DEPARTAMENTO
DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

RTP1: Sistemas Lineales

28/11/2014 Métodos Numericos

Integrante	LU	Correo electrónico
Dellanzo, Claudia Antonella	019/13	anto_tbdt@hotmail.com
De Rocco, Federico	403/13	fede.183@hotmail.com
Tallar, Nicolás	218/13	nicot.sanlorenzo@hotmail.com

Instancia	Docente	Nota
Primera entrega		
Segunda entrega		

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

Contents

1 Introducción **1**

2 Matriz Tradicional **2**
2.1 Creación 2
2.2 Eliminación Gaussiana 3
2.3 Resolución 4

3 Matriz Banda **6**
3.1 Creación 8
3.2 Eliminación Gaussiana 9
3.3 Resolución 11

4 Punto Crítico y Eliminación de Sanguijuelas **13**
4.1 Punto Crítico 13
4.2 Eliminación de Sanguijuelas 13

5 Experimentos **16**
5.1 Granularidad 16
5.2 Tiempo de cómputo 19
5.2.1 Análisis de la forma de representar la matriz 19
5.2.2 Granularidad 22
5.3 Punto Crítico y Eliminación de Sanguijuelas 23

6 Conclusion **27**

7 Modificaciones **28**

1 Introducción

En el siguiente trabajo se presentan distintas implementaciones para la resolución de sistemas de ecuaciones lineales, utilizando el algoritmo de eliminación gaussiana y el que se utiliza para la resolución de estos sistemas cuyas matrices sean bandas, además de versiones adaptadas de estos para la forma de representar la matrices banda.

El problema que se nos plantea es el de un parabrisas (de determinado largo y ancho) que esta siendo atacado por sanguijuelas que ocasionan que dentro de un determinado radio tenga una cierta temperatura, queriendo nosotros a partir de esta información obtener la temperatura de este. Además sabemos que este parabrisas contiene un sistema de refrigeración que ocasiona que la temperatura en los bordes sea de -100 °C, y que si un punto no esta en el borde ni esta afectado directamente por una sanguijuela cumple que (siendo *T(x,y)* la temperatura en el punto *(x,y)*):

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

Con el objetivo de resolver este problema realizamos una discretización del parabrisas, teniendo en cuenta solo los puntos de la forma *(j*h,i*h)* (dado un cierto $h \in \mathbb{R}_+$ pasado como parámetro) y mediante el método de diferencias finitas determinamos que la temperatura en un cierto punto del parabrisas cumple que (siendo $t_{i,j} = T(j*h,i*h)$):

$$t_{ij} = \frac{t_{i-1,j} + t_{i+1,j} + t_{i,j-1} + t_{i,j+1}}{4}. \tag{2}$$

Es por ello que luego de todos estos pasos nos queda un sistema de ecuaciones lineales (que posee todas las ecuaciones de las temperaturas de los puntos discretizados del parabrisas), el cual debemos resolver para obtener una aproximación de la temperatura de ellos. Uno de los puntos, en el cual es primordial saber la temperatura, es el punto crítico, el punto que se encuentra en el centro del parabrisas, debido a que si su temperatura supera los 235 °C, se romperá el parabrisas.

Durante este trabajo, nosotros planteamos dos diferentes alternativas para resolver este sistema (haciendo luego un análisis para realizar una comparación entre ambas), variando la estructura para representar la matriz asociada a este así como los algoritmos utilizados. Además nos centramos en el estudio de cómo se comporta este sistema y qué dificultades encontramos al tratar de resolverlo, con cuyo fin realizamos una serie de experimentos, pensando previamente a ellos en hipótesis y en ver cuál iba a ser el resultado y luego comparando esto con lo obtenido. Por último buscamos una forma eficiente de eliminar sanguijuelas (es decir que no elimine sanguijuelas de más), analizando su comportamiento y su eficiencia, también mediante el uso de experimentos.

2 Matriz Tradicional

2.1 Creación

El objetivo de este algoritmo es crear la matriz A del sistema $Ax = b$, el cual será utilizado para hallar las temperaturas de todos los puntos discretizados del parabrisas, conteniendo las ecuaciones lineales con las temperaturas de cada punto discretizado del parabrisas. Para obtener estos puntos vamos recorriendo las filas del parabrisas de abajo hacia arriba (y cada una de ellas de izquierda a derecha). Por cada longitud h (pasada como parámetro de esta función) recorrida en el parabrisas, tomaremos esa posición como uno de nuestros puntos discretizados. De esta forma vamos obteniendo estos puntos de manera ordenada para así luego formar la matriz A que contendrá información sobre cada uno de ellos.

Este algoritmo va recorriendo estos puntos discretizados hallados y colocando en la matriz A determinados valores para cada uno de ellos (los cuales explicaremos posteriormente), cuyas dimensiones son A $\in \mathbb{R}^{n*m \times n*m}$, siendo $n = \frac{a}{h} + 1$ y siendo $m = \frac{b}{h} + 1$ (a el ancho en metros del parabrisas, b el largo en metros y h la longitud de cada intervalo de discretización), donde las filas de A representan los puntos discretizados y las columnas los coeficientes que acompañan a las incógnitas de la ecuación para obtener la temperatura de cada punto discretizado.

De esta forma obtenemos el sistema $Ax = b$, en donde la $fila_k(A)$ tiene los coeficientes acompañan a las incógnitas de la ecuación para obtener la temperatura de cada uno de los puntos del sistema, en el orden en el que los recorremos explicado anteriormente. Luego, si este punto se encuentra en la fila i y la columna j del parabrisas discretizado, la ecuación que le corresponde es (si es que no se encuentra en el borde ni posee ninguna sanguijuela arriba):

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

Lo que realizamos como paso siguiente es pasar el 4 multiplicando hacia la izquierda y luego el $4 * t_{ij}$ restando a derecha, obteniendo lo siguiente:

$$0 = t_{(i)(j-1)} + t_{(i)(j+1)} + t_{(i+1)(j)} + t_{(i-1)(j)} - 4t_{ij}.$$

De esta forma, ahora podemos describir el b_i, que es el valor asignado a dicha ecuación, el cuál será -100, t ó 0 dependiendo de si el punto al que discretiza la fila i es un punto del parabrisas del borde, tiene sanguijuela englobándolo o si no es ninguno de los anteriores, respectivamente. También podemos describir el x_k que representa a la incógnita t_{ij} del sistema hallada a través de la función *OrdenIncog*. Esta función, dado un punto *i,j* de la discretización y el n obtenido anteriormente (cantidad de puntos dicretizados sobre el ancho del parabrisas), nos devuelve un número que representa la posición en la que se encuentra ese punto buscado a través de la operación $(n * i) + j$. Es decir que si buscamos, por ejemplo, el punto $(1, 2)$ de la discretización (indexando desde 0) y tenemos que hay 5 elementos discretizados por ancho del parabrisas ($n = 5$), luego el séptimo elemento del vector x representará a la incógnita del punto discretizado buscado t_{12}.

Cuando nos encontramos con un punto borde t_{ij}, la ecuación de su temperatura estará dada por $t_{ij} = -100$, luego en el punto de la matriz que representa a dicho punto debemos colocar un 1 puesto que es el coeficiente de dicha ecuación. Lo mismo sucede en los puntos en los que hay sanguijuelas, supongamos que en el punto de la fila i y columna j del parabrisas discretizado se encuentra una de ellas, luego su ecuación de temperatura será $t_{ij} = t$, siendo t la temperatura que ejerce la sanguijuela sobre el parabrisas, debiendo colocar un 1 en el punto de la matriz que lo representa (obtenido nuevamente con *OrdenIncog*). Siguiendo estos pasos, en todos aquellos puntos que sean bordes o que contengan una sanguijuela afectándolos, en nuestra matriz veremos solo un 1 en su fila, luego si en todos los puntos hubiesen sanguijuelas o fuesen puntos bordes, tendríamos una matriz con unos en su diagonal debido a la forma que hemos elegido ordenar las incógnitas, que se encuentra relacionada a cómo se fueron tomando los puntos discretizados del parabrisas. Esta es de la siguiente manera: supongamos que tenemos una matriz $M \in \mathbb{R}^{2\times 2}$ que representa los puntos discretizados del parabrisas, $M = \begin{pmatrix} t_{11} & t_{12} \\ t_{21} & t_{22} \end{pmatrix}$. Entonces, nuestra matriz A tendrá en la primer fila los coeficientes que acompañan a las incógnitas de la ecuación de temperatura para el punto discretizado t_{11}, la segunda la de t_{12}, la tercera la del t_{21} y la cuarta la del t_{22}

Ahora veamos el caso en el que hay un punto no borde o sin sanguijuelas. Por la ecuación de Laplace vista, la temperatura de estos puntos estará dada por el promedio de las temperaturas de los cuatro puntos que la rodean (por izquierda y derecha, por arriba y abajo). Sabemos que las ecuaciones de las temperaturas de los puntos que lo rodean por izquierda y derecha en nuestra

matriz son aquellos que se encuentran por encima y por debajo, por ejemplo supongamos que el punto buscado es el t_{ij}, luego el punto que se encuentra a su izquierda lo encontramos en el $t_{(i)(j-1)}$ y el de su derecha es el $t_{(i)(j+1)}$, y, por como fue creada nuestra matriz, estos puntos se encuentran encima y debajo del punto buscado en la matriz A. Ahora basta con encontrar dónde se encuentran los otros puntos restantes. El punto que se encuentra encima del buscado está en $t_{(i-1)(j)}$ y el punto que está debajo es el $t_{(i+1)(j)}$. Luego, con nuestra función OrdenIncog, dado el i y el j podemos encontrar la posición de estos puntos en la matriz. Como los coeficientes que acompañan a estas incógnitas son 1's, basta con poner 1's en estos cuatro puntos de la matriz puesto que tenemos la ecuación de la forma:

$$0 = t_{(i)(j-1)} + t_{(i)(j+1)} + t_{(i+1)(j)} + t_{(i-1)(j)} - 4t_{ij}.$$

Para colocar el coeficiente -4 que acompaña a la incógnita t_{ij} en la matriz, solo debemos colocarlo en la fila y columna dada por la función OrdenIncog aplicada a i y a j (que se encontrará sobre la diagonal). Supongamos que OrdenIncog nos devuelve el valor l, luego tendremos en la fila y columna l un -4 y un 1 en la columna l-1, l+1, l+n y l-n de la fila l.

De esta forma hemos llegado a tener la matriz completa con todos los coeficientes que acompañan a las incógnitas de las ecuaciones de las temperaturas de cada punto discretizado, con una diagonal completa con 1's y -4's, así no encontraremos ningún cero en ella, con el resto completo de 0's salvo por como mucho 4 valores por fila, puesto que el valor de cada ecuación depende como mucho de 5 incógnitas y, por ende, solo tendremos, como mucho, 5 coeficientes de las incógnitas representados. Entonces dado los parámetros de entrada podemos crearnos una matriz que posee representados todos los puntos discretizados del parabrisas junto con los respectivos coeficientes de cada ecuación que define su temperatura.

