DEPARTAMENTO
DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

# Sanguijuelas

4 de septiembre de 2014

Métodos Numéricos
Trabajo Práctico Nro. 1


| Integrante | LU | Correo electrónico |
|---|---|---|
| Martin Carreiro | 45/10 | martin301290@gmail.com |
| Kevin Kujawski | 459/10 | kevinkuja@gmail.com |
| Juan Manuel Ortíz de Zárate | 403/10 | jmanuoz@gmail.com |






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

# Índice

**1. Resumen** **3**

**2. Introducción teórica** **4**

2.1. Algoritmo de eliminación gaussiana . . . . . 4

2.2. Matriz banda . . . . . 5

**3. Desarrollo** **6**

3.1. Matriz Banda . . . . . 6

3.1.1. Optimización espacial . . . . . 6

3.1.2. Optimización temporal . . . . . 7

3.1.3. Ejemplo . . . . . 7

3.2. Eliminación Gaussiana . . . . . 10

3.3. Soluciones . . . . . 10

3.4. Solución Greedy + Local Search . . . . . 10

3.5. Solución Random . . . . . 12

**4. Resultados** **13**

4.1. Resultados Random . . . . . 13

4.2. Resultados Greedy . . . . . 18

**5. Discusión** **20**

5.1. Espiral vs centrico . . . . . 20

**6. Conclusiones** **21**

6.1. Random vs Greedy . . . . . 21

**7. Referencias** **22**

# 1. Resumen

Mediante el manejo de matrices se buscará modelar y solucionar un problema de la realidad.Cómo esta esta compuesta de infinitas variables dicha modelización implicará una inevitable discretización. Es decir trabajar con una cantidad acotada de variables del problema (sólo las relevantes). Si bien el problema en cuestión consiste en decidir cual/es de las sanguijuelas que están adheridas al parabrisa se debe eliminar la modelización del mismo no es trivial. Es más, podría decirse que este proceso es mucho mas complejo y costoso que la solución en sí. ¿Por qué? porque la creación de la matriz que represente al parabrisa y el cálculo de las temperaturas en cada punto, si no se usa buen método, podría llegar a demorar mucho tiempo, tirar overflows o directamente nunca terminar.

Por esto es que a lo largo de este tp haremos mucho foco en como calcular las temperaturas, como optimizar el espacio ocupado por la matriz obtenida y como optimizar lo mas posible todas las operaciones matriciales.

# 2. Introducción teórica

Dado que la modelización utlizada para este problema es una matriz y que en base a algunos datos que nos dan de ella (posición de las saniguijuelas, dimensiones y temperatras de los bordes) tenemos que calcular los valores de muchas celdas, lo que hay que resolver es un sistema de ecuaciones. Lo primero que se nos viene a la mente cuando tenemos esto es el método de eliminación gaussiano. Este transforma la matriz de coeficientes en una matriz triangular superior y luego mediante back substitution se pueden obtener todos sus valores.

Veamos ahora más en detalle como funciona cada uno de estos algoritmos.

## 2.1. Algoritmo de eliminación gaussiana

Es importante en este punto aclarar que para poder hacer estos cálculos la martiz debe ser cuadrada (7). Los pasos a seguir en este algoritmo son:

- Ir a la columna no cero extrema izquierda
- Si el primer renglón tiene un cero en esta columna, intercambiarlo con otro que no lo tenga
- Luego, obtener ceros debajo de este elemento delantero, sumando múltiplos adecuados del renglón superior a los renglones debajo de él
- Cubrir el renglón superior y repetir el proceso anterior con la submatriz restante. Repetir con el resto de los renglones (en este punto la matriz se encuentra en la forma de escalón)
- Comenzando con el último renglón no cero, avanzar hacia arriba: para cada renglón obtener un 1 delantero e introducir ceros arriba de éste sumando múltiplos correspondientes a los renglones correspondientes

Ejemplo:

$$\begin{cases} 2x & +y & -z & = & 8 \\ -3x & -y & +2z & = & -11 \\ -2x & +y & +2z & = & -3 \end{cases}$$

Figura 1: Sistema de ecuaciones de 3x3

$$\begin{cases} 2x & +y & -z & = & 8 \\ & \frac{1}{2}y & +\frac{1}{2}z & = & 1 \\ & 2y & +z & = & 5 \end{cases}$$

Figura 2: Sistema de ecuaciones con ceros en la primera columna

$$\begin{cases} 2x & -2z & = & 6 \\ & \frac{1}{2}y + \frac{1}{2}z & = & 1 \\ & -z & = & 1 \end{cases}$$

Figura 3: Sistema de ecuaciones triangulado

En esta última imagen podemos observar que el valor de Z ya lo tenemos. Por lo tanto si reemplazamos el mismo en la ecuación arriba obtenemos Y y lo mismo con la primer ecuación.

La complejidad computacional de esto es aproximadamente $n^3$. Esto es, el número de operaciones requeridas es $n^3$ si el tamaño de la matriz es n x n. Por lo tanto puede volverse un cálclo sumamente grande si la matriz es de dimensiones importantes. Por esto decidimos pensar un poco mejor la situación y encontrarle la vuelta para resolverlo en un tiempo razonable.

