Precondicionamiento y su utilidad en la generación de mallas con procesos iterativos

Universidad Católica San Pablo
Maestría en Ciencias de la Computación

20 de septiembre de 2017

Índice General

Introducción

El interes fundamental de este trabajo, se centra en mostrar maneras de como acelerar métodos iterativos para la solución numérica Ax = b, con la matriz A no singular de $\mathbb{R}^{nxn}$ y $b \in \mathbb{R}^n$. la acelaración se consigue con precondicionamiento que será explicado con mas detalles es los siguientes puntos de este trabajo. Con algún detalle, se consideran precondicionamientos basados en métodos iterativos estacionarios y en factorización LU incompleta

Otro punto a tratar en este trabajo, es la importancia de los metodos iterativos en la generación de mallas, como caso particular se mostrará la generación de malla en 2D. La técnica fundamental para generar este tipo de malla, es la deformación de una malla cartesiana inicial y su posterior alineación con sus fronteras internas.

1 Precondicionamiento

1.1 Historia

El término **precondicionamiento** parece haber sido utilizado por primera vez en 1948 por **Alan Turing**, en el artículo: *El efecto de los errores de redondeo en los métodos de solución directa.* Sin embargo, el primer uso del término en relación con los métodos iterativos se encuentra en un documento de **D. Evans** sobre la aceleración que aplicaba el ruso **Chebyschev** en el método de SSOR en 1968.

Lamberto Cesari en 1937 tenía un concepto de precondicionamiento como un medio para poder reducir el **número condicionante** y así mejorar la convergencia de un proceso iterativo, su idea fue usar un polinomio p(A) de bajo grado aplicada al sistema lineal

$$Ax = b \tag{1}$$

así

$$p(A)Ax = p(A)b$$

sería un sistema lineal precondicionado, donde A es una matriz de orden n simétrica y definida positiva.

El precondicionar un sistema lineal es una de las principales fuentes para obtener resultados más eficientes computacionalmente. Es así que en los últimos años se ha desarrollado una mayor investigación con respecto a los métodos de solución directa e incluso sobre el subespacio de Krylov.

1.2 Formulación Matemática

El término **precondicionamiento** se refiere a transformar el sistema lineal (1) en otro con propiedades más favorables en cuanto a la solución iterativa y que siga manteniendo la solución **x**. Un precondicionador es una matriz **M** que efectúa tal transformación. Para la matriz A del sistema (1), el número condicionante **cond(A)** se define como:

$$cond(A) = ||A||\ ||A^{-1}||$$

Luego de pre multiplicar al sistema lineal por la matriz **M** se debería verificar lo siquiente

$$MAx = Mb \quad , \ cond(MA) < cond(A)$$

Esta desigualdad es fundamental para garantizar al favorecimiento de los métodos iterativos, siempre que sea convergente. El mejor precondicionador para el sistema (1) sería $M = A^{-1}$ pues con $cond(A^{-1}A) = cond(I) = 1$ sería el óptimo y así el sistema convergería en tan sólo una interación, pero el problema es calcular el coste computacional de A^{-1}, esto equivaldría a calcular A^{-1} por algún método directo. Es por ello que se debe buscar un $M \approx A^{-1}$ sin coste elevado.

Para garantizar la convergencia de la matriz A, es suficiente mostrar que su radio espectral ρ es menor a la unidad, matemáticamente $\rho(A) < 1$.

Si en la ecuación (1) hacemos $A = M - N$ donde M es una matriz no singular, entonces

$$\begin{gathered}(M-N)x = b \\ Mx = Nx + b \\ x = M^{-1}Nx + M^{-1}b \\ x = M^{-1}(M-A) + M^{-1}b \\ x = (I - M^{-1}A)x + M^{-1}b\end{gathered}$$

donde I es la matriz indentidad de orden n. En este caso para que el sistema linea converja, se debe cumplir que $\rho(I - M^{-1}A) < 1$. Además mientras el radio espectral sea más cerca a cero entonces la velocidad de convergencia será mayor.