Luego falta armar el vector de resultados, es decir el b de la ecuación $Ax = b$. Este estará formado como un vector de $\mathbb{R}^{n*m}$, conteniendo en cada fila el resultado correspondiente a la ecuación que representa. Por ejemplo, dada la ecuación de temperatura t_{ij}, para saber a qué está igualada basta usar la función OrdenIncog con el i y el j para hallar en que posición del vector se encuentra dicho valor. Cuando colocamos un punto borde, en el vector resultante deberemos colocar un -100 en la posición debida, cuando colocamos un punto con sanguijuelas debemos colocar un t y cuando nos encontramos en el caso restante deberemos colocar un 0.

A continuación se presentará un pseudocódigo de lo hablado anteriormente, siendo A y b la matriz y vector, respectivamente, del sistema $Ax = b$, y $n = \frac{a}{h} + 1$ (con a el ancho en metros del parabrisas y h la longitud del intervalo de cada intervalo de discretización):

```
for Cada fila discretizada del parabrisas do
    for Cada columna de la fila seleccionada de la discretización del parabrisas do
        if Es punto borde ∨ hay sanguijuela englobando ese punto then
            //Asumiendo que estamos en la fila k de A;
            Colocar 1 en la columna equivalente a la fila (A_kk = 1) y 0 en las restantes;
            if Es punto borde then
                b_k = −100;
            else
                b_k = t;
            end
        else
            //Asumiendo que estamos en la fila k de A;
            Colocar -4 en la columna equivalente a la fila (A_kk = −4);
            Colocar 1 en las columnas k + 1, k − 1, k − n, k + n de la fila k;
            Colocar 0 en el resto de las columnas de la fila k;
            b_k = 0;
        end
    end
end
```

Algorithm 1: Algoritmo de CrearMatriz

2.2 Eliminación Gaussiana

Es un método para poder triangular una matriz, es decir para que, dada una matriz, queden 0's debajo de toda la diagonal (sin incluirla) para que luego su resolución sea más fácil puesto que,

como cada columna representa al coeficiente de una incógnita, al tenerla triangulada, para obtener el valor de cada una de ellas solo basta ir despejando desde la última fila hasta la primera debido a que, a medida que vamos subiendo en la matriz, vamos teniendo cada vez más valores despejados. Para realizar este algoritmo, tuvimos en cuenta que, como trabajamos con aritmética finita, puede suceder que al realizar la resta para poner algún cero a los valores que están debajo de la diagonal, esta no quede con un cero sino que con algún valor muy pequeño. Por esto es que decidimos poner directamente aquellos ceros pues ya sabemos que ese valor pertenece a allí. Otro dato a tener en cuenta es cómo esta formada la matriz que triangularemos(la cual fue explicada anteriormente). Como su diagonal está formada por 1's y -4's, luego los valores que deberemos poner en 0 son aquellos que se encuentran en las filas que poseen un -4 puesto que en las demás ya hay 0's debajo de la diagonal. Ahora consideremos alguna fila de la matriz que sea de aquellas cuyo punto no era borde ni tenía una sanguijuela, es decir aquella en la que encontraremos un -4 en su fila, llamémosla i. También consideremos la fila j_1 y j_2 que serán aquellas filas menores a i ($j_1 < i$, $j_2 < i$) por la cual deberemos restársela a i para poner 0's debajo de su diagonal, y supongamos que aquellas filas fueron filas anteriormente trianguladas, es decir que sobre su diagonal (sin incluirla) quedaron dos unos (en este ejemplo supongamos que sucede esto, puede suceder que queden otros números allí puesto que pudieron haber sido modificados en pasos anteriores de la triangulación). Como al realizar la resta restaremos la fila i por algún múltiplo de la fila j_1 y j_2, la única forma de que quede un 0 en la diagonal es que esos múltiplos formen un 4 en combinación. Luego de estudiar el comportamiento de la matriz, pudimos ver que esto nunca puede suceder pues estas filas j_1 y j_2 contendrán un -4 o números fraccionarios en sus valores que se encuentren sobre la diagonal de la matriz, cuyas combinaciones nunca serán tales que nos anulen el valor de la diagonal. Por esta razón, luego de aplicar el algoritmo clásico de Eliminación Guassiana, nunca quedará un 0 en nuestra diagonal y podremos proceder a realizar la resolución de manera más sencilla. A continuación presentaremos el pseudocódigo del algoritmo descripto, siendo F una fila de la matriz y a la matriz:

for $i = 1...n-1$ **do**
  for $j = i+1...n$ **do**
    $m_{i,j} = a_{ji}/a_{ii}$;
    $F_j = F_j - (m_{ji} * F_i)$;
  end
end

Algorithm 2: Algoritmo de Eliminación Gaussiana

2.3 Resolución

Una vez triangulada la matriz, la dificultad de resolver el sistema es mucho menor. Sea A, x y b la matriz y vectores que obtuvimos al crear la matriz (del sistema $Ax = b$), nos crearemos un vector *res* en el cual devolveremos el resultado que buscamos. Deberemos avanzar por la diagonal de la matriz en sentido inverso y despejar la solución, puesto que, al estar triangulada la matriz, en el primer caso tendremos la fila con todos ceros en sus columnas menos en la última, en el segundo tendremos dos columnas no nulas (de las cuales de una ya sabemos su valor) y así sucesivamente. Sea b_n el último valor de nuestro vector b, sabemos que $A_{nn}x_n = b_n$, entonces $x_n = b_n/A_{nn}$ y aquí obtuvimos el último valor para nuestro vector *res*. Esto parece solo solucionarnos la última incógnita, pero para poder hacerlo para todos los valores de la diagonal basta con que cada vez que llegues a un resultado, hagamos el remplazo en todos los de mayor fila de los valores ya obtenidos y lo pases restando al vector b, es decir que $\forall j \in [i-1..0]$, $b_j = b_j - (A_{ji} * res_i)$. Luego, al ir avanzando por la diagonal de la matriz en sentido inverso, ya tendremos los valores de las incógnitas que se encuentran a su derecha (a la izquierda solo tendremos ceros) y solo basta poner en la posición correspondiente del vector *res* su valor, el cual será, suponiendo que estamos en la fila i, $\frac{b_i}{A_{ii}}$. En esta parte se podría complicar la resolución si hubiera ceros en la diagonal por la triangulación ya que esto nos provocaría que en un caso de las variables no se pueda encontrar un resultado. Pero como fue explicado anteriormente en el apartado de Eliminación Gaussiana, nunca nos encontraremos en tal caso, por lo que nuestro algoritmo nunca se encontrará con tal dificultad.

A continuación se presentará un pseudocódigo de lo descripto anteriormente, siendo A y b la matriz y vector, respectivamente, del sistema $Ax = b$, *res* el vector que contendrá el resultado final y n la cantidad de filas de la matriz A:

```
for i = n − 1..0 do
    res_i = b_i / A_ii
    for j =i−1..0 do
        b_j = b_j − (A_ji ∗ res_i)
    end
end
```

Algorithm 3: Algoritmo de Resolver

3 Matriz Banda

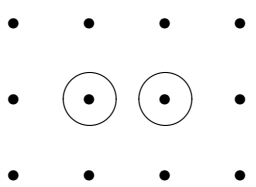
Al analizar la estructura que tiene la matriz de las ecuaciones lineales obtenida luego de la creación de esta, utilizando el método explicado anteriormente, se puede ver que como la funciones son de la forma:

$$t_{i,j} = t \quad t_{i,j} = \text{-}100 \quad o \quad t_{i-1,j} + t_{i,j-1} + t_{i,j+1} + t_{i+1,j} \text{-} 4 * t_{i,j} = 0$$

Luego, en una misma fila de la matriz de las ecuaciones lineales va a haber como máximo cinco coeficientes distintos de cero (los que acompañan a $t_{i-1,j}$, $t_{i,j-1}$, $t_{i,j}$, $t_{i,j+1}$, $t_{i+1,j}$). Es por ello que luego de realizar los algoritmos y de representar la matriz de la forma que normalmente se usaría nos pusimos a pensar de que manera se podría representar de forma tal que ahorremos tiempo y espacio.

La primera idea que tuvimos es de representarla utilizando una matriz perteneciente a $\mathbb{R}^{n*m\times 5}$ guardando en cada fila únicamente los coeficientes distintos de cero, pero nos dimos cuenta que de esta manera no se podía realizar la eliminación gaussiana al tratar de resolver el sistema. Esto se debe a que al realizar las restas entre las filas (durante este algoritmo) había coeficientes que no podiamos guardar debido a que no acompañaban a ninguno de las incógnitas mencionadas anteriormente.

Esto se puede ver en el caso que el parabrisas con el que trabajemos sea de la siguiente forma (siendo los círculos negros sin relleno las posiciones en las que estan ubicadas las sanguijuelas):



Para el caso anterior, la matriz obtenida mediante la aplicación del método utilizado para la creación de la matriz de ecuaciones lineales tradicional (el explicado anteriormente) es la siguiente:

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

En este caso, si representaramos esta matriz utilizando una perteneciente a $\mathbb{R}^{12\times 5}$ (tal como habiamos hablado previamente) deberíamos tener en la fila i guardado los valores de las columnas *i-3*, *i-1*, *i*, *i+1* y *i+3* de la matriz anterior (en el caso de que algunos de estos valores sea menor a 1 o mayor a *n*m*, en este caso 12, colocaremos un 0 en esa posición). Al realizar este método para la matriz mencionada anteriormente se obtiene la siguiente matriz:

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

Pero el problema es que al realizar la eliminación gaussiana de la primera de estas matrices obtenemos una que no puede ser guardada de la forma explicada anteriormente, debido a que en una de las posiciones que nosotros no guardaríamos hay valores distintos de cero. Esto se debe a que la matriz obtenida mediante la eliminación gaussiana es:

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -4 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -\frac{15}{4} & 1 & 0 & \frac{1}{4} & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Esta matriz no puede ser guardada en una perteneciente a $\mathbb{R}^{12\times 5}$ de la forma propuesta, debido a que en la fila *7* necesitariamos guardar el valor que esta en la columna *7+2*, el cual no guardaríamos con el método propuesto. Es por esto que decidimos no utilizar este método para ahorrar espacio al guardar las matrices con las que trabajamos y pensar en otro para usar.