## 2.2. Matriz banda

Analizando la matriz del sistema de ecuaciones que se genera a partir de la representación del Parabrisas observamos que presenta las caracteristicas de un tipo de matriz llamada "banda", la cual se distingue por tener todas las celdas en 0 salvo en la diagonal ensanchada, es decir, que p celdas a la izquierda y q a la derecha de la diagonal a lo sumo tienen algunos o todos los valores distintos de 0. En nuestro caso, al construir la matriz nos basamos en el tipo de celda, si es el borde frío o una sanguijuela el coeficiente respectivo sera un 1 en la diagonal igualado a la temperatura correspondiente, pero en el caso de ser una celda vacía en la diagonal irá un -4, a los costados un 1 y representando la fila de .ªrribaz .ªbajo"del parabrisas iran un 1 a "mçolumnas de distancia de la diagonal. Finalmente la matriz resultante quedará con una banda de ancho m+m, con "m.el ancho del parabrisas.

Aca podemos ver un ejemplo de una matriz banda:

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & 0 & 0 & 0 & 0 \\ a_{21} & a_{22} & a_{23} & a_{24} & 0 & 0 & 0 \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} & 0 & 0 \\ 0 & a_{42} & a_{43} & a_{44} & a_{45} & a_{46} & 0 \\ 0 & 0 & a_{53} & a_{54} & a_{55} & a_{56} & a_{57} \\ 0 & 0 & 0 & a_{64} & a_{65} & a_{66} & a_{67} \\ 0 & 0 & 0 & 0 & a_{75} & a_{76} & a_{77} \end{pmatrix}$$

Figura 4: Matriz banda 7x7

# 3. Desarrollo

## 3.1. Matriz Banda

Como explicamos en la introducción de la matriz banda, llegamos a la conclusión de que la matriz de ecuaciones de la representación del parabrisas tenía forma de la denominada "matriz bandaçuya respectiva banda era de tamaño m+m+1 con m siendo el ancho del parabrisas. Este es un hecho importante ya que las matrices banda tienen caracteristicas especiales de las cuales es posible la optimización temporal y espacial para resolver el sistema de ecuaciones con eliminación gausseana.

### 3.1.1. Optimización espacial

El problema de representar la matriz original es que la mayoría de los valores son 0 y solo importan los elementos de la banda de la matriz, por lo tanto una forma de optimizar espacialmente es solo guardar la banda, con lo que se logra reducir considerablemente el espacio en esta representación, ya que suponiendo que se tiene un parabrisas de n filas y m columnas, el tamaño de la matriz original sería de $(n*m)^2$, mientras que con la optimización de matriz banda quedaría de tamaño (n*m)*(2*m+1).

El método para construir la representación optimizada de la matriz banda es simple, se guardar la diagonal en una matriz, quedando en el centro los elementos de la diagonal dependiendo de lo que haya en esa posición, ya que en el caso de que sea vacía esta deberá tener coeficientes de las posiciones adyacentes. Además nos dimos cuenta que se podía representar la matriz de tal forma que presente una mejor optimización temporal, y consiste en no poner los coeficientes de las celdas vacias adyacentes si estas no son vacias, en tal caso al vector de resultados le restamos la temperatura de la misma, generando que luego no sea necesario considerar las posiciones no vacias para la eliminación gausseana a la hora de resolver el sistema.

```
por cada posición del parabrisas //O(N*M)
       pos = fila posicion + columna posicion * ancho;

              si en esa posición hay una SANGUIJUELA:

                     bandMatrix[pos][ancho] = 1;
                     resultados[pos] = ts;

              si esa posición es borde FRIO:

                     bandMatrix[pos][ancho] = 1;
                     resultados[pos] = -100;


              en caso de que esa posición sea VACIA:
                     bandMatrix[pos][ancho] = -4;
                     resultados[pos] = 0;

                     si la izquierda no es vacìa
                            resultados[pos] -= temperatura del de la izquierda;
                     else
                             bandMatrix[pos][ancho-1] = 1;
```

```
si la de arriba no es vacìa
        resultados[pos] -= temperatura del de arriba;
 else
        bandMatrix[pos][0] = 1;

 si la de la derecha no es vacìa
        resultados[pos] -= temperatura del de la derecha;
 else
        bandMatrix[pos][ancho+1] = 1;

 si la de abajo no es vacìa
        resultados[pos] -= temperatura del de abajo;
 else
        bandMatrix[pos][ancho*2] = 1;
```

