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

Chapter 1: Introducción Teórica

Habíendonos 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)

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

- las medidas de un hipotético parabrisas bidimensional cuyos márgenes se sabe que mantendrán una temperatura constante de -100C
- y ciertas posiciones del mismo que serán sometidas a una temperatura constante (también conocida) aplicada por una clase idílica 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 $\in \mathbb{R}^{nxn}$, 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

$$t_{ij} \;=\; \frac{t_{i-1,j} + t_{i+1,j} + t_{i,j-1} + t_{i,j+1}}{4}. \tag{1.2}$$

pensamos que podríamos disear 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:

$$0 \;=\; t_{i-1,j} + t_{i+1,j} + t_{i,j-1} + t_{i,j+1} - 4.t_{i,j+1} \tag{1.3}$$

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 $M \in \mathbb{R}^{16x17}$ 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 / $0 <= j <= 15$

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

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 estríamos 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,

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

Esto lo cumplen aquellos puntos discretizados que representan a alguna coordenada del contínuo 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 sustituída por la siguiente:

$$(p1 - c1)^2 + (p2 - c2)^2 \leq radio^2 \tag{1.6}$$

En relación a los casos en que al pasar del plano contínuo 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 diseados 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 produente 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 ← (k + 2) to (#filas − k) do
2     j ← 0                                  // j va a recorrer las columnas
3     while j < (posición elemento de la diagonal) do
4         com ← (i + j − k − 1)              // GUS: Poner aclaración abajo.
5         if (matriz(i, j) > ε) then
6             m ← 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 dividia al vector resultado por un coeficiente cuyo módulo no se encuentra acotado inferiormente, sino que también se repetía el procedimiento con los demas 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

Siendo A el ancho del parabrisas, B la altura, H el tamaño de la discretización, R el radio de las sanguijuelas, #S la cantidad de sanguijuelas:

2.1 Caso de test 1

En nuestro primer intento de test, elegimos los datos de entrada detallados abajo bajo la hipótesis de que cada método heurístico no iba a matar las mismas sanguijuelas, pero si la misma cantidad.

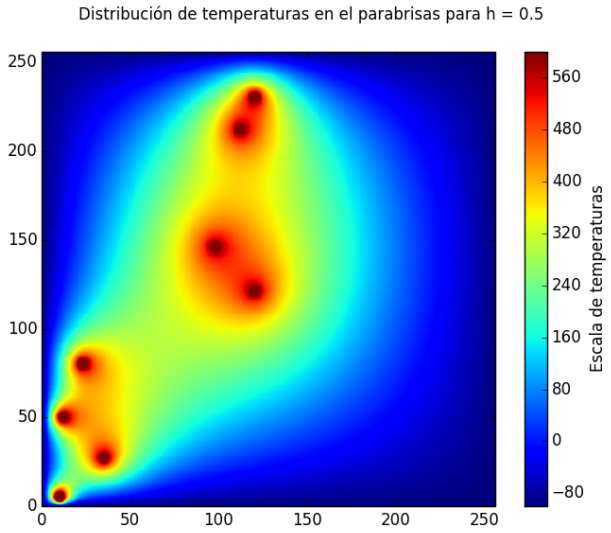


Figure 2.1: Los métodos de entrada para este caso fueron: A = 256, B = 256, H = 4, R = 3, Ts = 600, #S = 8

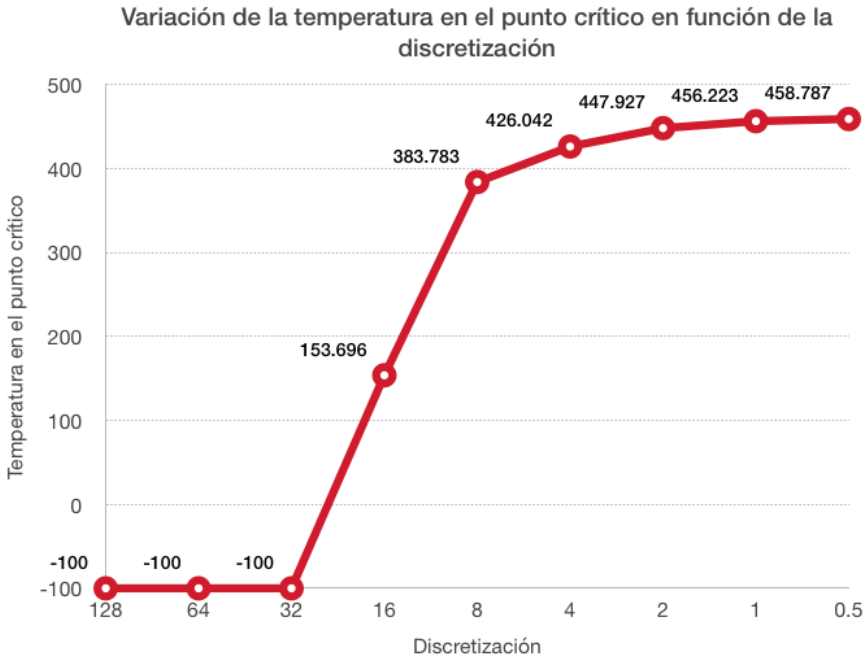


Figure 2.2: En el gráfico se puede apreciar cómo varía la temperatura del punto crítico a medida que se va modificando el h entre 128, 64, 32, 16, 8, 4, 2, 1, 0.5

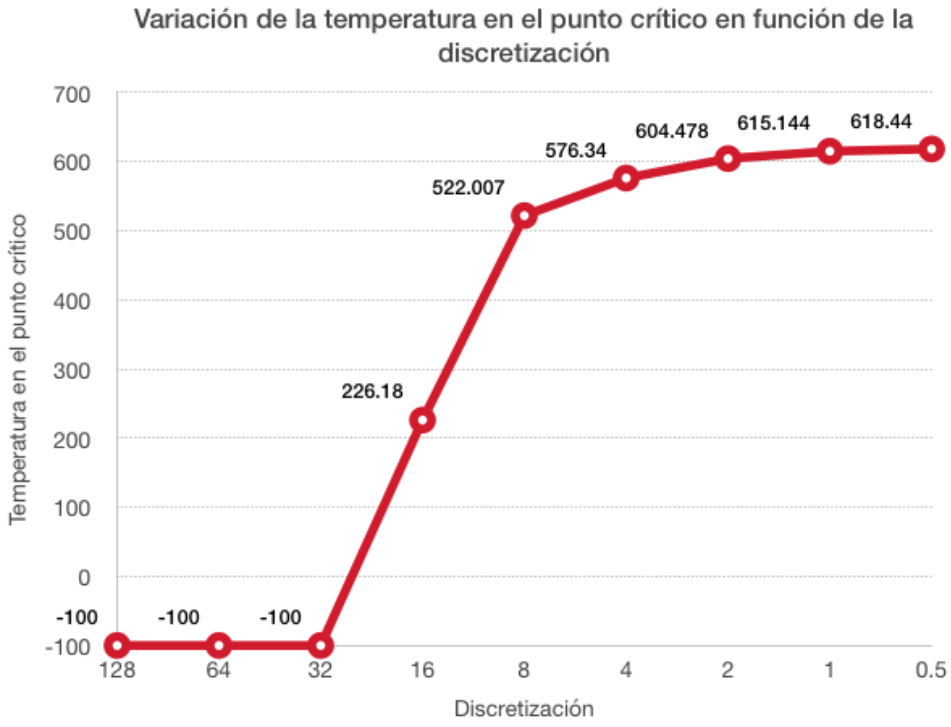


Figure 2.3: Si hacemos la misma prueba, pero aumentando la temperatura a 800 se obtiene un esquema similar