Los precondicionadores deben cumplir principalmente dos propiedades:

- Facilidad de implementación (bajo coste computacional).
- Debe mejorar la convergencia del sistema lineal.

Cabe resaltar que los precondicionadores no solamente se obtienen multiplicando por una matriz a la izquierda, sino también por la derecha e incluso por ambos lados.

1.3 Principales Precondicionadores

Dentro de los precondicionadores podemos distinguir dos grupos: los explícitos y los implícitos.

1. **Precondicionadores Explícitos:** Se busca encontrar una matriz M normalmente a partir de una factorización aproximada, como ejemplo de precondicionadores implícitos tenemos a Jacobi (diagonal), el SSOR, Gradiente Conjugado y las factorizaciones incompletas (Cholesky).

2. **Precondicionadores Implícitos:** Se construyen calculando directamente $M \approx A^{-1}$, aquí se encuentran los precondicionadores polinomiales e inversa aproximada de A.

2 Métodos Iterativos

La importancia de los métodos iterativos en álgebra lineal se deriva de un simple hecho: los métodos directos requieren de $O(n^3)$, mientras que los métodos iterativos pueden llegar a requerir solamente $O(n)$. Es así que para matrices con $n > 10^3$ se va volviendo intratable el no pensar resolverlo con un algoritmo iterativo. Aunque los métodos iterativos requieren menos almacenamiento no tienen la fiabilidad de los métodos directos en cuanto a precisiones de solución se requiere.

En algunas aplicaciones, los métodos iterativos fallan y es donde el precondicionamiento es necesario, lamentablemente no siempre es suficiente, para lograr la convergencia en un tiempo razonable. Mientras que los métodos directos están practicamente basados en alguna actualización de la eliminación gaussiana, mientras que los métodos iterativos comprende una gran variedad de técnicas, que van desde métodos verdaderamente iterativos, como el Jacobi clásico, Gauss-Seidel e iteraciones SOR al subespacio de Krylov , que teóricamente convergen en un número finito de pasos.

Por ejemplo, cuando A es definida positiva y simétrica (hermitiana en el caso complejo) el método del Gradiente Conjugado resuelve el sistema (1) muy rápido bajo ciertas condiciones con respecto a sus autovalores.

2.1 Precondicionadores

La velocidad de convergencia de los métodos iterativos depende de las propiedades espectrales de la matriz del sistema. Así si M es una matriz invertible que se aproxima de cierta manera a la matriz del sistema, A, los sistemas

$$Ax = B$$

$$M^{-1}Ax = M^{-1}B$$

tienen las mismas soluciones. No obstante, es posible que las propiedades espectrales de $M^{-1}A$ sean más favorables que las de A. Al sistema

$$M^{-1}Ax = M^{-1}B$$

se le llama sistema precondicionado por la izquierda. Hay otras posibles estrategias para el precondicionado de un sistema como usar un precondicionador por la derecha o combinar iteraciones de distintos métodos iterativos.

Una posible elección de M, que se denomina precondicionador del sistema, es el precondiconador de Jacobi donde M = diag(A).

Otro tipo de precondicionadores, en general, más eficientes son los basados en descomposiciones LU incompletas de la matriz A.

3 Generación de Mallas

En esta sección explicaremos conceptos previos para la generación de mallas, los cuales nos ayudaran a entenderentender la problemática generada, y cómo es que podemos aplicar un método de precondicionamiento para agilizar la generación de mallas.

3.1 ¿Qué es Generación de Mallas?

La idea principal en la generacion de mallas, esta dada en la idea de mallar un espacio dado algunos limites dado un dominio, por ejemplo el ala de un avion. Existen técnicas muy generales, capaces de generar mallas en geometrías complejas, como las de frente de avance o las de Delaunay-Voronoï, que presentan un coste computacional elevado. Por contra, métodos algebraicos generan la malla a una gran velocidad, pero son incapaces de mallar geometrías de cierta complejidad. La geometría del dominio, el coste computacional y la capacidad de control que queramos tener sobre la malla será