### 3.1.2. Optimización temporal

### 3.1.3. Ejemplo

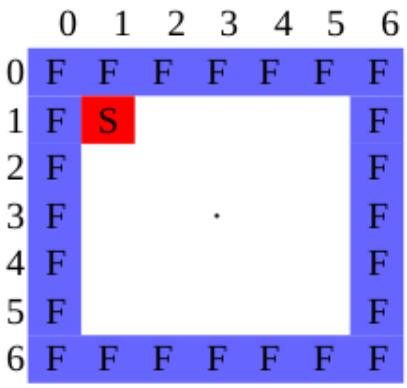



Figura 5: Vista de la representación de Parabrisas del ejemplo

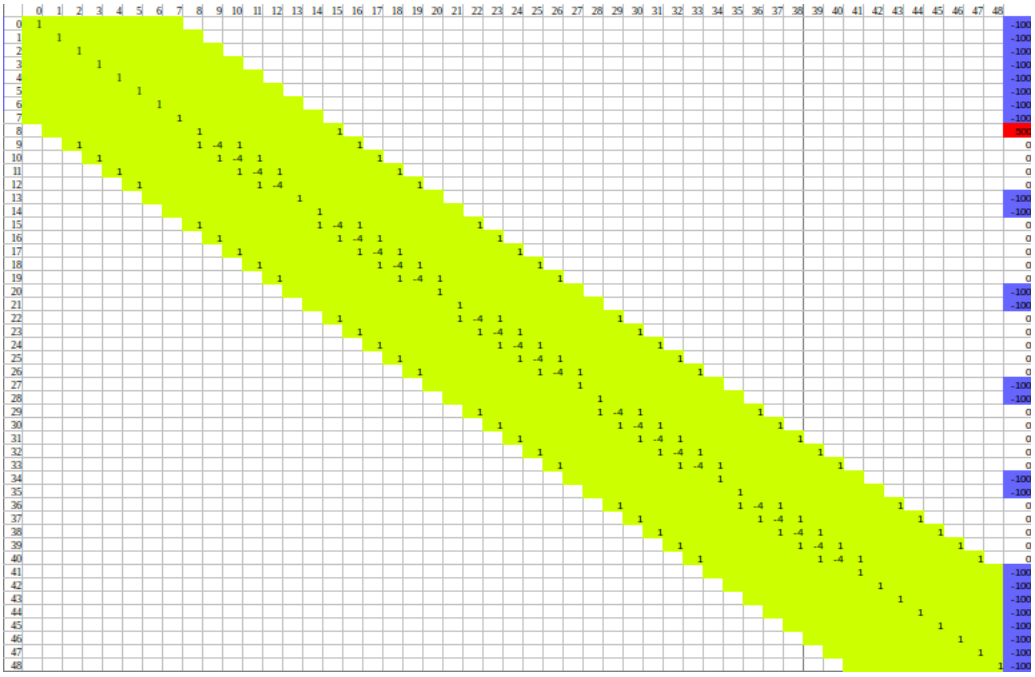
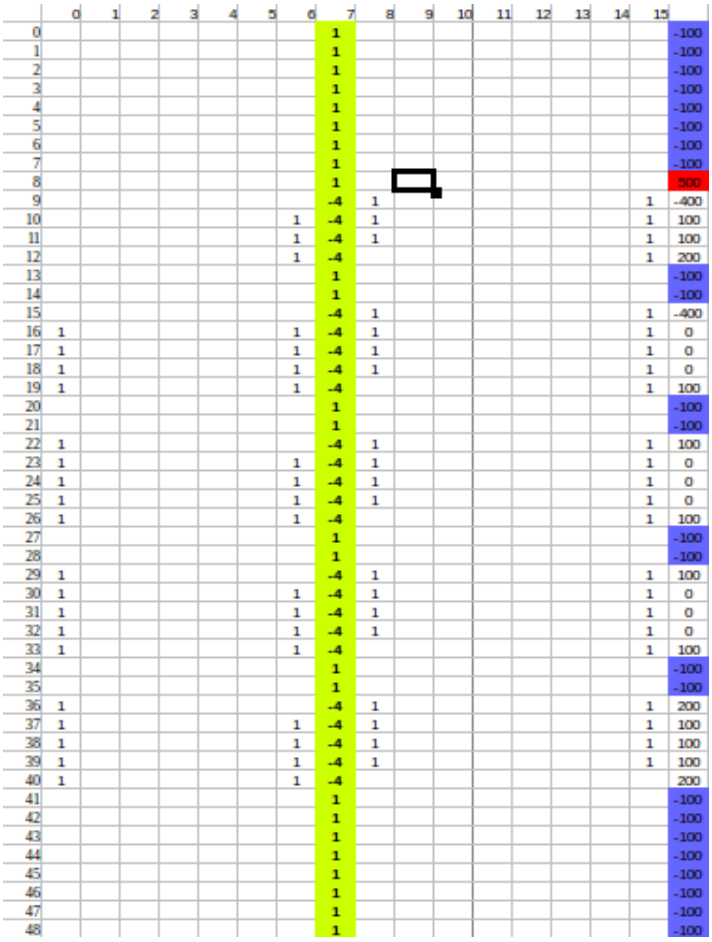


