Simulación de un fluido incompresible utilizando ecuaciones de Navier-Stokes

Edinson Orlando Dorado Dorado 1941966-3743, edinson.dorado@correounivalle.edu.co

23 de noviembre de 2025

Resumen

Este trabajo presenta la simulación numérica de un fluido incompresible en un canal bidimensional que contiene dos vigas fijas como obstáculos, utilizando las ecuaciones de Navier–Stokes discretizadas mediante diferencias finitas. El dominio físico original de 40×400 celdas se reduce a una malla computacional de 8×80 mediante un factor de escala $h = 5$, sobre la cual se formulan las ecuaciones de balance para la componente de velocidad v_x. Para resolver el sistema no lineal se emplea el método de Newton–Raphson, mientras que los sistemas lineales asociados se solucionan mediante métodos iterativos como SOR, Gauss–Seidel, Jacobi, Richardson, gradiente descendente y gradiente conjugado. Los resultados muestran que Newton–Raphson converge en pocas iteraciones y que SOR es el método lineal más eficiente, mientras que el gradiente conjugado no converge debido a que el Jacobiano no es simétrico definido positivo. Finalmente, se aplica una interpolación bidimensional mediante splines cúbicos naturales para prolongar la solución desde la malla reducida a la malla fina de 40×400, obteniendo un campo de velocidades suave que preserva las condiciones de frontera y la geometría del dominio físico.

1. Introducción

En este documento se detalla la solución de un problema de simulación de un fluido incompresible usando el lenguaje de programación Python. El problema consiste en simular un fluido incompresible que se deja fluir a través de una malla, la cual tiene 2 vigas que actúan como obstáculos. La solución que se documenta aquí hace uso de las ecuaciones de Navier-Stokes usando una discretización hecha con el método de diferencias finitas.

2. Materiales y Métodos

La situación que se quiere simular, es un fluido que recorre una malla que contiene 2 vigas fijas. El fluido entra a una velocidad por el extremo izquierdo, y se quiere entender cómo cambia la velocidad del fluido hasta llegar al extremo derecho.

2.1. Entendiendo el problema original

Para entender mejor el problema que se va a tratar, se va a explicar primero el problema original del cual se deriva. En la Figura 1 se pueden observar las condiciones de frontera y sus respectivos valores acordados para el problema original: el borde superior (Surface G), el borde inferior antes de la viga (E), el borde inferior después de la viga (A), la entrada del fluido (InletF), la salida del fluido (Outlet H), y por último los bordes de la viga (superior C, izquierdo D, derecho B). El problema original se puede encontrar en libro (Kincaid & Cheney, 2009). De este libro se obtienen las Figuras 1, 2 y 3.

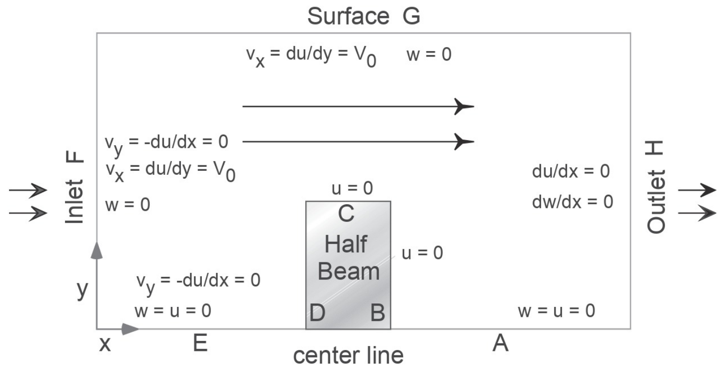

Figure 4.11. Boundary conditions for flow around a submerged beam. The flow is symmetric about the centerline, and the beam has length L in the x direction (along flow).

Figura 1: Condiciones de frontera en el problema original

De la Figura 1 se puede entender:

- **Surface G (techo,** $y = 0$**):** $\dfrac{\partial u}{\partial y} = V_0$ (Neumann con gradiente fijo).
- **InletF** ($x = 0$)**:** $\dfrac{\partial u}{\partial x} = 0$ (perfil "plano" a la entrada).
- **Outlet H** ($x = N_x - 1$)**:** $\dfrac{\partial u}{\partial x} = 0$ (salida convectiva).
- **Fondo E–A** ($y = N_y - 1$)**:** $u = 0$ (no deslizamiento).
- **Vigas/obstáculos:** $u = 0$ (no deslizamiento).

En la Figura 2 se muestran los valores exactos para las condiciones de frontera del problema original, a la izquierda para la velocidad, y en el centro para la vorticidad.

4.6.2 Flow over Submerged Beam

Boundary conditions for hydrodynamic flow can be challenging. For a submerged beam (Figure 4.11), those used in our simulation are:

$$
\begin{aligned}
u &= 0; & w &= 0 & &\text{Centerline EA} \\
u &= 0, & w_{i,j} &= -2(u_{i+1,j} - u_{i,j})/h^2 & &\text{Beam back B} \\
u &= 0, & w_{i,j} &= -2(u_{i,j+1} - u_{i,j})/h^2 & &\text{Beam top C} \\
u &= 0, & w_{i,j} &= -2(u_{i-1,j} - u_{i,j})/h^2 & &\text{Beam front D} \qquad (4.65) \\
\partial u/\partial x &= 0, & w &= 0 & &\text{Inlet F} \\
\partial u/\partial y &= V_0, & w &= 0 & &\text{Surface G} \\
\partial u/\partial x &= 0, & \partial w/\partial x &= 0 \quad \text{Outlet H} & &
\end{aligned}
$$

Figura 2: Condiciones de frontera en el problema original

Para que quede aún más claro el problema que se intenta solucionar en este documento, se muestra la Figura 3, en ella se detallan los valores de las condiciones de frontera del problema original, y en la parte central se muestra cómo el fluido se comporta. Se puede ver cómo la viga influye en el movimiento del fluido al hacer que este tenga que evadir ese obstáculo. También se puede observar cómo la velocidad del fluido cerca a la viga y cerca a los bordes es más baja en comparación con partes en donde el fluido está rodeado de más fluido.