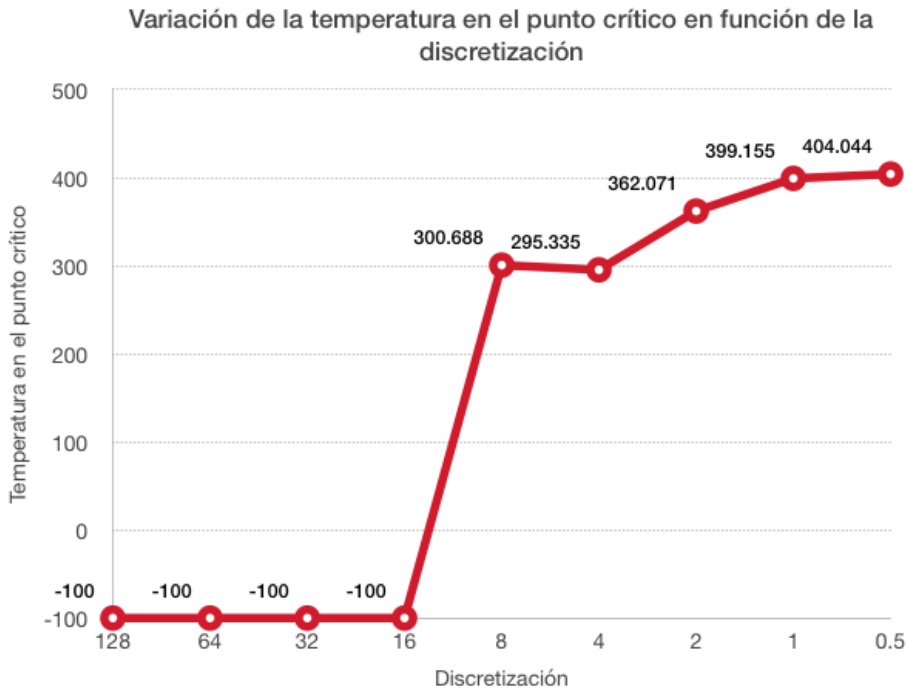


Figure 2.4: En cambio, si lo que hacemos es achicar el radio a 1.5, se obtiene

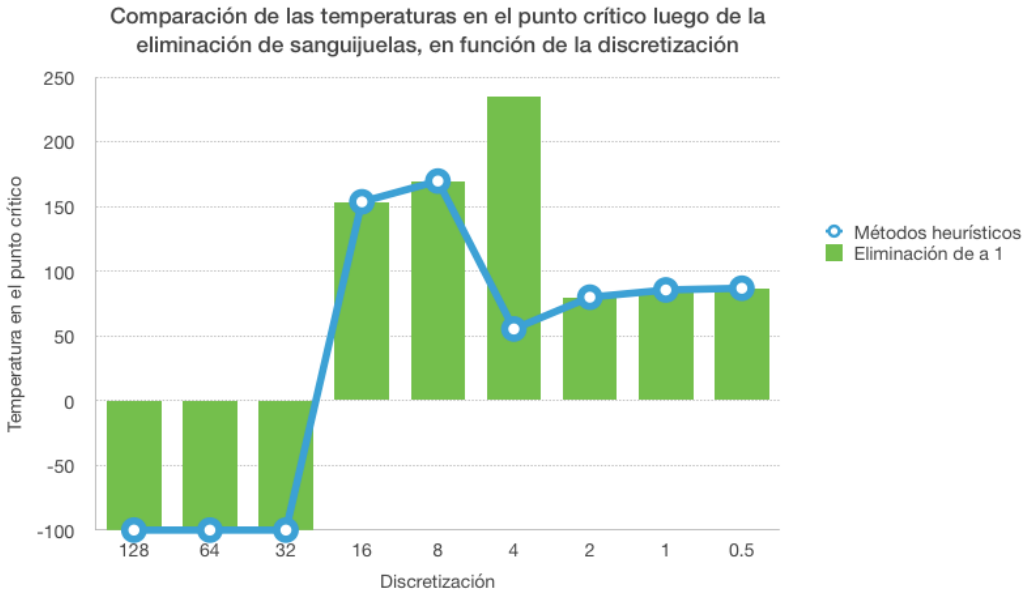

Comparación de las temperaturas en el punto crítico luego de la eliminación de sanguijuelas, en función de la discretización
Temperatura en el punto crítico
250
200
150
100
50
0
-50
-100
128
64
32
16
8
4
2
1
0.5
Discretización
Métodos heurísticos
Eliminación de a 1

Figure 2.5:

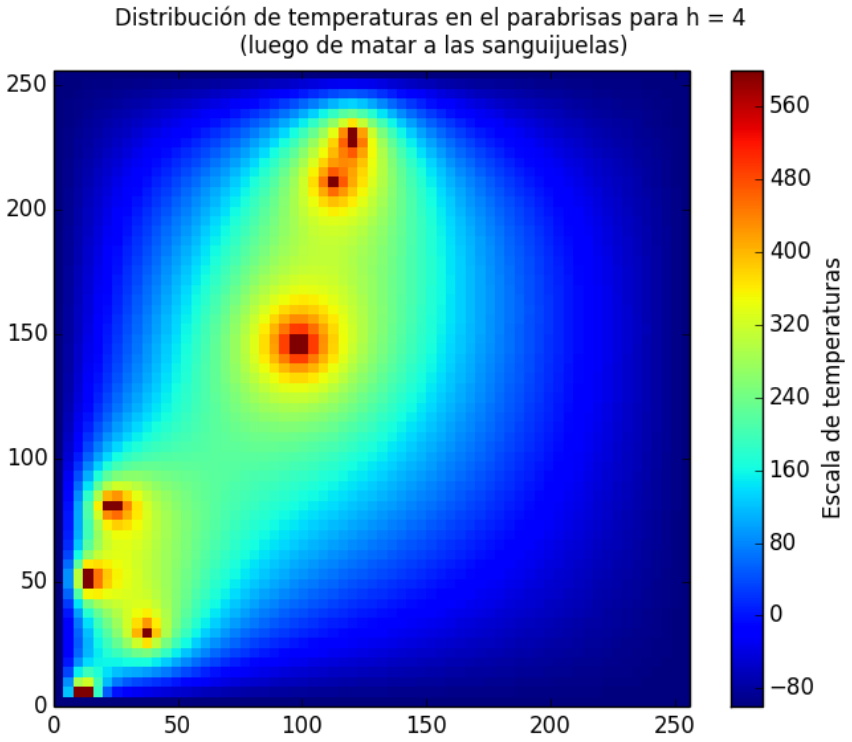


Figure 2.6: Cabe mencionar, el resultado de matar Sanguijuelas de a una, con un h=4:

