Recordemos que el métodos de gradientes conjugados es aplicable a matrices simétricas y definidas positivas. Veamos que resulta de la matriz del problema. Notemos que propondremos una forma de implementar las condiciones de borde de inmediato, ya que de lo contrario la forma de implementar las condiciones de borde podría ser inconsistente con la demostración de la correctitud del método.

Sea N_x y N_y la cantidad de puntos en cada dirección. Entonces $h_x = \frac{1}{N_x-1}$ y $h_y = \frac{1}{N_y-1}$. Consideremos, para el nodo $u_{i,j}$ con $0 \leq i \leq N_x - 1$, $0 \leq j \leq N_y - 1$ las siguientes expresiones

$$\alpha_N = \frac{\alpha_{i-\frac{1}{2},j}}{h_x^2}, \quad \alpha_S = \frac{\alpha_{i+\frac{1}{2},j}}{h_x^2}, \quad \alpha_E = \frac{\alpha_{i,j+\frac{1}{2}}}{h_y^2}, \quad \alpha_W = \frac{\alpha_{i-\frac{1}{2},j}}{h_x^2}$$

Además consideremos los casos especiales de las condiciones de borde.

1. Si $i - 1 = 0$: $\alpha_N = 0$
2. Si $i + 1 = N_x - 1$: $\alpha_S = 0$
3. Si $j - 1 = 0$: $\alpha_W = 0$
4. Si $j + 1 = N_y - 1$: $\alpha_E = 0$

Entonces, del problema resultan las ecuaciones interiores

$$(\alpha_W + \alpha_E + \alpha_N + \alpha_S + 1)u_{i,j} - \alpha_W u_{i,j-1} - \alpha_E u_{i,j+1} - \alpha_N u_{i-1,j} - \alpha_S u_{i+1,j} = f_{i,j}$$

En cambio si el nodo es del borde, esto es, cumple que $i = 0$ ó $i = N_x - 1$ ó $j = 0$ ó $j = N_y - 1$ entonces consideramos la ecuación

$$u_{i,j} = 0$$

Ahora veamos que independiente de como aplanemos el vector de nodos la matriz del problema va a ser simétrica. Notemos que en la ecuación (además de si mismo) solo aparecen nodos vécinos y que no son de borde. Sean u_1, u_2 dos nodos vecinos, ninguno de borde y con coordenadas (i_1, j_1) y (i_2, j_2) respectivamente. Entonces en la ecuación de la fila de u_1 el nodo u_2 aparece acompañado del escalar

$$\frac{\alpha_{\frac{i_1+i_2}{2},\frac{j_1+j_2}{2}}}{\mathbb{1}_{(i_1=i_2)}h_y^2 + \mathbb{1}_{(j_1=j_2)}h_x^2}$$

El cual claramente corresponde de igual forma al acompañante de u_1 en la fila del nodo u_2, concluyendo que la matriz es simétrica.

Para ver que es definida positiva consideremos la ecuación anterior. Para esto veamos el discos de Gershgorin para el nodo $u_{i,j}$, esto corresponde a

$$D_{i,j} = \overline{\mathcal{B}(\alpha_W + \alpha_E + \alpha_N + \alpha_S + 1, \alpha_W + \alpha_E + \alpha_N + \alpha_S)} \subseteq [1, \infty)$$

Por tanto, como los valores propios deben pertenecer a algún disco y los valores de los discos son mayores o iguales a uno se concluye que los valores propios de l amtriz son positivos, y por tanto, la matriz es definida positiva.

Como la matriz es definida positiva y simétrica, podemos usar el método de gradientes conjugados.

