rectangular de una nave en el momento de equilibrio. Este parabrisas recibe frío en sus bordes y calor a partir de $sanguijuelas$ circulares.

Este problema puede expresarse como un sistema de ecuaciones, con una ecuación por cada punto cuya temperatura se busca determinar. El objetivo de este trabajo, por lo tanto, es usar métodos numéricos para la resolución de sistemas de ecuaciones, observándolos como una matriz multiplicada por un vector incógnita; de este producto se obtiene un vector resultado, utilizando el lenguaje C++.

Un sistema de ecuaciones puede representarse como un producto de matrices. Si x es el vector incógnita, resolver el sistema se reduce a encontrar un x tal que $Ax = b$.

Dos sistemas de ecuaciones se consideran equivalentes si y solo si tienen el mismo conjunto de soluciones. Por lo visto en clase, la suma de un múltiplo de una fila a otra y el intercambio de filas no afectan a la solución del sistema, por lo que se pueden encontrar sistemas equivalentes mediante el uso de esta propiedad.

Para resolver el sistema de ecuaciones del problema (expresado en forma de una matriz) se utiliza el método conocido como 'Eliminación Gaussiana', que recurre a la suma de un múltiplo de una fila a otra e intercambio entre filas de la misma matriz (con un multiplicador que sirva para reducir los valores por debajo de la diagonal a 0). De esta forma, el algoritmo nos permite triangular una matriz de forma relativamente eficiente (es decir llegar de una matriz común a una triangular superior equivalente).

Una vez triangulada la matriz, puede hallarse el valor de la última posición del vector incógnita, luego usarla para obtener la anteúltima, y así sucesivamente hasta completar todo el vector.

Otro objetivo del trabajo es aprovechar la estructura banda de una matriz. Una matriz banda es una matriz que, en cada fila, tiene ceros en todos los puntos salvo los que están a una distancia determinada de la diagonal. Esto permite evitar almacenar muchos ceros.

2. Desarrollo

Nuestro trabajo apunta a resolver el problema dividiéndolo en tres partes.

2.1. Representación y determinación de temperatura

La **Primera parte** consiste en encontrar la tempratura de cada punto del parabrisas representándolo como una matriz. Algunos puntos son conocidos, ya que el parabrisas tiene bordes con una temperatura constante, y fuentes de calor (sanguijuelas) con radio y temperatura a determinar por los parámetros r y T_s.

Como dato, se tiene que cada punto del parabrisas satisface la ecuación de Laplace:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0$$

Para poder modelar el parabrisas se toma una discretización, con granularidad variable. De esta forma, el parabrisas queda representado con una matriz de $a/h+1$ filas y $b/h+1$ columnas, siendo a el ancho en metros del parabrisas original, b su alto y h la granularidad elegida. Al ancho de esta matriz lo llamaremos m y a su alto n.

Aplicando el modelo de aproximación por diferencias finitas se puede ver que la temperatura de cada punto desconocido puede expresarse como el promedio de cada uno de los puntos adyacentes. Es decir, la temperatura de cada punto del parabrisas se puede expresar como una ecuación lineal: [1]

$$\begin{cases} \frac{1}{4}x_{i+1,j} + \frac{1}{4}x_{i,j+1} + \frac{1}{4}x_{i-1,j} + \frac{1}{4}x_{i,j-1} - x_{i,j} = 0 & \text{si el punto es desconocido} \\ x_{i,j} = -100 & \text{si el punto es borde} \\ x_{i,j} = T_s & \text{si el punto es sanguijuela} \end{cases}$$

Considerando todos los puntos desconocidos de la matriz, se obtiene entonces un sistema de ecuaciones lineales de $mn \times mn$.

Nuestro problema entonces se reduce a plantear una resolución. Por ejemplo, sea un parabrisas discretizado como una matriz de 3x3, sin sanguijuelas:

$$\begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix}$$

El sistema de ecuaciones queda planteado de esta forma:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{4} & 0 & \frac{1}{4} & -1 & \frac{1}{4} & 0 & \frac{1}{4} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_{11} \\ x_{12} \\ x_{13} \\ x_{21} \\ x_{22} \\ x_{23} \\ x_{31} \\ x_{32} \\ x_{33} \end{bmatrix} = \begin{bmatrix} -100 \\ -100 \\ -100 \\ -100 \\ 0 \\ -100 \\ -100 \\ -100 \\ -100 \end{bmatrix}$$

Para almacenar la matriz se usaron dos métodos. El primero es un vector de filas, donde cada fila es un vector con todos sus valores. Para el segundo se aprovecha la estructura banda de la matriz para

[1] Consideramos que un punto 'es sanguijuela"si la distancia al centro de una sanguijuela es menor o igual al radio de las sanguijuelas. La distancia entre el punto $p = (p_x, p_y)$ y la sanguijuela con centro en $s = (s_x, s_y)$ se calcula como $\|p - s\|_2 = \sqrt{(p_x - s_x)^2 + (p_y - s_y)2}$. Si el punto está en el borde, no es sanguijuela, pero si el centro de la sanguijuela está en el borde, sigue modificando los puntos de alrededor.

almacenar, de cada fila, solo los elementos que forman parte de la banda. Al triangular, la matriz mantiene su ancho de banda.