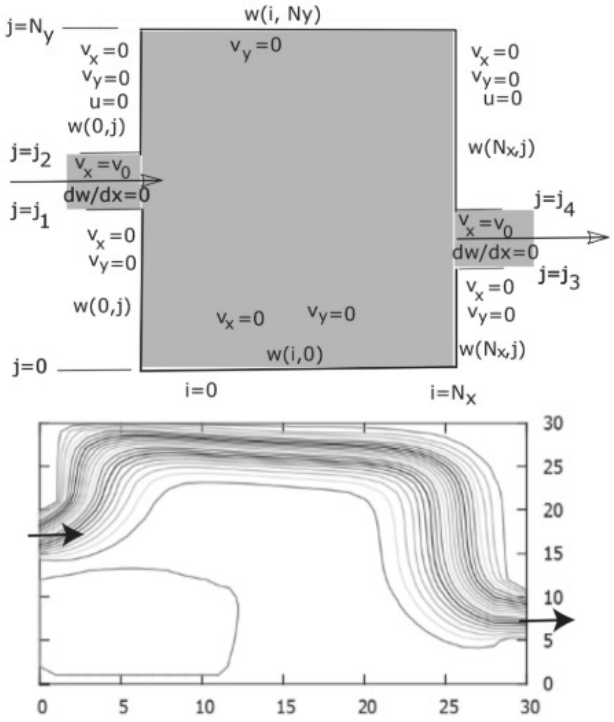

Figure 4.14. *Left:* A tank filled with fluid entering on the left and leaving on the right. Various boundary conditions are indicated. The cavity has inflow at the left between J_1 and J_2 and outflow at the right surface between J_3 and J_4. *Right:* The contour lines of constant u for the inflow and outflow tank.

The fluid tends to stick to the side of the tank, and so:

$$v_y = -\partial u/\partial x = 0, \qquad w(0,y) = -2[u(0,y) - u(h,y)]/h^2. \tag{4.77}$$

Except at hole, the fluid cannot move down along the bottom of the tank ($y = b$), and

Figura 3: Comportamiento del fluido en el problema original

Una vez entendido el problema original, se procede a entender el problema propio de este documento.

2.2. Malla del problema a solucionar

Se tiene una malla de dimensiones 40×400 unidades. Debido a la complejidad computacional que estas dimensiones significarían, se decidió dividir estas dimensiones entre 5 ($h = 5$), por lo que se reduce el problema a tener una malla de 8×80.

En cuanto a las vigas, se decidió tener 2 vigas de dimensiones 20×50, pero aplicando $h = 5$ quedan de 4×10.

La posición de las vigas, teniendo en cuenta que la primera celda en la esquina superior izquierda es la celda $(0, 0)$ o $C_{0,0}$, son: la primera viga se encuentra entre los puntos ($C_{7,34}$, $C_{4,34}$, $C_{4,43}$, $C_{7,43}$) y la segunda viga se encuentra entre los puntos ($C_{0,70}$, $C_{3,70}$, $C_{0,79}$, $C_{3,79}$).

En la Figura 4 se presenta la malla junto con sus dos vigas. Las zonas coloreadas corresponden a grupos de celdas que comparten el mismo número y disposición de vecinos, con valores variables. La malla completa y otros detalles adicionales pueden consultarse en el siguiente enlace: https://docs.google.com/spreadsheets/d/1oWH20P9R6IXGds46f7QT3c3LLVYNJoU4q6nNZu-FzOI/edit?gid=0#gid=0.

Figura 4: Malla del problema a solucionar

2.3. Condiciones de frontera para el problema a solucionar

En el cuadro 1 se muestran las condiciones de frontera acordadas para el problema a solucionar en este documento. En este cuadro se puede observar cómo InletF es la única frontera con velocidad igual a 1, y cómo el resto de fronteras son iguales a 0. Que la frontera InletF sea la única frontera con velocidad igual a 1 quiere decir que por ese lado es donde entra el fluido.

Cuadro 1: Condiciones de frontera del problema actual

Condición	**Borde / Superficie**
$u = 0$	Centerline EA
$u = 0$	Surface E
$u = 0$	Surface A
$u = 0$	Beam back B
$u = 0$	Beam top C
$u = 0$	Beam front D
$u = 0$	Centerline JK
$u = 0$	Beam front J
$u = 0$	Beam front I
$u = 0$	Beam front K
$u = 1$	InletF
$u = 0$	Surface G
$u = 0$	Outlet H

Aclaración: no se considerará la vorticidad; por tanto, no se incluyen los términos asociados a w.

2.4. Ecuaciones de Navier-Stokes

Al tener un problema de fluidos, las ecuaciones que se deben entender para solucionarlo son las ecuaciones de Navier-Stokes.

2.5. Discretización del problema

Para discretizar la ecuación de Navier-Stokes, se usan las siguientes 2 ecuaciones que son las equivalencias para discretizar primeras y segundas derivadas.

La siguiente ecuación se usa para discretizar primeras derivadas.

$$\frac{\partial F}{\partial x} \approx \frac{F(x + \Delta x, y) - F(x - \Delta x, y)}{2\Delta x} \tag{1}$$

La siguiente ecuación se usa para discretizar segundas derivadas.

$$\frac{\partial^2 F}{\partial x^2} \approx \frac{F(x + \Delta x, y) - 2F(x, y) + F(x - \Delta x, y)}{\Delta x^2} \tag{2}$$

Al reemplazar las dos equivalencias anteriores en la ecuación de Navier-Stokes se obtiene la siguiente función, en donde.

$v^x_{i,j}$ es una variable, mientras que $v^y_{i,j}$ se considera constante.

$$v^x_{i,j} = \frac{1}{4}\left\{ v^x_{i+1,j} + v^x_{i-1,j} + v^x_{i,j+1} + v^x_{i,j-1} - \frac{h}{2} v^x_{i,j}\left(v^x_{i+1,j} - v^x_{i-1,j}\right) - \frac{h}{2} v^y_{i,j}\left(v^x_{i,j+1} - v^x_{i,j-1}\right) \right\} \tag{2}$$

2.5.1. 9 ecuaciones para 9 zonas en la malla

La malla acordada se puede dividir en 9 zonas, cada una definida por el número de vecinos en donde el fluido cambia de velocidad, y definida por la posición de estos vecinos.