Al analizar lo ocurrido con la primera de las ideas, nos dimos cuenta de que era necesario tener representados todos los coeficientes que acompañen a las incógnitas que están entre las que teníamos representadas en la idea anterior. Esto quiere decir que, dado el orden que establecimos entre las incógnitas para la construcción de la matriz tradicional (para el cual construimos la función *ordenIncog* y del cual hablamos cuando explicamos como realizamos la construcción de la matriz de forma tradicional), estando en la fila i representamos los coeficientes que acompañan a las incógnitas que estan en la posición desde *i-n* hasta *i+n* (incluyendo *i-n* y *i+n*), siendo la incógnita x un $t_{i,j}$ tal que *ordenIncog* aplicado a este punto dé como resultado x. Realizamos esto teniendo en cuenta que si $i < n$ o $i+n > m*n$ entonces, como va a haber lugares en la fila que no representen a un coeficiente, a estos le ponemos el valor cero. Luego para obtener una fila de la matriz banda lo que hacemos es a partir de la misma fila de la matriz tradicional tomar las n columnas antes y después de la que posee el valor de la diagonal (incluyendo a este último). En conclusión la matriz que se obtiene posee *2*n+1* columnas por lo que esta es pertenece a $\mathbb{R}^{n*m\times 2*n+1}$ y posee en la columna n los coeficientes que acompañan a la diagonal de la matriz (si es que numeramos la primera columna a partir del cero).

Una propiedad importante por la cual podemos únicamente guardar estos valores de la matriz y no tener nunca ningún problema al realizar la eliminación gaussiana, como el que teniamos cuando guardabamos solo 5 valores de cada fila, es que la utilización de este método mantiene la banda de las matrices. Es decir, que si una matriz solo poseía n valores a la izquierda y derecha de la diagonal distintos de cero (con el resto de la matriz llena de ceros), entonces luego de cada paso de eliminación gaussiana, sigue valiendo esto. Es por esta razón que al realizar la eliminación gaussiana a una matriz que podiamos guardar de la forma planteada, se sigue podiendo guardar (es decir que todos los coeficientes que no guardamos en la matriz valen cero).

Con el fin de poder implementar esta forma de almacenar y manipular matrices tuvimos que realizar unas modificaciones a los algortimos que permiten pasar del parabrisas, con sanguijuelas en determinados puntos, a la temperatura en cada punto del parabrisas (cambiando lo realizado para cuando manipulamos la matriz de forma tradicional). Estos son los algoritmos de creación, eliminación gaussiana y de resolver el sistema para la matriz banda, manteniendo siempre las mismas ideas que se implementaron en los algortimos para la matriz tradicional pero adaptándolos para poder usarlos para esta matriz (principalmente realizando un cambio en los índices).

3.1 Creación

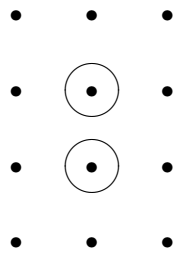
El objetivo de este algoritmo es que dados los datos de entrada del programa (a,b, la posición de las sanguijuelas entre otros) obtener la matriz banda que guarda todas las ecuaciones lineales del sistema, realizando lo explicado previamente. Para esto, realizamos lo siguiente:

INPUT: $a \in \mathbb{R}, b \in \mathbb{R}, h \in \mathbb{R}, r \in \mathbb{R}, t \in \mathbb{R}, x \in \mathbb{R}^k . y \in \mathbb{R}^k$
OUTPUT: $A \in \mathbb{R}^{m*n \times 2*n+1}, b \in \mathbb{R}^{m*n}$
$m = \frac{b}{h} + 1;$
$n = \frac{a}{h} + 1;$
for *Cada fila i discretizada del parabrisas (empezando de la de más abajo y moviendose para arriba)* **do**
  for *Cada columna j de la fila seleccionada de la discretización del parabrisas (comenzando desde la de más a la izquierda y moviendose para la derecha)* **do**
    $\alpha = ordenIncog(i, j, n);$
    if *En ese punto del parabrisas hay una sanguijuela* $\vee$ *esta en el borde del parabrisas* **then**
      $A_{\alpha,n} = 1;$
      if *Ese punto esta en el borde del parabrisas* **then**
        $b_\alpha = -100;$
      else
        $b_\alpha = t;$
      end
    else
      $A_{\alpha,0} = 1;$
      $A_{\alpha,n-1} = 1;$
      $A_{\alpha,n} = -4;$
      $A_{\alpha,n+1} = 1;$
      $A_{\alpha,2*n} = 1;$
      $b_\alpha = 0;$
    end
  end
end

Algorithm 4: Algoritmo de crear matrices banda

Entonces lo que hacemos es, al igual que para la matriz tradicional, recorrer todos los puntos del parabrisas y ver que ecuación corresponde para ese punto. Para obtener esta, vemos si es que hay una sanguijuela en este punto o si se trata de un punto en el borde del parabrisas, y de esa forma obtenemos los coeficientes que acompañan a las incógnitas, los cuales ubicamos en la matriz banda de la forma explicada anteriormente. Al ir recorriendo los puntos en el mismo orden que estan ordenadas las incógnitas en nuestra matriz, cuando modificamos la fila correspondiente a ese punto correctamente para cada uno de ellos, queda la matriz banda construida de forma correcta. Para esto aprovechamos que la clase vector viene inicializada en 0 y luego sólo modificamos los valores que son distintos de cero (cinco como mucho), viendo la posición que deben ocupar en base al punto que estamos analizando y por como es la estructura de la matriz banda.

Un ejemplo de como funciona este método se puede ver si poseemos el siguiente parabrisas (siendo los puntos negros los obtenidos mediante la discretización y los círculos sin relleno la posición de las sanguijuelas):



En este caso obtendríamos las siguientes matrices:

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

Como se puede observar en el ejemplo, para poder guardar menos espacio de la matriz lo único que hacemos es únicamente guardar la banda de la matriz, lo cual sería en nuestro caso guardar 3 de los valores a la izquierda de la diagonal y 3 a la derecha para cada fila (en general serían n). De esta forma guardamos únicamente los coeficientes necesarios para operar con la matriz sin problemas y ahorrar el máximo espacio posible.

3.2 Eliminación Gaussiana

Luego de ver cómo crear la matriz banda, nos pusimos a ver como haríamos para solucionar el sistema de ecuaciones lineales representado en la matriz banda. Para esto lo que hacemos es utilizar el algoritmo clásico de eliminación gaussiana (el mismo que utilizamos para la matriz tradicional) salvo que adaptado a que se trata de una matriz banda, además de realizar una pequeña modificación a este. Con este fin, implementamos el siguiente algoritmo:

INPUT/OUTPUT: $A \in \mathbb{R}^{m*n \times 2*n+1}, b \in \mathbb{R}^{m*n}$
INPUT: $n \in \mathbb{N}, m \in \mathbb{N}$;
$f = 0$;
$c = 0$;
while $f < n * m$ **do**
  //Buscamos una columna ¸en la fila f en la que el coeficiente sea distinto de 0;
  while $c < n \wedge A_{f,c} == 0$ **do**
    $c = c + 1$;
  end
  if $c < n$ **then**
    //Hacemos que el coeficiente en $A_{f,c}$ *sea 0, calculando* $Fila_f = Fila_f - m * Fila_{f-(n-c)}$ *(teniendo en cuenta que las filas estan almacenadas de manera distinta a la tradicional)*;
    $m = \frac{A_{f,c}}{A_{f-(n-c),n}}$;
    $A_{f,c} = 0$;
    for $i = (c+1)...(c+n)$ **do**
      $A_{f,i} = A_{f,i} - (m * A_{f-(n-c),n+(i-c)})$;
    end
    $b_f = b_f - (m * b_{f-(n-c)})$;
    $c = c + 1$;
  else
    $f = f + 1$;
    $c = 0$;
  end
end

Algorithm 5: Eliminación Gaussiana matriz Banda

Tal como explicamos antes, el objetivo de este algoritmo es de llevar a cabo la eliminación gaussiana para matrices representadas mediante la estructura alternativa presentada. Este posee una pequeña diferencia en comparación con el implementado para las matrices representadas tradicionalmente y con el algoritmo de eliminación gaussiana, la cual esta explicada en la sección de mejoras.

Como el objetivo de la eliminación gaussiana es que la matriz pase a ser triangular superior, esto se logra con las matrices banda cuando las primeras n columnas quedan con sus valores en cero. Con este fin, en nuestro algoritmo iremos focalizandonos en cada una de las filas, poniendo en cero los valores necesarios antes de pasar a la siguiente. Para hacer esto, en cada paso vamos a poner un nuevo coeficiente de la matriz en cero (de la fila con la que estamos trabajando) si es que hay alguno que debamos poner en cero (es decir alguno localizado en las primeras n columnas). Si esto no es cierto, pasamos a la siguiente fila y seguimos con la ejecución del algoritmo hasta haber recorrido todas las filas. Para poner un cero en la posición *(f,c)* de la matriz banda, lo que hacemos es realizar la siguiente cuenta:

$$m = \frac{A_{f,c}}{A_{f-(n-c),n}}$$
$$Fila_f = Fila_f - m * Fila_{f-(n-c)}$$

Esta se realiza teniendo en cuenta que las filas de las que hablamos son las que representarían las guardadas en la matriz banda, es decir, tomando los valores guardados y agregando los ceros a izquierda y derecha que le corresponderían tener si representaramos a la matriz de forma tradicional. Entonces, al hacer esta cuenta debemos restar el elemento i de la fila f de A con el elemento *n+(i-c)* de la fila *f-(n-c)* (siendo este último multiplicado por m).

Con esta cuenta nosotros queremos realizar algo similar a lo que se hace en cada uno de los pasos de la eliminación gaussiana tradicional, es decir dada la posición *(i,j)* en la que nos encontramos en la matriz, encontrar la fila j para asi poder restar ambas filas y que quede un cero en esta posición de la matriz (todo esto considerando la matriz que se obtendría si se buscaría como representar tradicionalmente la matriz banda). Pero como en este caso no queremos realizar este pasaje de como esta representada la matriz (porque sino no aprovecharíamos al máximo los beneficios de la utilización de las matrices banda), tenemos que buscar alguna forma de ver que fila debemos utilizar. Para esto nos aprovechamos que el número de filas de esta forma de representar la matriz

es el mismo que si utilizaramos el método tradicional. Entonces, si quisieramos que quede un cero en la posición *(f,c)* de la matriz banda, si c fuera igual a n entonces j sería igual a f (porque como *c==n (i,j)* pertenece a la diagonal de la matriz representada de forma tradicional). Entonces, la diferencia entre c y n es igual que la diferencia entre j y f por lo que llegamos a la conclusión que la fila que debemos utilizar con el objetivo de que haya un cero en la posición *(f,c)* es la fila *f-(n-c)*. Entonces, luego de haber encontrado esta fila, lo único que debemos hacer para que quede un cero en esta posición es restar ambas filas (teniendo en cuenta que estas no estan representadas tradicionalmente) multiplicando los valores de la fila *f-(n-c)* por un m tal que al hacer la operación quede un cero donde deseamos (esto se logra con el m mostrado anteriormente). Como las filas no estan representadas tradicionalmente, debimos encontrar una manera de modificar los indices que utilizamos para poder realizar la resta deseada. Para realizar esta, lo primero que hicimos fue, como sabiamos que luego de esta operación iba a quedar un cero en la posición *(f,c)*, poner un cero ahí sin realizar la cuenta. Para hacer el resto de las cuentas aprovechamos que lo contenido en la columna c de la fila f y lo de la columna n de la fila *f-(n-c)* quedarían en la misma columna si se representaría tradicionalmente la matriz (por ser *f-(n-c* la columna en la que estaría lo que actualmente esta en la columna *(f,c)* entonces el elemento de la diagonal de esta fila también lo estaría). Entonces para hacer la resta, solo basta restar lo que esta en la posición *(f,i)* con lo que esta en la posición *(f-(n-c), n+(i-c))* (con i $\in c+1, .., c+n$).

