DEPARTAMENTO DE COMPUTACION

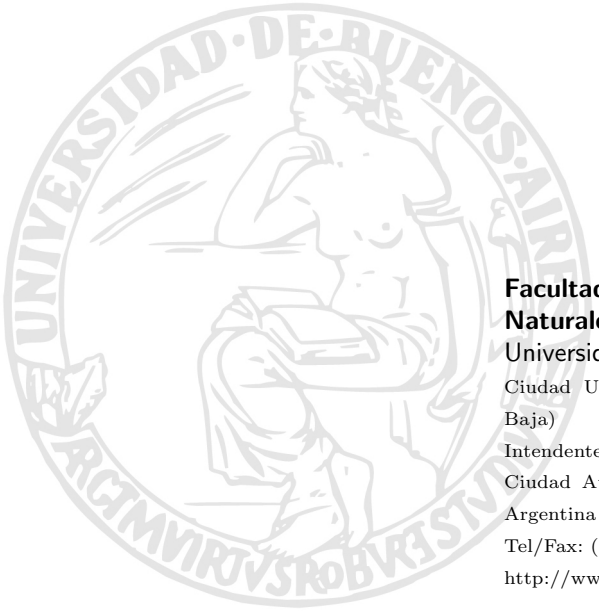
Facultad de Ciencias Exactas y Naturales - UBA

# Métodos Numéricos

## TP 1: "No creo que a él le gustara eso"

1 de septiembre de 2014


| Integrante | LU | Correo electrónico |
|---|---|---|
| Fosco, Martin Esteban | 449/13 | mfosco2005@yahoo.com.ar |
| Minces Müller, Javier Nicolás | 231/13 | javijavi1994@gmail.com |
| Chibana, Christian Tomokazu De La Vega Jr. | | tomistomus@LaMilagrosa.com.jp |


**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

# 1. Resumen

En este trabajo implementamos una solución al problema planteado en el tp, encontrar una manera de determinar qué sanguijuelas pegadas al parabrisas son peligrosas y eliminarlas, usando una representación matricial del sistema de ecuaciones que me permite hallar la temperatura del parabrisas en cada punto (con una precisión determinada por la granularidad) tomando como incógnitas a, justamente, cada punto del parabrisas.
Para resolver dicho sistema de ecuaciones recurrimos a técnicas matemáticas basadas en métodos de resolución de sistemas, y dividimos en distintos módulos (structs) con funciones específicas al parabrisas y a la matriz del sistema de ecuaciones para facilitar la comprensión (tanto de los docentes como de nosotros mismos) de la implementación hecha.

## 2. Introducción Teórica

El objetivo de este trabajo era resolver un problema, que puede dividirse en dos partes. La primera parte del problema consistía en hallar una forma de representar el calor causado por una sanguijuelas en un parabrisas rectangular en un momento determinado, representando dicho parabrisas como una matriz, utilizando el lenguaje C++. Algunos puntos eran conocidos, ya que el parabrisas tenía bordes con una temperatura constante, y fuentes de calor (sanguijuelas) con radio y temperatura variable. Como dato, se tenía que cada punto del parabrisas satisfacía la ecuación de Laplace.

Para poder modelar el parabrisas se tomó una discretización, con granularidad variable. De esta forma, el parabrisas quedó representado con una matriz de a/h+1 filas y l/h+1 columnas, siendo a el ancho en metros del parabrisas original, l su largo y h la granularidad elegida y l su largo. A esta matriz la llamaremos "matriz parabrisas".

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

Considerando todos los puntos desconocidos de la matriz, se obtuvo entonces un sistema de ecuaciones lineales de a+l x a+l, siendo a y l el ancho y el largo de la matriz parabrisas, respectivamente.

Para resolver nuestro sistema de ecuaciones (expresado en forma de una matriz) se usó el método conocido como .Eliminación Gaussiana", que nos permite triangular una matriz de forma relativamente eficiente (es decir llegar de una matriz común a una triangular superior) recurriendo a la resta entre filas de la misma matriz (con un multiplicador que sirva para reducir los valores por debajo de la diagonal a 0).

Una vez triangulada la matriz, pueden hallarse los valores del vector incógnita más fá reemplazando de abajo hacia arriba con los valores conocidos.

La segunda parte del problema consistía en encontrar un algoritmo que permitiera elegir qué sanguijuelas eliminar para que la temperatura del punto crítico del vidrio se mantuviera por debajo de 235º, eliminando la menor cantidad posible. El algoritmo

## 3. Desarrollo

Nuestro problema entonces se reducía a plantear una resolución En un principio pensamos en utilizar una sola estructura para el problema, una matriz que generara la matriz a partir de los datos leídos en el archivo de entrada (que nos indica los datos relevantes para hallar la temperatura en cada punto del parabrisas). El problema que encontramos era que al crear la matriz, solo guardábamos sus celdas, perdiendo los datos originales del problema. De esta forma, no podíamos generar la matriz parabrisas sin el saber su ancho, para devolver los datos de la forma requerida. Guardar estos datos como parte de la estructura hubiera sido una solución forzada, que hubiera hecho que la matriz dejara de ser una estructura genérica para volverse específica de este problema. Por eso, optamos por una estructura parabrisas, que guardara todos los datos relevantes.

Para almacenar la matriz se usaron dos métodos. El primero fue un vector de filas, donde cada fila era un vector con todos sus valores. Para el segundo aprovechamos la estructura banda de la matriz (¿explicar/definir?/desarrollar) para almacenar, de cada fila, solo los elementos que forman parte de la banda. Al triangular, la matriz mantiene su ancho de banda.

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

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

Al implementar esto, hubo que modificar el algoritmo que crea la matriz a partir de un parabrisas para que solo definiera los elementos de la banda.También hubo que modificar indirectamente la eliminación gaussiana, ya que al restar filas solo debían tenerse n cuenta en las celdas que formaran parte de la banda.

# 4. Resultados

Tiempo de ejecución según la granularidad hacer grafico con esto: h banda sin banda
0.5 223.142 -
1 13.774 462.923
2 0.893 13.245
2.5 0.38 4.374
5 0.037 0.153
10 0.009 0.062

# 5. Discusión

Algo de factorización LU? no tengo idea de qué va acá

# 6. Conclusiones

La banda es más rápida (¿cuanto?) y ocupa menos memoria (¿cuanto?)

Reducir n veces la granularidad implica aumentar el tiempo de ejecución n6 veces en el caso de la matriz común y n4 en el caso de la banda

El tiempo de ejecución no depende de la cantidad de sanguijuelas, salvo porque puede implicar más puntos determinados.

Algo del algoritmo de eliminación. Por ejemplo, que al eliminar cada sanguijuela baja la temperatura, salvo que haya dos en exactamente el mismo lugar. Eso habla bien del algoritmo. Algo de cuánto tarda, por ejemplo, comparando con un algortimo que asegure ser óptimo.

Qué tal si vemos cuanto tardan las diferentes partes del programa.

# 7. Apéndice A

# 8. Referencias