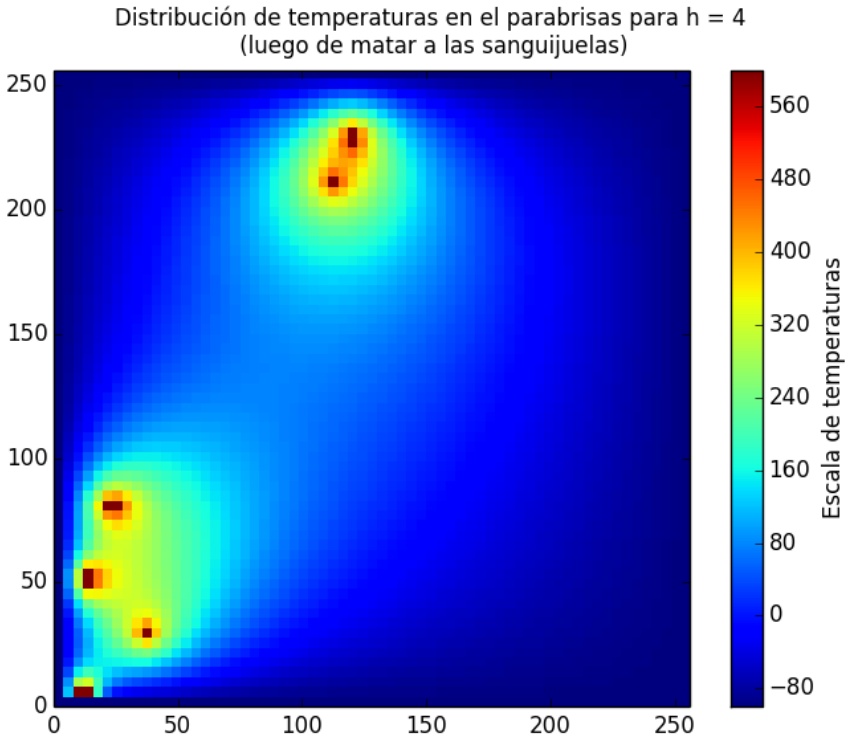


Figure 2.7: El resultado de matar sanguijuelas mediante un algoritmo eurístico (da igual cual), con h=4

2.2 Caso de test 2

Elegimos los datos de entrada detallados abajo bajo la hipótesis de que eligiendo estos parámetros las sanguijuelas borradas iba a diferir respecto al algoritmo eurístico usado.

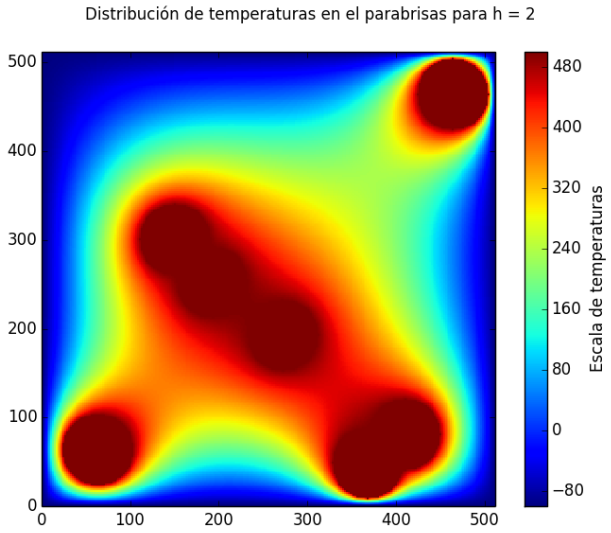
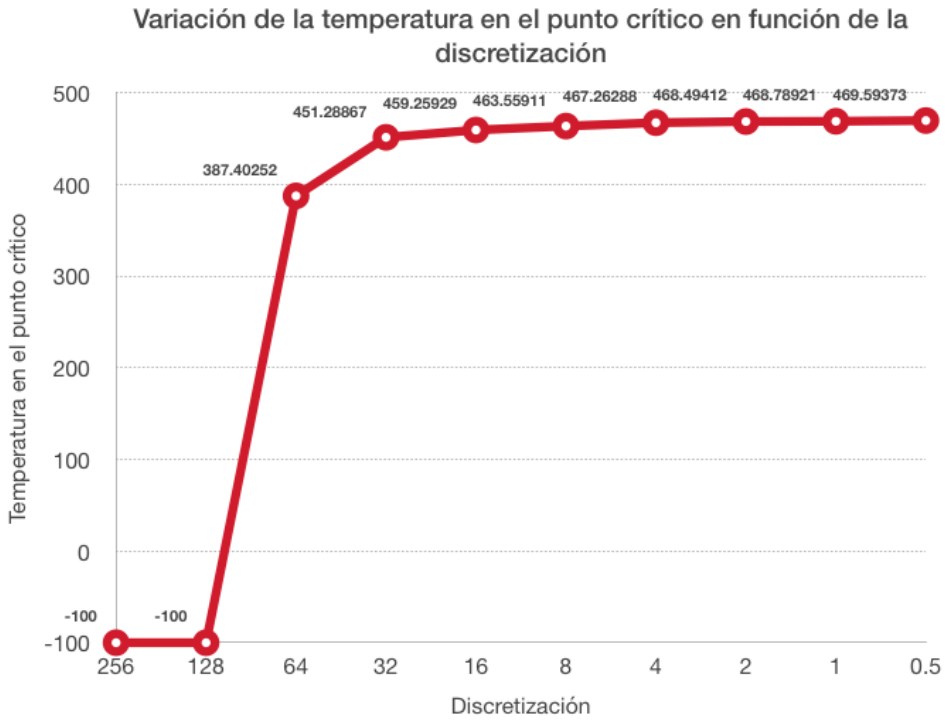


Figure 2.8: Los métodos de entrada para este caso fueron: A = 512, B = 512, H = 0.5, R = 40, Ts = 500, #S = 7



No todos los algoritmos eliminan la misma cantidad de sanguijuelas.