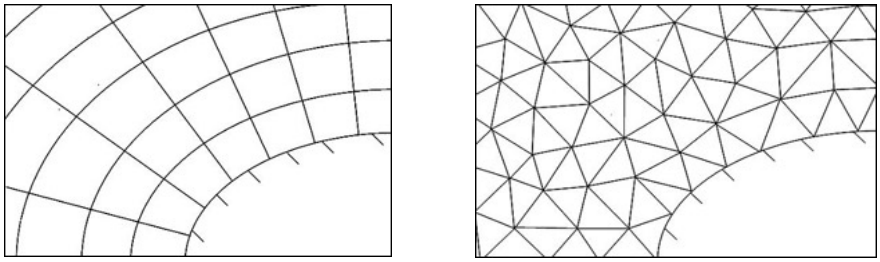
lo que nos haga decidir por unos u otros métodos.

Las mallas se clasifican de acuerdo a su estructura en dos tipos: mallas estructuradas y mallas no estructuradas. La diferencia básica entre una malla estructurada y una malla no estructurada radica en la forma de agrupar los datos que describen la malla. Estas a su vez se subdividen por los métodos empleados en su generación.

- Métodos de generación de malla estructurada:
  - Algebraicos.
  - Basados en EDPs.
  - Superposición-deformación de retícula.
  - Crecimiento estructurado.
- Métodos de generación de malla no estructurada:
  - Inserción de nodos y posterior conexión: Delaunay.
  - Generación simultánea de nodos y conectividad: Frente de avance.
  - Métodos Multibloque.

Una malla estructurada consiste generalmente de cuadriláteros, los cuales son formados por un grupo de coordenadas y conexiones que naturalmente son mapeados en elementos de una matriz. Los puntos vecinos dentro de la malla en el espacio físico son los elementos vecinos en la matriz de la malla. De esta manera, un arreglo bidimensional x(i,j) puede usarse para almacenar las coordenadas de los puntos para una malla en 2D.

En una malla no estructurada los puntos no pueden ser representados de la misma manera que en las mallas estructuradas, entonces se requiere proveer información adicional para describir la malla. Para representar cualquier punto particular, la conexión con otros puntos puede definirse en la matriz de conectividad.



a) b)

Figura 1: a) Malla estructurada b) Malla no estructurada.

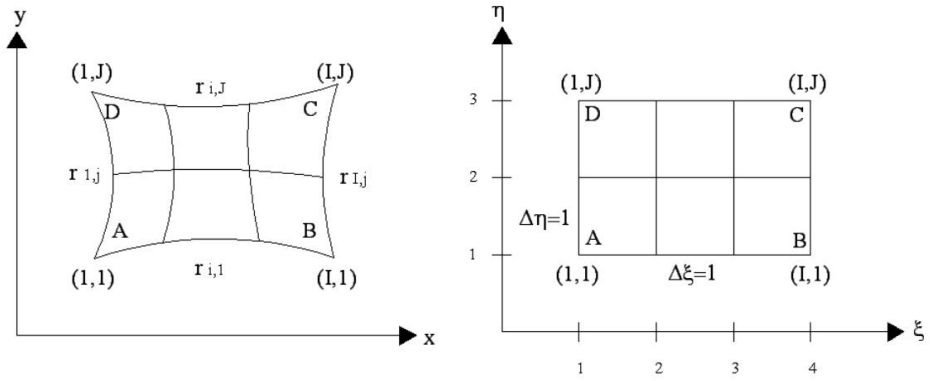
El uso de una malla no estructurada es factible en la discretización de geometrías complicadas. Sin embargo, la carencia de una estructura global en una malla no estructurada hace que la aplicación de los algoritmos de solución de barrido de línea sean más difíciles de aplicar que en las mallas estructuradas, impactando directamente en el desarrollo del código computacional haciéndolo más complejo e incrementando el tiempo de cálculo en la solución de problemas en la que se aplique este tipo de discretización.

