

Universidad de Buenos Aires
Facultad de Ciencias Exactas y Naturales
Departamento de Computación
Métodos Numéricos

Trabajo Práctico Número 1: "No creo que a él le gustara eso"

4 de Septiembre de 2014

Autores

Aldasoro, Agustina agusaldasoro@gmail.com

Bouzón, María Belén belenbouzon@gmail.com

Cairo, Gustavo Juan gjcairo@gmail.com

Resumen

En esta presentación se intentará implementar, optimizar y analizar el comportamiento de un conjunto de algoritmos que faciliten la estimación de la temperatura en el punto central de una superficie rectangular, partiendo del conocimiento de la temperatura en determinados puntos de la misma. Asimismo, se discutirán diferentes alternativas algorítmicas que permitan garantizar que dicho punto no exceda los 235 grados centígrados.

Palabras Clave

- Algoritmo de Eliminación Gaussiana
- Back Substitution
- Matriz Banda

Contents

1	Introducción Teórica	1
1.1	Planteo del Sistema de Ecuaciones	2
1.2	Planteo acerca de las estructura internas de las matrices	4
1.3	Planteo en relación al empleo de nuestras estructuras en el problema presentado	9
1.4	Planteo en relación a la implementación del Algoritmo de Eliminación Gaussiana y back substitution	13
2	Resultados	16
3	Discusión	17
4	Conclusiones	19
5	Apéndices	20
5.1	Apéndice A	20
5.2	Apéndice B	27
	Bibliography	34
6	Referencias	36

Chapter 1: Introducción Teórica

Habiéndonos sido proporcionados:

- la ecuación del calor que modela de forma genérica el comportamiento de la temperatura T en un punto $(x,y) \in \mathbb{R}^2$ luego de haber consolidado un estado estacionario (figura 1.1)

$$\frac{\partial^2 T(x,y)}{\partial x^2} + \frac{\partial^2 T(x,y)}{\partial y^2} = 0. \quad (1.1)$$

- las medidas de un hipotético parabrisas bidimensional cuyos márgenes se sabe que mantendrán una temperatura constante de -100°C
- y ciertas posiciones del mismo que serán sometidas a una temperatura constante (también conocida) aplicada por una clase idflica y mutante de filo anélidos llamados hirudíneos (comúnmente conocidos como "sanguijuelas"),

nuestros desafíos consisten en:

- Proveer un algoritmo que - a partir de la consideración de los datos recién citados - aproxime la temperatura esperada en condiciones de estabilidad en una cantidad finita de puntos del parabrisas.

- Optimizar dicho algoritmo aprovechando los beneficios potencialmente provistos por la estructura del sistema de ecuaciones planteado.
- Diseñar e implementar una función algorítmica que garantice, a partir de la eliminación de la menor cantidad posible de sanguijuelas, la perdurabilidad del parabrisas (esto es, el resguardo de su punto crítico a una temperatura inferior a los 235 grados Celcius).

Como punto de partida y linea de desarrollo que nos permita conseguir estos objetivos, haremos uso e implementaremos en C++ el Algoritmo de Eliminación Gaussiana, poniendo en discusión posteriormente posibles modificaciones del mismo que permitan adecuarlo a diversos contextos y necesidades.

Dicho algoritmo transforma un sistema lineal $Ax = b$, $A \in \mathbb{R}^{n \times n}$, en uno equivalente $Ux = y$ - donde U es una matriz triangular superior - permitiendo aplicar posteriormente el algoritmo de sustitución regresiva que culmina en la resolución total del sistema.

1.1 Planteo del Sistema de Ecuaciones

Al comenzar a plantear un sistema de ecuaciones que pudiera resultar efectivo para la resolución de los problemas presentados, se puso en cuestión inicialmente el que se desarrolla a continuación:

Considerando la ecuación de aproximación por diferencias finitas de la ecuación

del calor

$$t_{ij} = \frac{t_{i-1,j} + t_{i+1,j} + t_{i,j-1} + t_{i,j+1}}{4}. \quad (1.2)$$

pensamos que podríamos diseñar una matriz que tuviera para todo i,j una ecuación asociada en función de las otras posiciones y que estuviera extendida con cada valor de t_{ij} en el vector independiente (es decir, un sistema de ecuaciones expresadas como en la figura 2.1). Rápidamente descartamos esa posibilidad, ya que nos encontrábamos situando en el vector de términos independientes, varios que eran ciertamente dependientes.