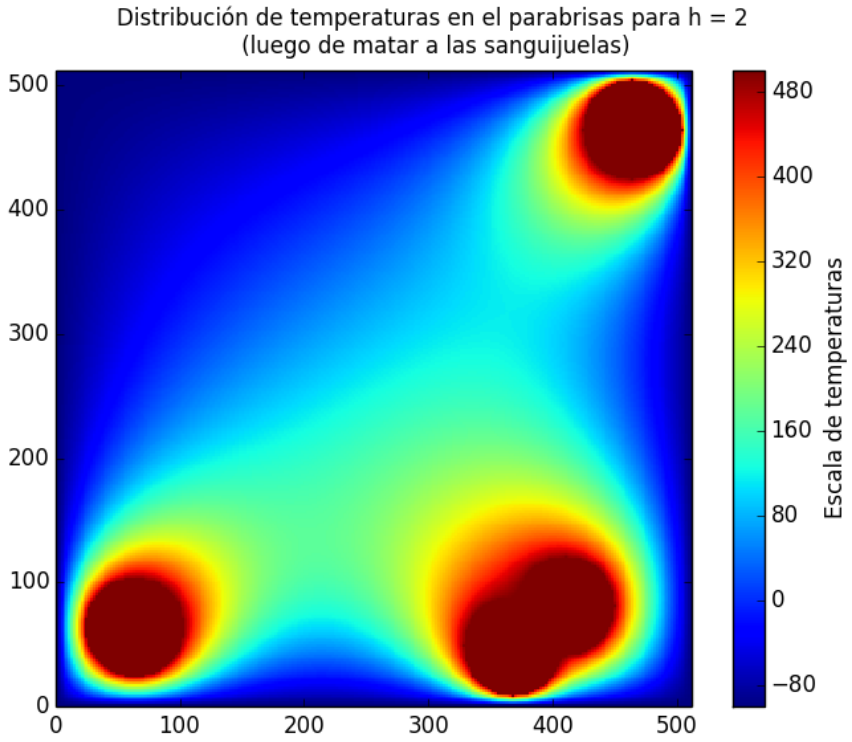


Figure 2.9: El resultado de matar sanguijuelas mediante el algoritmo de a 1

En este caso, los dos algoritmos de eliminación heurísticos no elimiman a las mismas sanguijuelas, pero si la misma cantidad.

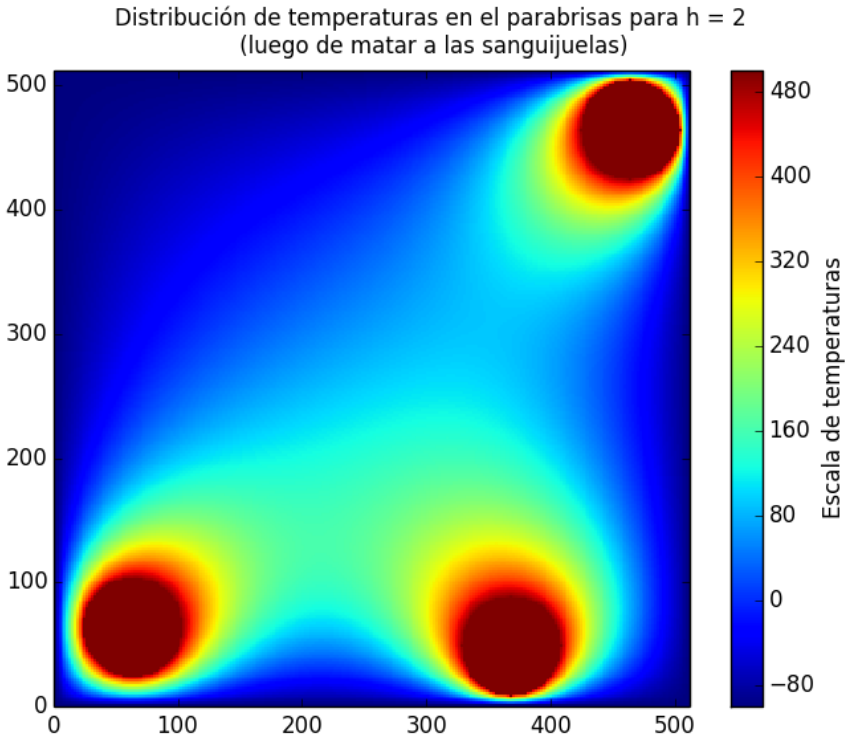


Figure 2.10: El resultado de matar sanguijuelas mediante el algoritmo de 25%

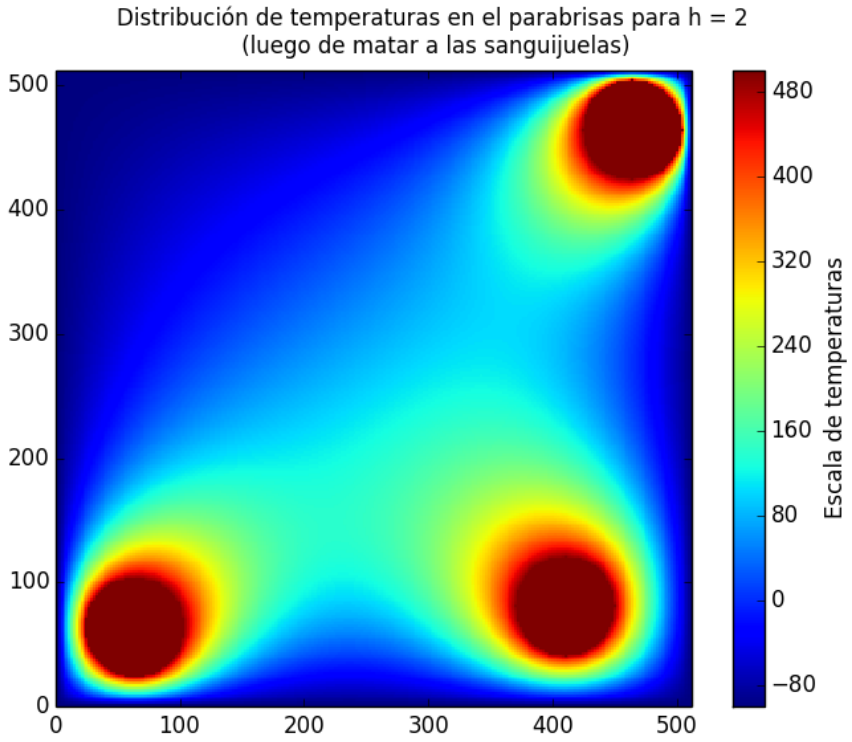


Figure 2.11: El resultado de matar sanguijuelas mediante el algoritmo de proximidad

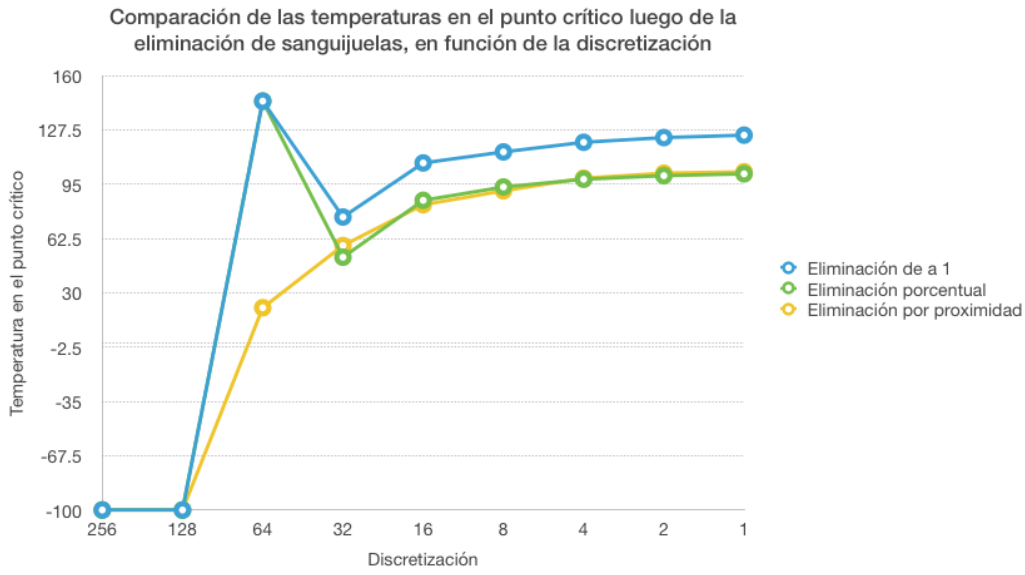


Figure 2.12:

2.3 Caso de test 3

Pusimos una temperatura suficientemente chica (apenas arriba del límite) y colocamos una sola sanguijuela cerca del punto crítico y las demás cerca de los bordes. Nuestra hipótesis fue que al influir mayoritariamente una sola sanguijuela en la temperatura del punto crítico, el algoritmo exacto iba a funcionar mejor que cualquier eurístico, ya que se matarían sanguijuelas innecesariamente.