Figura 6: Matriz de ecuaciones del ejemplo



| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | | | | | | | | 1 | | | | | | | | | -100 |
| 1 | | | | | | | | 1 | | | | | | | | | -100 |
| 2 | | | | | | | | 1 | | | | | | | | | -100 |
| 3 | | | | | | | | 1 | | | | | | | | | -100 |
| 4 | | | | | | | | 1 | | | | | | | | | -100 |
| 5 | | | | | | | | 1 | | | | | | | | | -100 |
| 6 | | | | | | | | 1 | | | | | | | | | -100 |
| 7 | | | | | | | | 1 | | | | | | | | | -100 |
| 8 | | | | | | | | 1 | | | | | | | | | 500 |
| 9 | | | | | | | | -4 | 1 | | | | | | | 1 | -400 |
| 10 | | | | | | | 1 | -4 | 1 | | | | | | | 1 | 100 |
| 11 | | | | | | | 1 | -4 | 1 | | | | | | | 1 | 100 |
| 12 | | | | | | | 1 | -4 | | | | | | | | 1 | 200 |
| 13 | | | | | | | | 1 | | | | | | | | | -100 |
| 14 | | | | | | | | 1 | | | | | | | | | -100 |
| 15 | | | | | | | | -4 | 1 | | | | | | | 1 | -400 |
| 16 | 1 | | | | | | 1 | -4 | 1 | | | | | | | 1 | 0 |
| 17 | 1 | | | | | | 1 | -4 | 1 | | | | | | | 1 | 0 |
| 18 | 1 | | | | | | 1 | -4 | 1 | | | | | | | 1 | 0 |
| 19 | 1 | | | | | | 1 | -4 | | | | | | | | 1 | 100 |
| 20 | | | | | | | | 1 | | | | | | | | | -100 |
| 21 | | | | | | | | 1 | | | | | | | | | -100 |
| 22 | 1 | | | | | | | -4 | 1 | | | | | | | 1 | 100 |
| 23 | 1 | | | | | | 1 | -4 | 1 | | | | | | | 1 | 0 |
| 24 | 1 | | | | | | 1 | -4 | 1 | | | | | | | 1 | 0 |
| 25 | 1 | | | | | | 1 | -4 | 1 | | | | | | | 1 | 0 |
| 26 | 1 | | | | | | 1 | -4 | | | | | | | | 1 | 100 |
| 27 | | | | | | | | 1 | | | | | | | | | -100 |
| 28 | | | | | | | | 1 | | | | | | | | | -100 |
| 29 | 1 | | | | | | | -4 | 1 | | | | | | | 1 | 100 |
| 30 | 1 | | | | | | 1 | -4 | 1 | | | | | | | 1 | 0 |
| 31 | 1 | | | | | | 1 | -4 | 1 | | | | | | | 1 | 0 |
| 32 | 1 | | | | | | 1 | -4 | 1 | | | | | | | 1 | 0 |
| 33 | 1 | | | | | | 1 | -4 | | | | | | | | 1 | 100 |
| 34 | | | | | | | | 1 | | | | | | | | | -100 |
| 35 | | | | | | | | 1 | | | | | | | | | -100 |
| 36 | 1 | | | | | | | -4 | 1 | | | | | | | 1 | 200 |
| 37 | 1 | | | | | | 1 | -4 | 1 | | | | | | | 1 | 100 |
| 38 | 1 | | | | | | 1 | -4 | 1 | | | | | | | 1 | 100 |
| 39 | 1 | | | | | | 1 | -4 | 1 | | | | | | | 1 | 100 |
| 40 | 1 | | | | | | 1 | -4 | | | | | | | | | 200 |
| 41 | | | | | | | | 1 | | | | | | | | | -100 |
| 42 | | | | | | | | 1 | | | | | | | | | -100 |
| 43 | | | | | | | | 1 | | | | | | | | | -100 |
| 44 | | | | | | | | 1 | | | | | | | | | -100 |
| 45 | | | | | | | | 1 | | | | | | | | | -100 |
| 46 | | | | | | | | 1 | | | | | | | | | -100 |
| 47 | | | | | | | | 1 | | | | | | | | | -100 |
| 48 | | | | | | | | 1 | | | | | | | | | -100 |

Figura 7: Matriz banda del ejemplo

### 3.2. Eliminación Gaussiana

A modo de comparación recreamos el clásico método de resolución de Eliminación Gaussiana que consta de la eliminación progresiva de variables en el sistema de ecuaciones, hasta tener sólo una ecuación con una incógnita. Una vez resuelta esta, se procede por sustitución regresiva hasta obtener los valores de todas las variables.

```
Class Windshield{
    resolveByGaussianElimination(){
         gaussianElimination();
         backSubstitution();
    }
    gaussianElimination(){
        for rows k already been reduced
            iMax = findPivot();
            Swap rows k and imax;
            Force 0's in column A[k+1..n-1][k];
    }
    backSubstitution(){
        for row from last to first:
            calculateFromNextOneExceptForLast()
    }
}
```

### 3.3. Soluciones

Para resolver el problema que se nos pide, planteamos distintos tipos de solución. Inicialmente encaramos el problema de manera exacta, pero lo deshechamos inmediatamente ya que sabiendo que es un problema de decisión, el tiempo para determinar cuál es el número mínimo de sanguijuelas a eliminar es exponencial en la cantidad de estas. Es por eso que decidimos realizar una heurística, ya que el tiempo es preciado cuando el parabrisas de nuestra nave tiene poco tiempo antes de que se rompa en caso de estar en situación crítica. Para modo de comparación tomamos dos tipos de algoritmos para establecer que es una mejor solución.