1. **Todos los vecinos son celdas en donde el fluido varía**
   La ecuación discretizada queda intacta.

$$v_{i,j}^x = \frac{1}{4}\left\{ v_{i+1,j}^x + v_{i-1,j}^x + v_{i,j+1}^x + v_{i,j-1}^x - \frac{h}{2}v_{i,j}^x\left(v_{i+1,j}^x - v_{i-1,j}^x\right) - \frac{h}{2}v_{i,j}^y\left(v_{i,j+1}^x - v_{i,j-1}^x\right)\right\} \quad (3)$$

2. **El vecino de arriba es una viga o una frontera**
   El término que representa la celda de arriba se reemplaza por cero.

$$v_{i,j}^x = \frac{1}{4}\left\{ v_{i+1,j}^x + v_{i,j+1}^x + v_{i,j-1}^x - \frac{h}{2}v_{i,j}^x\left(v_{i+1,j}^x\right) - \frac{h}{2}v_{i,j}^y\left(v_{i,j+1}^x - v_{i,j-1}^x\right)\right\} \quad (4)$$

3. **Los vecinos de arriba e izquierda son vigas o fronteras**
   Los términos que representan las celdas de arriba e izquierda se reemplazan por 0.

$$v_{i,j}^x = \frac{1}{4}\left\{ v_{i+1,j}^x + v_{i,j+1}^x - \frac{h}{2}v_{i,j}^x\left(v_{i+1,j}^x\right) - \frac{h}{2}v_{i,j}^y\left(v_{i,j+1}^x\right)\right\} \quad (5)$$

4. **Los vecinos de arriba y derecha son vigas o fronteras**
   Los términos que representan las celdas de arriba y derecha se reemplazan por 0.

$$v_{i,j}^x = \frac{1}{4}\left\{ v_{i+1,j}^x + v_{i,j-1}^x - \frac{h}{2}v_{i,j}^x\left(v_{i+1,j}^x\right) - \frac{h}{2}v_{i,j}^y\left(-v_{i,j-1}^x\right)\right\} \quad (6)$$

5. **El vecino de la izquierda es una viga o frontera**
   El término que representa la celda de la izquierda se reemplaza por 0.

$$v_{i,j}^x = \frac{1}{4}\left\{ v_{i+1,j}^x + v_{i-1,j}^x + v_{i,j+1}^x - \frac{h}{2}v_{i,j}^x\left(v_{i+1,j}^x - v_{i-1,j}^x\right) - \frac{h}{2}v_{i,j}^y\left(v_{i,j+1}^x\right)\right\} \quad (7)$$

6. **El vecino de la derecha es una viga o frontera**
   El término que representa la celda de la derecha se reemplaza por 0.

$$v_{i,j}^x = \frac{1}{4}\left\{ v_{i+1,j}^x + v_{i-1,j}^x + v_{i,j-1}^x - \frac{h}{2}v_{i,j}^x\left(v_{i+1,j}^x - v_{i-1,j}^x\right) - \frac{h}{2}v_{i,j}^y\left(-v_{i,j-1}^x\right)\right\} \quad (8)$$

7. **El vecino de abajo es una viga o una frontera**
   El término que representa la celda de abajo se reemplaza por cero.

$$v_{i,j}^x = \frac{1}{4}\left\{ v_{i-1,j}^x + v_{i,j+1}^x + v_{i,j-1}^x - \frac{h}{2}v_{i,j}^x\left(-v_{i-1,j}^x\right) - \frac{h}{2}v_{i,j}^y\left(v_{i,j+1}^x - v_{i,j-1}^x\right)\right\} \quad (9)$$

8. **Los vecinos de abajo e izquierda son vigas o fronteras**
   Los términos que representan las celdas de abajo e izquierda se reemplazan por 0.

$$v_{i,j}^x = \frac{1}{4}\left\{ v_{i-1,j}^x + v_{i,j+1}^x - \frac{h}{2}v_{i,j}^x\left(-v_{i-1,j}^x\right) - \frac{h}{2}v_{i,j}^y\left(v_{i,j+1}^x\right)\right\} \quad (10)$$

9. **Los vecinos de abajo y derecha son vigas o fronteras**
   Los términos que representan las celdas de abajo y derecha se reemplazan por 0.

$$v_{i,j}^x = \frac{1}{4}\left\{ v_{i-1,j}^x + v_{i,j-1}^x - \frac{h}{2}v_{i,j}^x\left(-v_{i-1,j}^x\right) - \frac{h}{2}v_{i,j}^y\left(-v_{i,j-1}^x\right)\right\} \quad (11)$$

2.6. Métodos para solucionar sistemas no lineales

Debido a que en el sistema se tiene una multiplicación de al menos 2 variables, el sistema es no lineal.

2.6.1. Método de Newton-Raphson

El método de Newton-Raphson consiste en usar la siguiente fórmula:

$$\mathbf{x}_{n+1} \;=\; \mathbf{x}_n \;-\; J_F\big(\mathbf{x}_n\big)^{-1}\, F\big(\mathbf{x}_n\big). \tag{12}$$

Para el problema de este documento, lo que se hizo primero fue crear una matriz de alto (8) por ancho (80), en donde cada una de las 640 celdas tuviera el nombre: pared, viga, InletF, o fluido. Luego para obtener el vector X en la iteración actual se recorre la matriz y se añade al vector X vacío cada celda de tipo fluido, resultando en un vector unidimensional de longitud 414.

Debido a que en la iteración 0 los valores de todas las celdas de tipo fluido son 0, entonces el vector X en la iteración 0 es un vector lleno de 0.