Para nuestro proposito se plantea estudiar mallas estructuradas, la cual a la vez se utilizaran los métodos algebraicos y los métodos basados en PDEs

3.2 Métodos para Generación de Mallas Estructuradas

Generalmente, para fenómenos físicos que tienen lugar en geometrías sencillas, por ejemplo, una cavidad rectangular y la sección transversal de un tubo, se elige un sistema coordenado ortogonal adecuado. Estos sistemas coordenados no necesitan transformación alguna, es decir, el plano físico es igual al plano computacional. Sin embargo, cuando la geometría ya no puede ser representada por los sistemas coordenados ortogonales, es necesario realizar una transformación que permita mapear la región física arbitraria a una región computacional regular.

Para ilustrar los conceptos básicos del mapeo se considera un dominio físico bidimensional en coordenadas cartesianas x, y y un dominio computacional en coordenadas cartesianas ξ, η. La transformación entre las coordenadas x, y y ξ, η debería ser tal que las fronteras del dominio físico deben coincidir con las coordenadas curvilíneas.



a) b)

Figura 2: a) Plano Físico b) Plano Computacional.

Existen muchos métodos para la generación de mallas estructuradas disponibles en la literatura. Fundamentalmente, estos pueden clasificarse en algebraicos y diferenciales. Los algebraicos emplean diferentes tipos de interpolación, así como, expresiones analíticas exactas y son bastante versátiles y rápidos. Los diferenciales, llamados así por que emplean sistemas de ecuaciones diferenciales, son más generales, en contrapartida, requieren un tiempo de calculo mayor y una elaboración matemática más compleja que los métodos algebraicos. .

3.2.1 Métodos Algebraicos Basados en Interpolación

En estos métodos, se usan ecuaciones algebraicas para relacionar los puntos de malla del dominio físico con el dominio computacional. Una vez definida la geometría del dominio físico y el dominio computacional construido, la malla en el dominio físico será obtenida por ecuaciones algebraicas. La base de estos métodos es el uso de técnicas de interpolación y relaciones algebraicas exactas, lo cual permite la obtención rápida de la malla. Esta es la mayor ventaja de los métodos algebraicos.

3.2.2 Métodos Basados en PDEs

En estos métodos se resuelve un sistema de EDP's para localizar los puntos en el interior del dominio físico, esto es, para generar la malla, pueden clasificarse en sistemas elípticos, parabólicos e hiperbólicos, de acuerdo al tipo

de ecuación diferencial que se utiliza para generar la malla.

3.2.3 Relación de Transformación

A continuacion se muestra la relación de transformación entre ambos planos:

$$\xi = \xi(x,y,z), \eta = \eta(x,y,z), \zeta = \zeta(x,y,z)$$

$$\begin{bmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} & \frac{\partial u}{\partial z} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} & \frac{\partial v}{\partial z} \\ \frac{\partial w}{\partial x} & \frac{\partial w}{\partial y} & \frac{\partial w}{\partial z} \end{bmatrix} = \begin{bmatrix} \frac{\partial u}{\partial \xi} & \frac{\partial u}{\partial \eta} & \frac{\partial u}{\partial \zeta} \\ \frac{\partial v}{\partial \xi} & \frac{\partial v}{\partial \eta} & \frac{\partial v}{\partial \zeta} \\ \frac{\partial w}{\partial \xi} & \frac{\partial w}{\partial \eta} & \frac{\partial w}{\partial \zeta} \end{bmatrix} \begin{bmatrix} \frac{\partial \xi}{\partial x} & \frac{\partial \xi}{\partial y} & \frac{\partial \xi}{\partial z} \\ \frac{\partial \eta}{\partial x} & \frac{\partial \eta}{\partial y} & \frac{\partial \eta}{\partial z} \\ \frac{\partial \zeta}{\partial x} & \frac{\partial \zeta}{\partial y} & \frac{\partial \zeta}{\partial z} \end{bmatrix}$$

$$J = \begin{bmatrix} \frac{\partial \xi}{\partial x} & \frac{\partial \xi}{\partial y} & \frac{\partial \xi}{\partial z} \\ \frac{\partial \eta}{\partial x} & \frac{\partial \eta}{\partial y} & \frac{\partial \eta}{\partial z} \\ \frac{\partial \zeta}{\partial x} & \frac{\partial \zeta}{\partial y} & \frac{\partial \zeta}{\partial z} \end{bmatrix} \quad J^{-1} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial x}{\partial \eta} & \frac{\partial x}{\partial \zeta} \\ \frac{\partial y}{\partial \xi} & \frac{\partial y}{\partial \eta} & \frac{\partial y}{\partial \zeta} \\ \frac{\partial z}{\partial \xi} & \frac{\partial z}{\partial \eta} & \frac{\partial z}{\partial \zeta} \end{bmatrix}$$

