Informe Final SCN

Jhorman Gomez[1] Nicolas Salazar[2]

2326867[1] 2328060[2]

Universidad del Valle

Facultad de Ingeniería

Escuela de Ingeniería de Sistemas y Computación

Santiago de Cali, Mayo de 2025

Abstract

Este trabajo presenta una simulación numérica de la vorticidad de un fluido incompresible mediante la discretización de las ecuaciones de Navier-Stokes. El problema se modeló sobre una malla rectangular, donde la velocidad en cada punto se calcula como un promedio de las velocidades en las celdas adyacentes, ajustado por un término no lineal. Para resolver el sistema no lineal asociado, se empleó el método de Newton-Raphson, mientras que para el cálculo del vector de corrección de Newton-Raphson se exploraron métodos iterativos como Richardson, Jacobi, Gauss-Seidel, y métodos de Krylov como Gradiente Conjugado y Gradiente Descendiente.

Introducción

El presente trabajo se enmarca en el campo de la dinámica de fluidos, específicamente en la simulación numérica de la vorticidad de un fluido incompresible a través de la discretización de las ecuaciones de Navier-Stokes. El objetivo principal de este estudio es modelar numéricamente el comportamiento de la vorticidad en un medio continuo, explorando y aplicando distintos métodos numéricos para su resolución.

El objetivo específico planteado es comparar distintos métodos númericos para la solución de sistemas lineales, con el propósito de identificar cuáles son aplicables a nuestro problema y determinar aquel que presenta un mejor desempeño en términos del número de iteraciones requeridas para alcanzar la convergencia. Los métodos evaluados incluyen Richardson, Jacobi, Gauss-Seidel, Gradiente Descendiente y Gradiente Conjugado.

A lo largo del desarrollo del proyecto, se concluyó que el método de Gauss-Seidel presenta un mejor comportamiento para este problema en particular, logrando una convergencia más rápida con un menor número de iteraciones. Adicionalmente, se identificó que los métodos de Krylov no pueden aplicarse en este contexto, debido a que la matriz generada en el sistema lineal asociado al método de Newton-Raphson no es simétrica, siendo este un requisito para su implementación.

Contenido

Primer Avance

Durante el primer avance se tuvo como objetivvo definir parámetros como la dimensión, el step h, las ecuaciones y condiciones de frontera de la malla.

Para los parámetros iniciales decidimos que usariamos los siguientes valores:

$$Nx = 60,\ Ny = 20,\ h = 1$$

N_x y N_y representando el número de columnas y filas respectivamente, generando así un sistema de 1200 ecuaciones, una cantidad que creemos manejable y por ende el por qué de estas dimensiones.

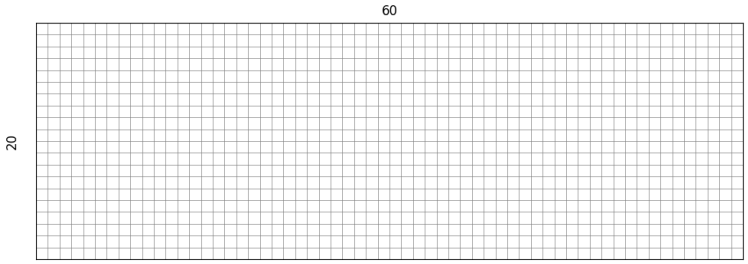

Figura 1: Primer Malla

Posteriormente definimos nuestras condiciones de frontera sobre la malla:

Borde superior

$$V_x = 0,\ v_y = 0 \quad i = 0, \quad 0 \leq j <= 59$$

Borde inferior

$$v_x = 0,\ v_y = 0 \quad i = 19, \quad 0 \leq j <= 59$$

Borde izquierdo

$$v_x = 1,\ v_y = 0 \quad j = 0, \quad 0 < i < 19$$

Borde derecho

$$v_x = 1,\ v_y = 0 \quad j = 59, \quad 0 < i < 19$$

Por último, partiendo de la siguiente ecuación:

$$r = \frac{1}{4}[v^x_{(i+1,j)} + v^x_{(i-1,j)} + v^x_{(i,j+1)} + v^x_{(i,j-1)}] + N - v^x_{(i,j)}$$

consideraremos que este residuo r es 0 obteniendo entonces:

$$v^x_{(i,j)} = \frac{1}{4}[v^x_{(i+1,j)} + v^x_{(i-1,j)} + v^x_{(i,j+1)} + v^x_{(i,j-1)}] + N$$

$$N = -\frac{h}{8} v^x_{(i,j)} (v^x_{(i+1,j)} - v^x_{(i-1,j)}) - \frac{h}{8} v^y_{(i,j)} (v^y_{(i,j+1)} - v^y_{(i,j-1)})$$

consideraremos entonces que la velocidad en una celda cualquiera será dada por un promedio de las velocidades de las celdas adyacentes más un factor no lineal.