Para crear el vector unidimensional F en cada iteración, se recorre la matriz en busca de celdas tipo fluido y para cada celda se le aplica la fórmula siguiente y se anexa al final del vector F:

$$0 \;=\; v^x_{i,j} \;-\; \frac{1}{4}\left(v^x_{i+1,j} + v^x_{i-1,j} + v^x_{i,j+1} + v^x_{i,j-1} - \frac{h}{2}\, v^x_{i,j}\big(v^x_{i+1,j} - v^x_{i-1,j}\big) - \frac{h}{2}\, v^y_{i,j}\big(v^x_{i,j+1} - v^x_{i,j-1}\big)\right). \tag{13}$$

Para crear el Jacobiano inverso, primero se halla el Jacobiano y luego usando np.linalg.inv se halla el Jacobiano inverso. Para hallar el Jacobiano se usan las siguientes derivadas, las cuales se obtienen de derivar por cada variable la ecuación (2):

$$\frac{\partial F_{i,j}}{\partial u^x_{i,j}} = 1 - \frac{\partial\,\mathrm{RHS}_{i,j}}{\partial u^x_{i,j}} = 1 - \frac{1}{4}\left(-\frac{h}{2}(U - D)\right) = 1 + \frac{h}{8}\,(U - D), \tag{14}$$

$$\frac{\partial F_{i,j}}{\partial U} = -\frac{\partial\,\mathrm{RHS}_{i,j}}{\partial U} = -\frac{1}{4}\left(1 - \frac{h}{2}\, u^x_{i,j}\right), \tag{15}$$

$$\frac{\partial F_{i,j}}{\partial D} = -\frac{\partial\,\mathrm{RHS}_{i,j}}{\partial D} = -\frac{1}{4}\left(1 + \frac{h}{2}\, u^x_{i,j}\right), \tag{16}$$

$$\frac{\partial F_{i,j}}{\partial R} = -\frac{\partial\,\mathrm{RHS}_{i,j}}{\partial R} = -\frac{1}{4}\left(1 - \frac{h}{2}\, v^y_{i,j}\right), \tag{17}$$

$$\frac{\partial F_{i,j}}{\partial L} = -\frac{\partial\,\mathrm{RHS}_{i,j}}{\partial L} = -\frac{1}{4}\left(1 + \frac{h}{2}\, v^y_{i,j}\right). \tag{18}$$

El Jacobiano por ser de dimensiones número de variables X número de funciones, y como cada variable es una celda de tipo FLUID, entonces las dimensiones del Jacobiano para este caso son: 414 X 414.

Finalmente, se resuelve iteración por iteración, y se actualiza el vector X, y la matriz de valores en sus celdas tipo FLUID. Ya al actualizarse tanto la matriz como el vector X para esa iteración, para la siguiente iteración se recalcula el Jacobiano inverso, el vector F, y así sucesivamente hasta que:

$$\|\Delta\mathbf{x}_k\|_2 = \|\mathbf{x}_{k+1} - \mathbf{x}_k\|_2 < \tau_{\text{abs}}, \quad \text{con } \tau_{\text{abs}} = 10^{-5}. \tag{19}$$

Cuando se llegue a la iteración que cumpla esa condición, el algoritmo se detiene, debido a que es la aproximación que se quiere.

2.7. Métodos para solucionar sistemas lineales

Los métodos iterativos generan una secuencia de aproximaciones $\mathbf{x}^{(k)}$ que convergen hacia la solución exacta del sistema $A\mathbf{x} = \mathbf{b}$. La matriz cuadrada A es la matriz Jacobiana de dimensiones $n \times n$,

el vector columna b es el vector $-F$, el cual en cada celda i tiene la evaluación del vector $-F$ de cada elemento i del vector $\mathbf{x}_0$. El vector $\mathbf{x}_0$ contiene todas las n aproximaciones iniciales.

$$A\mathbf{x} = \mathbf{b} \quad \Longrightarrow \quad J\mathbf{x}_0 = -\mathbf{F} \tag{20}$$

2.7.1. Método de Richardson

Partimos de $A\mathbf{x} = \mathbf{b}$ y corregimos la aproximación con el residuo:

$$\mathbf{x}^{(k)} \;=\; \mathbf{x}^{(k-1)} \;+\; \alpha\,\big(\mathbf{b} - A\,\mathbf{x}^{(k-1)}\big), \tag{21}$$

que equivale a la forma estacionaria

$$\mathbf{x}^{(k)} \;=\; (I - \alpha A)\,\mathbf{x}^{(k-1)} \;+\; \alpha\,\mathbf{b}.$$

Cómo saber si converge (regla simple). Sea $T_\alpha := I - \alpha A$. El error cumple $\mathbf{e}^{(k)} = T_\alpha^k \mathbf{e}^{(0)}$. *Converge* para cualquier arranque si y sólo si

$$\rho(T_\alpha) \;=\; \rho(I - \alpha A) \;<\; 1,$$

donde $\rho(\cdot)$ es el radio espectral, es decir, el mayor valor absoluto de los autovalores.

Caso más usado: A simétrica definida positiva (SPD). En este caso los autovalores de A son reales positivos, $0 < \lambda_{\text{mín}} \le \lambda_i \le \lambda_{\text{máx}}$, y los de T_α son $1 - \alpha\lambda_i$. De aquí salen reglas **claras y prácticas**:

- **Condición de convergencia:** $0 \;<\; \alpha \;<\; \dfrac{2}{\lambda_{\text{máx}}(A)}$.
- **Buena elección (casi óptima):** $\alpha^\star = \dfrac{2}{\lambda_{\text{mín}}(A) + \lambda_{\text{máx}}(A)}$.
- **Velocidad de convergencia:** razón geométrica $\dfrac{\kappa(A) - 1}{\kappa(A) + 1}$, con $\kappa(A) = \dfrac{\lambda_{\text{máx}}(A)}{\lambda_{\text{mín}}(A)}$ (número de condición espectral).

Notas mínimas. (i) Con α *constante*, Richardson es un método *estacionario*. (ii) Si α varía con k, la condición se evalúa sobre T_{α_k} en cada paso.

2.7.2. Método de Jacobi

El método de Jacobi es un algoritmo iterativo utilizado para resolver sistemas lineales de la forma $A\mathbf{x} = \mathbf{b}$. Consiste en descomponer la matriz de coeficientes como $A = D + L + U$, donde D es la parte diagonal, L la parte inferior estricta y U la parte superior estricta. A partir de esta descomposición se obtiene la forma iterativa:

$$\mathbf{x}^{(k)} = D^{-1}\big(\mathbf{b} - (L + U)\mathbf{x}^{(k-1)}\big) \tag{22}$$

equivalentemente,

$$x_i^{(k)} \;=\; \frac{1}{a_{ii}}\left(b_i \;-\; \sum_{j \neq i} a_{ij}\, x_j^{(k-1)}\right). \tag{23}$$

En cada iteración, el método calcula una nueva aproximación $\mathbf{x}^{(k)}$ utilizando únicamente los valores de la iteración anterior $\mathbf{x}^{(k-1)}$. Esto lo hace sencillo de implementar, pero generalmente más lento que otros métodos como Gauss–Seidel, ya que no aprovecha los valores recién actualizados.