Luego volvimos a la ecuación de la que habíamos partido, que nos indicaba la propiedad que inexorablemente cumplía el comportamiento de la temperatura en función de la posición consultada y de la distribución de las sanguijuelas. Le restamos a ambos lados t_{ij} y, despejando la ecuación, obtuvimos una nueva expresión igualada a cero, que determinará finalmente la estructura de nuestro sistema de ecuaciones, dado por:

$$0 = t_{i-1,j} + t_{i+1,j} + t_{i,j-1} + t_{i,j+1} - 4.t_{i,j} \quad (1.3)$$

para todos aquellos puntos a los que no se les aplica una temperatura constante (es decir, no pertenecen al borde del parabrisas ni se encuentran ocupados por sanguijuelas).

$T_{ij} = -100$ para todos los puntos del borde.

$T_{ij} = T_s$ para los puntos ocupados por sanguijuelas, siendo T_s la temperatura aplicada por las sanguijuelas.

1.2 Planteo acerca de las estructura internas de las matrices

Para representar matrices consideramos inicialmente la estructura clásica: *vector < vector < double >>*.

Esta implementación permite acceder al elemento M_{ij} mediante la sentencia *matriz[i][j]*.

Almacenamos cada punto discretizado del parabrisas como una incógnita de la matriz, es decir si tenemos una discretización de $n + 1$ filas por $m + 1$ columnas, tendremos una matriz de $(n + 1).(m + 1)$ filas y $(n + 1).(m + 1)$ columnas. Si quiero acceder al elemento de la fila i , columna j del parabrisas discretizado: considero la incógnita de la columna de la matriz $(k.i) + j$, siendo $k = \frac{a}{h} + 1$.

En cuanto al almacenamiento de nuestra matriz banda, al considerar su morfología nos pareció una idea plausible representarla conservando únicamente, los elementos no nulos de la bandas, de la diagonal principal y los términos independientes. Viéndolo gráficamente, si la matriz fuera la siguiente:

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Se almacenaría de esta forma:

$$\begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & -4 & 1 & 1 \\ 1 & 1 & -4 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

Posteriormente advertimos que aquella estructura podría traernos serios inconvenientes al momento de utilizarla como parámetro en el algoritmo de eliminación gaussiana e intentar hallar la solución al sistema, dado que los ceros que se encuentran a la derecha de la diagonal y a la izquierda son potencialmente modificables durante este procedimiento, y dichos cambios serían imperceptibles a pesar de ser determinantes a la hora de hallar la solución del sistema.

Ejemplo:

Consideremos la siguiente matriz $M \in \mathbb{R}^{16 \times 17}$ producida a partir de un parabrisas cuadrado de 3 metros de lado, sin sanguijuelas, discretizado definiendo arbitrariamente $h=1$.

$$\left(\begin{array}{cccccccccccccccc|c} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -100 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -100 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -100 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -100 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -100 \\ 0 & 1 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -100 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -100 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -100 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -100 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -100 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -100 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -100 \end{array} \right)$$

En este caso, si la Eliminación Gaussiana se estuviera encargando de anular los valores que se encuentran por debajo de la diagonal principal en la quinta columna, eventualmente el resultado en la novena fila será, para todo $j / 0 \leq j \leq 15$

$$(a'_{9,j}) = (a_{9,j}) - 4^{-1} * (a_{5,j}) \quad (1.4)$$

Es decir que el elemento $a'_{9,6}$ deberá pasar a valer -4^{-1} , pero a causa de un error en

la estructura de representación estábamos omitiendo la relevancia de su modificación y arribando, en consecuencia, a un resultado incorrecto.