Para aprovechar la estructura banda se pensó en guardar únicamente un vector de valores booleanos que permitiera saber para cada celda de la matriz si su valor era conocido o desconocido, ya que solo con ese dato era posible saber el valor de toda la fila de la otra matriz. Sin embargo, esta estructura no puede mantenerse al triangular, ya que el valor de la fila toma valores que no dependen únicamente de la celda que representa.

Finalmente, para implementar la matriz banda, no se creó otra estructura, sino que se modificó la estructura matriz ya existente, de forma que incluyera un booleano que indicara si era una matriz banda, y un entero para el ancho de banda. De esta forma, todas las operaciones de la matriz podían utilizarse para la matriz banda, con la única excepción de la función que permite definir un valor en una celda y la que busca el valor de una celda determinada, en las que hubo que separar en dos casos.

Al implementar esto, hubo que modificar el algoritmo que crea la matriz a partir de un parabrisas para que solo definiera los elementos de la banda. También hubo que modificar indirectamente la eliminación gaussiana, ya que al restar filas solo debían tenerse en cuenta las celdas que formaran parte de la banda. El ancho de banda es $2 \times m + 1$.

Se trabaja con tolerancia, forzando un 0 cuando el valor de una celda es menor a 10^{-10} en módulo.

Es particularmente necesario para el problema calcular la temperatura del $puntocrítico$, que se define como el centro del parabrisas. Sin embargo, este punto puede no pertenecer a la discretización: si a/h o b/h no es divisible por 2, entonces no hay una celda que se corresponda con el punto crítico. Una posibilidad para calcular su temperatura es, entonces, hacer un promedio de los cuatro puntos que lo rodean.

2.2. Análisis de complejidades y comportamientos esperados

La **Segunda parte** se trata de experimentar con estos algoritmos desarrollados y comparar el tiempo y espacio ocupados por cada implementación. Se esperaba que la banda se desempeñara de forma más eficiente, ya que realiza menos operaciones, midiendo el orden de complejidad asintótico (O) en peor caso de cada parte de la implementación.

Operación	Costo matriz común	Costo matriz banda
Cargar los datos	O(k)	O(k)
Generar la matriz	O(m^2n^2k)	O(m^2nk)
Triangular la matriz	O(m^3n^3)	O(m^3n)
Generar el vector de resultados	O(m^2n^2)	O(m^2n^2)
Imprimir el vector de resultados	O(mn)	O(mn)

El costo de generar la matriz incluye la cantidad de sanguijuelas k porque es necesario determinar, para cada punto, si está en una sanguijuela.

Se tiene que lo que determina el costo total es la triangulación de la matriz. Por eso, el algoritmo banda es temporalmente más eficiente que el algoritmo de eliminación gaussiana común. Este es el algoritmo de triangulación de una matriz común:

```
EliminacionGaussiana(Matriz m):
for i=(1:columnas)                                                    
    for j=(i+1:filas)                                                 
        if (|A_ii| ≥ tol ∧|A_ji| ≥ tol)                               
            mult = A_ji / A_ii                                        
            restarFilas(A_j,A_i*mult)                          O(mn)  
        endif                                                         
    endfor                                          O(mn) · O(m)=O(m²n²)
endfor                                              O(m²n²) · O(m)=O(m³n³)
```

Y este es el algoritmo de triangulación de una matriz banda:

```
EliminacionGaussiana(Matriz m):
for i=(1:columnas)
    for j=(i+1:min{filas,i+anchoBanda})
        if (|A_ii| ≥tol ∧|A_ji| ≥ tol)
            mult = A_ji / A_ii
            restarFilas(A_j,A_i*mult)                                   O(m)
        endif
    endfor                                                  O(m) · O(m)=O(m^2)
endfor                                                      O(mn) · O(m^2)=O(m^3n)
```

La operación que resta filas se ejecuta hasta $((mn)^2)/2$ veces en el caso de la matriz común, ya que puede ser necesario ejecutarla una vez para generar un 0 en cada celda por debajo de la diagonal. En el caso de la matriz banda, se ejecuta hasta $m + 1$ veces por cada columna, es decir, $m^2n + mn$ veces.

Esta operación tiene como complejidad $m \times n$ en el caso de la matriz común, y $2m+1$ en el caso de la matriz banda, ya que solo debe restar las celdas que forman parte de la banda, cuya cantidad corresponde con el ancho de banda. La complejidad total de la triangulación es, entonces, O($(mn)^3$) en el caso de la matriz común y O(m^3n) en el caso de la banda.

Considerando $m \sim n >> k$

Operación	Costo matriz común	Costo matriz banda
Cargar los datos	O(k)	O(k)
Generar la matriz	O(m^4)	O(m^4)
Triangular la matriz	O(m^6)	O(m^4)
Generar el vector de resultados	O(m^4)	O(m^4)
Imprimir el vector de resultados	O(m^2)	O(m^2)
Tiempo total de ejecución	O(m^6)	O(m^4)

Al reducirse la granularidad, debería aumentar el tiempo de ejecución. Esto es porque reducir a la mitad la granularidad, por ejemplo, implica duplicar tanto el alto como el ancho de la matriz que representa la discretización del parabrisas.