Forma estacionaria. La ecuación (22) puede escribirse en la forma general de punto fijo:

$$\mathbf{x}^{(k)} = T_J\,\mathbf{x}^{(k-1)} + \mathbf{c}_J,$$

donde

$$T_J = -D^{-1}(L+U), \qquad \mathbf{c}_J = D^{-1}\mathbf{b}.$$

Esta forma se usa para analizar la convergencia, ya que el método converge si el radio espectral cumple $\rho(T_J) < 1$; en tal caso, el error se reduce en cada iteración.

Convergencia. El método de Jacobi converge si la matriz A es *diagonalmente dominante* o *simétrica definida positiva*. En tales casos, las iteraciones sucesivas se aproximan al valor real de $\mathbf{x}$ con un error que disminuye de forma geométrica.

2.7.3. Método de Gauss–Seidel

El método de Gauss–Seidel es un algoritmo iterativo para resolver $A\mathbf{x} = \mathbf{b}$ que, a diferencia de Jacobi, *usa inmediatamente* los valores recién actualizados dentro de cada iteración. Con la descomposición $A = D + L + U$ (diagonal D, parte estrictamente inferior L, y estrictamente superior U), la forma iterativa matricial es:

$$\mathbf{x}^{(k)} = (D+L)^{-1}\big(\mathbf{b} - U\,\mathbf{x}^{(k-1)}\big), \tag{24}$$

equivalentemente, en forma componente:

$$x_i^{(k)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j<i} a_{ij}\,x_j^{(k)} - \sum_{j>i} a_{ij}\,x_j^{(k-1)}\right). \tag{25}$$

Forma estacionaria. La ecuación (24) puede expresarse como una iteración de punto fijo:

$$\mathbf{x}^{(k)} = T_{GS}\,\mathbf{x}^{(k-1)} + \mathbf{c}_{GS},$$

donde

$$T_{GS} = -(D+L)^{-1}U, \qquad \mathbf{c}_{GS} = (D+L)^{-1}\mathbf{b}.$$

Esta forma permite analizar la convergencia del método mediante el radio espectral $\rho(T_{GS})$. Si $\rho(T_{GS}) < 1$, el error se reduce en cada iteración y el proceso converge a la solución exacta.

Convergencia. El método de Gauss–Seidel converge si la matriz A es *diagonalmente dominante* (estricta) o *simétrica definida positiva*. En general, la convergencia está garantizada cuando el radio espectral cumple $\rho(T_{GS}) < 1$. Suele converger más rápido que el método de Jacobi, al aprovechar los valores recién actualizados dentro de cada iteración. Además, la versión SOR ($\omega \in (1,2)$) puede acelerar aún más la convergencia.

2.7.4. Método de Sobrerrelajación Sucesiva (SOR)

El método SOR es una extensión de Gauss-Seidel que introduce un parámetro de relajación $\omega > 0$ para acelerar (o estabilizar) la convergencia en la solución de $A\mathbf{x} = \mathbf{b}$, con la descomposición $A = D+L+U$ (diagonal D, parte estrictamente inferior L, y estrictamente superior U). Su forma iterativa matricial es

$$\mathbf{x}^{(k)} \;=\; (D+\omega L)^{-1}\Big[\,\omega\,\mathbf{b} \;-\; \big(\omega U + (\omega-1)D\big)\,\mathbf{x}^{(k-1)}\Big]. \tag{26}$$

Equivalente en componentes,

$$x_i^{(k)} \;=\; (1-\omega)\,x_i^{(k-1)} \;+\; \frac{\omega}{a_{ii}}\left(b_i \;-\; \sum_{j<i} a_{ij}\,x_j^{(k)} \;-\; \sum_{j>i} a_{ij}\,x_j^{(k-1)}\right). \tag{27}$$

Forma estacionaria. Puede escribirse como $\mathbf{x}^{(k)} = T_\omega\,\mathbf{x}^{(k-1)} + \mathbf{c}_\omega$, con

$$T_\omega \;=\; (D+\omega L)^{-1}\big[(1-\omega)D - \omega U\big], \qquad \mathbf{c}_\omega \;=\; \omega\,(D+\omega L)^{-1}\mathbf{b}.$$

En la práctica no se forma $(D+\omega L)^{-1}$; en cada iteración se resuelve un sistema triangular inferior con la misma matriz $D+\omega L$.

Elección de ω y convergencia.

- $\omega = 1$ recupera Gauss–Seidel; $0 < \omega < 1$ (*subrelajación*) puede estabilizar; $1 < \omega < 2$ (*sobrerelajación*) suele acelerar.
- El criterio general es $\rho(T_\omega) < 1$. En particular, si A es simétrica definida positiva, SOR converge para $0 < \omega < 2$.
- El ω óptimo depende de A; en problemas elípticos discretizados típicos (p. ej., Poisson), valores en [1,1, 1,9] suelen ser efectivos.

Criterio de paro. Detener cuando $\|\mathbf{b} - A\mathbf{x}^{(k)}\|_2 \le \varepsilon$ o $\|\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\|_2 \le \varepsilon$. Requisito básico: $a_{ii} \neq 0\ \forall i$.

2.7.5. Método de Gradiente Descendente

El método de gradiente descendente busca el mínimo de una función moviéndose en la dirección opuesta al gradiente, es decir, en la dirección de mayor descenso. En el contexto de un sistema cuadrático $A\mathbf{x} = \mathbf{b}$, equivale a minimizar

$$f(\mathbf{x}) = \tfrac{1}{2}\mathbf{x}^\top A\mathbf{x} - \mathbf{b}^\top\mathbf{x}.$$

Iteración básica.

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \alpha\,\nabla f(\mathbf{x}^{(k)}) = \mathbf{x}^{(k)} - \alpha(A\mathbf{x}^{(k)} - \mathbf{b}),$$

donde $\alpha > 0$ es el tamaño de paso o *tasa de aprendizaje*.

Interpretación. En cada iteración se avanza en la dirección contraria al gradiente (pendiente más pronunciada), con un paso controlado por α. Si α es muy grande, el método puede divergir; si es muy pequeño, converge lentamente.