Teniendo en cuenta este inconveniente, resolvimos escoger una opción similar que a nivel espacial era peor que la recién mencionada pero seguía aventajando a la opción clásica. Esta tercera alternativa consistió en computar para cada fila de la matriz original sólo los elementos que se encontraran entre el extremo izquierdo de la banda y el derecho (inclusive). La matriz presentada con anterioridad quedaría almacenada de la siguiente manera:

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Este modo junto con el clásico fueron los que decidimos implementar para experimentar y contrastar sus comportamientos en relación a los algoritmos que desarrol-

laríamos.

1.3 Planteo en relación al empleo de nuestras estructuras en el problema presentado

Al momento de recibir por parámetro las coordenadas del centro de cada sanguijuela, nuestra matriz de posiciones debía adaptar sus valores de acuerdo al mapeo que se haría de cada una de ellas en el nuevo modelo discreto de representación. En función de la longitud de su radio, la ubicación de su centro y la medida de discretización escogida, los mencionados especímenes podrían pasar a estar representados en un único punto, en varios o incluso en ninguno. En principio, para cada sanguijuela debíamos recorrer posición a posición de la matriz discretizada averiguando si pertenecía al sector abarcado por la misma. Para evitarlo, calculamos previamente, a partir de los parámetros, un sector cuadrado de puntos fuera de los cuales no existiría la posibilidad de que una sanguijuela en particular tuviera injerencia directa. Determinado este sector, sólo deberíamos modificar los puntos p pertenecientes al mismo que cumplieran, a su vez,

$$\sqrt{(p1 - c1)^2 + (p2 - c2)^2} \leq radio \quad (1.5)$$

Esto lo cumplen aquellos puntos discretizados que representan a alguna coordenada del continuo cuya distancia al centro $C=(c1,c2)$ de la sanguijuela es menor o igual al radio de la misma. A dichas posiciones se les asignará posteriormente la

temperatura T_s en la matriz de temperaturas, mientras que a las restantes les corresponderá la ecuación de la figura 1.1.

Teniendo en cuenta el empleo subyacente de la aritmética finita, consideramos que planteando una ecuación equivalente elevando al cuadrado a ambos lados de la desigualdad - eliminando el cálculo de la raíz cuadrada - podríamos lograr una mayor precisión y una menor probabilidad de error. Por ende, la ecuación (2.4) fue sustituida por la siguiente:

$$(p1 - c1)^2 + (p2 - c2)^2 \leq radio^2 \quad (1.6)$$

En relación a los casos en que al pasar del plano continuo al discreto la presencia de las sanguijuelas no se veía reflejada en ningún punto, nos pareció una buena práctica pensar algún criterio para diferenciar y advertir al usuario sobre aquellas discretizaciones que se pudieran considerar como "buenas" de otras que pudieran favorecer la aparición de casos patológicos en los cuales el cálculo de la estimación de la temperatura fuera groseramente inexacto. Algunos de estos casos se mencionan a continuación:

-La sanguijuela no tiene ningún punto discretizado en su interior.

A fines de preservar el parabrisa, si esto ocurre en puntos cerca del borde no nos es de mayor interés. En cambio, si la ubicación de la sanguijuela es cerca del punto crítico podría influir notablemente en su temperatura. Cabe recalcar, que hay que tener en cuenta que esto varía de acuerdo al radio y la temperatura de las sanguijuelas.

-La cantidad de sanguijuelas que no tiene ningún punto discretizado en su interior y están cerca del punto crítico es alta.

Podría pasar que haya una cantidad "alta" de sanguijuelas cerca del borde, pero ninguna tenga un punto en su interior que este discretizado. Las sanguijuelas están, sólo que nuestra discretización no les permite influir en la temperatura del punto central.

-Existe una sanguijuela que no tiene ningún punto discretizado en su interior, pero el punto crítico si pertenece a él.

Ejemplo: Hay una sola sanguijuela, que dado por su radio y por su centro, tiene en su interior al punto crítico. En teoría, el punto crítico tendría el valor T_s , pero una vez más la discretización dada por el h no nos permitiría verlo.

ACLARACIÓN:

Si la temperatura T_s pasada por parámetro es menor que 235 grados centígrados, no existe siquiera la posibilidad de que el parabrisas pase a ser inestable.