3.3 Sobre los Métodos Usados

En nuestra implementación decidimos seguir el procedimiento propuesto en varios libros y papers. Seguimos el siguiente procedimiento:

* **Uso de un generador algebraico**
  Usamos un generador basado en la Interpolación Transfinita. La malla generada se usará como condición inicial para nuestro generador basado en PDEs.

* **Discretización de las ecuaciones del generador elíptico**
  El generador basado en PDEs usado es el Elíptico, cuyas ecuaciones son discretizadas y formadas en un método iterativo para calcular un grid por medio de la Iteración de Picard.

* **Refinamiento de la malla**
  En este último paso usamos el generador Elíptico discretizado para así poder refinar la malla inicial generada por el generador algebraico.

Acontinuación pasamos a describir los métodos usados.

3.3.1 Método Algebraico de Interpolación Transfinita

En nuestra implementación usamos el generador algebraico basado en Interpolación Transfinita. Este puede ser formulado como sigue :

$$\begin{gathered} x(\xi,\eta) = (1-\xi)x_l + \xi x_r + (1-\eta)x_b + \eta x_t - \dots \\ (1-\eta)(1-\xi)x_b(0) - (1-\xi)\eta x_t(0) - \dots \\ (1-\eta)\xi x_b(1) - \eta\xi x_t(1) \\ y(\xi,\eta) = (1-\xi)y_l + \xi y_r + (1-\eta)y_b + \eta y_t - \dots \\ (1-\eta)(1-\xi)y_b(0) - (1-\xi)\eta y_t(0) - \dots \\ (1-\eta)\xi y_b(1) - \eta\xi y_t(1) \end{gathered}$$

Donde, $x_l(\eta)$, $y_l(\eta)$, $x_r(\eta)$, $y_r(\eta)$, $x_b(\xi)$, $y_b(\xi)$, $x_t(\xi)$, $y_t(\xi)$ son las fronteras de la geometría expresadas como función de ξ y η, las coordenadas en el espacio de cómputo.
En nuestro caso, cargamos la geometría de ejemplo mostrada en la siguiente figura :