Convergencia. Si A es simétrica definida positiva (SPD), el método converge para

$$0 < \alpha < \frac{2}{\lambda_{\text{máx}}(A)}.$$

El valor que minimiza el número de iteraciones (paso óptimo) es

$$\alpha^\star = \frac{2}{\lambda_{\text{mín}}(A) + \lambda_{\text{máx}}(A)}.$$

La velocidad de convergencia depende del número de condición $\kappa(A) = \lambda_{\text{máx}}/\lambda_{\text{mín}}$.

Relación con otros métodos. El gradiente descendente es la base de muchos algoritmos de optimización. En particular, el **método del gradiente conjugado** mejora su desempeño generando direcciones conjugadas que evitan el zigzagueo y logran convergencia mucho más rápida para sistemas SPD.

2.7.6. Método de Gradiente Conjugado (CG)

Problema y requisitos. Resuelve sistemas lineales simétricos definidos positivos (SPD):

$$A\mathbf{x} = \mathbf{b}, \quad A = A^\top,\ \mathbf{v}^\top A\mathbf{v} > 0\ \forall\, \mathbf{v} \neq \mathbf{0}.$$

Equivale a minimizar la energía cuadrática

$$\phi(\mathbf{x}) = \tfrac{1}{2}\,\mathbf{x}^\top A\mathbf{x} - \mathbf{b}^\top \mathbf{x}.$$

Idea central. Construye aproximaciones en el subespacio de Krylov $\mathcal{K}_k(A, \mathbf{r}_0) = \mathrm{span}\{\mathbf{r}_0, A\mathbf{r}_0, \ldots, A^{k-1}\mathbf{r}_0\}$, siguiendo direcciones *conjugadas en A*:

$$\mathbf{p}_i^\top A\,\mathbf{p}_j = 0 \quad (i \neq j),$$

lo que evita "deshacer" progreso como en descenso más pronunciado.

Iteración (forma básica). Con $\mathbf{r}_k = \mathbf{b} - A\mathbf{x}_k$ y $\mathbf{p}_0 = \mathbf{r}_0$:

$$\alpha_k = \frac{\mathbf{r}_k^\top \mathbf{r}_k}{\mathbf{p}_k^\top A\,\mathbf{p}_k}, \qquad \mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k\,\mathbf{p}_k,$$

$$\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k\,A\mathbf{p}_k, \qquad \beta_k = \frac{\mathbf{r}_{k+1}^\top \mathbf{r}_{k+1}}{\mathbf{r}_k^\top \mathbf{r}_k},$$

$$\mathbf{p}_{k+1} = \mathbf{r}_{k+1} + \beta_k\,\mathbf{p}_k.$$

En aritmética exacta, CG *resuelve en a lo sumo n pasos*. En práctica, se detiene cuando $\|\mathbf{r}_k\|_2 \le \varepsilon\|\mathbf{b}\|_2$.

Convergencia. En la norma A, el error decrece geométricamente con la condición espectral $\kappa(A) = \lambda_{\text{máx}}/\lambda_{\text{mín}}$:

$$\|\mathbf{e}_k\|_A \ \le\ 2\left(\frac{\sqrt{\kappa(A)}-1}{\sqrt{\kappa(A)}+1}\right)^k \|\mathbf{e}_0\|_A, \quad \mathbf{e}_k = \mathbf{x}_k - \mathbf{x}^*.$$

Cuanto más bien condicionada esté A, más rápido converge.

Preacondicionado (PCG). Si $M \approx A$ es SPD y fácil de invertir, definir $\mathbf{z}_k = M^{-1}\mathbf{r}_k$ y usar

$$\alpha_k = \frac{\mathbf{r}_k^\top \mathbf{z}_k}{\mathbf{p}_k^\top A\,\mathbf{p}_k}, \qquad \beta_k = \frac{\mathbf{r}_{k+1}^\top \mathbf{z}_{k+1}}{\mathbf{r}_k^\top \mathbf{z}_k}, \qquad \mathbf{p}_{k+1} = \mathbf{z}_{k+1} + \beta_k\,\mathbf{p}_k.$$

PCG mejora la tasa de convergencia al reducir $\kappa(M^{-1}A)$.

Coste y memoria. Cada iteración requiere un producto matriz–vector $A\mathbf{p}_k$ y pocos productos escalares; la memoria es $O(n)$ (almacena unos pocos vectores).

2.8. Interpolación con Splines Cúbicos

El procedimiento numérico descrito hasta ahora entrega una solución para la velocidad $v_{i,j}^x$ únicamente en la malla reducida de 8×80 celdas. Sin embargo, el problema físico original se planteó sobre una malla más fina de 40×400 celdas. Con el fin de recuperar esta resolución espacial (para análisis y visualización), se realiza una **interpolación bidimensional** sobre las regiones de fluido utilizando *splines cúbicos naturales*, sin modificar las condiciones de frontera ni los valores impuestos en las paredes y vigas.

En términos generales, el procedimiento consta de cuatro etapas:

1. reconstruir el campo de velocidades en la malla *malla gruesa* 8×80,
2. refinar en la dirección x (horizontal) por segmentos contiguos de fluido,
3. refinar en la dirección y (vertical) sobre esos segmentos,
4. combinar el campo refinado con la geometría fina 40×400 para obtener el campo final.

Reconstrucción del campo en la malla gruesa. El método SOR proporciona un vector solución $\mathbf{x}_j$ que contiene la velocidad v^x únicamente en las celdas etiquetadas como `FLUID`. Para poder interpolar en dos dimensiones, primero se reconstruye una matriz

$$U_{\text{malla gruesa}} \in \mathbb{R}^{8\times 80},$$

tal que:

- en cada celda `FLUID` se coloca el componente correspondiente de $\mathbf{x}_j$, respetando el orden de ensamblaje;
- en las celdas `VELOO` (entrada) se asigna directamente la velocidad impuesta $u = 1$;
- en paredes y vigas (`OWALL`, `BEAM`) se asigna el valor de frontera correspondiente (en este caso, $u = 0$).

De esta forma se obtiene un campo coherente con la solución numérica en toda la malla reducida.