La idea de informar al usuario sobre estos casos patológicos fue descartada debido a que en el contexto del presente trabajo cobra relevancia el estudio del comportamiento de las estructuras y algoritmos diseñados en un plano teórico global, en detrimento del análisis de los casos particulares.

Sin embargo, sin perder de vista la proyección de nuestras implementaciones en un hipotético caso real, nos pareció sensato idear una forma prouyente de estimar la temperatura en el punto crítico cuando el mismo no se hallara discretizado. Inicialmente se nos presentaron las siguientes opciones:

- Adjudicarle la temperatura del punto discretizado más próximo.
- Asignarle la temperatura del promedio de los cuatro puntos discretizados más cercanos.
- Otorgarle la temperatura del promedio de una cantidad de puntos discretizados cercanos proporcional a la cantidad total.

Decidimos resolverlo del siguiente modo:

- Si el punto crítico está discretizado, se le otorga el valor que se calculó en la discretización.
- Si el punto crítico está ubicado en el medio del cuadrante armado por la discretización, es decir es el único caso en que está a la misma distancia de los cuatro puntos discretizados más cercanos, se le otorga el promedio de estos cuatro valores.
- Si el punto crítico está a la misma distancia de dos puntos discretizados, se le otorga el promedio de estos dos valores.
- En cualquier otro caso, se le otorga al punto crítico el valor del punto discretizado más cercano.

PODEMOS PONER LOS DIBUJITOS SI NOS SOBRA TIEMPO. Just saying.

1.4 Planteo en relación a la implementación del Algoritmo de Eliminación Gaussiana y back substitution

A la hora de pensar maneras de optimizar el cómputo de los cálculos vinculados a la matriz, tomamos en cuenta la posibilidad de particionarla de forma tal que quedara dividida en patrones que pudieran repetirse, de modo que los mismos pudieran ser resueltos algorítmicamente una única vez para luego replicar su resultado en más de una oportunidad. Concluimos, finalmente, que este mecanismo resultaría aplicable sólo en casos muy específicos para los cuales no existe ningún indicio de que su probabilidad de ocurrencia pudiera ser significativa.

Habiendo descartado dicha posibilidad, decidimos optar por implementar una adaptación del algoritmo de Eliminación Gaussiana que procurara aprovechar las particularidades de la morfología de la estructura de las matrices "Banda". Dicho algoritmo se implementó de la siguiente manera:

Algorithm 1.1: EGbanda

```

input : Matriz
output: Void
1 for  $i \leftarrow (k + 2)$  to  $(\#filas - k)$  do
2    $j \leftarrow 0$  // j va a recorrer las columnas
3   while  $j < (\text{posición elemento de la diagonal})$  do
4      $com \leftarrow (i + j - k - 1)$  // GUS: Poner aclaración abajo.
5     if  $(matriz(i, j) > \varepsilon)$  then
6        $m \leftarrow matriz(i, j) / matriz(com, PosDiagonal)$ 
7       Resta Corrida sin la columna Resultado (fila i, fila com)
8        $matriz(i, PosResult) -= matriz(com, PosResult)$ 

```

Teniendo en cuenta que bajo las condiciones dadas en nuestro sistema de ecuaciones, es imposible que un elemento de la diagonal principal sea nulo o quede nulo después de triangular. Decidimos omitir el paso de la Eliminación Gaussiana donde se asegura que al momento de triangular no quede un elemento nulo en la diagonal, mediante un intercambio de filas.

Asimismo, luego de esquematizarlo, nos planteamos la posibilidad de optimizar la eficiencia del algoritmo y minimizar su error aplicando pivoteo parcial. Sin embargo, dicha implementación tendría un costo extra en complejidad temporal y no influiría en nuestros resultados dado al origen de la matriz. Creímos razonable, en consecuencia, no implementarlo de esa forma en esta oportunidad.