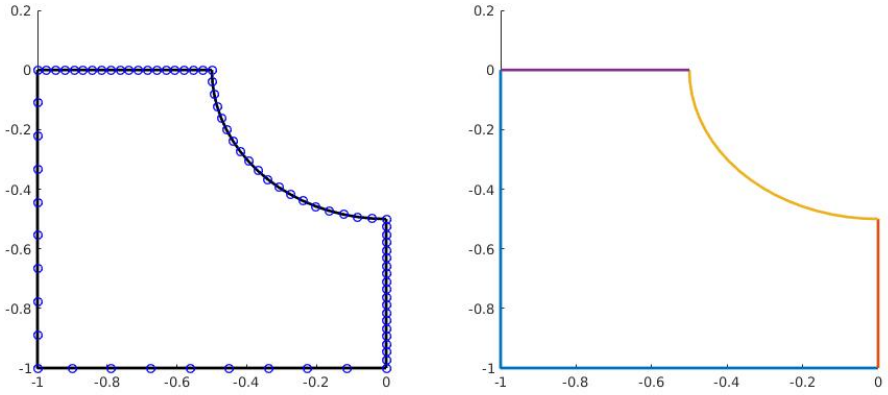


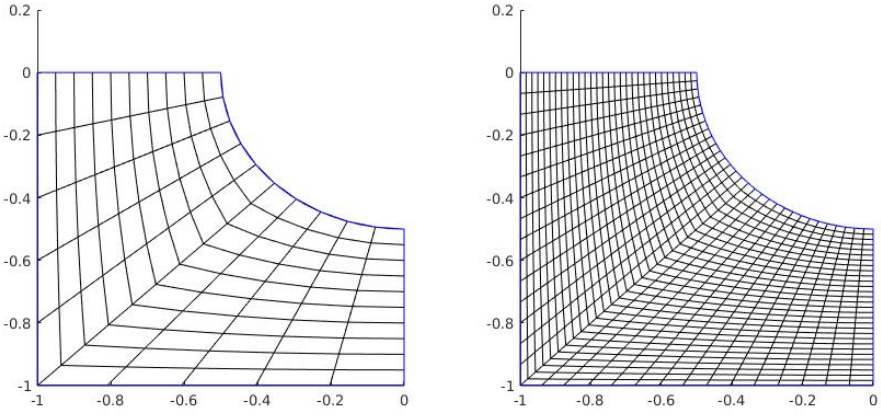
Figura 3: Geometría de prueba

Esta geometría está dada en forma de puntos, por lo que no tenemos expresiones analíticas para las fronteras. Para esto, expresamos las fronteras como funciones lineales en trozos, haciendo que cada frontera sea definida por un conjunto de subfunciones tipo segmento de recta definidas por los

puntos que definen la frontera. Esto se expresa como lo siguinte:

$$x(q) = \Big\{ x_i(q) = x(i) + q\{x(i+1) - x(i)\}$$
$$y(q) = \Big\{ y_i(q) = y(i) + q\{y(i+1) - y(i)\}$$
$$q = \xi, \eta$$

Al aplicar este método a la geometría dada, obtenemos los siguientes resultados.



a) b)

Figura 4: Grid generado para la geometría de prueba usando el método algebraico, size de a)10 y b)30

3.3.2 Método Basado en PDEs Elípticas

Para poder refinar la malla generada por el generador algebraico usamos un generador basado en PDEs. El generador es del tipo elíptico, el cuál define la nueva grilla como la solución a la siguiente PDE :

$$\xi_{xx} + \xi_{yy} = 0$$
$$\eta_{xx} + \eta_{yy} = 0$$

Éstas PDEs las transformamos del espacio x, y al espacio ξ, η, lo cuál nos deja un problema de frontera (con fronteras $x(\xi, \eta)$, $y(\xi, \eta)$) definido por

las siguientes ecuaciones :

$$(x_{\eta\eta}^2 + y_{\eta\eta}^2)x_{\xi\xi} - 2(x_\xi x_\eta + y_\xi y_\eta)x_{\xi\eta} + (x_{\xi\xi}^2 + y_{\xi\xi}^2)x_{\eta\eta}$$
$$(x_{\eta\eta}^2 + y_{\eta\eta}^2)y_{\xi\xi} - 2(x_\xi x_\eta + y_\xi y_\eta)y_{\xi\eta} + (x_{\xi\xi}^2 + y_{\xi\xi}^2)y_{\eta\eta}$$

Al discretizarlas obtenemos las siguientes ecuaciones :

$$\alpha_{ij}(x_{i+1,j} - 2x_{i,j} + x_{i-1,j}) + \gamma_{ij}(x_{i,j+1} - 2x_{i,j} + x_{i,j-1}) - \ldots$$
$$0.5\beta_{ij}(x_{i+1,j+1} - x_{i+1,j-1} - x_{i-1,j+1} + x_{i-1,j-1}) = 0$$
$$\alpha_{ij}(y_{i+1,j} - 2y_{i,j} + y_{i-1,j}) + \gamma_{ij}(y_{i,j+1} - 2y_{i,j} + y_{i,j-1}) - \ldots$$
$$0.5\beta_{ij}(y_{i+1,j+1} - y_{i+1,j-1} - y_{i-1,j+1} + y_{i-1,j-1}) = 0$$