Entonces, luego de que se complete este algoritmo, la matriz banda quedara con las n primeras filas llenas de ceros, es decir que quedará triangular superior (si es que consideramos como sería la matriz tradicional a partir de la matriz banda que queda).

Un detalle importante que tuvimos en cuenta a la hora de hacer este algoritmo es que debido a que estamos trabajando con aritmética finita es posible que cuando restemos números iguales, pero que contienen operaciones en el medio, el resultado puede no ser cero, asi que puede haber problemas y quede en alguna de las primeras filas un valor distinto de cero. Es por ello que decidimos que en vez de realizar una operación que sabemos que tiene resultado cero, le asignamos cero a donde debemos, siendo este el caso cuando restamos las filas en nuestro algoritmo.

3.3 Resolución

El último algoritmo necesario para poder obtener las temperaturas del parabrisa es el de resolver, este algoritmo lo que hace es que a partir de una matriz triangular superior se pueda obtener las temperaturas de cada punto del parabrisas, adaptado esto a como están representadas las matrices en este caso. Para esto tenemos como objetivo que queden unos en los coeficientes de la diagonal (es decir unos en la fila n de la matriz banda) y ceros en el resto de la matriz. Entonces, si logramos esto mediante operaciones con las filas, obtendríamos en el vector b (que utilizamos para guardar el resultado de cada ecuación representada en la matriz, en el mismo orden que en el que están en ella) las temperaturas de cada uno de los puntos obtenidos al realizar la discretización de la matriz. Con este objetivo, implementamos la siguiente función:

INPUT: $n \in \mathbb{N}, m \in \mathbb{N}$;
INPUT/OUTPUT: $A \in \mathbb{R}^{m*n \times 2*n+1}, b \in \mathbb{R}^{m*n}$
for *Cada fila i de la matriz A (empezando desde la última y subiendo con cada ciclo)* **do**
  //Como en la fila i ya despejamos casi todas las incognitas, queda en esta la equación $A_{i,n}x_i = b_i$ *por lo que solo debemos despejar esta*;
  $b_i = \frac{b_i}{A_{i,n}}$;
  $A_{i,n} = 1$;
  //Teniendo el valor de la incógnita, lo despejamos en el resto de las ecuaciones;
  for *Cada fila j de la matriz A (desde la i-1 hasta la primera)* **do**
    if *El coeficiente que acompaña a la incognita en la fila i esta en la banda* **then**
      $b_j = b_j - (A_{j,n+(i-j)} * b_i)$;
      $A_{j,n+(i-j)} = 0$;
    end
  end
end

Algorithm 6: Algoritmo para resolver matrices banda

En esta, utilizamos el mismo algoritmo que para la matriz tradicional, despejando las incógnitas desde la última fila hasta la primera teniendo en cuenta que los indices que utilizamos se tienen

que modificar por la estructura con la que representamos las matrices. Esto lo tuvimos que tomar en cuenta para poder obtener el valor de $T_{j,i}$ en la estructura utilizada al realizar el despeje de las incógnitas (con T la matriz obtenida si representamos la matriz banda tradicionalmente). Algo que aprovechamos para la realización de esto es que siempre valía $i \geq j$, por lo que la columna a la que debemos acceder en la matriz banda es mayor o igual a n. Es por esto que la distancia a la diagonal de la columna a la que queremos acceder se puede obtener mediante el cálculo de *i-j*. Esto se debe a que vale que si *i=j+k* entonces, como *(j,j)* pertenece a la diagonal, k sería cuantas columnas más a la derecha de la diagonal queremos acceder. Es por esta razón que para acceder a $T_{j,i}$, debemos acceder al valor de *(j, n+(i-j))* de la matriz banda.

En conclusión, mediante este método, logramos encontrar las temperaturas de cada uno de los puntos del parabrisas discretizado, quedando estos en b.

Algo que sigue teniendo importancia para la realización de este algoritmo (al igual que para el que utilizamos en la matriz tradicional) es que luego de aplicar la función de eliminación gaussiana (llamada *EGBanda*) no quedan ceros en la diagonal por como son las filas de nuestra matriz (es decir en la columna n de nuestra matriz). Si esto no ocurriera, entonces habría un problema en nuestro algoritmo al realizar la siguiente cuenta:

$$b_i = \frac{b_i}{A_{i,n}}$$

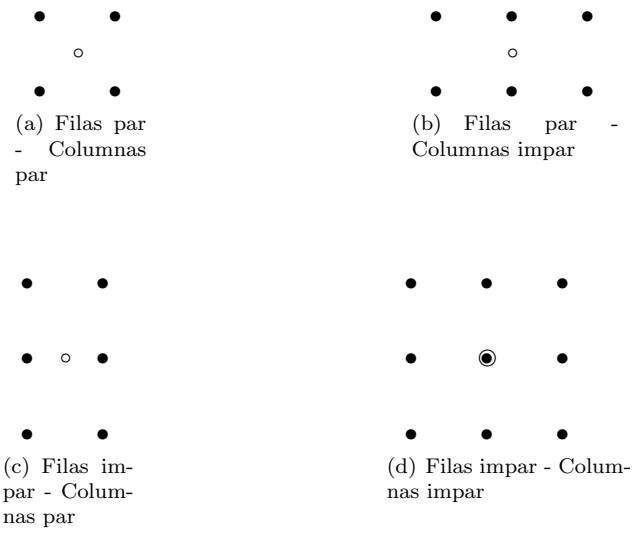
Esto se debe a que si existiera algun i para el cual $A_{i,n}$ fuera cero entonces estariamos dividiendo por cero al realizar la cuenta descripta anteriormente.

Como esto no ocurre, entonces la función puede ser ejecutada siempre sin ningún problema, logrando que siempre quede en b las temperaturas para cada uno de los puntos del parabrisas discretizado.

4 Punto Crítico y Eliminación de Sanguijuelas

4.1 Punto Crítico

El punto crítico es aquel que se encuentra en el centro del parabrisas. Para estimar su temperatura hemos decidido tomar dos caminos. En el primero, vemos si el punto crítico se encuentra dentro de uno de los puntos de la discretización, en cuyo caso la temperatura buscada es la temperatura del punto encontrado. Este caso se da cuando la cantidad de filas y columnas de la matriz que contiene a todos los puntos de la discretización es impar. Luego, basta con tomar el punto del medio de la matriz para estimar la temperatura del punto crítico. El segundo camino es el caso contrario al anterior, en el cual hemos decidido tomar el promedio de las temperaturas de los puntos de la matriz (la que contiene las discretizaciones) cercanos al punto crítico, teniendo en cuenta de que estos puntos cercanos varían dependiendo el tamaño de la matriz. Cuando nos encontramos en el caso en el que la cantidad de filas y columnas de la matriz es par, el punto crítico se encuentra en el centro de cuatro puntos. Luego, basta con tomar el promedio de las temperaturas de estos cuatro puntos para estimar la del punto crítico. El caso restante se da cuando la cantidad de filas es impar y la de las columnas es par, o viceversa (cantidad de filas par y cantidad de columnas impar). Cuando nos encontramos aquí, el punto crítico tiene dos puntos cercanos, los cuales se encuentran vertical u horizontalmente dependiendo del caso en el que nos encontremos. Luego basta con tomar el promedio entre ambos puntos para estimar la temperatura del punto crítico. Luego, como hemos cubierto todos los posibles casos que se pueden dar para estimar la temperatura del punto crítico, podemos decir que tenemos una estimación para éste. Todos estos casos nombrados se pueden observar en la siguiente figura, en la cual los círculos vacíos representan al punto crítico:



(a) Filas par - Columnas par

(b) Filas par - Columnas impar

(c) Filas impar - Columnas par

(d) Filas impar - Columnas impar

Figure 1: Ejemplos puntos críticos

Luego de hablar la forma que usamos para obtener esta temperatura, nos gustaría mencionar el porqué de nuestra decisión. Si bien cuando el punto crítico cae dentro de nuestros puntos discretizados es claro porqué hemos elegido la temperatura de ese punto, de los restantes casos (los cuales son 3) hemos decidido tomar el promedio porque consideramos que es una buena estimación de lo buscado puesto que, por ejemplo, para encontrar la temperatura de algún punto que no sea borde ni contenga sanguijuela lo que hacíamos era tomar el promedio de los puntos que se encuentran rodeándolos. Luego, al tomar el promedio de los puntos que se encuentren más cercanos a éste, creemos que es una buena forma de aproximar la temperatura que puede llegar a tener el punto crítico.

4.2 Eliminación de Sanguijuelas

Cuando el parabrisas de la nave El Pepino Marino está siendo atacado por sanguijuelas mutantes, puede suceder que la temperatura del punto crítico sea mayor o igual a 235 °C. Cuando esto sucede,

el parabrisas no resiste la temperatura y se destruye. Por esta razón existe el método de eliminación de sanguijuelas, el cual consiste en verificar si la temperatura del punto crítico está en estado grave (es decir sobrepasa los límites propuestos) y, si lo hace, destruir la menor cantidad posible para que el parabrisas no se desruya. No debemos eliminar de más ya que eliminarlas consume energía y necesitamos que el capitán Guybrush Threewood llegue a destino. A continuación hablaremos del método elegido para realizar lo propuesto, seguido de un pseudocódigo.