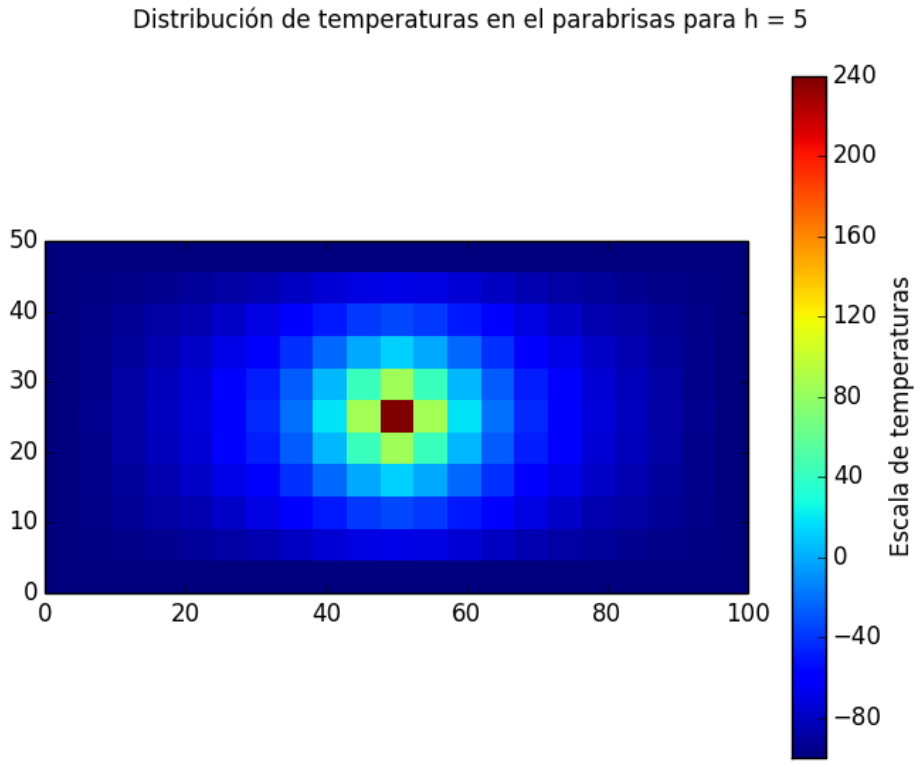


Figure 2.13: Los métodos de entrada para este caso fueron: A = 100, B = 50, H = 5, R = 1, Ts = 240, #S = 13