Donde:

$$\xi = \frac{i}{N_\xi}, \eta = \frac{j}{N_\eta}$$
$$\alpha_{ij} = 0.25((x_{i,j+1} - x_{i,j-1})^2 + (y_{i,j+1} - y_{i,j-1})^2)$$
$$\beta_{ij} = 0.25((x_{i,j+1} - x_{i,j-1})(x_{i+1,j} - x_{i-1,j}) + (y_{i,j+1} - y_{i,j-1})(y_{i+1,j} - y_{i-1,j}))$$
$$\gamma_{ij} = 0.25((x_{i+1,j} - x_{i-1,j})^2 + (y_{i+1,j} - y_{i-1,j})^2)$$

Para resolver estas ecuaciones y transformarlas a un sistema lineal hacemos uso de la iteración de Picard, haciendo que los coeficientes α, β, γ sean dependientes de la malla actual, mientras que los otros términos sean dependientes de la malla siguiente, teniendo lo siguiente (caso x) :

$$\alpha_{ij}^k(x_{i+1,j}^{k+1} - 2x_{i,j}^{k+1} + x_{i-1,j}^{k+1}) + \gamma_{ij}^k(x_{i,j+1}^{k+1} - 2x_{i,j}^{k+1} + x_{i,j-1}^{k+1}) - \ldots$$
$$0.5\beta_{ij}^k(x_{i+1,j+1}^{k+1} - x_{i+1,j-1}^{k+1} - x_{i-1,j+1}^{k+1} + x_{i-1,j-1}^{k+1}) = 0$$

Con lo cuál podemos formar el sistema $Az = b$, donde A es la forma matricial de las ecuaciones anteriores luego de aplicar el stencil mostrado en la figura siguiente, z es la representación en vector columna de la grilla en la siguiente iteración-refinamiento y b se obtiene de aplicar las condiciones de frontera.

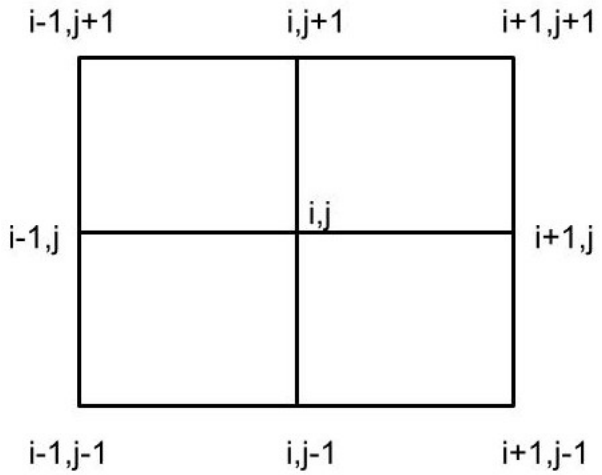
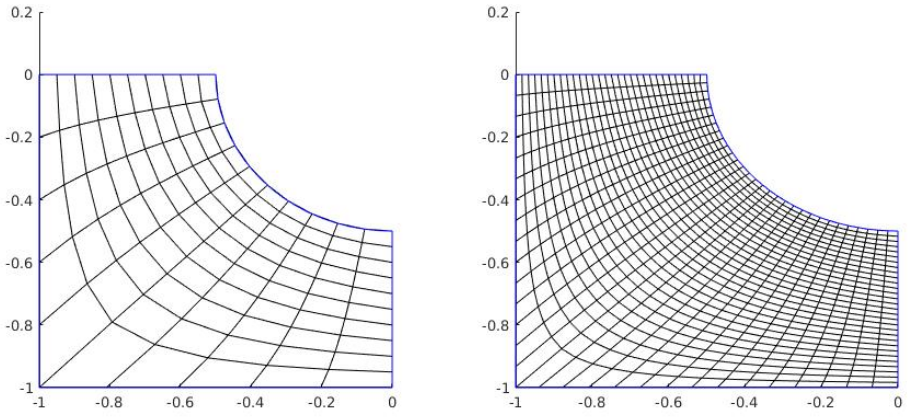


Figura 5: Stencil a aplicar a las ecuaciones discretizadas del generador elíptico

Implementamos el generador elíptico en MATLAB, lo cuál nos dio los siguientes resultados al usar la malla del generador algebraico de la geometría de ejemplo como condición inicial.