Si se considera $m = a/h + 1$, es posible reescribir los órdenes de complejidad de los algoritmos en función de a, y h. El algoritmo de triangulación de la matriz común cuesta O($\frac{a^6}{h^6}$) y el de la matriz banda, O($\frac{a^4}{h^4}$).

La cantidad de sanguijuelas no debería afectar el tiempo de ejecución, salvo que sea muy grande, comparable con m o n.

2.3. Algoritmo de eliminación de sanguijuelas

La **Tercera parte** del problema consiste en encontrar un algoritmo que permita elegir qué sanguijuelas eliminar para que la temperatura del punto crítico del vidrio se mantenga por debajo de 235º, evitando eliminar más sanguijuelas de las estrictamente necesarias. El algoritmo ultilizado fue heurístico: se elimina siempre la sanguijuela más cercana al centro. Elegimos hacerlo de esta manera porque la sanguijuela que más influye en la temperatura de un punto es la más cercana a él, siendo el borde fijo. En caso de que dos sanguijuelas sean equidistantes al centro, se elimina la última que se agregó.

Una opción más provechosa en términos del uso de láseres sería usar un algoritmo que antes de decidir si eliminar una sanguijuela simulara el estado del punto crítico posterior a su eliminación, haciendo esta simulación con todas las sanguijuelas (o con todas dentro un determinado radio); luego podría elegir una sanguijuela a eliminar observando en cuál estado posterior de la simulación el punto crítico baja más.

Un ejemplo en el que este último algoritmo se desempeña mejor sería un caso en el que hubiera diez sanguijuelas muy cerca una de la otra y del punto crítico y una no tan cerca, pero en el que eliminar esta sanguijuela solitaria bastaría para reducir la temperatura del punto crítico a un valor óptimo. El algoritmo que elegimos eliminaría las 10 sanguijuelas más cercanas, y recalcularía todo el sistema cada

vez. El algoritmo más eficiente en uso de lásers, en cambio, notaría que eliminando una sola sanguijuela puede alcanzar un resultado aceptable, y al ver que eliminando una sola de las sangujuelas más cercanas no conseguiría nada, elegiría entonces a la más lejana, matando una sola sanguijuela.

Sin embargo, nos inclinamos por el algoritmo heurístico debido a que consideramos que el costo del algoritmo más inteligente tanto en complejidad de programación, temporal y espacial sería mucho mayor, debido a que debería correr once veces la simulación para decidirse por una sanguijuela, mientras que el de fuerza bruta sólo una. Ademas, este es un caso particular en el que, por ejemplo, el radio de las sanguijuelas sea lo suficientemente pequeño como para que las cercanas al centro no afecten ningún punto de la grilla, mientras que la más alejada, por hallarse más cerca su centro de un punto de la grilla, modifique la temperatura del centro.

Decidimos implementar el algoritmo de forma que imprimiera en un archivo separado, cuyo nombre se pasa por parámetro, una lista de las coordenadas de las sanguijuelas eliminadas, en orden, junto con la temperatura en el punto crítico después de cada eliminación. Dado que después de cada eliminación era necesario recalcular todo el sistema, usamos para esto el algoritmo de matriz banda.

3. Resultados

Los tests utilizados fueron los siguientes, cambiando la granularidad:

test	ancho	alto	radio	T_s	#sang
test1	100	100	10	500	6
test2	100	100	10	500	1
test3	120	120	10	500	6
test4	120	120	10	500	1

Se toma 100 y 120 como ancho y alto para el parabrisas porque deben ser números que permitan variar la granularidad. Es necesario recordar que la granularidad debe dividir a ambos valores. Se varía la cantidad de saguijuelas para observar si este parámetro afecta el tiempo de ejecución. No se varía su temperatura, ya que no debería afectarla, mientras que variar el radio debería tener un efecto similar a variar la cantidad de saguijuelas.

Uno de los tests tiene una de las sanguijuelas sobre el punto crítico. Según la granularidad que se tome, podría no reflejarse esto en el resultado. Además, la intención es que un cambio en el tamaño de parabrisas, al cambiar la posición del punto crítico, cambie su temperatura.

Los siguentes son los mapas de temperaturas para cada test, generados con el archivo $graphsol.py$, provisto por la cátedra, con distintas granularidades: 1, 2, 2.5, 4, 5, 10 para los primeros dos tests y 1, 2, 3, 4, 5, 6 para los últimos dos.