seleccionar que sanguijuelas matar, pensamos una solución Greedy (7) + Local Search . Esta consiste, a grandes rasgos, en ir matando las sanguijuelas del medio (ya que son las que mayor temperatura generan en el punto crìtico) hasta que el punto central este por debajo del 235 C$^{o}$. Para tener algún parámetro de referencia pensamos en buscar otra solución posible, y asì compararlas y chequear que esta sea mejor. Esta solución alternativa consiste básicamente en seleccionar aleatoriamente que sanguijuelas aniquilar. Procederemos ahora a explicar en detalle las implementaciones de estas soluciones.

### 3.4. Solución Greedy + Local Search

Esta solución consiste, a grandes rasgos, en ir matando las sanguijuelas del medio (ya que son las que mayor temperatura generan en el punto crítico) hasta que el punto central este por debajo del 235 C$^{o}$. Finalmente, tratar de realizar una búsqueda local intentado volcar de vuelta en el parabrisas sanguijuelas que habíamos decidido sacar y viendo si era realmente necesario eliminarla para poder salir del estado crítico. La decisión de hacer LocalSearch está dada para el caso de que exista una sanguijuela en el centro y dos a la misma distancia. Pero de un lado haya una linea de muchas sanguijuelas, por lo que es muy probable que con sacar la del lado de la linea fuera suficiente.

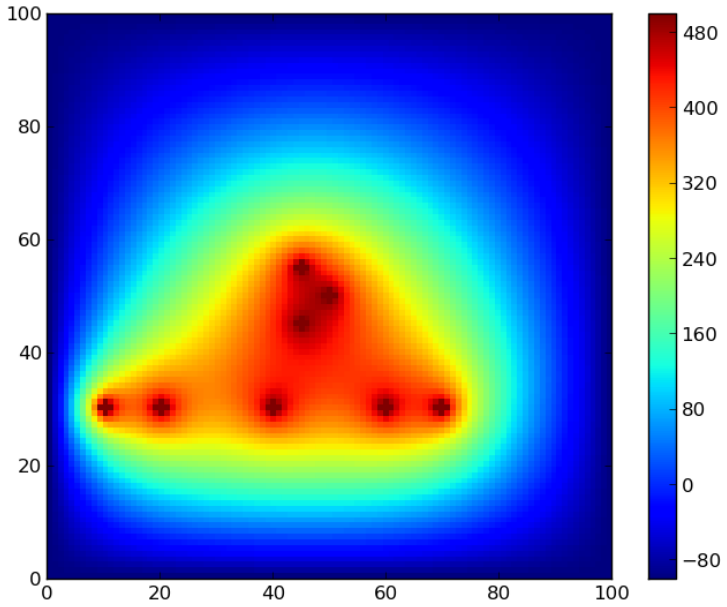



Figura 8: Matriz banda del ejemplo

Pseudocodigo:

```
Class Windshield{
    greedySolution(){
        orderLeachesByDistanceToCenter()
         while(!this->isCooledDown(){
             leachesToRemove(this->removeFirstNotRemovedLeachOrderedCentrically())
         }

         localSearchTryToReturnLeaches(leachesToRemove);

         return leachesToRemove;
    }
    isCooledDown(){
         recalculateByBandMatrix();
         return (matrix.centerPoint < Ts)
    }
    orderLeachesByDistanceToCenter(){
        sort(leaches).by(lambda {|leach,otherLeach| leach.distanceToCenter < otherLeach.distanceT
    }

    localSearchTryToReturnLeaches(leachesToRemove){
        for (leach in leachesToRemove){
            putBackLeachInWindshield(leach);
            if (isCooledDown){
                leachesToRemove.erase(leach)
            }else{
                takeOutLeachFromWindshield(leach);
```

```
            }
        }
    }
}
```

### 3.5. Solución Random

En esta solución seleccionamos de nuestro array de posiciones de sanguijuelas (que es el array recibido por parametro, osea con las posiciones sin discretizar) una al azar y la eliminamos. Luego ejecutamos de nuevo el cálculo de las temperaturas y chequeamos si el punto crìtico esta por debajo del 235 C$^{o}$.Si no lo está, elegimos otra al azar y repetimos el proceso hasta que lo esté.

Pseudocodigo:

```
Class Windshield{
    randomSolution(){
         while(!this->isCooledDown(){
             this->randomKill()
         }
    }
    isCooledDown(){
         return (matrix.centerPoint < Ts)
    }
    randomKill(){
         randomRemove(posSanguijuelas)
         this->recalculateTemps()
    }
}
```

Vale aclarar que el recalculateTemps utiliza el metodo band matrix, ya que este es màs rápido.