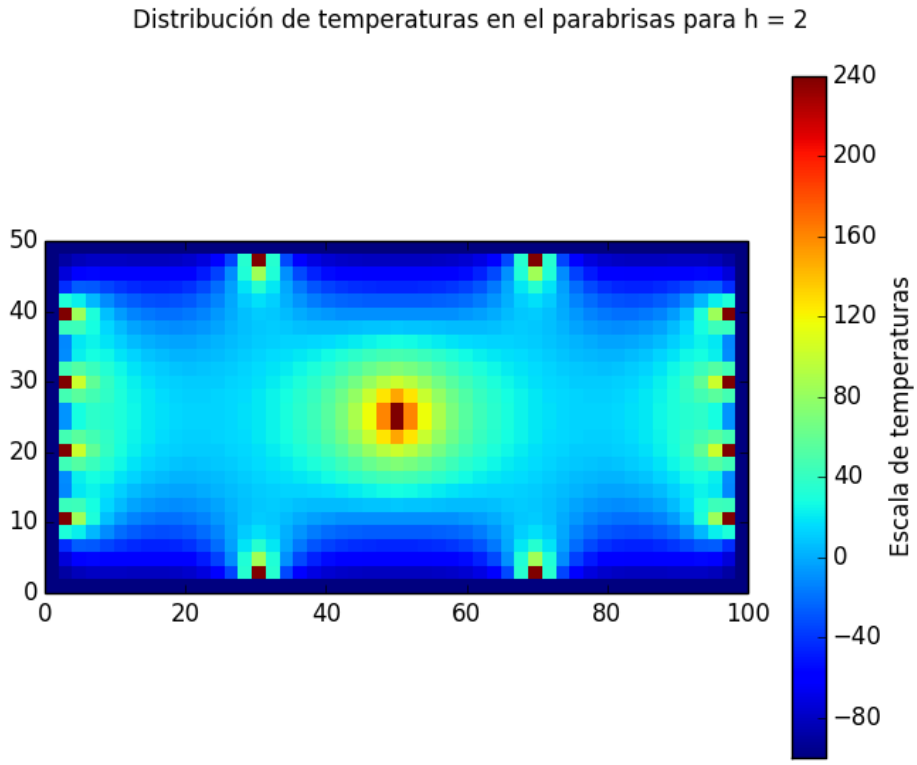


Figure 2.14: El mismo sistema cambiando el h

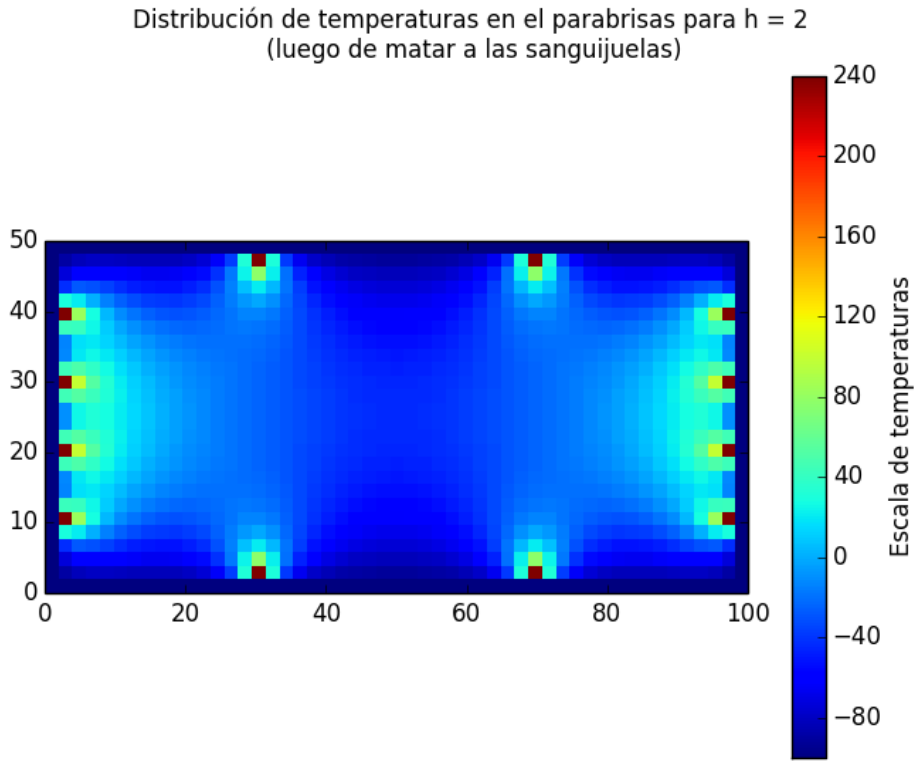


Figure 2.15: Sistema luego de matar con el algoritmo de a 1

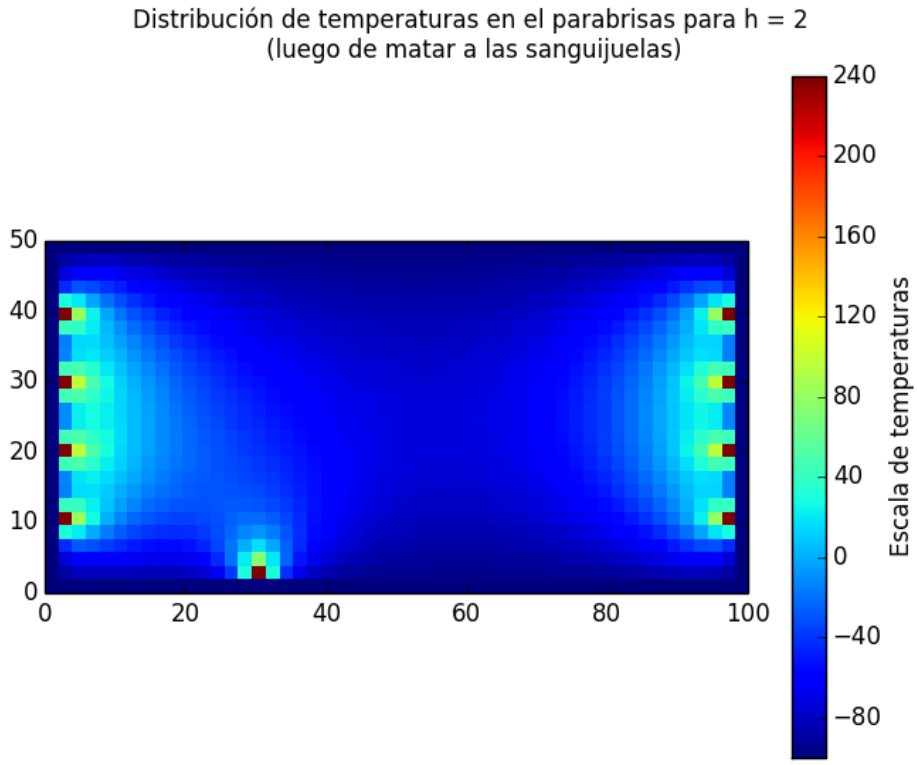


Figure 2.16: Sistema luego de matar con algún algoritmo eurístico

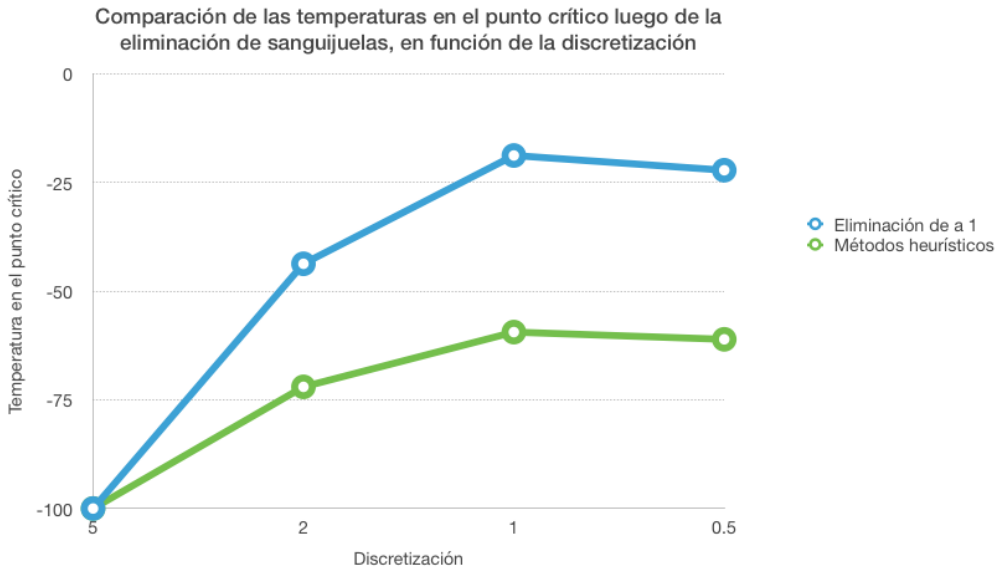


Figure 2.17:

2.4 Caso de test 4

Chapter 3: Discusión

Los resultados presentados en la sección anterior son el producto de un intento de contrastar diversas hipótesis que fuimos formulando conforme nos dedicábamos al diseo, la implementación y el desarrollo del presente trabajo.
A continuación intentaremos presentar un esbozo de las mismas, junto con una posible interpretación de los resultados obtenidos.

3.1 Test 1

Hipótesis: Cada método heurístico no mataría las mismas sanguijuelas, pero si la misma cantidad.
Discusión: Sin embargo, en todos los casos -excepto uno- ambos algoritmos eurísticos matan a las mismas sanguijuelas.
Viendo el gráfico de distribución de las sanguijuelas con la discretizacion h= 4, junto a los resultados obtenidos luego de la eliminación, se puede ver que es el ejemplo de un caso en que no convendría usar el método de eliminar de a una. Puesto que al matarla el centro queda a una temperatura riesgosa de 234.551, muy cercana a la temperatura de muerte. Si se discretizara un poco mas, el algoritmo mataría a dos sanguijuelas. Es decir, que se esta pasando por alto que en realidad el centro está

realmente a una temperatura superior a lo que se considera estable. En este caso, se hacen evidentes los peligros de utilizar una mala discretización.

3.2 Test 2

Hipótesis: Las sanguijuelas borradas iban a diferir respecto al algoritmo eurístico usado. El algoritmo del 25 % iba a ser óptimo.

Discusión: Como muestra en la figura 2.12, contrastando con la hipótesis, la temperatura del punto crítico después de resolver el problema, variando el h, se va a estabilizar antes utilizando la función de proximidad. Luego, parecen converger todas a lo mismo.