El primer paso del método es asegurarse de que la temperatura del parabrisas está en estado grave. Para eso calculamos las temperaturas y vemos cuál es la del punto crítico. Una vez asegurado esto, el paso siguiente es decidir cuáles sanguijuelas eliminaremos. Lo primero que nos fijamos es si hay alguna sanguijuela sobre el punto crítico (este caso puede darse por lo explicado anteriormente sobre cómo estimar la temperatura del punto crítico). Si sucede esto, lo que debemos hacer es eliminar todas las sanguijuelas que se encuentren sobre el punto crítico y recalcular su temperatura. Si la temperatura sigue siendo mayor o igual a 235 °C, luego proseguiremos con el algoritmo. Para esto, nos posicionaremos en el punto crítico del parabrisas (el punto central). Sea h la longitud de cada intervalo de discretización (la misma h que recibimos como parámetro de entrada), consideraremos inicialmente un radio que sea igual a h. Con este radio lo que haremos es ir recorriendo cada punto que se encuentra a un radio menor o igual a éste desde el punto crítico y fijándonos si hay alguna sanguijuela que lo afecta, puesto que estas sanguijuelas son las que están provocando que el punto crítico tenga dicha temperatura. Esto es realizado con el algoritmo dameSang, que además de devolverme un int con la cantidad de sanguijuelas que se encuentran dentro del radio, coloca un valor particular en aquellas sanguijuelas que se encuentran más lejanas al radio para así saber que estas sanguijuelas no tienen que ser eliminadas puesto que no son en las que estamos interesados actualmente (puede suceder que estas sanguijuelas estén afectando la temperatura del punto crítico, pero nos encargaremos de ellas posteriormente). Luego de haber encontrado todas las sanguijuelas que se encuentran dentro del radio desde el punto crítico, eliminaremos una y recalcularemos todas las temperaturas para ver si la del punto crítico disminuyó de los 235 °C. Si exitosamente logramos que sea menor a esa temperatura, el algoritmo terminará, sino seguiremos ejecutándolo buscando más sanguijuelas y eliminándolas si es necesario. Si ya eliminamos todas las sanguijuelas más cercanas al punto crítico (esto fue asegurado al tomar inicialmente el radio igual a h) y la temperatura no disminuyó del valor deseado, o no encontramos ninguna sanguijuela dentro de ese radio, luego lo que haremos será agrandar el radio en $\frac{h}{4}$ para encontrar las siguientes sanguijuelas más cercanas al punto crítico (fuera de las que ya eliminamos, es decir buscaremos de las que no fueron eliminadas las siguientes más cercanas). La razón por la que se aumenta el radio en $\frac{h}{4}$ es debido a que de esta forma se pueden obtener los siguientes puntos más cercanos de la discretización (a esta conclusión se llegó después de haber experimentado con distintos tamaños y viendo qué puntos discretizados obteníamos), si seleccionásemos un número mayor, por ejemplo h, estaríamos eligiendo más puntos de los que queremos, puesto que queremos seleccionar solo los más cercanos que se encuentran a la misma distancia a medida que aumentamos el *radio*. Luego, el algoritmo seguirá ejecutándose hasta que la temperatura haya bajado de los 235 °C; siempre que encuentre una sanguijuela la eliminará, recalculará las temperaturas para hallar la del punto crítico y seguirá eliminado si es necesario. Finalmente devolveremos el vector resultante que fuimos obteniendo a través de las iteraciones para poder obtener el vector con las temperaturas buscadas.

A continuación presentaremos un pseudocódigo que resume aquello de lo que hablamos anteriormente:

```
Calcular temperaturas de los puntos discretizados;
Calcular temperatura del punto critico;
if Hay Sanguijuelas en el punto critico then
    Eliminarlas todas;
    Calcular temperaturas de los puntos discretizados;
    Calcular temperatura del punto crtico;
end
Radio = h;
while Temperatura es mayor o igual a 235°C do
    Buscar sanguijuelas que se encuentren a radio radio;
    if Si hay alguna then
        Eliminar una;
        Calcular temperaturas de los puntos discretizados;
        Calcular temperatura del punto critico;
        if Si no quedaron mas sanguijuelas dentro de este radio then
            Radio = Radio + h/4;
        end
    else
        Radio = Radio + h/4;
    end
    Devolver las temperaturas finales halladas;
end
```

Algorithm 7: Algoritmo de Eliminacion de Sanguijuelas

5 Experimentos

Luego de la realización de todos los algoritmos necesarios para la resolución del sistema de ecuaciones lineales (teniendo en cuenta las dos posibles estructuras para representar la matriz) pasamos a estudiar su comportamiento y a realizar ciertas comparaciones importantes como analizar las dos formas de representar la matriz.

5.1 Granularidad

En primer lugar empezamos analizando nuestro programa estudiando el impacto que el parámetro de entrada h tiene sobre los resultado de nuestro programa, es decir ver como la granularidad de nuestra discretización ocasiona que obtengamos resultados diferentes. Con el objetivo de ver esto realizamos una serie de experimentos, en cada uno de los cuales íbamos variando la granuralidad manteniendo todos los demás parámetros intactos, y observando principalmente en estos el efecto que este cambio tiene en las temperaturas obtenidas como resultado y en la temperatura del punto crítico.

Entre otros algunos de los experimentos realizados fueron:

- Primer experimento:

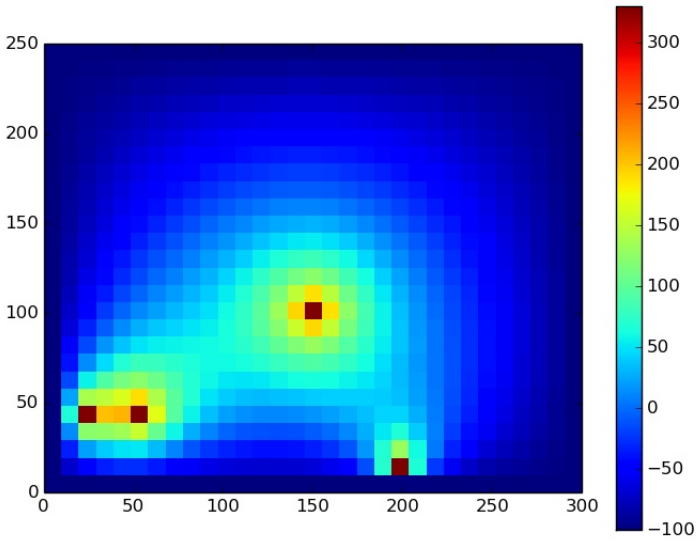


Figure 2: Datos usados: a=300, b=250, h=10, r=5, t=330 y k=5.

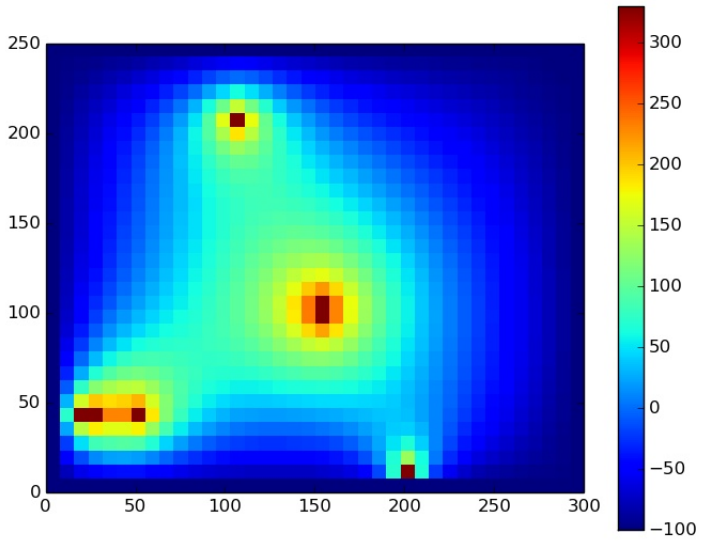


Figure 3: Datos usados: a=300, b=250, h= 8, r=5, t=330 y k=5.

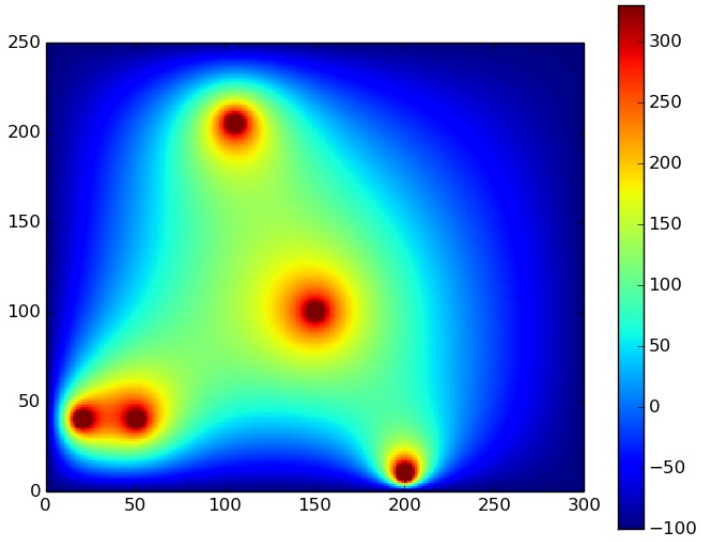


Figure 4: Datos usados: a=300, b=250, h=1, r=5, t=330 y k=5.

- Segundo experimento:

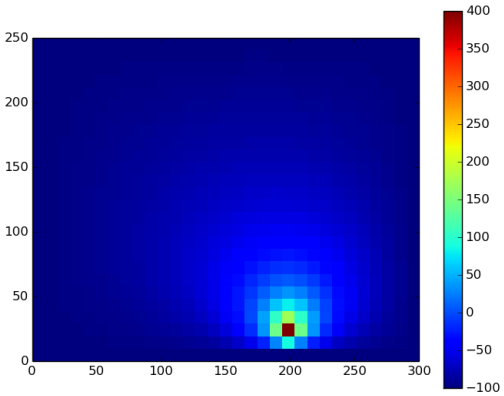


Figure 5: Datos usados: a=300, b=250, h=10, r=5, t=400 y k=4.

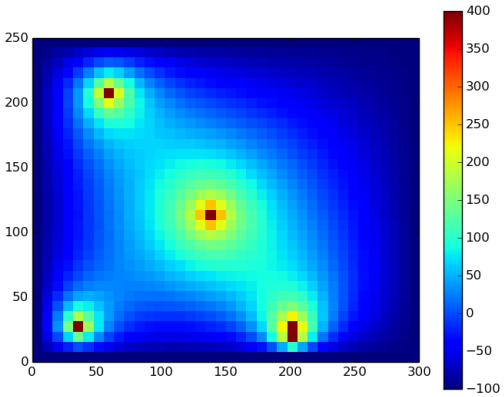


Figure 6: Datos usados: a=300, b=250, h=8, r=5, t=400 y k=4.

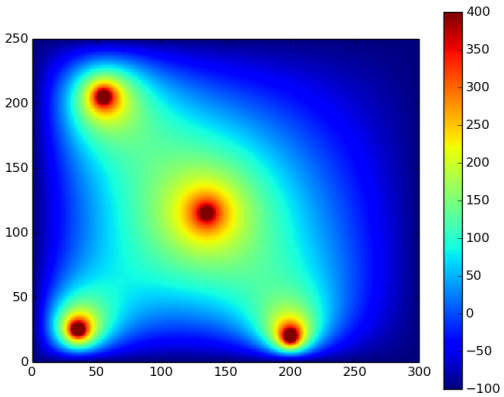


Figure 7: Datos usados: a=300, b=250, h=1, r=5, t=400 y k=4.