# 4. Resultados

## 4.1. Resultados Random

A continuación observaremos una matriz en la que la única sanguijuela que se debe eliminar es la del centro. Esta se encuentra exactamente en el punto central. Adicionalmente hay otras 4 sanguijuelas en las puntas. Matar a estas no soluciona nada ya que la del centro esta aplicando una temperatura constante de 400 C°.

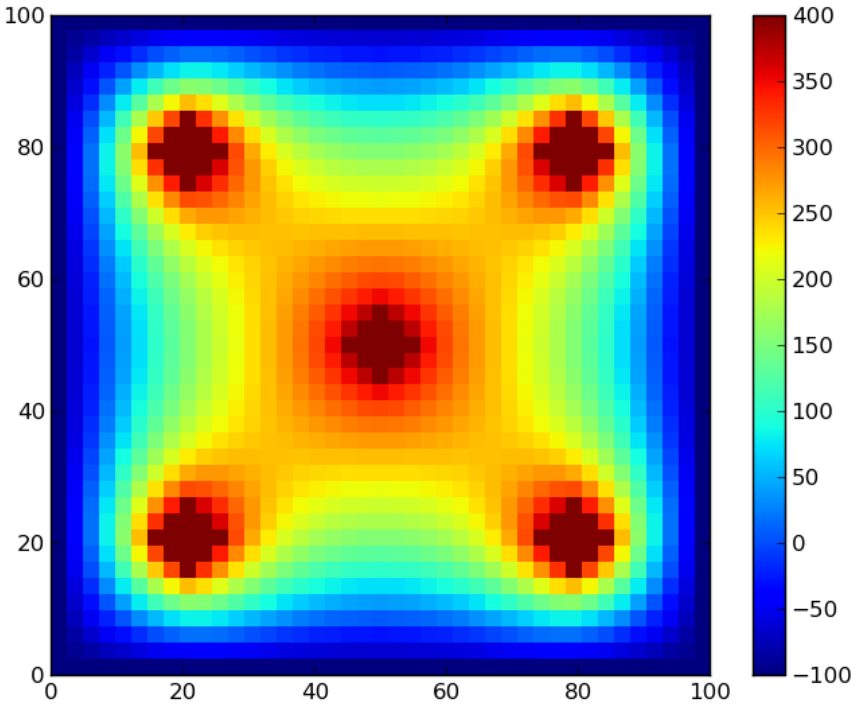



Figura 9: Parabrisas con 5 sanguijuelas

Ahora veamos que obtenemos al aplicarle distintas veces el algoritmo de solución random explicado en el desarrollo (3.5).