Una observación de mayúscula importancia fue notar que el primer algoritmo que desarrollamos con los fines de resolver el sistema triangulado mediante E.G, generaba resultados con errores porcentualmente grandes en relación a los tolerables. Luego de un análisis teórico del problema, nos dimos cuenta de que nos encontrábamos dividiendo a cada fila por el coeficiente ubicado en la diagonal para asegurarnos de que al final de la pasada por esa fila, la misma quedara como el vector canónico igualado a su resultado. Esto acarreaba error porque hacia menos exacta la cuenta ya que no solo se dividía al vector resultado por un coeficiente cuyo módulo no se encuentra acotado inferiormente, sino que también se repetía el procedimiento con los demás coeficientes que luego habría que triangular.

Por lo tanto, decidimos ir triangulando por filas, es decir, despejar cada elemento desde la última fila hacia la primera, suplantando en la ecuación superior los valores

obtenidos. Únicamente después de este procedimiento dividiríamos por el coeficiente principal, de modo que quede la matriz Identidad igualada a su resultado.

Chapter 2: Resultados

Cito:

Deben incluir los resultados de los experimentos, utilizando el formato mas adecuado para su presentacion. Deberan especificar claramente a que experiencia corresponde cada resultado. No se incluyan aqui corridas de maquina

No es aceptable incluir tablas de resultados numericos siempre que sea posible su graficacion. Esto no significa que se deban mostrar los resultados exclusivamente en forma de graficos. Se debe buscar la manera de mostrar la informacion importante en la mejor forma de visualizacion. Este hecho sera tenido en cuenta para la evaluacion de la presentacion de los informes. Es conveniente que el proceso de experimentacion no se realice en forma independiente de la presentacion de los resultados, de manera que los resultados parciales obtenidos en los primeros experimentos vayan orientando la experimentacion posterior.

Chapter 3: Discusión

Cito:

Se incluire aqui un analisis de los resultados obtenidos en la seccion anterior (se analizara su validez, coherencia, etc.). Deben analizarse como mnimo los items pedidos en el enunciado. No es aceptable decir que los resultados fueron los esperados, sin hacer clara referencia a la teora a la cual se ajustan. Ademias, se deben mencionar los resultados interesantes y los casos "patologicos" encontrados.

La seccion de discusion debe estar relacionada indefectiblemente con el desarrollo y los resultados presentados en las secciones anteriores. No se aceptaran afirmaciones que no esten respaldadas por los datos presentados

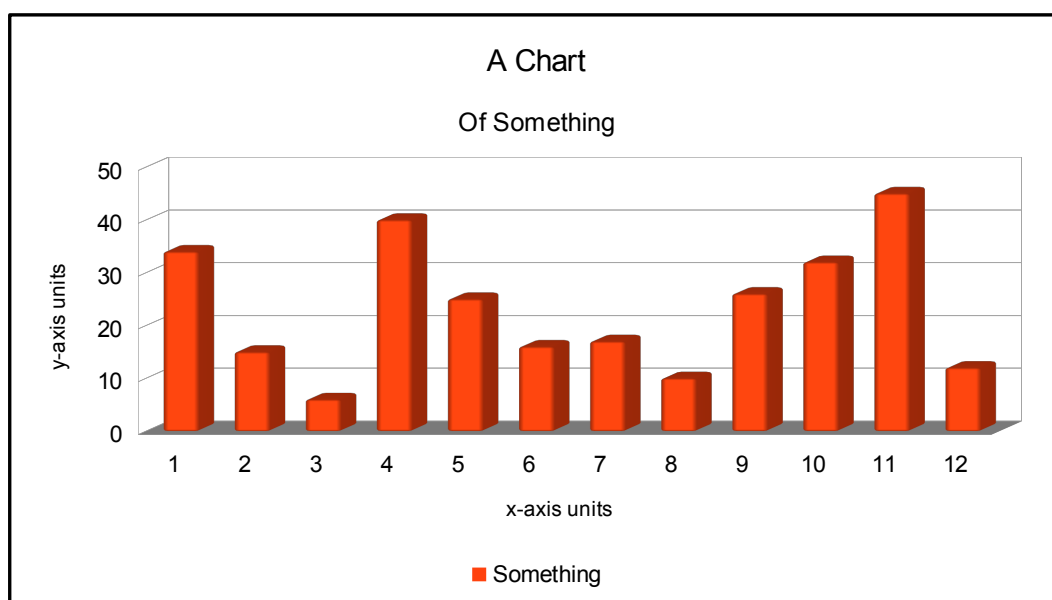


Figure 3.1: A Chart.

