**PRECONDICIONAMIENTO: UTILIDAD EN LA GENERACIÓN DE MALLAS CON PROCESOS ITERATIVOS**

**Christian Córdova, Wilbert Pumacay, José Castro, Joel Gallegos**

**Filiación 1,** [**christian.cordova@ucsp.edu.pe**](mailto:christian.cordova@ucsp.edu.pe)**, Filiación 2,** [**wilbert.pumacay@ucsp.edu.pe**](mailto:wilbert.pumacay@ucsp.edu.pe)**, Filiación 3,** [**jose.castro@ucsp.edu.pe**](mailto:jose.castro@ucsp.edu.pe)**, Filiación 4,** [**joel.gallegos@ucsp.edu.pe**](mailto:joel.gallegos@ucsp.edu.pe)

**RESUMEN**

El interés fundamental de este trabajo se centra en como acelerar métodos iterativos para la solución numérica de un sistema de ecuaciones mediante precondicionamiento. 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. Una de sus aplicaciones es la solución numérica de ecuaciones diferenciales parciales, las cuales generalmente generan sistemas sparse.

Otro punto a tratar en este trabajo es la importancia de los métodos iterativos, que realizan mucho menos pasos con respecto a los métodos directos. Una aplicación directa es la generación de mallas. Como caso de prueba se mostrará la generación de mallas en 2D y se verá la importancia del uso de precondicionamiento respecto al número de iteraciones total usado para generar la malla.

PALABRAS CLAVES: precondicionamiento, malla, métodos iterativos, **key 4, key 5.**

**PRECONDITIONING: USEFULNESS IN THE MESH GENERATION WITH ITERATIVE PROCESSES**

**ABSTRACT**

The fundamental interest of this work focuses on how to accelerate iterative methods for the numerical solution of a system of linear equations via preconditioning. Preconditioning a linear system is one of the main sources for obtaining more efficient computational results. Thus, in recent years, more research has been developed with respect to direct solution methods. One of its applications is the numerical solution of partial differential equations, which usually generate sparse systems.

Another point to be addressed in this paper is the importance of iterative methods, that perform much less steps with respect to direct methods. A direct application is the generation of meshes. As a test case we will show the generation of meshes in 2D and then see the importance of preconditioning respect to the total number of iterations used to generate the mesh.

**KEY WORDS:** Preconditioning, mesh, iterative methods, **key word 4, key word 5**

1. **INTRODUCCIÓN**

**…Type here something…**

1. **PRECONDICIONAMIENTO**

El término precondicionamiento se refiere a transformar un sistema original de ecuaciones lineales en otro con propiedades más favorables en cuanto a la solución iterativa y que siga manteniendo la solución. Un precondicionador es una matriz *M* que efectúa tal transformación.Sea *A* la matriz de orden del siguiente sistema

El número condicionante *cond(A)* se define como:

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

Esta desigualdad es fundamental para garantizar al favorecimiento de los métodos iterativos, siempre que sea convergente. El mejor precondicionador para el sistema (1) sería pues como

Sería el óptimo y así el sistema convergería en tan solo una iteración, pero el problema es calcular el coste computacional de esto equivaldría a calcular por algún método directo. Es por ello que se debe buscar un sin coste elevado.

Para garantizar la convergencia de la matriz A, es suficiente mostrar que su radio espectral sea menor a la unidad, matemáticamente .Si en la ecuación (1) hacemos donde es una matriz no

singular, entonces

Donde es la matriz identidad de orden . En este caso para que el sistema lineal converja, se debe cumplir que

Además, mientras el radio espectral sea más cerca de cero entonces la velocidad de convergencia será mayor.

Los precondicionadores deben cumplir principalmente dos propiedades [1]:

* 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. **MÉTODOS ITERATIVOS**

La importancia de los métodos iterativos en álgebra lineal se deriva de un simple hecho: los métodos directos requieren de , mientras que los métodos iterativos pueden llegar a requerir solamente , Es así que para matrices con se va volviendo intratable el no pensar resolverlo con un algoritmo iterativo. Aunque los métodos iterativos requieren menos almacenamiento no tienen la habilidad 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 prácticamente basados en alguna actualización de la eliminación gaussiana, mientras que los métodos iterativos comprenden 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 \_nito 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.

1. **GENERACIÓN DE MALLAS**

La idea principal en la generación de mallas, está dada en la idea de mallar un espacio dado algunos límites de un dominio, por ejemplo, el ala de un avión. Existen métodos de generación de mallas estructuradas y no estructuradas. Para nuestro fin se realizó los métodos de generación de mallas estructuradas, dentro del cual se analizó los métodos Algebraicos y los Basados en EDPs. Es necesario mencionar que las mallas estructuradas tienen un orden al momento de ser generadas, lo cual nos permite hacer un precondicionamiento para acelerar el proceso de generación de la malla.