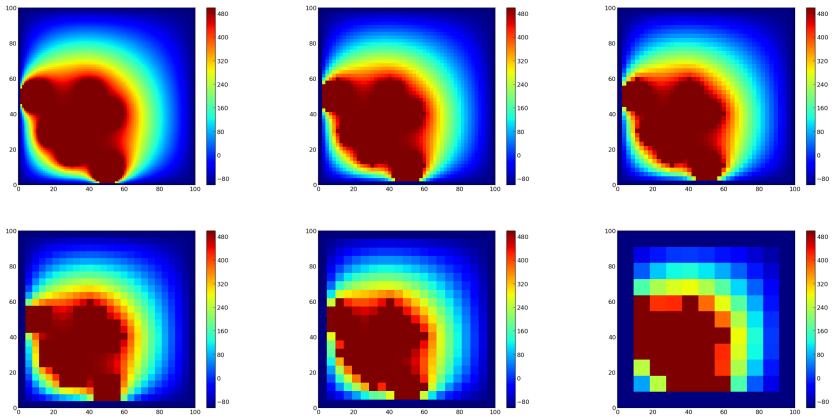


Figura 1: Test 1

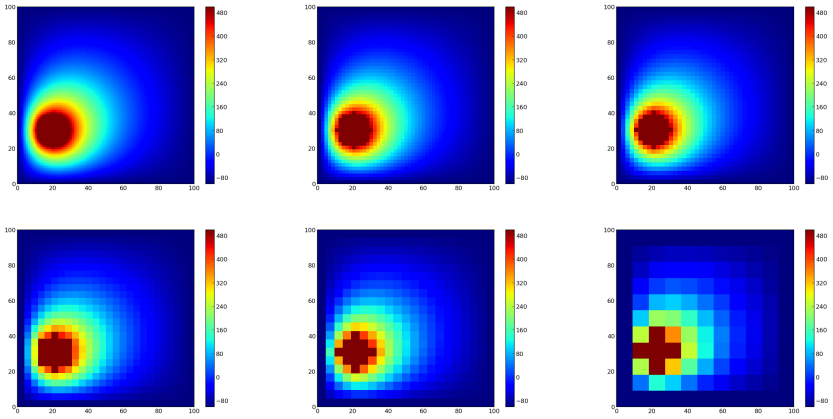
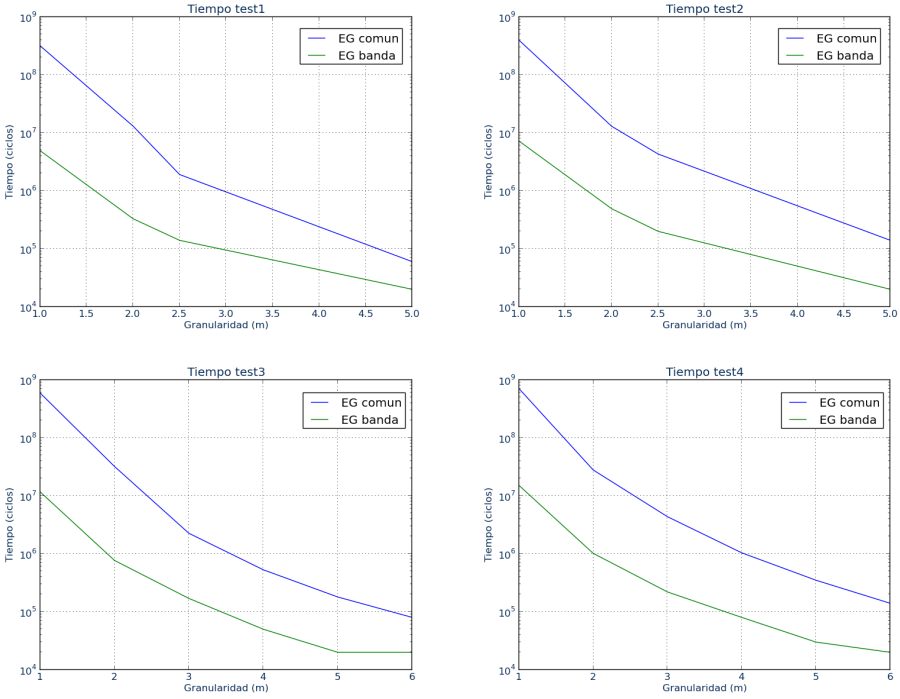


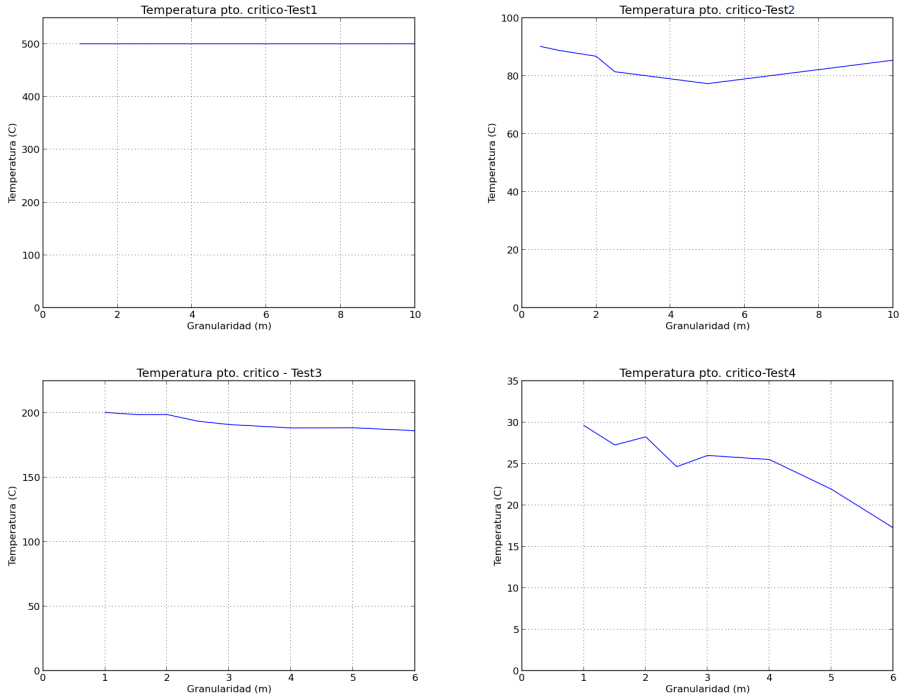
Figura 2: Test 2

Los mapas de temperatura para los test 3 y 4 son muy similares a los anteriores, ya que el único parámetro que cambia es el ancho.