Teniendo en cuenta las condiciones de frontera mencionadas, los parámetros iniciales definidos y la ecuación para la velocidad de cada celda, la malla estará conformada por bordes con valores constantes y celdas internas que, dependiendo de su posición en la malla interior (centro, borde o esquina), tendrán una ecuación diferente:

Borde superior interno

$$v^x_{(1,j)} = \frac{1}{4}[v^x_{(2,j)} + v^x_{(1,j+1)} + v^x_{(1,j-1)}] - \frac{1}{8} v^x_{(1,j)} \cdot v^x_{(2,j)} - \frac{1}{8} v^y_{(1,j)} (v^y_{(1,j+1)} - v^y_{(1,j-1)})$$

$$1 < j < 58$$

Borde inferior interno

$$v^x_{(18,j)} = \frac{1}{4}[v^x_{(17,j)} + v^x_{(18,j+1)} + v^x_{(18,j-1)}] - \frac{1}{8} v^x_{(18,j)} (-v^x_{(17,j)}) - \frac{1}{8} v^y_{(18,j)} (v^y_{(18,j+1)} - v^y_{(18,j-1)})$$

$$1 < j < 58$$

Borde izquierdo interno

$$v^x_{(i,1)} = \frac{1}{4}[v^x_{(i+1,1)} + v^x_{(i-1,1)} + v^x_{(i,2)} + 1] - \frac{1}{8} v^x_{(i,1)} (v^x_{(i+1,j)} - v^x_{(i-1,j)}) - \frac{1}{8} v^y_{(i,1)} (v^y_{(i,2)} - v^y_{(i,0)})$$

$$1 < i < 18$$

Borde derecho interno

$$v^x_{(i,1)} = \frac{1}{4}[v^x_{(i+1,1)} + v^x_{(i-1,1)} + v^x_{(i,2)} + 1] - \frac{1}{8} v^x_{(i,1)} (v^x_{(i+1,j)} - v^x_{(i-1,j)}) - \frac{1}{8} v^y_{(i,1)} (v^y_{(i,2)} - v^y_{(i,0)})$$

$$1 < i < 18$$

Esquina superior izquierda
Esquina superior derecha
Esquina inferior izquierda
Esquina inferior derecha
Centro

$$v^x_{(i,j)} = \frac{1}{4}[v^x_{(i+1,j)} + v^x_{(i-1,j)} + v^x_{(i,j+1)} + v^x_{(i,j-1)}] + N$$

$$N = -\frac{h}{8} v^x_{(i,j)} (v^x_{(i+1,j)} - v^x_{(i-1,j)}) - \frac{h}{8} v^y_{(i,j)} (v^y_{(i,j+1)} - v^y_{(i,j-1)})$$

$$2 < i < 18,\ 2 < j < 58$$

Segundo Avance

Para el segundo avance, el objetivo fue aplicar el método de **Newton-Raphson** para solucionar el sistema de ecuaciones asociado al problema. Dicho sistema está conformado por ecuaciones de la forma:

$$F_{i,j} = v^x_{(i,j)} - \frac{1}{4}\left[v^x_{(i+1,j)} + v^x_{(i-1,j)} + v^x_{(i,j+1)} + v^x_{(i,j-1)}\right] - N$$

donde N representa el término no lineal que depende de las diferencias de velocidades adyacentes y se define como:

$$N = -\frac{h}{8}v^x_{(i,j)}\left(v^x_{(i+1,j)} - v^x_{(i-1,j)}\right) - \frac{h}{8}v^y_{(i,j)}\left(v^y_{(i,j+1)} - v^y_{(i,j-1)}\right)$$

El sistema de ecuaciones generado tiene un total de $(N_x - 2)\cdot(N_y - 2)$ ecuaciones, es decir, 1044 ecuaciones en nuestro caso, considerando las condiciones de frontera.

El método de **Newton-Raphson** para sistemas de ecuaciones de varias variables se expresa mediante la siguiente ecuación iterativa:

$$\vec{x}^{(k+1)} = \vec{x}^{(k)} + H$$

donde:

- $\vec{x}$ es el vector de incógnitas que representa las velocidades en cada celda interna de la malla.
- H es el incremento calculado en cada iteración para aproximarnos a la solución.