Refinamiento en la dirección x con splines cúbicos. Sea r_x el factor de refinamiento en la dirección horizontal. En este trabajo se utiliza $r_x = 5$, por lo que cada intervalo entre columnas de la malla gruesa se subdivide en 5 columnas finas. El tamaño de la malla fina en x queda entonces

$$N_x^{\text{fine}} = 80 \cdot r_x = 400.$$

Para cada fila de la malla gruesa i, se identifican los *segmentos contiguos de fluido* en esa fila, es decir, rangos de columnas $(j_{\text{ini}}, j_{\text{fin}})$ donde todas las celdas son `FLUID` y están rodeadas por paredes, vigas o fronteras donde la velocidad ya es conocida. Dentro de cada segmento se toman los puntos de la malla gruesa

$$x_k = j_{\text{ini}} + k, \quad u_k = U_{\text{malla gruesa}}(i, x_k), \qquad k = 0, \ldots, n-1,$$

y se construye un spline cúbico natural

$$S_i(x) \approx u(x), \tag{28}$$

que interpola la velocidad en la dirección x para esa fila.

El spline $S_i(x)$ se evalúa luego en los puntos finos generados al subdividir cada intervalo $[x_k, x_{k+1}]$ en r_x subintervalos uniformes. De este modo se obtiene una matriz intermedia

$$U_{\text{inter}} \in \mathbb{R}^{8\times 400},$$

que contiene un refinamiento horizontal del campo, pero aún mantiene la resolución original en la dirección vertical.

Es importante resaltar que en esta etapa **solo se interpola sobre celdas** `FLUID`; las celdas de pared, vigas o entrada se dejan intactas, respetando así las condiciones de frontera.

Refinamiento en la dirección y con splines cúbicos. De forma análoga, se define un factor de refinamiento vertical r_y (en este caso $r_y = 5$), lo que lleva a una altura fina de

$$N_y^{\text{fine}} = 8 \cdot r_y = 40.$$

Para cada columna fina J (ya refinada en x), se determina la columna de la malla gruesa asociada mediante la relación

$$j_{\text{malla gruesa}} = \left\lfloor \frac{J}{r_x} \right\rfloor.$$

En esa columna de la malla gruesa $j_{\text{malla gruesa}}$ se buscan de nuevo segmentos contiguos de fluido $(i_{\text{ini}}, i_{\text{fin}})$ en la dirección vertical. A partir de los valores intermedios

$$y_k = i_{\text{ini}} + k, \quad u_k = U_{\text{inter}}(y_k, J),$$

se construye un spline cúbico natural

$$T_J(y) \approx u(y), \tag{29}$$

que se evalúa en los puntos refinados obtenidos al subdividir cada intervalo $[y_k, y_{k+1}]$ en r_y subintervalos. El resultado es una matriz fina

$$U_{\text{fine}} \in \mathbb{R}^{40\times 400},$$

que representa el campo de velocidad interpolado únicamente en regiones de fluido, con resolución equivalente a la malla original del problema.

Combinación con la geometría fina. Para mantener la coherencia con la definición geométrica del problema, se genera también una malla de clases en alta resolución mediante la misma rutina de construcción de malla, pero usando las dimensiones 40×400. Esto produce una matriz de nombres donde cada celda fina se clasifica nuevamente como `FLUID`, `VELOO`, `OWALL` o `BEAM`.

Finalmente, se construye una matriz de visualización U_{plot} sobre la malla fina combinando:

- los valores interpolados U_{fine} en las celdas de tipo `FLUID`,
- la condición de entrada $u = 1$ en las celdas `VELOO`,
- el valor de pared/obstáculo (por ejemplo, $u = 0$) en las celdas `OWALL` y `BEAM`.

De esta forma, la solución obtenida en la malla reducida mediante SOR se *prolonga* a la malla física de 40×400 mediante splines cúbicos naturales, respetando la estructura de fronteras y obstáculos del problema original y permitiendo analizar el campo de velocidades con la resolución espacial deseada.

3. Resultados

Los métodos iterativos generan una secuencia de aproximaciones $\mathbf{x}^{(k)}$ que convergen hacia la solución exacta del sistema $A\mathbf{x} = \mathbf{b}$. Para determinar cuándo detener el proceso iterativo, se emplea un **criterio de paro** basado en el residuo:

$$\mathbf{r}^{(k)} = \mathbf{b} - A\,\mathbf{x}^{(k)}.$$

El método se considera convergente cuando la norma del residuo cae por debajo de una tolerancia predefinida ε:

$$\boxed{\|\mathbf{b} - A\,\mathbf{x}^{(k)}\|_2 < \varepsilon.}$$

Por último cabe destacar que para todos los métodos implementados se usó la variable $v^{y}_{i-1,j} = 4$. Los resultados mostraron que cuando esta variable crece el fluido avanza más en la malla, pero también se concluyó que cuando esta variable crece la matriz deja de ser diagonalmente dominante, por lo que este valor elegido es uno de los mayores valores que esta variable puede tomar.

3.1. Sistemas no lineales

El cuadro 2 resume el desempeño del método iterativo de Newton-Raphson registrando el omega usado, el error final por residuo, el número de iteraciones requeridas y el tiempo total de ejecución. Se observa que el método convergió en un tiempo menor a 1 segundo.

Cuadro 2: Desempeño del método de Newton-Raphson

Método	**Tolerancia** (ε)	$\boldsymbol{\omega}$	**Error por residuo**	**Iteraciones**	**Tiempo (s)**
Newton-Raphson	10^{-5}	–	$1{,}87 \times 10^{-11}$	5	0.9

A continuación se muestra la malla reducida de 8X80 que se usó en esta subsección (la malla original es de 40X400).

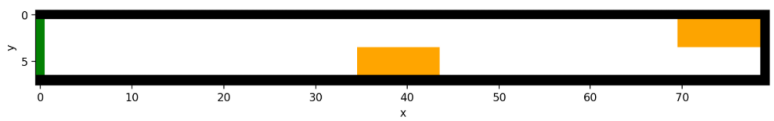

Figura 5: Malla 8X80

A continuación se muestra el rango de color para la velocidad final que se usa en la visualización del resultado de esta sección.

Figura 6: Rango de color para la velocidad

En la siguiente figura se muestra la solución del problema usando una malla de 8X80 y el método de Newton-Raphson.