A continuación se presenta el tiempo de ejecución según la granularidad. Las mediciones fueron efectuadas con instrucciones de la librería $ctime$. Realizar varias mediciones produce una diferencia de hasta 0.5s, por lo que se hizo un promedio de varias ejecuciones; sin embargo, no se puede garantizar que los resultados para los casos de mayor granularidad sean confiables. Notar que la escala es logarítmica:

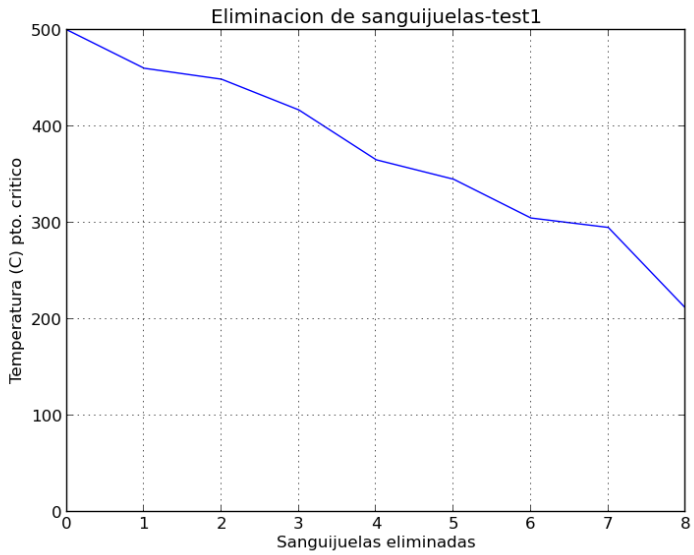


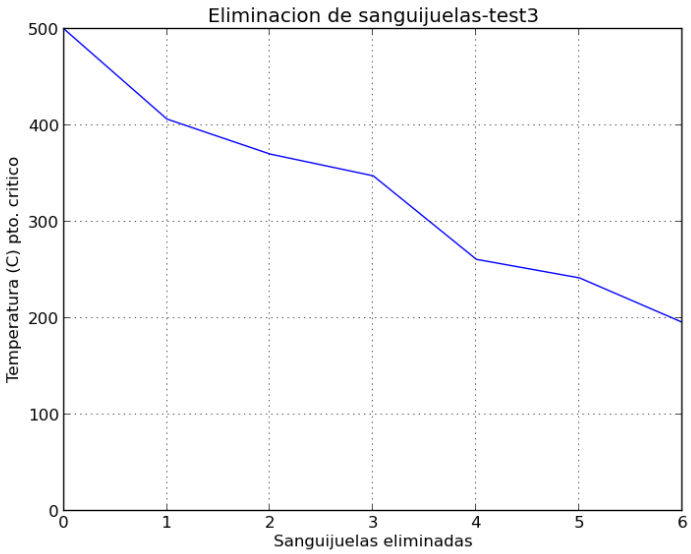
Lo que sigue es la temperatura del punto medio según la granularidad.



Para el algoritmo de eliminación de sanguijuelas se tomaron los tests 1 y 3, se fijó la granularidad en 2 y se agregaron más sanguijuelas, de forma que el algoritmo tuviera que repetirse un número mayor de veces.

test	ancho	alto	h	radio	temp	#sang
test1	100	100	2	10	500	15
test3	120	120	2	10	500	15





En cuanto al tiempo de ejecución, se obtuvieron estos resultados que presentamos junto a los del test correspondiente, sin eliminación de sanguijuelas:

test	tiempo sin elim	tiempo con elim	# elim	$\frac{\text{tiempo con elim}}{\text{tiempo sin elim}}$
test1	0,33	5,71	8	17,30
test3	1,02	8,87	6	8,69

4. Discusión

Teniendo en cuenta que la matriz cuadrada es de $(m \times n) \times (m \times n)$ y que la matriz banda solo debe conservar $(m \times n) \times (2 \times m + 1)$ valores, la matriz banda ocupa menos memoria cuando $n > 2$. La diferencia escala con n. En el caso asintótico, ocupa $O(m^2n^2)$ la matriz cuadrada y $O(m^2n)$ la matriz banda.

Como se puede ver en los gráficos de tiempo de ejecución, el algoritmo de eliminación gaussiana banda fue más eficiente que el común. Sin embargo, la diferencia escala con a, no con a^2, por lo que la aproximación no fue muy buena.

La cantidad de sanguijuelas afectó al tiempo total de ejecución. En algunos casos lo aumentó y en otros lo redujo. Si bien al analizar el orden de complejidad en peor caso parecería que solo debería aumentarlo, hay que tener en cuenta que agregar una sanguijuela implica cambiar algunos valores de la matriz de 1/4 a 0. Esto reduce el tiempo de triangulación, que es el que domina la función.