Chapter 4: Conclusiones

Cito:

Esta seccion debe contener las conclusiones generales del trabajo. Se deben mencionar las relaciones de la discusion sobre las que se tiene certeza, junto con comentarios y observaciones generales aplicables a todo el proceso. Mencionar tambien posibles extensiones a los metodos, experimentos que hayan quedado pendientes. Se pide certeza y rigor logico de la discusion, el analisis y las conclusiones. Se pondra especial enfasis en que las conclusiones se desprendan de las experiencias realizadas.

Chapter 5: Apéndices

5.1 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°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°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:

$$\frac{\partial^2 T(x, y)}{\partial x^2} + \frac{\partial^2 T(x, y)}{\partial y^2} = 0. \quad (5.1)$$

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

$$T(x, y) = -100^{\circ}\text{C} \quad \text{si } x = 0, a \text{ ó } y = 0, b. \quad (5.2)$$

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 $i = 0, 1, \dots, n$ y $j = 0, 1, \dots, m$, llamemos $t_{ij} = T(x_j, y_i)$ al valor (desconocido) de la función T en el punto $(x_j, y_i) = (jh, ih)$, 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:

$$t_{ij} = \frac{t_{i-1,j} + t_{i+1,j} + t_{i,j-1} + t_{i,j+1}}{4}. \quad (5.3)$$

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 , h , 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 destruidas 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 $i = 1, \dots, k$, 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.

5.2 Apéndice B

Códigos fuente de funciones relevantes desde el punto de vista numérico:

Algorithm 5.1: EGcomún

```

input : Matriz
output: Void

1 for  $i \leftarrow 0$  to columnafinal do
2   for  $j \leftarrow 0$  to (últimafila -  $k$ ) do
3     if (matriz( $i, j$ ) >  $\varepsilon$ ) then
4        $m \leftarrow \text{matriz}(i, j) / \text{matriz}(i, i)$ 
5       fila  $j \leftarrow \text{fila } j - m.(\text{fila } i)$ 

```

Algorithm 5.2: EGbanda

```

input : Matriz
output: Void

1 for  $i \leftarrow (k + 2)$  to (#filas -  $k$ ) do
2    $j \leftarrow 0$  //  $j$  va a recorrer las columnas
3   while  $j < (\text{posición elemento de la diagonal})$  do
4      $com \leftarrow (i + j - k - 1)$  // GUS: Poner aclaración abajo.
5     if (matriz( $i, j$ ) >  $\varepsilon$ ) then
6        $m \leftarrow \text{matriz}(i, j) / \text{matriz}(com, \text{PosDiagonal})$ 
7       Resta Corrida sin la columna Resultado (fila  $i$ , fila  $com$ )
8        $\text{matriz}(i, \text{PosResult}) -= \text{matriz}(com, \text{PosResult})$ 

```

- **PosResult** es la posición de la columna de resultados.
- Decidimos hacer la triangulación por filas en vez de por columnas.
- La resta se debe hacer corrida y no simplemente restar los vectores fila como en la eliminación Gaussiana común porque las columnas no coinciden fila a fila. Entonces, se hace coincidir la posición de la diagonal del vector superior con su posición

correspondiente en el inferior y se comienza la resta, la cual se puede ir haciendo iterativamente ya que las posiciones de las columnas de al lado son consecutivas. En el caso de la columna de resultados, no es la misma situación, se resta aparte.