A pesar de haber realizado varios experimentos más aparte de estos 2, elegimos mostrar estos, con esos parámetros utilizados, debido a que la información mostrada por los otros experimentos hechos corrobora lo que concluimos a partir de estos, sin aportar más información relevante. Es

por esto que lo ocurrido en los experimentos mostrados es lo que en general sucedío con los demás. En estos, seleccionamos los paramétros de forma tal que se puedan observar claramente el impacto que tiene realizar modificaciones en el h, principalmente eligiendo la cantidad y posición de las sanguijuelas, junto con su radio con este objetivo en mente. Aparte, elegimos los diferentes h con el fin de que se pueda observar mejor el impacto que tiene realizar modificaciones en este, mostrando lo ocurrido con el mismo experimento con distintos valores para este paramétro.

Al analizar los experimentos realizados pudimos ver que la granularidad afecta notablemente las temperaturas obtenidas. Una de las razones de esto es que, tal como se puede observar en el experimento 1, puede ocurrir que al ser el h demasiado grande hay sanguijuelas que no se toman en cuenta porque no afectan a ninguno de los puntos obtenidos a partir de la discretización (lo cual sucede en este experimento con la sanguijuela que se encuentra más arriba que todas las otras). Además pudimos ver que cuanto menor es el h, mayor es la cantidad de puntos que se toman en cuenta en la discretización por lo tanto es más cercano a lo que realmente ocurre con el parabrisas. La razón de esto último es que al realizar las diferencias finitas para obtener las ecuaciones lineales, se puede observar que cuanto menor es el h, más cerca son las aproximaciones de la derivada con este método por lo que más cercana es la temperatura obtenida. En el experimento 2 se puede observar claramente que ocurre esto, debido a que en este cuando $h=10$ la temperatura del punto crítico es completamente errónea debido a que el radio afectado por cada sanguijuela es menor al h.

Como vimos, el cálculo de la temperatura mejora para todos los puntos dentro de la discretización realizada al disminuir el h, siendo especialmente importante que lo mismo ocurra con el punto crítico. La razón por la que la temperatura de este punto es tan importante es que si esta es demasiado alta se romperá el parabrisas, por lo que es de mucha importancia que su cálculo sea efectuado correctamente. El motivo por el cual mejora su estimación si disminuye el h es que, al igual que con los demás puntos, esto ocasiona que se tomen una mayor cantidad de puntos en la discretización ocasionando que más cercano se encuentre el parabrisas discretizado al parabrisas real.

En conclusión, pudimos observar en los experimentos realizados acerca de la granularidad utilizada que el uso de h distintos, pueden tener un impacto notable sobre lo cercano que se encuentra el modelo que estamos representando (la discretización realizada) de lo que realmente ocurre. Esto se debe a que cuanto menor sea el valor de este parámetro, mejores serán las estimaciones realizadas, por lo que no solo sera más cercana la temperatura calculada para cada punto discretizado con respecto a la del parabrisas real, sino que también del punto crítico.

5.2 Tiempo de cómputo

Luego de los anteriores experimentos, pasamos a centrarnos en analizar diferentes aspectos de nuestros algoritmos realizacionados con el tiempo que estos tardan en ejecutarse. Primero, al ya tener dos formas alternativas de representar las matrices con las que trabajamos pasamos a comparar estas en base al tiempo y al espacio que cada una de ellas requiere para su funcionamiento, centrandonos especialmente en ver si es que pudimos aprovechar que las matrices con las que trabajamos son matrices banda. Es decir, tratamos de ver si es que mediante nuestra representación alternativa de las matrices de este estilo pudimos ahorrar tiempo y/o espacio en nuestros algoritmos. Aparte, como ya analizamos el impacto que la granularidad tiene sobre la calidad de las temperaturas obtenidas, quisimos ver como este modifica el tiempo de ejecución utilizado. Con el objetivo de realizar apropiadamente estos experimentos, repetimos 25 veces cada uno, calculando el promedio de los tiempos en segundos obtenidos (sacando para esto 4 outliers, es decir los 2 menores valores obtenidos y los 2 mayores). La razón por la que realizamos de esta forma nuestros experimentos es que puede ocurrir que alguna de las veces que los realizamos haya algo externo al programa que estamos corriendo (otro programa o un aumento o disminución de la velocidad del CPU) que ocasione que alguna de las mediciones realizadas sea muy distinta del valor real, estas las quitamos al sacar los outiliers.

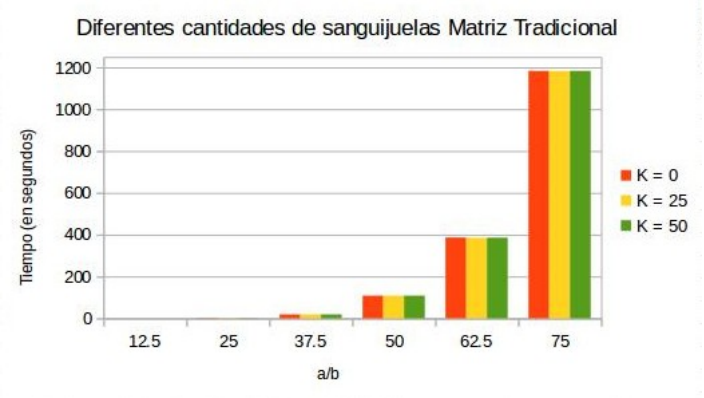
5.2.1 Análisis de la forma de representar la matriz

Empezamos primero comparando las dos maneras alternativas que tenemos de representar las matrices y los algoritmos que utilizamos para trabajar con ellas. Para esto, comparamos dos distintos aspectos de estos, el tiempo y el espacio requerido para que estos funcionen. Empezamos primero analizando el tiempo, para lo cual realizamos una serie de experimentos, variando diferentes

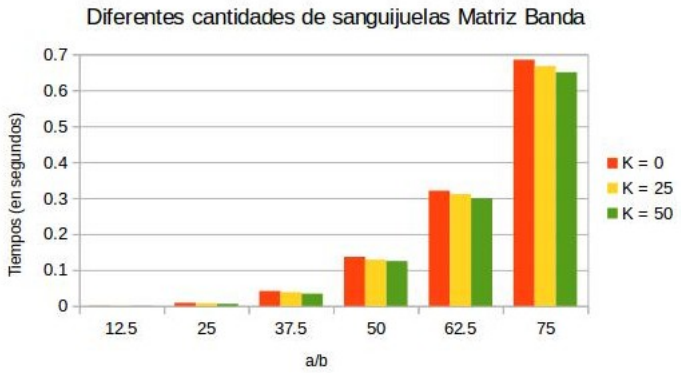
parámetros del programa, centrandonos principalmente en modificar la cantidad de sanguijuelas y el tamaño del parabrisas con el que trabajamos, manteniendo los demas parámetros constantes. La razón por la que realizamos esto es que, al analizar los algoritmos utilizados pudimos ver que la temperatura no modifica en nada el tiempo de cómputo debido a que solo ocasiona un cambio en el valor inicial de b. Por otro lado, el radio de las sanguijuelas no lo modificamos debido a que la cantidad de puntos del parabrisas discretizado afectados por sanguijuelas la modificamos con la cantidad de estas últimas, no con su tamaño. Además, el tamaño de las matrices con las trabajamos lo modificamos realizando cambios con el a/b y no con el h por lo que este lo mantenemos constante. Esto se debe a que el h modificaría el tamaño al aumentar o disminuir los valores de n y m, lo cual explicamos con más detalle al realizar los experimentos de tiempo con respecto a la granularidad.

Teniendo en cuenta lo anterior, realizamos diferentes experimentos, variando en estos el tamaño de las matrices con las que trabajamos (cambiando el valor de a y de b) así como también la cantidad de sanguijuelas y el método para representar las matrices. En estos, con el fin de que las sanguijuelas esten distribuidas uniformemente en el parabrisas, establecimos su posición aleatoriamente. Al leer los algoritmos implementados para cada uno de los métodos realizados para representar las matrices, nosotros preveíamos que al realizar estos experimentos iba a haber una gran diferencia en el tiempo que tarda en realizarse cada uno de los experimentos para cada uno de estos. La razón por la que pensabamos esto es que, a pesar de que parece no haber tanta diferencia entre lo realizado en el algoritmo para resolver las matrices, en los otros dos si que hay diferencias notorias. En primer lugar, la principal diferencia en el algoritmo de creación de la matriz es que para las matrices tradicionales no aprovechamos el hecho que los vectores vienen inicializados en cero, por lo que debimos recorrer toda la matriz para incializarla, algo que no realizamos para la matriz banda. Por otro lado, debido al menor tamaño de las matrices banda, en el algoritmo de eliminación gaussiana debimos realizar menos operaciones puesto que solo operamos con los coeficientes guardados en la matriz en ambos casos (que en el caso de la matriz banda es una cantidad mucho menor). Aparte, en el caso de la matriz banda, hay veces que no realizamos ninguna operación muy costosa si no es necesario, es decir si un coeficiente ya esta en cero no realizamos ninguna operación con este, mientras que para las matrices tradicionales siempre realizamos una resta entre las filas. Además, la resta de filas es una operación mucho más costosa en el caso de tratarse de una matriz tradicional debido a que las filas en estas son de largo $n*m$ mientras que para las matrices bandas es de $2*n+1$, aparte de que al realizar la resta para estas últimas solo restamos n coeficientes (debido a que aprovechamos que gran parte de la matriz se encuentra en cero).