Como esperábamos, reducir la granularidad aumentó el tiempo de ejecución. Se puede ver que una reducción de la granularidad del 50 % llevó a un aumento del tiempo de ejecución del orden de 2^4 en la matriz banda y hasta 2^5 en la matriz común, aproximadamente; es decir, lo esperado en el primer caso y la mitad de lo esperado en el segundo:

Matriz banda

test	h=1	h=2	$\frac{h=1}{h=2}$
test1	4,94	0,33	14,96
test2	7,24	0,49	14,77
test3	11,76	0,77	15,27
test4	15,2	1,02	14,90

Matriz común

test	h=1	h=2	$\frac{h=1}{h=2}$
test1	318,48	10,43	30,52
test2	398,78	12,95	30,79
test3	596,97	32,19	18,54
test4	711,23	27,96	25,42

Reducir la granularidad llevó a resultados más exactos, como se puede ver en los mapas de temperatura. Con los gráficos de temperatura del punto crítico se puede ver que el valor de ese punto parece ir tendiendo hacia un determinado límite; si fuera posible tomar un límite con granularidad tendiente a 0 entonces se podría encontrar el valor del punto crítico en el caso continuo, en otras palabras, cuanta más precisa la granularidad, más confiable y cercano al real es el resultado.

En un caso la temperatura se mantuvo contanste en T_s. Esto se debe a que el punto crítico queda bajo una sanguijuela, algo que no varió al cambiar la granularidad. Creemos que la pérdida de precisión al aumentar el valor de h en los otros casos provoca tanto ascensos como descensos en el valor de la temperatura; sin embargo, en la mayoría de los casos se ve un descenso, ya que los bordes ocupan una proporción mayor de la matriz.

En cuanto al algoritmo de eliminación, se pude ver que cada sanguijuela eliminada redujo la temperatura del punto crítico. Esto no garantiza que el algoritmo sea óptimo, ya que es evidente que si hubiera dos sanguijuelas en el mismo punto, esto no se daría. Pero dejando ese caso a un lado, se puede decir que el algoritmo es bueno, ya que no desperdicia disparos en sanguijuelas que no afectan la temperatura del punto crítico.

En cuanto al tiempo de ejecución de este algoritmo, no coincide exactamente con el producto de la cantidad de sanguijuelas y el tiempo que se tarda en volver a calcular el sistema, si bien se aproxima. Esto se debe a que, como ya se dijo, recalcular el sistema con menos sanguijuelas no necesariamente cuesta lo mismo, ya que hay más posiciones de la matriz cuya temperatura no se conoce.

El cálculo de qué sanguijuela eliminar podría optimizarse si se guardara un vector con la distancia al centro de cada sanguijuela, de forma que no habría que volver a calcular la de todas cada vez que se busca una a eliminar. Esta solución, sin embargo, ocuparía más espacio de memoria y no tendría un gran impacto en el tiempo total.

5. Conclusiones

Podemos, al final de nuestro trabajo, sacar varias conclusiones respecto al enfoque que le dimos a este problema (y el que suponemos que era esperado que le diéramos).

Primero, si se aumenta la granularidad, se pierde exactitud. Como consecuencia, el valor del punto crítico puede alejarse considerablemente del real, de forma que podría ser mayor a 235 cuando el real no lo es. Además, si el diámetro de las sanguijuelas es menor a la granularidad, una sanguijuela podría no cubrir ningún punto de la grilla, quedando el mapa de temperaturas como si ésta no estuviera, dando como resultado una temperatura menor en el punto crítico a la real. Es decir que es conveniente elegir una granularidad menor al diámetro de las sanguijuelas. Sin embargo, esto no siempre es posible, ya que el radio podría ser 0.

Teniendo esto en cuenta, junto al impacto de la granularidad en el tiempo de ejecución, podemos decir que obtuvimos los mejores resultados con granularidad alrededor de 1/50 del lado, con un parabrisas cuadrado. Por supuesto, hay factores que modifican esto, como cuán precisa debe ser la medición del valor del punto crítico.

Podemos sacar otras conclusiones al ver las diferencias que surgen de implementar una matriz como una banda o como una común.

Principalmente, es destacable la importancia que tiene el no guardar más datos de los necesarios (o de guardarlos de manera inteligente) para cubrir todas las funciones esperadas de nuestra estructura de manera satisfactoria, ya que es evidente que si bien ambas implementaciones cumplen con lo necesario para considerarse correctas, la eficiencia, tanto a nivel temporal como espacial, de la matriz banda sobrepasa por mucho a aquella de la matriz común. Habiendo dicho esto, uno podría pensar que la matriz banda sería deseable por sobre la común en todos los casos, pero podemos hacernos la pregunta de si no hay algo que estamos perdiendo al obtener esta mayor eficiencia.
Salta entonces a la vista que la matriz banda se diferencia de la común en que está mucho más especializada, es decir que al aprovechar el hecho de que trabaja con una matriz con forma de banda, al asumir que los puntos alejados del centro a partir de una determinada distancia valen 0 estamos justamente reduciendo la cantidad de casos a los que la matriz banda es aplicable. La matriz banda trabaja mejor con todos los casos que puede, pero es aplicable a menos casos, mientras que la común cambia velocidad y memoria por una versatilidad mucho mayor.
En otras palabras, si quisiéramos usar la banda para alguna otra utilidad, distinta del propósito de este trabajo, tendríamos primero que comprobar si es aplicable a nuestros experimentos, mientras que con la común no habría requisitos para representar en forma de matriz lo que sea que necesitemos.