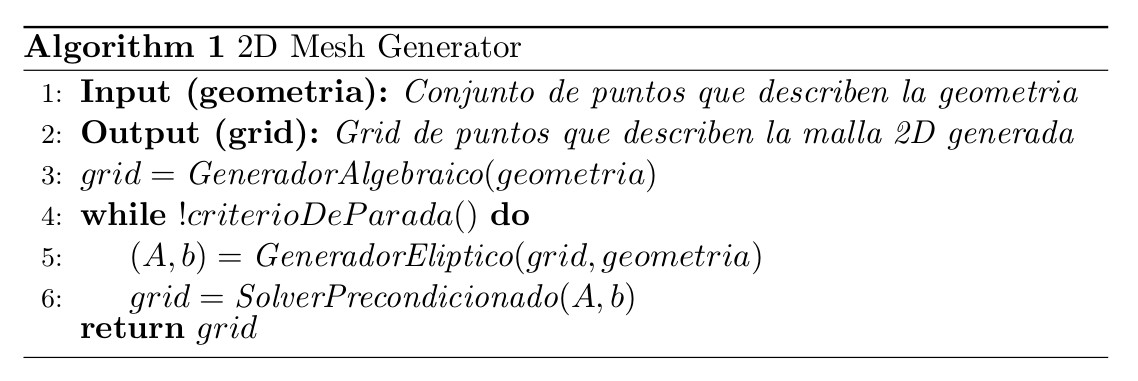
Cuando se quiere representar una geometría que no se puede representar en el sistema de coordenadas ortogonales, es necesario hacer una trasformación que permita mapear la región física a una región computacional.

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

|  |  |
| --- | --- |
|  |  |
| a) | b) |
| *Fig. 1: a) Plano Físico b) Plano Computacional* | |

**Generación de mallas estructuradas**

El problema a tratar fue el de la generación de mallas estructuradas, el cuál puede ser resuelto usando diferentes técnicas de generación. Las usadas en nuestro caso fueron los **Generadores Algebraicos basados en Interpolación Transfinita** y los **Generadores basados en EDPs Elípticas**.

****En la siguiente figura se puede ver el procedimiento seguido para la generación final de una malla 2D.

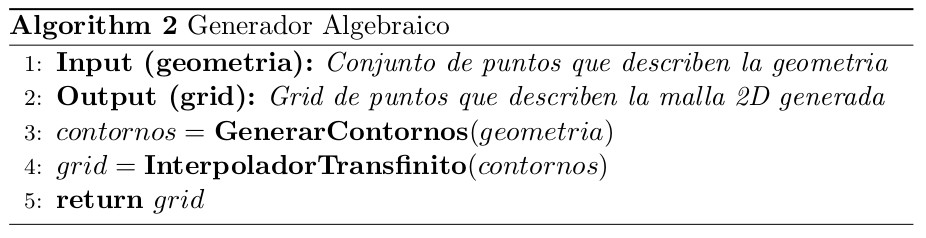
*Algoritmo para la generación de mallas 2D.*

**Generador Algebraico basado en Interpolación Transfinita**

El generador algebraico usado fue el Interpolación Transfinita, el cual genera la malla a partir de funciones interpoladas en base a la geometría dada como base. Esta geometría se usa como contorno, siendo *xl, xr, xb* y *xt* los contornos definidos por la geometría. El grid resultante es una interpolación de estos contornos en el espacio de computo , como se muestra en la siguiente ecuación.

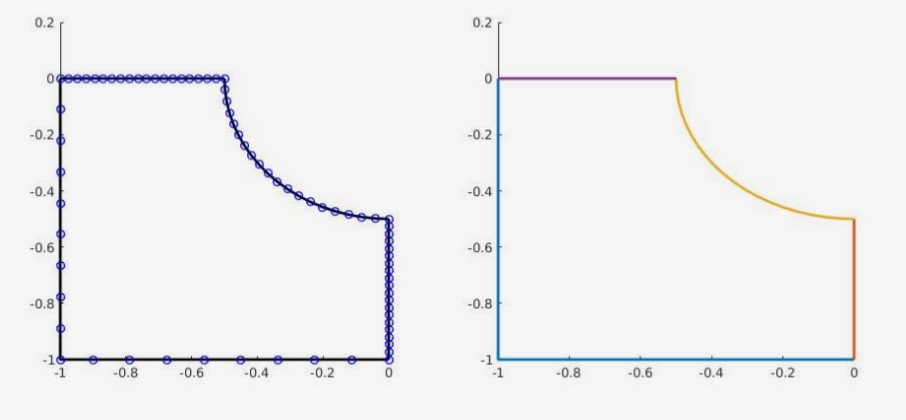
La geometría de entrada es dada en forma de puntos, lo cual es un pequeño inconveniente al querer usar este generador, debido a que se espera que los contornos sean expresados en forma explícita como una función de las coordenadas de cómputo . Para esto, generamos los contornos como funciones lineales por trozos del tipo siguiente.

El generador algebraico puede ser resumido en el siguiente pseudocódigo.