3.3 Test 3

Hipótesis: Pusimos una temperatura suficientemente chica (apenas arriba del límite) y colocamos una sola sanguijuela cerca del punto crítico y las demás cerca de los bordes. Nuestra hipótesis fue que al influir mayoritariamente una sola sanguijuela en la temperatura del punto crítico, el algoritmo exacto iba a funcionar mejor que cualquier eurístico, ya que se matarían sanguijuelas innecesariamente.

Discusión: Todos los algoritmos tardan el mismo tiempo, pero en este caso los algoritmos heuristicos son mucho peores que el de a uno: se mata cuatro veces mas

sanquijuelas que para el otro.

3.4 Test 4

Hipótesis: Discusión:

3.5 Correcta discretización

Uno de los primeros dilemas presentados fue: cómo podemos hacer frente a unos parámetros que no permiten plasmar fielmente la realidad en el modelo presentado?. Esta pregunta nació a partir de una idea que todos los integrantes del grupo inferimos desde la presentación del enunciado del trabajo práctico: en una analogía con las sumas de Riemann, donde una longitud ínfima de base de los rectángulos permite una mejor aproximación al valor real de una integral definida (de hecho, su valor se encontrará con el límite de dicha base tendiendo a cero), del mismo modo, la magnitud de error que produzca la estimación de la temperatura en un punto dado de la superficie estará íntimamente vinculada con la discretización que el usuario decida escoger.

Esta dependencia y sus riesgos se ilustran en los resultados bla bla y bla [FIGURA DE 4 SANGUIJUELITAS DEJANDO EN CRISIS AL SISTEMA PERO PASANDOSE POR ALTO]

En dicho ejemplo se aprecia la forma en que al utilizar una granularidad grosera el sistema informa de una temperatura que indica indefectiblemente que no se ha hallado sanguijuela alguna. Sin embargo, al comenzar a refinar la discretización la temperatura se dispara ante la detección de sanguijuelas que inicialmente habían pasado inadvertidas, progresivamente pasando a retornar temperaturas cada vez más estabilizadas, es decir, convergiendo a un único valor mucho más coherente con el sistema desde el punto de vista teórico.

3.6 Dependencia entre el radio y el valor de discretización

En estrecha relación con lo antedicho, deducimos también la dependencia existente entre el radio de las sanguijuelas y el valor de la discretización a partir del cual las temperaturas del punto crítico se aproximan lo suficiente a un valor de convergencia. Para definir la validez de nuestros razonamientos, realizamos dos pruebas con instancias cuyos parámetros eran idénticos, a excepción del radio de las sanguijuelas.

Ver **figure 2.4** (Caso de test 1)

De los resultados arrojados se puede interpretar que, si se quiere aproximar una temperatura en algún punto, la granularidad que se debe tomar en el caso de que el radio de las sanguijuelas sean muy pequeas debe ser más fina que aquella necesaria

para lograr un resultado aceptable en un contexto de sanguijuelas de gran tamao.

3.7 Desventajas de un h óptimo

Ahora bien, si en varios contextos la calidad de los resultados parece crecer a medida que aumenta la granularidad es decir, a medida que el h se achica - entonces por qué no se decide optar siempre por hacer empleo de la mayor granularidad posible? La respuesta está en el deseo de los rencorosos: tarde o temprano todo se paga.
Parte del costo de hacer que el modelo aproxime lo más fielmente posible a la realidad se visibiliza en las figuras.

BLABLABLABLA.

All se puede apreciar la abismal diferencia entre el tiempo de cmputo demandado por dos parabrisas idnticos modelados con granularidades dismiles.
Es necesario aclarar que la eleccción del h más chico no es arbitraria. Si se elige un nuevo h muy cercano al anterior, es muy probable que no exista una diferencia significantemente posible. Pero en el ejemplo, se optó por tomar un h que sea la mitad del anterior. Esto hace que se duplique la cantidad de columnas y la cantidad de filas, por lo tanto la cantidad de puntos discretizados va a ser cuatro veces la anterior.

3.8 Solución a problemas de tiempo de cómputo

Pero quién dijo que todo está perdido? Nosotros venimos a ofrecer una solución. Visto que las matrices con más posiciones demandan tiempos de procesamiento mayores y considerando que nos interesa poder obtener resultados de calidad, nos aventuramos a comprobar que existe al menos una manera de comprimir lo suficiente la estructura de una matriz Banda como para garantizar una precisión aceptable de los resultados sin comprometer demasiado el tiempo requerido por la computadora para procesarla. Esta forma de almacenar la matriz fue explicada en la sección Desarrollo del presente trabajo, y fue puesta a prueba en numerosas oportunidades para cerciorarnos de que la nueva estructura realmente aventajara a la matriz clásica generada con vectores de vectores.

Los resultados de estas pruebas se observan en las figuras

BLA Y BLA (x ej el de matriz vacia y las q puedan usarse para comparar TIEMPO en funcion de BANDA/COMUN aplicando o sin aplicar algoritmos)

las cuales ponen de manifiesto el tiempo empleado en la resolución y estabilización del sistema para los distintos algoritmos de eliminación de sanguijuelas en función de la estructura de base utilizada.

3.9 Gasto de energía vs gasto de tiempo

Una vez hubimos terminado de corroborar el costo temporal del cálculo de las temperaturas en las diferentes estructuras implementadas, comenzamos a idear formas algorítmicas que, de ser necesario, nos permitieran eliminar una cantidad de sanguijuelas que estuvieran impidiendo que el punto central del parabrisas se encontrara por debajo de los 235 grados C.

En vistas a optimizar el cuidado de los recursos energéticos de la nave, la primera idea que se nos ocurró fue la de fijarse qué sanguijuela se encuentra más prxima al punto crítico, eliminarla, recalcular el sistema y averiguar si el sistema está estable para decidir si seguir matando más sanguijuelas es una necesidad.

Como intuímos que iba a ser un procedimiento lento, decidimos pensar otras alternativas que pudieran emplearse en contextos de emergencia.

La presencia de un rango que acotara las temperaturas posibles de las sanguijuelas o algún dato sobre la posible distribución de las mismas hubiera facilitado la elección de un algoritmo que fuera más eficiente en determinado contexto. Sin embargo, al no contar con estos datos debimos apelar a la intuición y al desarrollo de algoritmos más generalizables a diversos contextos aunque menos específicos y, por ende, tal vez menos competente bajo determinadas condiciones.

Una de estas alternativas consistió en buscar y eliminar cada vez el 25% de las sanguijuelas que se encontraran en el parabrisas. Las más prximas al punto crítico seran las que desaparecerían y luego se recalcularía el sistema para evaluar la necesidad de aplicar la eliminación una nueva vez.

Este algoritmo en particular se basó fuertemente en el comportamiento del calor. Sabemos que mientras todas las fuentes de calor apliquen la misma temperatura, las más cercanas al punto crítico serán las que mayor injerencia tengan en la fluctuación de su temperatura.

Que la cantidad de sanguijuelas a eliminar sea proporcional a la cantidad de sanguijuelas totales nos pareció una idea sensata que se desprendía de saber que a mayor cantidad de ellas, mayor sería la temperatura en general (sin superar en ningún punto a T_s).