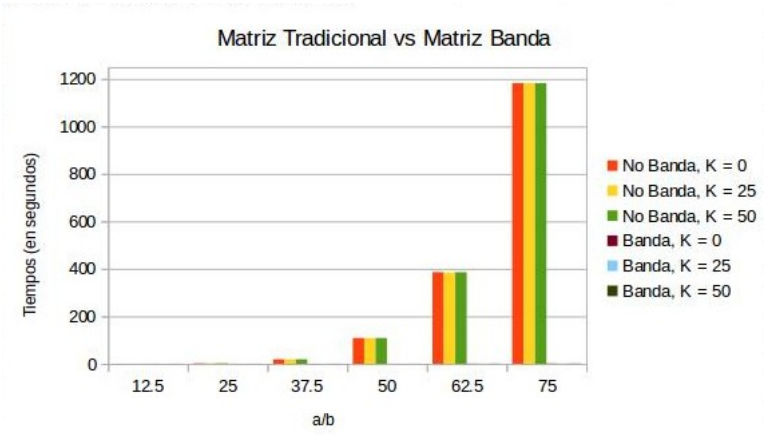
Por estas razones es que nosotros creíamos que iba a haber una gran diferencia de tiempo al comparar el tiempo de cómputo requerido para ambos métodos, sin importar el valor de los demas parámetros. Luego de realizar una serie de experimentos, algunos de los resultados obtenidos fueron los siguientes (estos no fueron los únicos realizados pero si los que reflejan claramente lo ocurrido en general:



(a) Matriz Tradicional



(b) Matriz Banda



(c) Matriz Tradicional vs Matriz Banda

Figure 8: Tiempos Matriz Tradicional vs Matriz Banda

Tal como se puede en los gráficos, nuestra hipótesis era correcta, debido a que se puede ver que hay una amplia diferencia en el tiempo tardado para cada los dos métodos, pudiendo no verse el tiempo tardado por en el caso de la matriz banda en comparación con el uso de la matriz tradicional. Por otro lado, también se puede observar que parece haber una relación cuadrática entre el tamaño de las matrices y el tiempo que tardan los algoritmos en realizarse, debido a que este va aumentando muy rápidamente a medida que aumentamos el tamaño de las matrices, siendo esto más notorio para la representación tradicional de las matrices. Aparte, algo muy importante que se puede observar en los gráficos es que a medida que vamos aumentando la cantidad de sanguijuelas, hay una disminución en el tiempo de cómputo requerido para realizar las operaciones con la matriz banda (aunque esta cantidad no es tan grande), manteniendose este valor con poca variación para la matriz tradicional.

Luego de ese análisis pasamos a ver que tal era el ahorro de espacio utilizando el método de representación alternativo para las matrices en vez del tradicional. Al ver los algoritmos vimos que para la matriz tradicional, como guardamos todos los datos, esta pertenece a $\mathbb{R}^{n*m \times n*m}$ es decir que debe utilizar $(n^2)*(m^2)$ lugares para guardar la información. A diferencia de esta, la matriz banda es de $\mathbb{R}^{n*m \times 2*n+1}$ entonces ocupa $n*m*(2*n+1)$ (lo que es igual a $2*(n^2)*m+n*m = O(n^2*m)$. Luego representando la matriz de esta forma se ahorra memoria.

En conclusión pudimos ver, luego de este análisis, que la representación alternativa de las matrices supera en todo aspecto a la tradicional, siendo mucho mejores sus algortimos en cuanto al tiempo y al espacio, sin importar el tamaño de matriz con el que trabajemos. Es por esto que recomendamos la utilización de esta forma alternativa de representar la matriz antes que la tradicional. Por otro lado, también pudimos una ver disminución en el tiempo requerido a medida que aumentabamos la cantidad de sanguijuelas en el parabrisas para las matrices banda (sin modificar ningún otro parámetro), así como también la existencia de una relación cuadrática entre los tiempos requeridos por cada uno de los métodos y el tamaño de las matrices.

5.2.2 Granularidad

Luego, continuamos analizando que ocurría al modificar la granularidad con la que trabajamos en cada una de las formas de representar la matriz. Con este fin, en nuestros experimentos dejamos fijos la cantidad, posición y radio de cada una de las sanguijuelas (tomando siempre 25 sanguijuelas de radio 1 con su posición elegida aleatoriamente), haciendo lo mismo con la temperatura de las sanguijuelas pero cambiando el tamaño del parabrisas con el que trabajamos como así también la granularidad. La razón por la que utilizamos una cantidad de sanguijuelas constante es que, como vimos en el experimento anterior, la cantidad con la que trabajemos tiene muy poca influencia en el tiempo que tardan los algoritmos. Además elegimos su posición aleatoria debido a que de esta manera las sanguijuelas estan distribuidas uniformemente en el parabrisas (lo que igual no tiene tanta importancia por lo anteriormente mencionado). Por otro lado, mantuvimos constante la temperatura que las sanguijuelas ocasionan que tenga el parabrisas debido a que esta no influye en el trabajo de nuestros algoritmos.

Teniendo en cuenta todo lo anterior, realizamos una serie de experimentos viendo en estos que ocurría al trabajar con parabrisas con distintas discretización (es decir con un h distinto), analizando esto para ambas formas de representar la matriz y diferentes tamaños de esta. Nuestra hipótesis al realizar estos experimentos era que sin importar como vayamos variando estos últimos (es decir la forma de representar la matriz y su tamaño), siempre iba a ocurrir que el tiempo de cómputo aumente a medida que disminuyamos el h (si es que mantenemos todos los demas parámetros constantes). La razón por la que pensabamos esto es que el tamaño de las matrices con las que trabajamos es inversamente proporcional al h, lo que se debe a que el tamaño de las matrices sea $\mathbb{R}^{n*m \times n*m}$ o $\mathbb{R}^{m*n \times 2*n+1}$, valiendo:

$$m = \frac{b}{h} + 1$$
$$n = \frac{a}{h} + 1$$

Luego, al disminuir h aumentan n y m por lo que a su vez también lo hace el tamaño de la matriz en ambos casos, lo cual ocasionaría que el tiempo de cómputo requerido para trabajar con esta sea mayor. Es por esta razón que nosotros supusimos que un h menor iba a ocasionar que, para ambas formas de representar la matriz y para cualquier tamaño, el tiempo que los algortimos tarden en ejecutarse sea mayor. Luego para probar si nuestra hipótesis era correcta realizamos una serie de experimentos, siendo algunos de ellos:

(a) Matriz Tradicional

(b) Matriz Banda

Figure 9: Granularidad

A pesar de haber realizado otros experimentos aparte de los mostrados en el informe, en todos los realizados obtuvimos siempre los mismos resultados, por lo que se puede observar en los casos mostrados lo que en general ocurre. Entonces, la razón por la que en estos elegimos h=*1.5*, h=*1.75* y h=*2* es debido a que muestran lo que ocurren para diferentes h y porque si variabamos demasiado su valor entonces la diferencia en tiempo se hacia tan grande que no se podia observar con claridad lo que ocurría con los tiempos (al menos en los gráficos, en los valores obtenidos los resultados eran los mismos).

Al analizar lo ocurrido, pudimos ver que ocurría lo que nosotros habiamos pensado, es decir que sin importar el tamaño de la matriz con la que trabajemos ni si esta es la matriz banda o matriz tradicional, siempre aumenta el tiempo de cómputo a medida que disminuimos el h (a pesar de que algunos de estos no se pueden ver con claridad en los gráficos). Otra cosa importante de mencionar es que este experimento sigue confirmando más lo ocurrido en el experimento de comparación de las distintas formas de representar las matrices. Esto se debe a que hay una amplia diferencia entre el tiempo que tarda en realizarse los experimentos entre ambas formas de representar la matriz, siendo siempre el tiempo superior para la representación tradicional de la matriz, ocurriendo esto aún cuando comparamos para diferentes h con el mismo tamaño.

5.3 Punto Crítico y Eliminación de Sanguijuelas

En este apartado se discutirá sobre el comportamiento de la estimación del punto crítico y el algoritmo de eliminar sanguijuelas. Si bien ya hemos explicado como realizamos ambos métodos, nos gustaría discutir sobre si son eficientes o no, y cuáles son sus ventajas y desventajas.

Luego de los experimentos realizados, se puede ver que al tener intervalos de discretización más pequeños, la temperatura de cada punto va a ser más acertada y que al estimar la temperatura del punto crítico vamos a estar más cerca del valor real. Por ejemplo, si es que nos encontramos en el caso en el que el punto crítico cae dentro de unos de los puntos discretizados, si nuestros intervalos de discretizar son pequeños, vamos a estar muy cerca del valor real, pero si estos intervalos son

muy grandes, entonces no van a estar muy bien representados los valores de las temperaturas y la del punto crítico no va a ser muy parecida al valor real buscado. Algo similar sucede en el caso en el que el punto crítico no cae dentro de los puntos de discretización. Como estamos tomando el promedio, si nuestro intervalo es tan pequeño que cada valor de la temperatura de cada punto se acerca mucho al de la realidad, luego al tomar el promedio de estos puntos también estaremos cerca de encontrar el valor de la temperatura del punto crítico. En cambio, si el intervalo de discretización es muy grande, como los puntos no están reflejando muy bien la temperatura real, entonces la temperatura del punto crítico tampoco va a estar muy bien reflejada. Por ende, se puede concluir que la temperatura del punto crítico que estimamos depende más del tamaño del intervalo de discretización que del método propuesto.

Cuando nos encontramos en el algoritmo de eliminación de sanguijuelas, es un método aproximado para realizar lo pedido y no exacto. Uno de las desventajas que tenemos, lo cual afecta en la exactitud, es que se puede dar el caso en el que, cuando el punto crítico no cae sobre un punto de la discretización, puede suceder que dado 2 puntos cercanos (cuando m era par y n era impar o viceversa, m era impar y n era par. También puede suceder esto cuando son 4 los puntos cercanos, solo exhibimos un ejemplo.), llamémoslos x e y, x tenga muchas sanguijuelas que lo engloben mientras que y solo contenga una. Dependiendo de como estén pasadas en el .in que tomamos como archivo de entrada estas sanguijuelas, nuestro algoritmo lo que puede llegar a hacer es primero eliminar todas las sanguijuelas que se encuentren sobre el punto x menos una, en este caso cada punto x y y tienen solo una sanguijuela actualmente, y lo que sucedió es que la temperatura de nuestro punto crítico todavía no disminuyó, dado que las sanguijuelas encimadas no acumulan su temperatura, y recién en el próximo paso disminuir al eliminar la siguiente sanguijuela. Por esta razón es que este método no es exacto y, a veces, no muy eficiente. Una forma de corregir este problema sería primero verificando de todos los puntos cercanos al punto crítico, aquel que tenga menos sanguijuelas englobándolo y primero eliminar todas esas; de esta forma nos aseguramos que el punto crítico disminuya su temperatura lo más rápido posible para que así el parabrisas no se destruya. También otra forma de solucionarlo podría ser parándonos en cualquiera de estos puntos cercanos y eliminar todas las sanguijuelas que estén afectándolo actualmente en un paso. Otro problema que tiene nuestro algoritmo es que repetitivamente vamos buscando aquellas sanguijuelas que se encuentran dentro del radio r, lo que implica que si nos dieron x sanguijuelas cuya ubicacíon es dentro de este radio, entonces x veces estaremos recorriendo los vectores que contienen las sanguijuelas buscando aquellas que se encuentren cercanas y siempre encontraríamos las mismas. Esto se podría resolver modificando un poco nuestro algoritmo de la siguiente manera (reemplazando nomás lo que se encuentra dentro del *while*):

entero cantidadSanguijuelas;
Buscar sanguijuelas que se encuentren a radio *radio* y colocar la cantidad encontrada en el entero cantidadSanguijuelas;
while *Hay sanguijuelas en el radio* **do**
    Eliminar una;
    Disminuir cantidadSanguijuelas;
    Calcular temperaturas de los puntos discretizados;
    Calcular temperatura del punto critico;
    if *Temperatura* $< 235°C$ **then**
        cantidadSanguijuelas = 0;
    end
end
Radio = Radio + $\frac{h}{4}$;