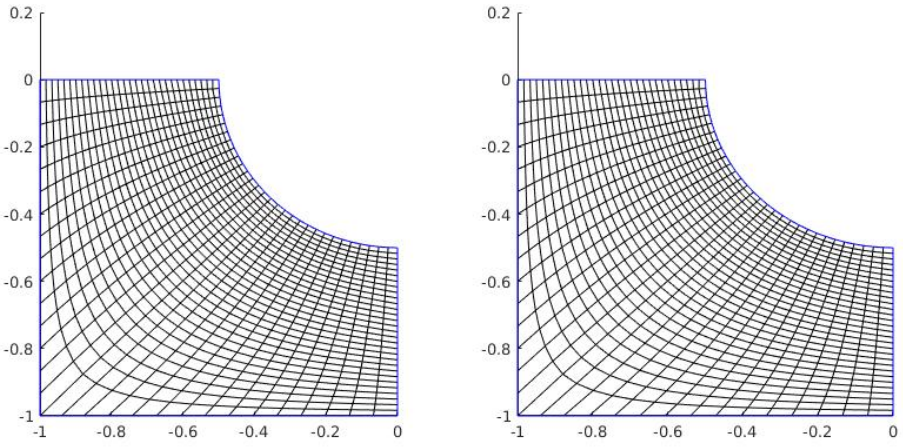


a) b)

Figura 6: Grid generado para la geometría de prueba usando el generador elíptico, size de a)10 y b)30

4 Resultados

En el desarrollo del presente trabajo verificamos la importancia de usar precondicionamiento al momento de resolver un sistema de ecuación lineal para la generación de una malla, para este propósito se uso el método de gauss seidel por bloque como matriz de precondicionamiento, y como se podrá observar en la siguiente figura ambas mallas son generadas correctamente con ambos métodos, con la diferencia que el método de gauss seidel por bloque converge con mucho menos iteraciones que el método de gauss seidel tradicional.



a) b)

Figura 7: a)Gauss seidel por bloque. Converge con 13 iteraciones b)Gauss seidel. Converge con 715 iteraciones

Referencias

[1] MICHELE BENZI: *Preconditioning Techniques for Large Linear Systems: A Survey* - Mathematics and Computer Science Department, Emory University, Atlanta, Georgias, 2002.

[2] YOUSEF SAAD: *Iterative Methods for Sparse Linear Systems* - second Edition, 2000.

[3] JOE F. THOMPSON, BHARAT K. SONI, NIGEL P. WEATHERILL: *Handbook of Grid Generation* - 1999.

[4] LLOYD N. TREFETHEN, DAVID BAU: *Numerical Linear Algebra* - 1997.

5 Anexos

5.1 Código - Gauss Seidel

```
function [x,err] = gs(A,b,x,xstar,MaxIt,tol)
%
% Solve Ax = b using Gauss-Siedel iteration
%
b = b(:); x = x(:); err = zeros(MaxIt,1);
DL = tril(A); U = -triu(A,1);
%disp(sprintf('x = '));
for k = 1:MaxIt
r = b - A*x;
if (norm(r)<tol*norm(b)), break; end
x = x + DL\r;
err(k) = norm(x-xstar);
%disp(sprintf('%12.4e ', x));
end
disp(sprintf('Number of iterations: %d ', k));
end
```

5.2 Código - Gauss Seidel por Bloque

```
function [x,err] = Bgs(A,b,x,xstar,MaxIt,tol)
%
% Solve Ax = b using Block Gauss-Siedel iteration
%
b = b(:); x = x(:); err = zeros(MaxIt,1);
m = round(size(b)/2);
n = size(b) - round(size(b)/2);
DL = [A(1:m,1:m) zeros(m,n); A(m+1:m+n,1:m) A(m+1:m+n,m
    +1:m+n)]; U = A - DL;
disp(sprintf('x = '));
for k = 1:MaxIt
  r = b - A*x;
  if (norm(r)<tol*norm(b)), break; end
  x = x + DL\r;
  err(k) = norm(x-xstar);
end
disp(sprintf('Number of iterations: %d ', k));
end
```