1 Introducción

El parabrisas la nave del capitán Guybrush Threepwood está siendo atacado por sanguijuelas mutantes. Dicho ataque consiste en aplicar altas temperaturas constantes sobre la superficie del mismo, con el objetivo de lograr romperlo, para poder lograr un ataque más mortífero. La superficie del parabrisas donde se aplica el calor es circular (la sopapa de ataque es circular).

Para defenderse de estos ataques Guybrush cuenta solamente con un sistema de refrigeración que aplica temperaturas de -100°C a los bordes del parabrisas. El parabrisas se romperá si alcanza una temperatura mayor o igual a los 235°C en el punto central (llamaremos a este punto, *punto crítico*).

Si el sistema de refrigeración no es suficiente para salvar el parabrisas, se puede utilizar un arma para destruir algunas sanguijuelas, pero se desea que sea la menor cantidad posible, siempre y cuando el parabrisas siga en pie, pues dicha arma consume energía que es de vital importancia.

1.1 Temperatura del Parabrisas

Para calcular las temperaturas en el parabrisas se aplicará el siguiente criterio:

En los bordes, como se explicó anteriormente, la temperatura será de -100°C, es decir sean x e y las coordenadas del parabrisas, y $T(x, y)$ la función que devuelve la temperatura en un determinado punto, sea b el ancho y a el alto:

$$T(x,y) = -100^{o}C \qquad si \quad x = 0 \quad \vee \quad x = b \quad \vee \quad y = 0 \quad \vee \quad y = a \quad (1)$$

La temperatura de los puntos que se encuentren dentro del perímetro de la sopapa circular de una sanguijuela será igual a la temperatura aplicada por dicha sanguijuela (T_s).

$$T(x,y) = T_s \qquad si \quad (x,y) \in PuntosSanguijuela \quad (2)$$

La temperatura en el resto de los puntos en el estado estacionario satisface la siguiente ecuación.

$$\frac{\delta^2 T(x,y)}{\delta x^2} + \frac{\delta^2 T(x,y)}{\delta y^2} = 0 \quad (3)$$

1.2 Problemas

En el siguiente trabajo veremos como la aritmética finita de las computadoras puede generar distintos problemas.

En principio deberemos representar al parabrisas, el cual está compuesto por infinitos puntos. Sabemos que no es posible representar en una computadora los infinitos puntos del mismo, por lo que se utilizará cierta discretización, la cual haremos variar con el motivo de estudiar el comportamiento de nuestro sistema.

Otro problema que deberemos afrontar es que al trabajar en la búsqueda de soluciones de problemas que conllevan a la utilización de gran cantidad de operaciones matemáticas de punto flotante, en cada operación se puede perder cierta precisión, y la acumulación de estos errores puede escalar hasta llegar a una solución no satisfactoria. Esta pérdida de precisión se debe nuevamente a la limitación de las computadoras para representar números infinitos.

2 Desarrollo

2.1 Planteo del Problema

Como dijimos anteriormente en nuestro problema teníamos un parabrisas con infinitos puntos, por ser una superficie continua, una forma de pensar el problema es discretizar estos puntos, y trabajar sobre ello. Una vez hecha esta operación, podemos modelar los puntos resultantes con una matriz, donde cada posición de esta matriz represente un punto en el parabrisas.

Al discretizar los puntos, la ecuación (3) para obtener temperaturas en cada punto del parabrisas continuo, se transforma en la siguiente ecuación por diferencias finitas.

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

Es decir que en el parabrisas discretizado, la temperatura en cada punto se calcula como el promedio de la temperatura de los puntos vecinos (los puntos que estan arriba, abajo, izquierda y derecha).

En nuestro problema nos interesa conocer el punto crítico (el centro del parabrisas), esto además significa conocer la temperatura de sus vecinos, y a su vez los vecinos necesitaran conocer la temperatura de sus otros vecinos. Es decir que en un principio es necesario calcular la temperatura de todos los puntos del parabrisas discretizado. Este problema es modelado mediante un sistema de ecuaciones, en el cual cada ecuación se corresponde con un solo punto del parabrisas. Dicho sistema de ecuaciones lo representamos en una matriz cuyo tamaño es *#puntos* x *#puntos*. Finalmente podemos calcular la temperatura en cada punto aplicando *eliminación gaussiana* sobre la matriz.

Una vez que sabemos qué temperatura hay en el punto crítico, podremos decidir que criterio utilizar para matar sanguijuelas, mediante distintas experimentaciones.

2.2 Nuestro sistema de ecuaciones.

Con el objetivo de calcular la temperatura de cada punto de nuestra matriz de $n \times m$, la cual es una representación discreta del parabrisas en nuestro problema inicial, planteamos un sistema de ecuaciones compuesto por $n*m$ incógnitas (una por cada punto). Con este motivo, numeramos los puntos (i, j) del parabrisas mediante la función:

$$F(i,j) = i * m + j, \quad F(i,j) \in [0, n * m) \tag{5}$$

Además, como se mencionó anteriormente, para cada punto $\alpha_{i,j}$ en el parabrisas se le corresponde una de las siguientes ecuaciones:

(6)

$$T_\alpha = -100 \qquad \text{si } (i,j) \text{ es un borde,} \tag{7}$$

$$T_\alpha = T_{Sanguijuela} \qquad \text{si } (i,j) \in \text{Sanguijuela,}$$

$$-4T_\alpha + T_{\alpha+1} + T_{\alpha-1} + T_{\alpha+m} + T_{\alpha-m} = 0 \qquad \text{Demás casos}$$