Luego de haber sacado conclusiones de comparar nuestras implementaciones en términos de versatilidad, velocidad y tiempo, queda también ver qué podemos sacar de la comparación entre complejidad de implementaciones; sin embargo no encontramos mucha diferencia entre nuestra implementación de matriz banda y de común, ya que nos limitamos a reducir el ancho de la matriz efectiva que guardábamos, incluso comparten funciones (con ligeras variaciones).

6. Apéndice A

Laboratorio de Métodos Numéricos - Segundo Cuatrimestre 2014
Trabajo Práctico Número 1: *"No creo que a él le gustara eso"*

Introducción

El afamado Capitán Guybrush Threepwood se encuentra nuevamente en problemas. El parabrisas de su nave El Pepino Marino está siendo atacado simultáneamente por varios dispositivos hostiles vulgarmente conocidos como *sanguijuelas mutantes*. Estos artefactos se adhieren a los parabrisas de las naves y llevan a cabo su ataque aplicando altas temperaturas sobre la superficie, con el objetivo de debilitar la resistencia del mismo y permitir así un ataque más mortífero. Cada sanguijuela consta de una *sopapa de ataque* circular, que se adhiere al parabrisas y aplica una temperatura constante sobre todos los puntos del parabrisas en contacto con la sopapa.

Para contrarrestar estas acciones hostiles, el Capitán Guybrush Threepwood solamente cuenta con el sistema de refrigeración de la nave, que puede aplicar una temperatura constante de -100^{o}C a los bordes del parabrisas. El manual del usuario de la nave dice que si el punto central del parabrisas alcanza una temperatura de 235^{o}C, el parabrisas no resiste la temperatura y se destruye. Llamamos a este punto el *punto crítico* del parabrisas.

En caso de que el sistema de refrigeración no sea suficiente para salvar el punto crítico, nuestro Capitán Guybrush Threepwood tiene todavía una posibilidad adicional: puede destruir algunas de las sanguijuelas, pero debe eliminar la menor cantidad posible de sanguijuelas dado que cada eliminación consume energía que puede ser vital para la concreción de la misión. La situación es desesperante, y nuestro héroe debe tomar una rápida determinación: debe decidir qué sanguijuelas eliminar de modo tal que el parabrisas resista hasta alcanzar la base más cercana.

El modelo

Suponemos que el parabrisas es una placa rectangular de a metros de ancho y b metros de altura. Llamemos $T(x, y)$ a la temperatura en el punto dado por las coordenadas (x, y). En el estado estacionario, esta temperatura satisface la ecuación del calor:

$$\frac{\partial^2 T(x,y)}{\partial x^2} + \frac{\partial^2 T(x,y)}{\partial y^2} = 0. \tag{1}$$

La temperatura constante en los bordes queda definida por la siguiente ecuación:

$$T(x, y) = -100^o\text{C} \quad \text{si } x = 0, a \text{ ó } y = 0, b. \tag{2}$$

De forma análoga es posible fijar la temperatura en aquellos puntos cubiertos por una sanguijuela, considerando T_s a la temperatura ejercida por las mismas.

El problema en derivadas parciales dado por la primera ecuación con las condiciones de contorno presentadas recientemente, permite encontrar la función T de temperatura en el parabrisas, en función de los datos mencionados en esta sección.

Para estimar la temperatura computacionalmente, consideramos la siguiente discretización del parabrisas: sea $h \in \mathbb{R}$ la granularidad de la discretización, de forma tal que $a = m \times h$ y $b = n \times h$, con $n, m \in \mathbb{N}$, obteniendo así una grilla de $(n+1) \times (m+1)$ puntos. Luego, para $i = 0, 1, \ldots, n$ y $j = 0, 1, \ldots, m$, llamemos $t_{ij} = T(x_j, y_i)$ al valor (desconocido) de la función T en el punto $(x_j, y_i) = (ih, jh)$, donde el punto $(0, 0)$ se corresponde con el extremo inferior izquierdo del parabrisas. La aproximación por *diferencias finitas* de la ecuación del calor afirma que:

$$t_{ij} \;=\; \frac{t_{i-1,j} + t_{i+1,j} + t_{i,j-1} + t_{i,j+1}}{4}. \tag{3}$$

Es decir, la temperatura de cada punto de la grilla debe ser igual al promedio de las temperaturas de sus puntos vecinos en la grilla. Adicionalmente, conocemos la temperatura en los bordes, y los datos del problema permiten conocer la temperatura en los puntos que están en contacto con las sanguijuelas.

Enunciado

Se debe implementar un programa en `C` o `C++` que tome como entrada los parámetros del problema (a, b, n, m, r, T_s y las posiciones de las sanguijuelas) que calcule la temperatura en el parabrisas utilizando

el modelo propuesto en la sección anterior y que determine a qué sanguijuelas dispararle con el fin de evitar que se destruya el parabrisas. El método para determinar las sanguijuelas que serán destruídas queda a criterio del grupo, y puede ser exacto o heurístico.