Algorithm 8: Mejora del algoritmo de Eliminacion de Sanguijuelas

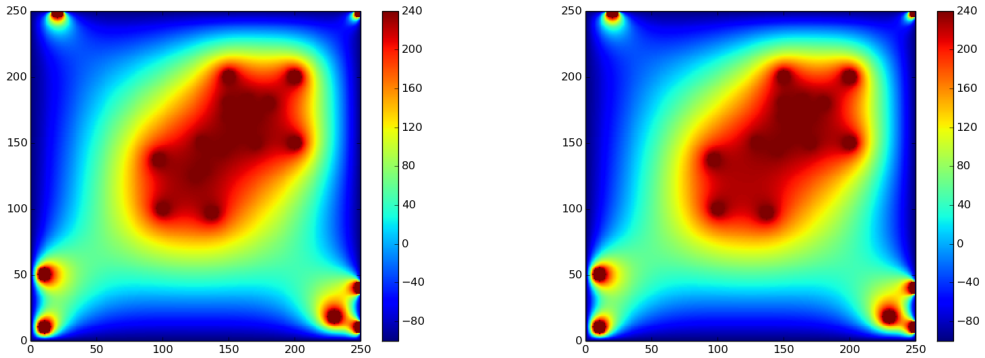
Las ventajas de nuestro método es que, al crearnos un radio alrededor del punto crítico e ir aumentándolo, esto nos asegura de ir agarrando primero los puntos más cercanos a éste, luego ir agarrando los siguientes más cercanos y etc. De esta forma, siempre vamos a ir eliminando las sanguijuelas que más directamente estén afectando la temperatura del punto crítico, para así poder eliminar todas las más cercanas necesarias para que vaya disminuyendo la temperatura. Cabe aclarar que si realizacemos lo de encontrar primero los puntos con menos sanguijuelas de aquellos puntos más cercanos con sanguijuelas, o eliminar de cada punto todas las sanguijuelas que lo afecten de un solo paso, luego nuestro método sería mucho más eficiente y podríamos colocar esto dentro de las ventajas de nuestro método puesto que estaríamos más acertados con cuáles

sanguijuelas eliminasemos.

Si bien como explicamos anteriormente este método puede no funcionar bien en algunos casos, si la distribución de las sanguijuelas fuese uniforme, habría menos probabilidad de caer en el caso en el que sobre uno de los puntos cercanos hubiese muchas sanguijuelas y sobre los otros pocas, y que nuestro algoritmo primero eliminase todas las sanguijuelas sobre ese punto que contiene mayor cantidad. Esto se debe a que estarían mejor distribuidas alrededor de todo el parabrisas, entonces al suceder esto la probabilidad de que muchas caigan sobre un punto sería baja y también la probabilidad de que aquellas que vinieron encimadas hayan sido dadas consecutivamente (puesto que nuestro algoritmo de eliminación va eliminando aquellas que primero encontró en en el vector x e y). Entonces si nos encontrasemos en este caso, nuestro algoritmo de eliminación de sanguijuelas sería mucho más efectivo ya que no caeríamos tan frecuentemente en aquellos casos en los que no funciona tan bien.

Luego de realizar el análisis anterior, vamos a hablar de los experimentos realizados sobre nuestro algoritmo de eliminar sanguijuelas para evaluar su funcionamiento. Lo que se buscaba observar era que efectivamente estuviese eliminando las sanguijuelas cercanas al punto crítico y no las que no lo afectaban, y además que la temperatura final del punto crítico no se encuentre en peligro. Por esto, esto es lo que esperamos ver en las imágenes.

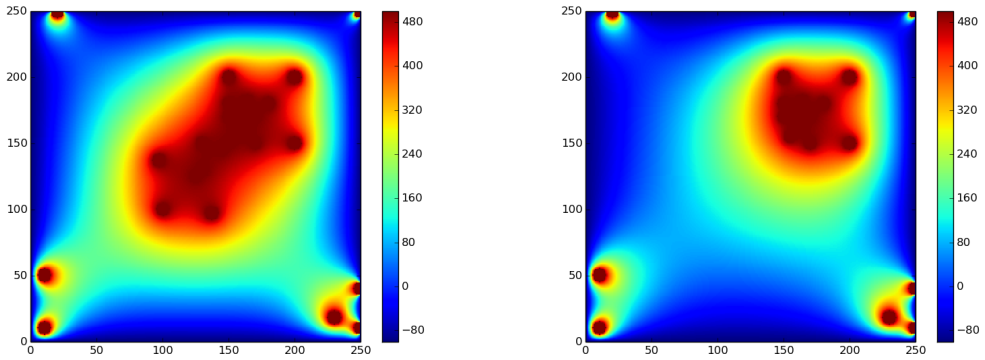
Hemos realizado este experimento con varias muestras distintas, decidiendo mostrar solamente 2 de ellas ya que reflejaban bien el comportamiento esperado del algoritmo. Para estos seleccionamos 30 sanguijuelas colocándolas en lugares particulares (en los bordes, cerca del punto crítico y algunas más concentradas en la región entre el punto crítico y los bordes) para ver si eran eliminadas o no. Además, en estas dos instacias que mostraremos, hemos cambiado el valor de la temperatura que ejerce cada sanguijuela (240 °C para el primer experimento y 500 °C para el segundo). Tomamos un ancho y largo del parabrisas de 250 metros, con una granularidad de 1 y un radio de 5 metros sobre el cual las sanguijuelas ejercen el calor. También hemos calculado la temperatura del punto crítico al finalizar la ejecución del algoritmo para evaluarla. Hemos obtenido los siguientes gráficos:



(a) Sin eliminar sanguijuelas

(b) Eliminando sanguijuelas

Figure 10: Experimento 1



(a) Sin eliminar sanguijuelas

(b) Eliminando sanguijuelas

Figure 11: Experimento 2

En el primero de los experimentos, en el cual las sanguijuelas ejercían una temperatura de 240 °C, podemos observar cómo cambio la temperatura de la zona en la que se encontraba el punto crítico (en la posición $(125.5; 125.5)$), el cual pasó de tener varias sanguijuelas cercanas que generaban una temperatura elevada allí a tener una zona más despejada. Luego del experimento la temperatura final del punto crítico fue de 226,52 °C, lo cual nos lleva a concluir que fueron eliminadas las sanguijuelas para que la temperatura disminuyera de los 235 °C.

En el segundo, en el cual las sanguijuelas ejercían una temperatura de 500 °C, se puede observar mejor que muchas más sanguijuelas fueron eliminados puesto que provocaban una mayor temperatura en el punto crítico. Se puede ver que este punto quedó totalmente despejado de sanguijuelas puesto que, al ejercer una temperatura tan alta, si hubiese alguna cercana provocaría que el punto crítico estuviese en problemas. Además, la temperatura final de este punto fue de 233,689 °C, lo cual efectivamente eliminó las necesarias para que no este más en peligro.

De ambos experimentos pudimos sacar las mismas conclusiones, las cuales fueron que se eliminaron correctamente todas las sanguijuelas debidas para que la temperatura del punto crítico bajase del valor pedido. En el segundo caso, esta eliminación fue mayor debido a que, como las sanguijuelas ejercían mayor temperatura, habían más afectando al punto crítico que debían ser eliminadas. Aparte, en ambos gráficos se pueden observar que todas las demás sanguijuelas de puntos lejanos se mantuvieron intactas, es decir que no fueron eliminadas por nuestro algoritmo, confirmando nuestras hipótesis iniciales. Por esto, y por los demás experimentos realizados que mostraron situaciones similares, podemos confirmar que nuestro algoritmo efectivamente elimina las sanguijuelas más cercanas al punto crítico que provocan que su temperatura se encuentre en estado grave, es decir mayor o igual a 235 °C, haciendo que nuestro método funcione correctamente.

6 Conclusion

En conclusión, a lo largo de este trabajo práctico realizamos distintas implementaciones de las matrices banda (junto con todos los algoritmos utilizados para resolver el sistema de ecuaciones lineales al que esta asociada) con el fin de poder solucionar el problema de como calcular la temperatura de los puntos del parabrisas. Además, realizamos análisis de estas (mediante una serie de experimentos), llegando a las conclusiones de que pudimos pensar una estructura alternativa para representar a la matriz banda, debido a la utilización de esta ocasiona que haya un ahorro en el tiempo y en el espacion utilizado. Además analizamos que ocurría en ambos casos cuando se modificaba la granularidad viendo que si h era mayor se perdían datos y que los resultados obtenidos se iban haciendo más lejanos de lo que realmente ocurría, perjudicando a su vez el cálculo de la temperatura en el punto crítico, aunque esto ocasionaba que el tiempo de cómputo sea menor. Con respecto al punto crítico, podemos concluir que la estimación de su temperatura depende más del tamaño de los intervalos de discretización que del método propuesto. Puede suceder que, al tener un intervalo muy grande, esta estimación no sea buena. Pero en el mejor caso nos estaremos acercando al valor real buscado. Si bien a veces este punto no cae dentro de los puntos discretizados, creemos que nuestra forma de obtenerlo es bastante eficaz. Con respecto al método de eliminación de sanguijuelas, ya hemos discutido sus ventajas y desventajas. Luego de la realización de varios experimentos podemos concluir que la eficacia de nuestro método está relacionada a la distribución de las sanguijuelas en el parabrisas. Puede suceder que nuestro método no sea muy eficiente como también serlo bastante. También podríamos lograr que este sea eficiente en la mayoría de los casos aplicando varias modificaciones en nuestro algoritmo, las cuales ya fueron nombradas anteriormente.

7 Modificaciones

Al realizar este trabajo práctico, realizamos algunas modificaciones a algunos de los algortimos comunmente usados por distintos motivos, por ejemplo, para que sean más fáciles de implementar. Las modificaciones que realizamos fueron:

- Realizamos una modificación al algoritmo de eliminación gaussiana al adaptar este para su uso en matrices banda, esta la realizamos con el objetivo de que sea más fácil de implementar. Lo que cambiamos es que en vez de realizar implementar este como normalmente es, es decir, en cada paso del algortimo poner en cero todos los valores debajo de la diagonal para cada columna (tal como lo hicimos para la eliminación gaussiana para cuando guardamos la matriz de la forma tradicional), nos centramos en cada paso en que esto suceda para cada fila. Entonces, en cada paso de este algoritmo, lo que hacemos es usar las filas que se encuentran más arriba de con la que estamos trabajando ahora para que en la fila en la que nos encontremos queden en ceros todos coeficientes que se encuentran anteriores a la diagonal. Utilizando la forma alternativa para representar las matrices, esto significa que luego de cada paso poseeremos una fila más de la matriz banda con los primeros n coeficientes en cero.