Algorithm 5.3: ResolverSistema

```

input : Matriz
output: Void

1 if (tipo == Matriz Común) then
2   matriz.EGcomún()
3    $i \leftarrow \# \text{filas}$ 
4   while ( $i \geq 0$ ) do
5     for  $j \leftarrow \text{última columna}$  to  $i$  do
6        $\text{vectorResultado}(i) -= \text{matriz}(i,j) * \text{matriz}(j, \text{PosResult})$ 
7        $\text{matriz}(i, \text{PosResult}) = \text{matriz}(i, \text{PosResult}) / \text{matriz}(i,j)$ 
8 else if (tipo == Matriz Banda) then
9   matriz.EGbanda()
10   $i \leftarrow (\# \text{filas} - k - 2)$  // Las últimas k-2 filas ya están
    trianguladas
11  while ( $i > k$ ) do
12     $f \leftarrow 1$ 
13    for  $j \leftarrow (k+2)$  to PosResult do
14       $\text{matriz}(i, \text{PosResult}) -= \text{matriz}(i,j) * \text{matriz}(i+f, \text{PosResult})$ 
15       $f++$ 
16     $\text{matriz}(i, \text{PosResult}) = \text{matriz}(i, \text{PosResult}) / \text{matriz}(i, \text{PosDiag})$ 
17     $i--$ 
```

- **PosResult** es la posición de la columna de resultados.
- **PosDiag** es la posición de la columna de la diagonal.
- Para optimizar tiempo de cómputo, a medida que se hace el Backward Sustitution, los únicos elementos modificados son los de la columna resultados. Es decir, los ceros

que habría que escribir en la matriz se omiten, ya que no se vuelve acceder a ellos.

- Una vez triangulada inferiormente la fila, se la divide por el elemento de la diagonal, así queda esa incógnita despejada. Esto también se hace solamente en la posición del resultado.

Algorithm 5.4: CrearParabrisas

input : a, b, r, h, temp, vect, tipo
output: Parabrisas

```

1 if (tipo == Matriz Común) then
2   Redimensiono la matriz con el tamaño que necesito
3   for  $i \leftarrow 0$  to  $\#columnas$  do
4     if (columna i es un elemento del Borde) then
5        $matriz(i,i) \leftarrow 1$ 
6        $matriz(i, PosResult) \leftarrow -100$ 
7   for  $i \leftarrow 0$  to  $tamaño(vect)$  do
8      $Ataque(vect[i])$ 
9   for  $i \leftarrow 0$  to  $\#columnas$  do
10    if ( $matriz(i,i) == 0$ ) then
11       $matriz(i,i) \leftarrow -4$ 
12       $matriz(i,i-1) \leftarrow 1$ 
13       $matriz(i,i+1) \leftarrow 1$ 
14       $matriz(i,i-\frac{a}{h}-1) \leftarrow 1$ 
15       $matriz(i,i+\frac{a}{h}+1) \leftarrow 1$ 
16 else if (tipo == Matriz Banda) then
17   Redimensiono la matriz con el tamaño que necesito
18   for  $i \leftarrow 0$  to  $\#filas$  do
19     if (elemento de la fila i, columna de la diagonal es un elemento del  

20       Borde) then
21        $matriz(i, PosDiag) \leftarrow 1$ 
22        $matriz(i, PosResult) \leftarrow -100$ 
23   for  $i \leftarrow 0$  to  $tamaño(vect)$  do
24      $Ataque(vect[i])$ 
25   for  $i \leftarrow 0$  to  $\#filas$  do
26     if ( $matriz(i, PosDiag) == 0$ ) then
27        $matriz(i, PosDiag) \leftarrow -4$ 
28        $matriz(i,0) \leftarrow 1$ 
29        $matriz(i, PosDiag - 1) \leftarrow 1$ 
30        $matriz(i, PosDiag + 1) \leftarrow 1$ 
        $matriz(i, PosResult - 1) \leftarrow 1$ 

```

Algorithm 5.5: Ataque

input : Parabrisas, posición: (p1, p2)