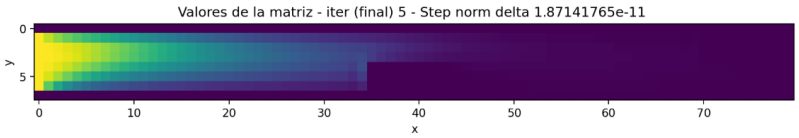

Figura 7: Resultado Newton-Raphson

3.2. Sistemas lineales

El cuadro 3 resume el desempeño de los métodos iterativos implementados. Cada algoritmo se ejecutó hasta cumplir el criterio de convergencia $\|\mathbf{b} - A\,\mathbf{x}^{(k)}\|_2 < \varepsilon$, registrando el omega usado, el error final por residuo, el número de iteraciones requeridas y el tiempo total de ejecución. Se observa que el método basado en gradiente descendente es el método que converge más lento, y también se observa cómo el método de gradiente conjugado no converge, esto es debido a que la matriz A (Jacobiano) no es simétrica positiva definida por lo que no se garantiza la convergencia de este método. En cuanto a los esquemas clásicos como Richardson, Jacobi, Gauss–Seidel y SOR se puede ver que todos convergen, siendo SOR el método que converge más rápido.

Cuadro 3: Comparación de desempeño de los métodos iterativos para solucionar sistemas lineales

Método	**Tolerancia** (ε)	$\boldsymbol{\omega}$	**Error por residuo**	**Iteraciones**	**Tiempo (s)**
Richardson	10^{-5}	0,5	$9{,}11 \times 10^{-6}$	205	29.97
Jacobi	10^{-5}	—	$9{,}76 \times 10^{-6}$	1452	234.85
Gauss–Seidel	10^{-5}	—	$9{,}99 \times 10^{-6}$	647	98.47
SOR ($\omega = 0{,}8$)	10^{-5}	0,8	$8{,}96 \times 10^{-6}$	31	4.60
Gradiente Descendente	10^{-5}	—	$9{,}86 \times 10^{-6}$	1907	297.62
Gradiente Conjugado	10^{-5}	—	$4{,}91 \times 10^{+4}$	12000	1131.92

A continuación se muestra la malla reducida de 8X80 que se usó en esta subsección (la malla original es de 40X400).

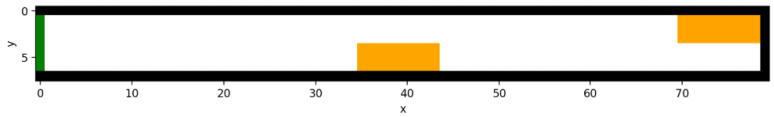

Figura 8: Malla 8X80

A continuación se muestra el rango de color para la velocidad final que se usa en la visualización del resultado de esta sección.

Figura 9: Rango de color para la velocidad

En la siguiente figura se muestra la solución del problema usando una malla de 8X80 y el método de SOR.

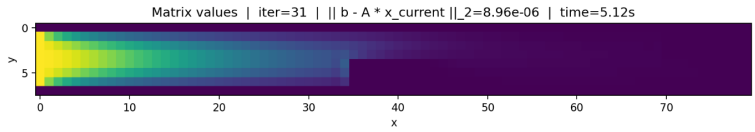

Figura 10: Resultado SOR

3.3. Splines

La solución numérica obtenida mediante los métodos iterativos se calcula sobre una malla reducida de 8×80, la cual proviene de escalar la malla física original de 40×400 mediante un factor de reducción $h = 5$. Este escalamiento disminuye significativamente el costo computacional, pero también produce una representación *pixelada* del campo de velocidades, ya que cada celda representa un bloque de 5×5 celdas del dominio original.

Con el fin de recuperar una visualización suave y detallada del flujo en la malla de alta resolución, se aplicó un procedimiento de **interpolación bidimensional mediante splines cúbicos naturales**. La idea consiste en "prolongar" la solución obtenida en la malla reducida hacia la malla completa, sin alterar las condiciones de frontera ni los valores impuestos en paredes, vigas u otros obstáculos.

El proceso se desarrolla en dos etapas principales:

1. **Interpolación horizontal**: Para cada fila de la malla gruesa se identifican los segmentos contiguos de celdas tipo `FLUID`. En cada uno de estos segmentos se construye un spline cúbico natural que interpola la velocidad en la dirección x. De esta forma, cada intervalo entre columnas de la malla gruesa se subdivide en cinco columnas finas, recuperando la resolución original horizontal.
2. **Interpolación vertical**: Una vez refinado el campo en la dirección x, se repite el procedimiento por columnas. Para cada columna fina se identifican los segmentos verticales de fluido y se construyen splines cúbicos naturales que permiten subdividir cada intervalo en cinco filas finas, restituyendo así la resolución vertical del dominio.

El resultado final es un campo continuo y suavizado sobre la malla de 40×400, que respeta estrictamente:

- los valores de frontera (`VELOO`),
- las regiones sólidas (`OWALL` y `BEAM`), que no se interpolan,
- y la estructura física original del problema.

De este modo, los splines cúbicos permiten reconstruir una representación visual coherente y de alta resolución del flujo, preservando la solución numérica obtenida en la malla reducida pero evitando el efecto pixelado inherente al escalamiento.

A continuación se muestra la malla original de 40X400.

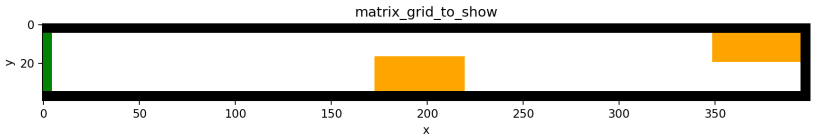

Figura 11: Malla 40X400

A continuación se muestra el rango de color para la velocidad final que se usa en la visualización del resultado de esta sección.

Figura 12: Rango de color para la velocidad

En la siguiente figura se muestra el resultado de interpolar el resultado obtenido usando una malla de 8X80 y el método SOR usando interpolación bidimensional mediante splines cúbicos naturales.

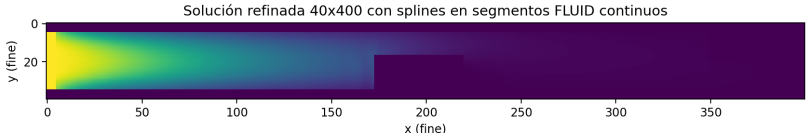

Figura 13: Resultado Splines

3.4. Repositorio en GitHub

El código fuente completo de este proyecto está disponible en el repositorio oficial de GitHub:

Repositorio: `navier-stokes-fluid-problem`