El incremento H se obtiene resolviendo el sistema lineal siguiente:

$$J(\vec{x}^{(k)})H = -F(\vec{x}^{(k)})$$

donde:

- $J(\vec{x})$ es la matriz jacobiana del sistema, que contiene las derivadas parciales de cada ecuación $F_{i,j}$ respecto a cada incógnita:

$$J(\vec{x}) = \begin{pmatrix} \frac{\partial F_1}{\partial x_1} & \frac{\partial F_1}{\partial x_2} & \cdots & \frac{\partial F_1}{\partial x_n} \\ \frac{\partial F_2}{\partial x_1} & \frac{\partial F_2}{\partial x_2} & \cdots & \frac{\partial F_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial F_n}{\partial x_1} & \frac{\partial F_n}{\partial x_2} & \cdots & \frac{\partial F_n}{\partial x_n} \end{pmatrix}$$

- $F(\vec{x})$ es la matriz de ecuaciones evaluada en el punto actual.

Teniendo en cuenta la teoría desarrollada previamente, procedemos al cálculo del **Jacobiano** de nuestra malla. Debido a la estructura de las ecuaciones, para cada una de ellas únicamente existen cinco derivadas parciales no nulas. Esto implica que el Jacobiano adopta una forma similar a una matriz diagonal acompañada por cuatro sub-diagonales.

Los cinco tipos de derivadas parciales no nulas corresponden a las contribuciones de las celdas adyacentes en las siguientes direcciones:

- Derivada con respecto a $v^x_{(i+1,j)}$ — Celda inferior.

$$-\frac{1}{4} + \frac{h}{2} v^x_{(i,j)}$$

- Derivada con respecto a $v^x_{(i-1,j)}$ — Celda superior.

$$-\frac{1}{4} - \frac{h}{2} v^x_{(i,j)}$$

- Derivada con respecto a $v^x_{(i,j+1)}$ — Celda a la izquierda.

$$-\frac{1}{4} - \frac{h}{2} v^y_{(i,j)}$$

- Derivada con respecto a $v^x_{(i,j-1)}$ — Celda a la derecha.

$$-\frac{1}{4} + \frac{h}{2} v^y_{(i,j)}$$

- Derivada con respecto a $v^x_{(i,j)}$ — Celda actual (término diagonal).

$$1 + \frac{h}{8}(v^x_{(i+1,j)} - v^x_{(i-1,j)})$$

obteniendo así:

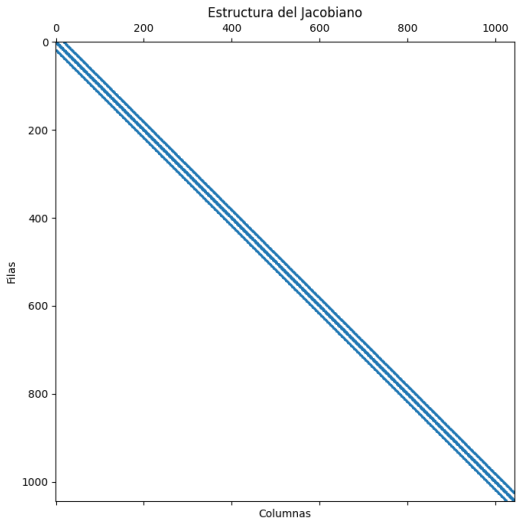

Figura 2: Jacobiano

Una vez definido el Jacobiano y su estructura, el siguiente paso es implementar el método de Newton-Raphson para resolver el sistema de ecuaciones no lineales. Para ello, utilizaremos Python y la función **dgesv** de la librería **scipy**, la cual emplea factorización LU para resolver el sistema.

Es importante mencionar que el valor inicial seleccionado para el paso de malla, $h = 1$, en el primer avance, fue modificado a $h = 0.001$. Este cambio se realizó tras observar que las simulaciones con el valor anterior no arrojaban resultados consistentes con el comportamiento esperado. Además, el nuevo valor permite una convergencia más rápida del sistema. Teniendo en cuenta lo anterior, esta fue la solución obtenida:

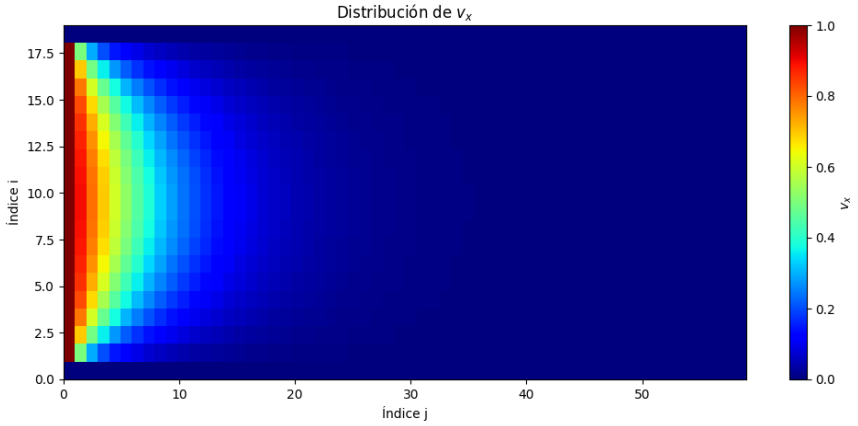

Figura 3: Primer Simulación

Tercer Avance

En el tercer avance se tiene como objetivo la implementación de los métodos iterativos **Richardson**, **Jacobi** y **Gauss-Seidel**, y los métodos de Krylov **Grad. Descendiente** y **Grad. Conjugado**, para el cálculo del vector de correción H.

Para los métodos iterativos mencionados, la ecuación de iteración es:

$$\vec{x}^{(k)} = (I - Q^{-1}A)\vec{x}^{(k-1)} + Q^{-1}b$$

donde:

- A en nuestro caso es el Jacobiano y b es el vector de términos independientes obtenido de la matriz de ecuaciones.
- Q varía dependiendo del método:
  - Para Richardson, Q es la matriz identidad.
  - Para Jacobi, Q es la matriz diagonal de A.
  - Para Gauss-Seidel, Q es la matriz triangular inferior de A.

estos métodos convergen bajo los siguientes supuestos:

- Según [1], para los métodos de Gauss-Seidel y Jacobi, basta con que la matriz A sea diagonal dominante.
- Según [1], para el método de Richardson, es necesario comprobar que $\|I - Q^{-1}A\| \leq 1$ para alguna norma dada.

recordemos que una matriz es diagonal dominante si:

$$|a_{ii}| \geq \sum_{j \neq i} |a_{ij}|$$

procedemos entonces a verificar estos supuestos. Al ejecutar la siguiente función en python sobre nuestra matriz Jacobiana:

```python
def isDiagonallyDominant(J):
    n = J.shape[0]

    for i in range(n):
        sumaFila = np.sum(np.abs(J[i, :])) - np.abs(J[i, i])
        if np.abs(J[i, i]) < sumaFila:
            print(f"Suma de los valores de la fila: {sumaFila}")
            print(f"Valor comparado: {J[i, i]}")
            return False

    return True
```

Figura 4: Función `isDiagonallyDominant`

obtenemos que existen valores en la diagonal que son menores que la suma de los elementos restantes de la fila. Sin embargo, la diferencia es mínima, lo que puede atribuirse a pérdidas de precisión debido al redondeo de números. En consecuencia, consideraremos que la matriz es diagonalmente dominante, y por lo tanto, podemos aplicar los métodos de **Jacobi** y **Gauss-Seidel**.

En cuanto a **Richardson**, al realizar la prueba de convergencia obtenemos el siguiente valor para la norma infinito:

$$\|I - Q^{-1}A\|_\infty = 1.0000415344381142$$

este valor es ligeramente diferente de 1, pero la diferencia es pequeña, lo que nuevamente atribuimos a la pérdida de precisión por redondeo. Por lo tanto, consideraremos que se cumple el supuesto y, en consecuencia, podemos aplicar el método de Richardson.

Una vez cubiertos los métodos iterativos y sus respectivos supuestos, pasamos al análisis de los métodos de Krylov, en particular, el **Gradiente Descendiente** y el **Gradiente Conjugado**. Para aplicar estos métodos es necesario garantizar que la matriz A cumple los siguientes requisitos:

- Es **simétrica**.
- Es **positiva definida**.

para verificar el primer supuesto, implementamos la siguiente función en Python:

```python
def isSymmetric(A):
    return np.allclose(A, A.T)
```

Figura 5: Función `isSymmetric`

al ejecutar esta función sobre nuestra matriz A, se obtiene que no es simétrica. Debido a esto, no se cumple el primer supuesto requerido para aplicar los métodos de Gradiente Descendiente y Gradiente Conjugado y por ende no podemos aplicarlos a nuestro problema.