output: Parabrisas

```

1 if (La sanguijuela no toca el borde) then
2   Cálculo de los bordes de la circunferencia
3   IntervaloX  $\leftarrow$  (p1 - radio, p1 + radio)
4   IntervaloY  $\leftarrow$  (p2 - radio, p2 + radio)
5   i  $\leftarrow$  (p1 - radio)
6   while ( $i \leq p1 + \text{radio}$ ) do
7     c  $\leftarrow$  (p2 - radio)
8     while ( $c \leq p2 + \text{radio}$ ) do
9       norma2  $\leftarrow$   $(i - p1)^2 + (c - p2)^2$ 
10      if ( $\text{norma2} \leq \text{radio}^2$ ) then
11        matriz(fila del elemento (i,c) , PosResult)  $\leftarrow$  temp
12        matriz(fila del elemento (i,c) , PosDiag)  $\leftarrow$  1
13      i  $\leftarrow$  (p1 + radio)
14    while ( $i \leq p1 + \text{radio}$ ) do
15      c  $\leftarrow$  p2 + radio
16      while ( $c \leq p2 + \text{radio}$ ) do
17        norma2  $\leftarrow$   $(i - p1)^2 + (c - p2)^2$ 
18        if ( $\text{norma2} \leq \text{radio}^2$ ) then
19          matriz(fila del elemento (i,c) , PosResult)  $\leftarrow$  temp
20          matriz(fila del elemento (i,c) , PosDiag)  $\leftarrow$  1
21    Si la sanguijuela modificó algún punto de la matriz, la agrego al vector
    de sanguijuelas Discretizadas

```

Algorithm 5.6: Cálculo del Punto Crítico

```

input : Matriz
output: double

1 PuntoCrítico = (p1,p2)
2 if (PuntoCrítico está discretizado) then
3   | devolver el valor de la matriz que corresponda
4 else
5   | Cálculo de los cuatro puntos que encierran al PuntoCrítico:
6   | pArrIzq = (filArri, colIzq)
7   | pArrDer = (filArri, colDer)
8   | pBajIzq = (filAba, colIzq)
9   | pBajDer = (filAba, colDer)
10  | if ( $p1 - promedio(filArri, filAba) \leq \varepsilon$ ) then
11  |   | if ( $p2 - promedio(filArri, filAba) \leq \varepsilon$ ) then
12  |   |   | return Promedio(pArrIzq, pArrDer, pBajIzq, pBajDer)
13  |   |   | else if (p2 está más cerca de filAba) then
14  |   |   |   | return Promedio(pBajDer, pBajIzq)
15  |   |   | else
16  |   |   |   | return Promedio(pArrIzq, pArrDer)
17  |   | else if ( $p2 - promedio(filArri, filAba) \leq \varepsilon$ ) then
18  |   |   | if (p1 está más cerca de colDer) then
19  |   |   |   | return Promedio(pArrDer, pBajDer)
20  |   |   | else
21  |   |   |   | return Promedio(pArrIzq, pBajIzq)
22  | else
23  |   | if (PuntoCrítico está en la mitad izquierda del cuadrado) then
24  |   |   | if (PuntoCrítico está en la parte superior) then
25  |   |   |   | return pArrIzq // Está en el segundo cuadrante
26  |   |   |   | else
27  |   |   |   |   | return pBajIzq // Está en el tercer cuadrante
28  |   |   | else
29  |   |   |   | if (PuntoCrítico está en la parte superior) then
30  |   |   |   |   | return pArrDer // Está en el primer cuadrante
31  |   |   |   | else
32  |   |   |   |   | return pBajDer // Está en el cuarto cuadrante

```

Algorithm 5.7: Matar Sanguijuelas de a 1

input : Parabrisas

output: Void

```

1 while (El parabrisas no sea estable) do
2   Elijo del vector de sanguijuelas discretizadas al elemento que su centro
   este más cerca del Punto Crítico.
3   Lo borro
4   Recalculo el parabrisas con las nuevas sanguijuelas

```

Algorithm 5.8: Matar 25% deSanguijuelas

input : Parabrisas

output: Void

```

1 while (El parabrisas no sea estable) do
2   Elijo del vector de sanguijuelas discretizadas al 25% de sus elementos,
   que su centro sean los más cerca del Punto Crítico.
3   Los borro
4   Recalculo el parabrisas con las nuevas sanguijuelas

```

Algorithm 5.9: Matar por Proximidad

input : Parabrisas

output: Void

```
1 while (El parabrisas no sea estable) do  
2   Elijo del vector de sanguijuelas discretizadas al elemento que su centro  
   este más cerca del Punto Crítico.  
3   Busco los elementos que sean el 25% del total, contando al ya obtenido,  
   que esten más cerca de este  
4   Los borro  
5   Recalculo el parabrisas con las nuevas sanguijuelas
```

Bibliography

Chapter 6: Referencias