Para resolver este problema, se deberá formular un sistema de ecuaciones lineales que permita calcular la temperatura en todos los puntos de la grilla que discretiza el parabrisas, e implementar el método de Eliminación Gaussiana (EG) para resolver este sistema particular. Dependiendo de la granularidad utilizada en la discretización, el sistema de ecuaciones resultante para este problema puede ser muy grande. Luego, es importante plantear el sistema de ecuaciones de forma tal que posea cierta estructura (i.e., una matriz banda), con el objetivo de explotar esta característica tanto desde la *complejidad espacial* como *temporal* del algoritmo.

En función de la implementación, como mínimo se pide:

1. Representar la matriz del sistema utilizando como estructura de datos los tradicionales arreglos bi-dimensionales. Implementar el algoritmo clásico de EG.

2. Representar la matriz del sistema aprovechando la estructura banda de la misma, haciendo hincapié en la complejidad espacial. Realizar las modificaciones necesarias al algoritmo de EG para que aproveche la estructura banda de la matriz.

En función de la experimentación, como mínimo deben realizarse los siguientes experimentos:

- Considerar al menos dos instancias de prueba, generando discretizaciones variando la granularidad para cada una de ellas y comparando el valor de la temperatura en el punto crítico. Se sugiere presentar gráficos de temperatura para los mismos, ya sea utilizando las herramientas provistas por la cátedra o implementando sus propias herramientas de graficación.

- Analizar el tiempo de cómputo requerido en función de la granularidad de la discretización, buscando un compromiso entre la calidad de la solución obtenida y el tiempo de cómputo requerido. Comparar los resultados obtenidos para las variantes propuestas en 1 y 2.

- Estudiar el comportamiento del método propuesto para la estimación de la temperatura en el punto crítico y para la eliminación de sanguijuelas.

Finalmente, se deberá presentar un informe que incluya una descripción detallada de los métodos implementados y las decisiones tomadas, incluyendo las estructuras utilizadas para representar la matriz banda y los experimentos realizados, junto con el correspondiente análisis y siguiendo las pautas definidas en el archivo `pautas.pdf`.

Programa y formato de archivos

El programa debe tomar tres parámetros (y en ese orden): el archivo de entrada, el archivo de salida y el método a ejecutar (0: EG común, 1: EG banda). El archivo de entrada contiene los datos del problema (tamaño del parabrisas, ubicación, radio y temperatura de las sanguijuelas) desde un archivo de texto con el siguiente formato:

```
(a)  (b)  (h)  (r)  (t)  (k)
(x1) (y1)
(x2) (y2)
...
(xk) (yk)
```

En esta descripción, $a \in \mathbb{R}_+$ y $b \in \mathbb{R}_+$ representan el ancho y largo en metros del parabrisas, respectivamente. De acuerdo con la descripción de la discretización del parabrisas, h es la longitud de cada intervalo de discretización, obteniendo como resultado una grilla de $n+1 \in \mathbb{N}$ filas y $m+1 \in \mathbb{N}$ columnas. Por su parte, $r \in \mathbb{R}_+$ representa el radio de las sopapas de ataque de las sanguijuelas (en metros), $t \in \mathbb{R}$ es la temperatura de ataque de las sopapas, y $k \in \mathbb{N}$ es la cantidad de sanguijuelas. Finalmente, para $i = 1, \ldots, k$, el par (x_i, y_i) representa la ubicación de la i-ésima sanguijuela en el parabrisas, suponiendo que el punto $(0, 0)$ corresponde al extremo inferior izquierdo del mismo.

El archivo de salida contendrá los valores de la temperatura en cada punto de la discretización utilizando la información original del problema (es decir, antes de aplicar el método de remoción de sanguijuelas), y será utilizado para realizar un testeo parcial de correctitud de la implementación. El formato del archivo de salida contendrá, una por línea, el indicador de cada posición de la grilla i, j junto con el correspondiente valor de temperatura. A modo de ejemplo, a continuación se muestran cómo se reportan los valores de temperatura para las posiciones $(3, 19)$, $(3, 20)$, $(4, 0)$ y $(4, 1)$.

```
...
3   19  -92.90878
3   20  -100.00000
4   0   -100.00000
4   1   60.03427
...
```

El programa debe ser compilado, ejecutado y testeado utilizando los *scripts* de *python* que acompañan este informe. Estos permiten ejecutar los tests provistos por la cátedra, incluyendo la evaluación de los resultados obtenidos e informando si los mismos son correctos o no. Es requisito que el código entregado pase satisfactoriamente los casos de tests provistos para su posterior corrección. Junto con los archivos podrán encontrar un archivo `README` que explica la utilización de los mismos.

Fecha de entrega

- *Formato electrónico:* Jueves 04 de Septiembre de 2014, hasta las 23:59 hs., enviando el trabajo (`informe + código`) a `metnum.lab@gmail.com`. El `asunto` del email debe comenzar con el texto `[TP1]` seguido de la lista de apellidos de los integrantes del grupo. Ejemplo: `[TP1] Díaz, Bianchi, Borghi`
- *Formato físico:* Viernes 05 de Septiembre de 2014, de 17:30 a 18:00 hs.

7. Referencias

Apuntes de clase
R. Burden y J.D.Faires, Análisis numérico, International Thomson Editors, 1998