Luego de descartar algunos algoritmos menos interesantes desde el punto de vista experimental, incluimos uno que tena una lógica más arbitraria que heurística. Esto lo hicimos para poder contrastar los resultados del mismo con los que habamos ideado. En principio supusimos que el primer algoritmo mencionado sería más exacto y que ahorraría más energía, pero tardaría más. En contraposición, los otros dos serían menos eficientes en términos de energía , pero tardarían menos.

[VER Y ANALIZAR LOS GRAFICOS QUE COMPARAN LOS ALGORITMOS. DESCRIBIRLOS. PONER LAS CONCLUSIONES EN CONCLUSIONES!] Idea: Podemos aclarar que mas proximo esperabamos que fuera cualquiera y dio mejor de lo esperado. Funcionan relativamente parecido cuando la discretizacion es grande, porque es un mecanismo paleativo pero en cuando se empieza a hacer una granularidad mas chiquita y la precision es mayor, el metodo de porcentaje es el mejor: Ya que vas a matar 4 en cualquiera de los dos casos, es preferible que mates aquellas que aumentan mas la temperatura del punto critico.

Chapter 4: Conclusiones

Con respecto a la elección de estructuras, destacamos que elegir almacenar la matriz como la matriz "banda" es lo más acertado en cuanto complejidad espacial. Ya que esta forma de implementación no tiene ningún tipo de restricción por sobre la manera convencional.
Nuestro algoritmo de Eliminaciń Gaussiana tiene ciertas mejoras en cuanto complejidad espacial.

Calcular, o mejor dicho, estimar la temperatura del punto crítico está íntimamente relacionado con el tamaño del h y del r. Por lo visto anteriormente, si un r es muy chico para determinado h, una sanguijuela podría desaparecer y así dejar de influir con su temperatura en el punto crítico. Cuanto más chico el h, más exacta la cuenta. Cuando h tiende a cero, el punto crítico va a tener su valor real. Por eso mismo es importante, a la hora de utilizar este método, saber elegir un h donde la temperatura en el punto crítico ya este estabilizada (Ver gráfico).

ponemos algo respecto del cambio de temperatura??

A la hora de elegir el "mejor" método para eliminar sanguijuelas nos parece importante establecer un equilibrio entre gasto de energía y gasto de tiempo. Por lo

tanto, consideramos que elegir el 25% de la cantidad de sanguijuelas pertenecientes es una cota acertada por lo demostrado en resultados.

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^{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^oC, 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:

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

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

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

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 $i = 0, 1, \ldots, n$ 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:

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

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 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.

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 ← 0 to columnafinal do
2     for j ← 0 to (últimafila − k) do
3         if (matriz(i, j) > ε) then
4             m ← matriz(i,j)/matriz(i,i)
5             fila j ← fila j - m.(fila i)
```

Algorithm 5.2: EGbanda

input : Matriz
output: Void

```
1 for i ← (k + 2) to (#filas − k) do
2     j ← 0                                  // j va a recorrer las columnas
3     while j < (posición elemento de la diagonal) do
4         com ← (i + j − k − 1)              // GUS: Poner aclaración abajo.
5         if (matriz(i, j) > ε) then
6             m ← matriz(i,j)/matriz(com,PosDiagonal)
7             Resta Corrida sin la columna Resultado (fila i, fila com)
8             matriz(i, PosResult) -= matriz(com, 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 ← #filas
 4     while (i ≥ 0) do
 5         for j ←última columna to i do
 6             vectorResultado(i) -= matriz(i,j) * matriz(j,PosResult)
 7         matriz(i,PosResult) = matriz(i,PosResult)/matriz(i,j)
 8 else if (tipo == Matriz Banda) then
 9     matriz.EGbanda()
10     i ← (#filas -k -2)              // Las últimas k-2 filas ya están
       trianguladas
11     while (i > k) do
12         f ← 1
13         for j ←(k+2) to PosResult do
14             matriz(i,PosResult) -= matriz(i,j)*matriz(i+f,PosResult)
15             f++
16         matriz(i, PosResult) = matriz(i,PosResult)/matriz(i,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 ← 0 to #columnas do
4          if (columna i es un elemento del Borde) then
5              matriz(i,i)← 1
6              matriz(i,PosResult)← −100
7      for i ← 0 to tamaño(vect) do
8          Ataque(vect[i])
9      for i ← 0 to #columnas do
10         if (matriz(i,i)==0) then
11             matriz(i,i)← −4
12             matriz(i,i-1)← 1
13             matriz(i,i+1)← 1
14             matriz(i,i-a/h-1)← 1
15             matriz(i,i+a/h+1)← 1
16 else if (tipo == Matriz Banda) then
17     Redimensiono la matriz con el tamaño que necesito
18     for i ← 0 to #filas do
19         if (elemento de la fila i, columna de la diagonal es un elemento del
           Borde) then
20             matriz(i,PosDiag)← 1
21             matriz(i,PosResult)← −100
22     for i ← 0 to tamaño(vect) do
23         Ataque(vect[i])
24     for i ← 0 to #filas do
25         if (matriz(i,PosDiag)==0) then
26             matriz(i,PosDiag)← −4
27             matriz(i,0)← 1
28             matriz(i,PosDiag - 1)← 1
29             matriz(i,PosDiag + 1)← 1
30             matriz(i,PosResult - 1)← 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 ← (p1 − radio, p1 − radio)
4      IntervaloY ← (p2 − radio, p2 − radio)
5      i← (p1 − radio)
6      while (i ≤ p1 + radio) do
7          c← (p2 − radio)
8          while (c ≤ p2 + radio) do
9              norma2 ← (i − p1)^2 + (c − p2)^2
10             if (norma2 ≤ radio^2) then
11                 matriz(fila del elemento (i,c) , PosResult) ← temp
12                 matriz(fila del elemento (i,c) , PosDiag) ← 1
13     i← (p1 − radio)
14     while (i ≤ p1 + radio) do
15         c← p2 − radio
16         while (c ≤ p2 + radio) do
17             norma2 ← (i − p1)^2 + (c − p2)^2
18             if (norma2 ≤ radio^2) then
19                 matriz(fila del elemento (i,c) , PosResult) ← temp
20                 matriz(fila del elemento (i,c) , PosDiag) ← 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) ≤ ε) then
11 |  |  if (p2 − promedio(filArri,filAba)) ≤ ε 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) ≤ ε) 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
   |  |  |  |                                  // Está en el segundo cuadrante
25 |  |  |  |  return pArrIzq
26 |  |  |  else
   |  |  |  |                                   // Está en el tercer cuadrante
27 |  |  |  |  return pBajIzq
28 |  |  else
   |  |  |                          // Está en la mitad derecha del cuadrado
29 |  |  |  if (PuntoCrítico está en la parte superior) then
   |  |  |  |                                   // Está en el primer cuadrante
30 |  |  |  |  return pArrDer
31 |  |  |  else
   |  |  |  |                                   // Está en el cuarto cuadrante
32 |  |  |  |  return pBajDer
```

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