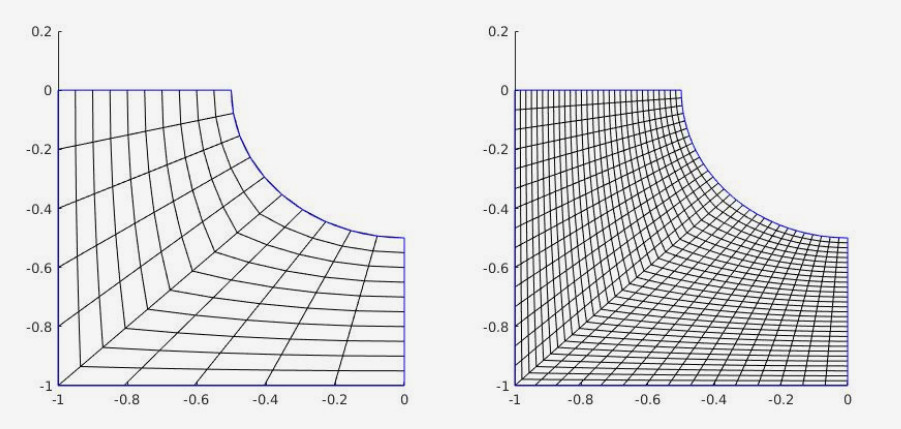
****

*Algoritmo del generador algebraico*

Finalmente, en las siguientes figuras se muestra la geometría de ejemplo a usar, y la malla resultante del generador algebraico.

****

*Fig. 2. Geometría usada de ejemplo*

****

*Fig. 3. Mallas resultantes del generador algebraico para tamaños de 10x10 y 30x30 respectivamente*

**Generador basado en Ecuaciones Diferenciales Parciales Elípticas**

El siguiente paso en el algoritmo es el generador basado en EDPs. El generador es basado en una EDP elíptica, el cual relaciona la malla en el espacio físico al espacio computacional usando la siguiente relación.

Transformando estas ecuaciones al espacio físico respecto al computacional, tenemos las siguientes ecuaciones resultantes.

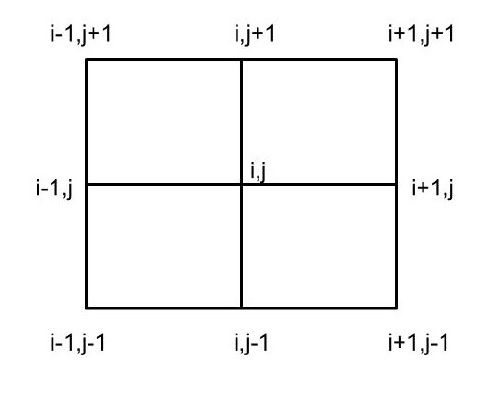
Luego de discretizar estas ecuaciones en el espacio computacional tendremos las siguientes ecuaciones :

Donde se discretizó el espacio computacional con la siguiente relación:

Lo cual genera una grilla de puntos de tamaño , siendo los coeficientes definidos en función de de la siguiente forma:

Para resolver estas ecuaciones generamos un sistema lineal usando la iteración de Picard, haciendo que los coeficientes no lineales sean constantes para una determinada iteración *k*. Las ecuaciones resultantes son las siguientes:

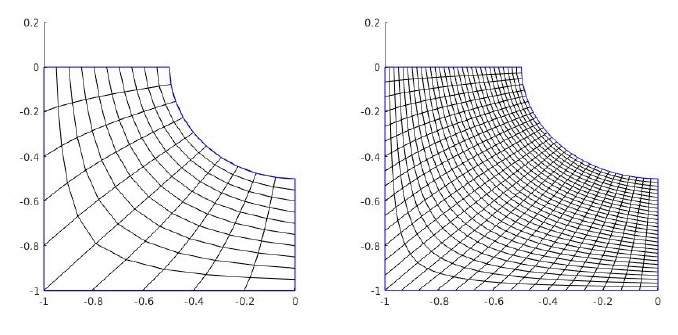
Con lo cual podemos generar el sistema lineal , siendo generados de las ecuaciones anteriores al aplicar un stencil del siguiente tipo:



*Fig. 4. Stencil a aplicar para generar el sistema lineal en el generador elíptico*

Se puede observar de la iteración de Picard que el grid es refinado en cada nueva iteración. Para esto, el grid inicial es el grid obtenido por el generador algebraico, como se resaltó en el algoritmo inicial.

Finalmente, como resultado del generador elíptico se obtuvieron las siguientes mallas refinados tomando como mallas iniciales las obtenidas del generador algebraico.



*Fig. 5. Mallas resultantes del generador elíptico*

1. **CONCLUSIONES**
2. **REFERENCIAS**
3. BENZI, Michele. “Preconditioning Techniques for Large Linear Systems: A Survey”. Mathematics and Computer Science Department, 2002.
4. SAAD, Yousef. Iterative Methods for Sparse Linear Systems. Second edition, 2000. 447pp
5. THOMPSON, Joe; SONI, Bharat; WEATHERILL, Nigel. Grid Generation. 1999. 1096pp
6. TEJA, Víctor. “Generación de Mallas Numéricas para Geometrías Irregulares y Complejas”, 2011