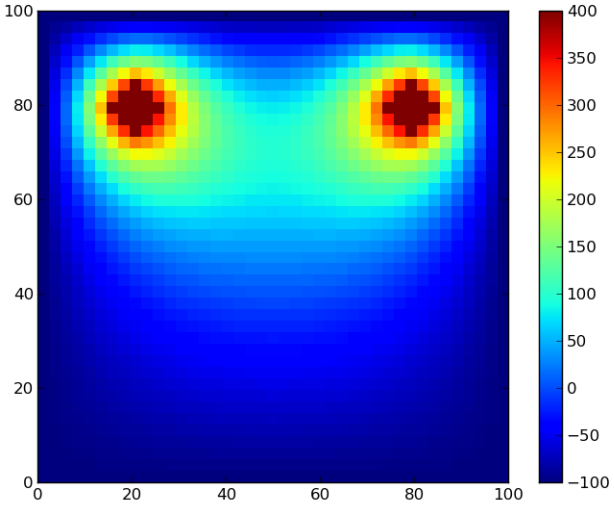



Figura 10: Resultado primer corrida

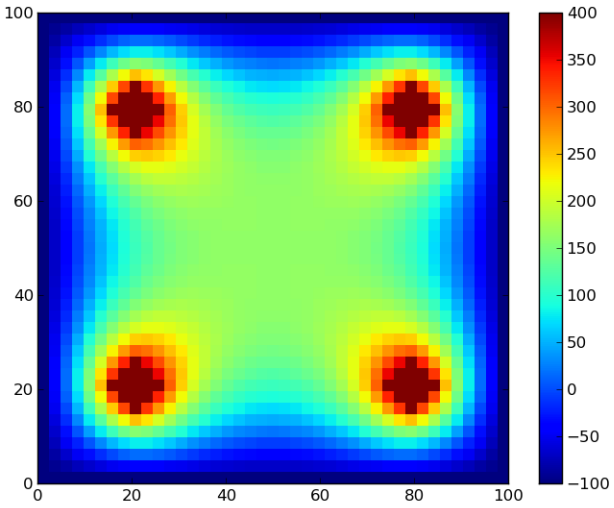



Figura 11: Resultado segunda corrida

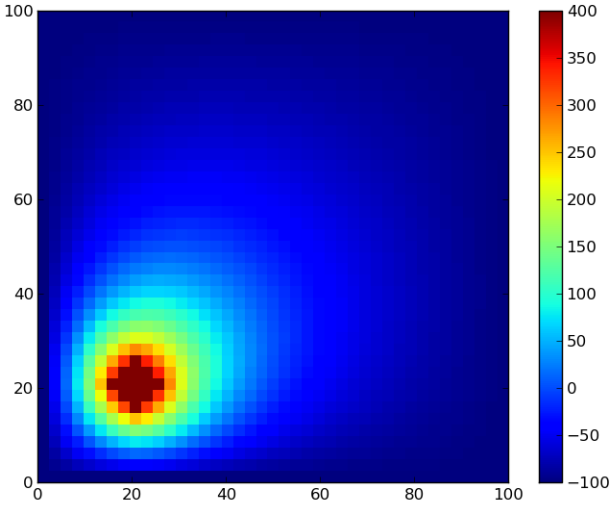



Figura 12: Resultado tercer corrida

Como podemos observar la solución óptima (la que menos sanguijuelas elimina), es la segunda. Pero como esta solución es completamente random en la primera corrida mata 2 sanguijuelas antes de elegir la correcta y en la tercera 4. Sólo en la segunda elije en el primer intento la sanguijuela correcta.

Veamos ahora cuanto demoró cada una de las soluciones. Es importante aclarar que acá el algoritmo utilizado para recalcular las temperaturas fue el gaussiano sin banda matriz. Para que la diferencia de tiempos sea mas clara.

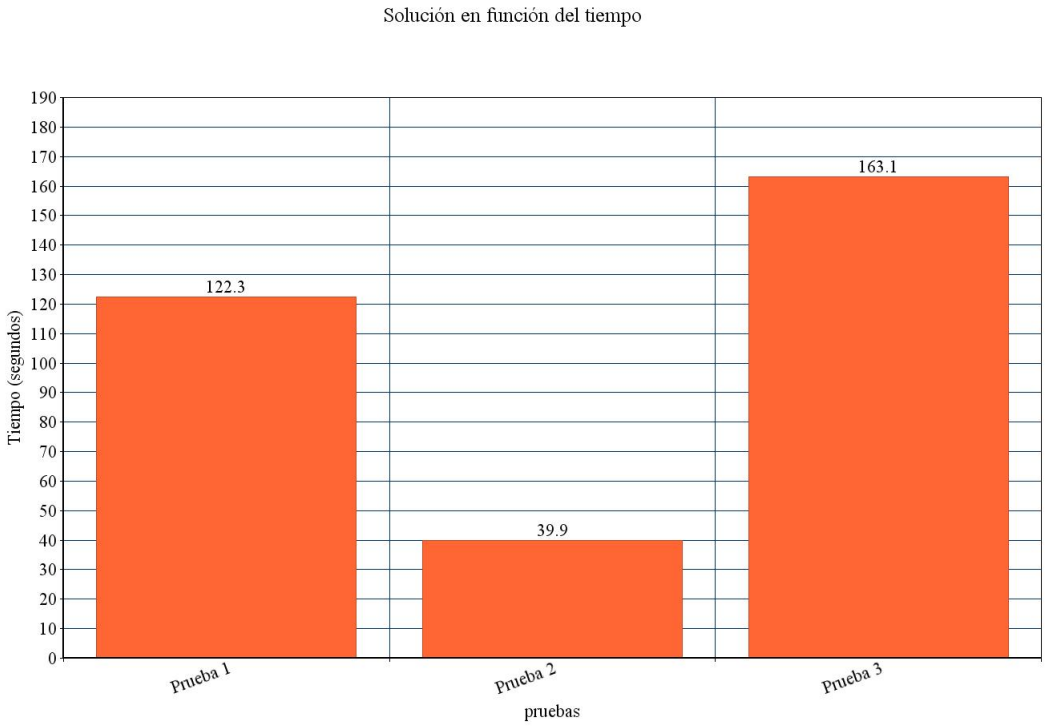



Figura 13: Tiempos de corrida de cada prueba

Acá podemos observar que, como era de esperarse, cuanto màs sanguijuelas elimina mas tiempo tarda, ya que debe recalcular las temperaturas.

Veamos ahora como se comporta con la matriz utilizada en la explicaciòn del algoritmo greedy (3.4).