Notemos que en el problema orignal la matriz sería de tamaño $(N_x \cdot N_y) \times (N_x \cdot N_y)$ con la matriz con a lo sumo 5 entradas por cada fila. Como esta matriz es muy sparse y tiene una estructura heredada del método de elementos finitos podemos usar el almacenamiento por stencil, donde cada entrada (i, j) con $0 \leq i \leq N_x - 1$ y $0 \leq j \leq N_y - 1$ representa un nodo. Para esto, se tiene que la matriz A la separamos en 5 matrices N, W, C, E, S de tamaño $N_x \times N_y$ cada una. Además, los vectores b, x, r, q, p del algoritmo los consideramos desaplanados, esto es, son matrices de tamaño $N_x \times N_y$. Por último, los escalares ρ y β siguen siendo escalares. Además, voy a reusar la memoria para no tener que guardar la información de todas las iteraciones del algoritmo, ya que solo necesitamos la anterior.

El algoritmo se encuentra implementado en C++ y adjunto a la entrega. Notemos que las condiciones de borde fueron implementadas como antes, esto es, forzar a dichos valores a ser 0 en la solución de la ecuación (considerando la ecuación $u_{ij} = 0$ para los nodos del borde), y además eliminando su aparición en las otras ecuaciones para que la matriz siga siendo simétrica.

Se testeó la implementación y efectivamente converge y las variables guardan lo que tienen que guardar. Por lo tanto, a partir de ahora voy a comentar la impresión en consola de los restos (ya que me molesta un poco la verdad). Además, se verifica que en los nodos del borde la solución nos queda nula (obedeciendo a las condiciones de borde).

Luego lo implementé en paralelo usando OpenMP. Para evitar las race conditions lo que hice fue en un principio adecuar el código de tal forma que en los bucles solo se hagan asignación de tal forma que se asignen cosas a la entrada (i, j) del bucle actual de forma independiente, por lo que la única variable cambiada será propia de la iteración actual del bucle y no se relaciona con otras iteraciones (aunque quizás el acceso a otras si pueda ser compartida, pero como dichas no son modificadas no son problemáticas). La única operación que hay que tener cuidado es la de reducción de suma, pero como OpenMP ofrece una forma sencilla para tratar con esto simplemente la ocupamos y nos escapamos de cualquier tipo de problema.

Y para la paralelización de los ciclos anidados en primera instancia opté por solamente paralelizar el ciclo externo. Esto se debe a que mi número de núcleos es pequeño comparado con el N_x con el que pretendo trabajar, por lo que aunque un hilo se quede esperando toda una fila a los otros sin ayudarle, esto porcentualmente no debiera ser mayor. La gran ventaja de esta decisión es que para una fila fija si hacemos variar las columnas los arreglos están almacenados de tal forma que toda esa memoria es contigua, y por tanto podemos aspirar a accesos de memoria más rápidos. Luego hice el experimento y haciendo la paralelización de los dos ciclos a la vez al parecer era más rápido, así que lo dejé así (quizás quedarse esperando no era tan marginal como lo pensaba).

Ahora hagamos un experimento de escalabilidad fuerte. Para esto consideraremos $N_x = 1000$, $N_y = 1000$ y 3000 iteraciones. La tabla del experimento se muestra a continuación.

Tamaño de Matriz	Iteraciones	Hilos	Tiempo	Aceleración	Eficiencia
1000×1000	3000	1	87s	1.00	1.00
1000×1000	3000	2	47s	1.85	0.925
1000×1000	3000	3	35s	2.48	0.83
1000×1000	3000	4	34s	2.55	0.63
1000×1000	3000	5	32s	2.71	0.54
1000×1000	3000	6	28s	3.10	0.51
1000×1000	3000	7	26s	3.34	0.47
1000×1000	3000	8	25s	3.48	0.43
1000×1000	3000	9	25s	3.48	0.38
1000×1000	3000	10	25s	3.48	0.34
1000×1000	3000	11	25s	3.48	0.31
1000×1000	3000	12	30s	2.90	0.24

Se puede ver que el algoritmo tiene una buena eficiencia en cuanto a la paralelización fuerte, y más importante, una gran aceleración (con más de 3 hilos no es buena pero debe ser porque el problema igual es pequeño o por temas de hardware). Esto nos permite hacer este algoritmo realmente escalable, ya que recordemos que estamos resolviendo un sistema de ecuaciones con 1 millón de ecuaciones en mi computador personal, viéndolo de esta forma resulta increíble que podamos hacer algo de ese tamaño.