Finalmente, representamos el sistema de las ecuaciones $(0, \ldots, n * m - 1)$ como la matriz extendida de $Ax = B$. A modo de ejemplo la representación del sistema de ecuaciones de un panel de 4×4 es:

$$\left(\begin{array}{cccccccccccccccc|c}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -100 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -100 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -100 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -100 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -100 \\
0 & 1 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & T_{Sang1} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -100 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -100 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & T_{Sang2} \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -100 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -100 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -100 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -100 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -100
\end{array}\right)$$

2.3 Eliminación Gaussiana

El método de *Eliminación Gauss-Jordan*, también conocido como *Eliminación Gaussiana*, como ya mencionamos anteriormente nos permite resolver un sistema de ecuaciones. En nuestro caso lo utilizamos para calcular la temperatura en cada punto del parabrisas.

Nuestro sistema de ecuaciones está representado con una matriz extendida A que contiene los coeficientes de las ecuaciones junto a una columna extra con sus soluciones, y un vector x que contiene las incógnitas. Luego nosotros queremos saber el resultado de $Ax = b$, y para esto es que usamos el método de *Eliminación Gaussiana*, que consiste en operar sobre la matriz A para obtener una matriz A' triangular superior. Estas operaciones son equivalentes a sumar o restar ecuaciones entre si (o múltiplos de las mismas). Una vez triangulada la matriz es fácil obtener las soluciones, mediante un algoritmo de *backward substitution* (sustitución hacia atrás), que se verá en la siguiente sección.

Para obtener ceros debajo de la diagonal (obtener la matriz triangular) lo que se hace es, a cada fila que se encuentra debajo de la "última fila triangular" se le resta esta última, multiplicada por el coeficiente c obtenido de la siguiente manera; c se obtiene mediante la división del valor que se encuentra en la columna en la cual queremos dejar ceros, y en la fila a la cual le vamos a sustraer la "última fila triangular", dividido el valor que se encuentra en la misma columna y en la "última fila triangular".

Donde "última fila triangular" es una fila que cumple:

- tiene ceros a la izquierda de la diagonal
- todas las filas anteriores tienen ceros a la izquierda de la diagonal
- la fila siguiente no tiene todos ceros a la izquierda de la diagonal

A continuación se muestra un ejemplo de la primer iteración de *Eliminación Gaussiana* en una matriz de 4×4

$$\begin{pmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{pmatrix}$$

El resultado luego de la primer iteración es el siguiente:
Para $c_1 = \frac{a_{10}}{a_{00}}$, $c_2 = \frac{a_{20}}{a_{00}}$, $c_3 = \frac{a_{30}}{a_{00}}$

$$\begin{pmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ 0 & a_{11} - c_1 * a_{01} & a_{12} - c_1 * a_{02} & a_{13} - c_1 * a_{03} \\ 0 & a_{21} - c_2 * a_{01} & a_{22} - c_2 * a_{02} & a_{23} - c_2 * a_{03} \\ 0 & a_{31} - c_3 * a_{01} & a_{32} - c_3 * a_{02} & a_{33} - c_3 * a_{03} \end{pmatrix}$$

A cada elemento de una fila $0 < j < 4$ se le resta el elemento correspondiente a la fila 0 multiplicado por un coeficiente (que se mantiene igual para toda la fila)

A continuación se muestra un pseudo-código del algoritmo.

Algorithm 1 Pseudo-código del algoritmo de eliminación Gaussiana

1: **for** cada columna de la matriz **do**
2: **for** cada fila debajo de la diagonal **do**
3: calculo el coeficiente correspondiente a la división del elemento de la columna actual y la fila actual, y el elemento de la diagonal
4: le resto la fila de la diagonal por el coeficiente a la fila actual
5: **end for**
6: lleno de ceros a la izquierda de la diagonal
7: **end for**

Notese que en la línea 6 agregamos ceros a la izquierda de la diagonal, ya que sabemos que matemáticamente deberíamos obtener dicho resultado. De esta manera nos ahorramos de calcular costosas operaciones de punto flotante, y en lugar de esto simplemente asignamos ceros a los lugares correspondientes de la matriz. Además omitimos cualquier error numérico que podrían traer estas operaciones.

A continuación se muestra un analisis de complejidad del algoritmo, sea n el ancho y el alto de la matriz (pues es una matriz cuadrada), sea j los elementos de la fila que se encuentran a la izquierda de la diagonal en cada iteración.

Algorithm 2 complejidad del algoritmo de eliminación Gaussiana

1: **for** chequear la guarda y aumentar un índice es O(1) **do**
2: **for** chequear la guarda y aumentar un índice O(1) **do**
3: O(1) pues es una división y una asignación
4: para cada elemento de la fila que se encuentra despues de la diagonal hago una multiplicacion y una resta agisnaciones O(2(n-j))
5: **end for**
6: O(j)
7: **end for**

El primer **for** de la línea 1 itera para todas las columnas excepto la ultima (que ya esta triangulada), es decir itera $n-1$ veces.

El segundo **for** en la línea 2 itera $n-i$ veces, donde i es la columna en la que nos encontramos en cada iteración, es decir en el peor caso itera $O(n-1)$

En la linea 4 obtenemos $O(2(n-j))$, en el peor caso $j=1$, luego esto se reduce a $O(n-1)$

en la línea 6 tenemos la complejidad $O(j)$, en este caso, el peor caso se da cuando $j=n-1$

Finalmente la complejidad total se resume a $O(n^3)$