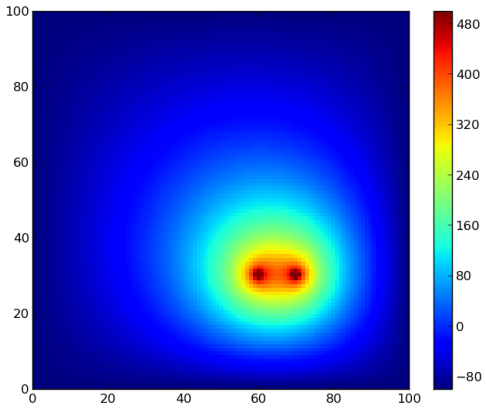



Figura 14: Resultado primer corrida

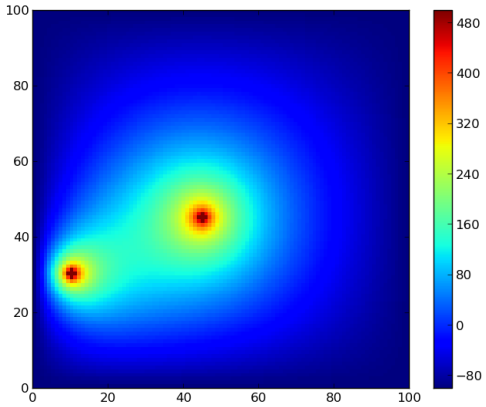



Figura 15: Resultado primer corrida

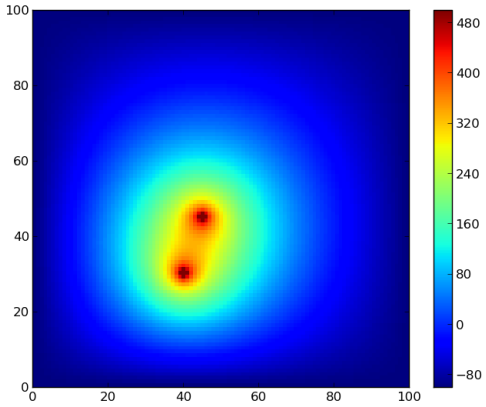



Figura 16: Resultado primer corrida

Como podemos ver en este caso el random sigue dando malas soluciones en promedio.

## 4.2. Resultados Greedy

Veamos ahora para la misma matriz como se comporta nuestro otro algoritmo.

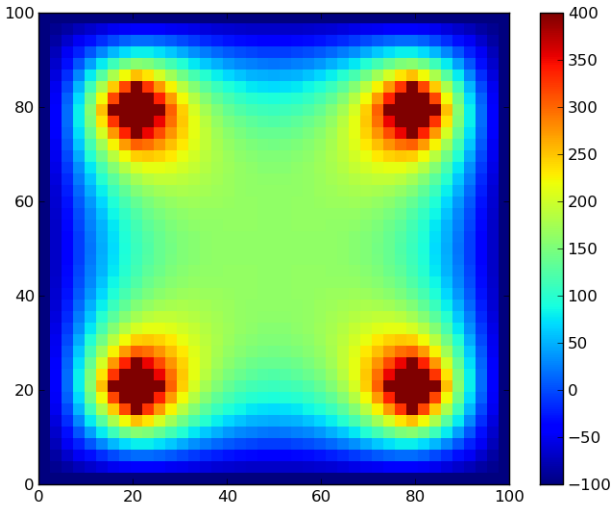



Figura 17: Resultado corrida con greedy

En todas las corridas obtuvimos el mismo resultado y demoraron lo mismo. Era de esperarse, ya que efectivamente en este caso, la sanguijuela a eliminar era la del medio.

# 5. Discusión

## 5.1. Espiral vs centrico

Durante el desarrollo del algoritmo Greedy + LocalSearch, nuestro primer acercamiento fue recorrer la matriz de forma espiral. El primer problema que encontramos fue que teníamos un cálculo muy grande para determinar dado un punto del parabrisas, qué sanguijuela era y que recorríamos partes de la matriz que no importaban (en especial los bordes). A su vez, tendíamos a alejarnos del objetivo principal que era preocuparnos por el centro. Es por eso que decidimos ordenar las sanguijuealas según su distancia al centro. Para ahorrar tiempo computacional, este dato está calculado en el momento de la creación de la sanguijuela para su posterior ordenamiento.

no contar los bordes ni las sanguijuelas para crear la matriz banda

¿porque random?

# 6. Conclusiones

## 6.1. Random vs Greedy

Como observamos en los resultados para el caso en el que la sanguijuela/s se encuentran en el medio claramente es mejor el greedy, ya que siempre mata a la sanguijuela correcta a diferencia del random que sólo lo hizo en uno de los 3 intentos. Por otro lado si tenemos en cuenta los tiempos que pueden llegar a demorar recalcular las temperaturas, esto le da todavìa una mayor importancia, ya que se reduce considerablemente el tiempo de ejecución.

Por otro lado en el segundo caso, donde es un poco mas conflictivo para la lógica de ir sacando los del medio, no sólo sigue siendo claramente mejor que el random sino que ademàs también da la mejor por solución. Por lo tanto podemos concluir que no sólo es un mejor algoritmo que el random sino que además se comporta de forma óptima para los casos que pensabamos iban a ser mas conflictivos.

# 7. Referencias

- Una matriz de n por m elementos, es una matriz cuadrada si el número de filas es igual al número columnas, es decir, n = m y se dice, entonces que la matriz es de orden n
- En matemáticas una matriz se le llama matriz banda cuando es una matriz donde los valores no nulos son confinados en un entorno de la diagonal principal, formando una banda de valores no nulos que completan la diagonal principal de la matriz y más diagonales en cada uno de sus costados.
- Un algoritmo Greedy (también conocido como ávido, devorador o goloso) es aquel que, para resolver un determinado problema, sigue una heurística consistente en elegir la opción óptima en cada paso local con la esperanza de llegar a una solución general óptima. Este esquema algorítmico es el que menos dificultades plantea a la hora de diseñar y comprobar su funcionamiento.