Una vez cubiertos los métodos y sus supuestos, procederemos a compararlos bajo el número de iteraciones que necesitaron para alcanzar una tolerancia de 1×10^{-6}. A continuación los parámetros y soluciones:

$$N_x = 60,\ N_y = 20,\ h = 0.001$$

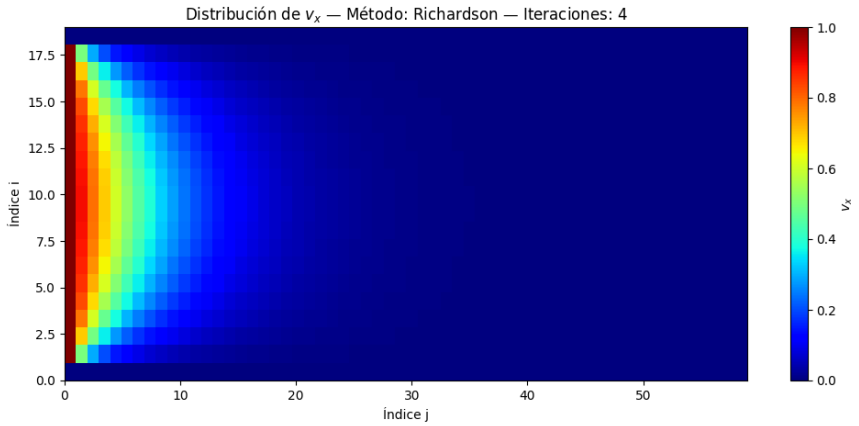

Figura 6: Solución mediante Richardson

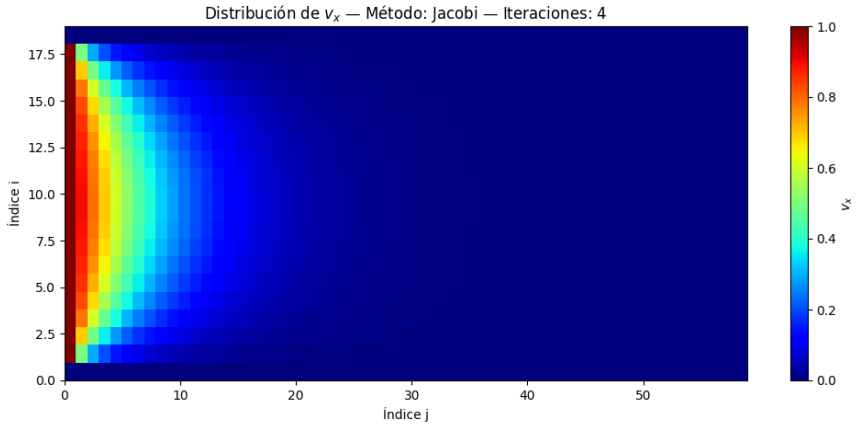

Figura 7: Solución mediante Jacobi

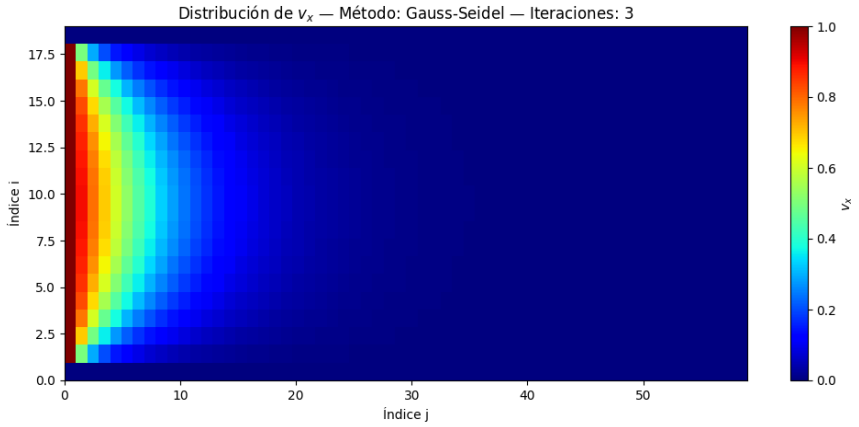

Figura 8: Solución mediante Gauss Seidel

Metodología

Conclusiones

Referencias

[1] David Kincaid and Ward Cheney. *Numerical Analysis: Mathematics of Scientific Computing*. The Sally Series; Pure and Applied Undergraduate

Texts, Vol. 2, Pacific Grove, CA, 3rd ed. edition, 2009. Theorems 1, 2, 6, p. 210-216.