Asociación Argentina
de Mecánica Computacional

Mecánica Computacional Vol XXVII, págs. 67-82 (artículo completo)
Alberto Cardona, Mario Storti, Carlos Zuppa. (Eds.)
San Luis, Argentina, 10-13 Noviembre 2008

# FLUJO DE STOKES: COMPARACIÓN DE SOLVERS DIRECTOS E ITERATIVOS

**Pablo A. Caron[a], Paulo F. Porta[a] y Daniel T.P. Köster[b]**

[a]*Investigación de Productos Químicos, DARMEX S.A.C.I.F.I., L.M. Drago 1555, B1852LGS Burzaco, Buenos Aires, Argentina, {pcaron,pporta}@darmex-int.com, http://www.darmex-int.com*

[b]*Numerical Analysis and Computational Mechanics Group, University of Twente, Faculty EEMCS, Dept. of Applied Mathematics, P.O. Box 217, 7500 AE Enschede, The Netherlands, kosterdtp@ewi.utwente.nl*

**Palabras clave:** Flujo de Stokes, Solver directo, Solver iterativo.

**Resumen.**

Cuando se resuelve un flujo de Stokes existen dos estrategias para la resolución del sistema de ecuaciones, resolver el sistema acoplado ($v$ y $p$ al mismo tiempo) o resolver el sistema segregado ($v$ y $p$ por separado).

Los métodos segregados calculan los dos vectores incógnita, $v$ y $p$, separadamente. Esta aproximación involucra la solución de dos sub-sistemas lineales de menor tamaño, uno para $v$ y otro para $p$; en algunos casos se resuelve un sistema reducido para una incógnita auxiliar. Estos sub-sistemas se pueden resolver con solvers iterativos, directos, o una combinación de ellos.

Los métodos acoplados resuelven el sistema de ecuaciones completo, sin usar explícitamente sistemas reducidos. Estos métodos incluyen tanto solvers directos como iterativos. Los últimos típicamente con alguna forma de precondicionamiento.

El objetivo del presente trabajo es comparar la performance de un solver directo, cuando resuelve el sistema acoplado, con una implementación del método de gradientes conjugados (CG), con los subsistemas resueltos utilizando solvers iterativos. Los cálculos se realizaron en una computadora secuencial. El sistema de ecuaciones del flujo de Stokes se ensambla con las librerías ALBERTA (http://www.alberta-fem.de/). Además de poseer herramientas para ensamblar los sistemas de ecuaciones, ALBERTA incluye varios solvers iterativos. Uno de estos se utiliza en la solución de los subsistemas del esquema iterativo. Para resolver el sistema acoplado se ensambla la matriz de Stokes y se utiliza el software UMFPACK (http://www.cise.ufl.edu/research/sparse/umfpack/) como solver directo.

Se presentan resultados comparativos de la performance de los dos solvers para el caso del flujo alrededor de una esquina.

# 1. INTRODUCCIÓN

## 1.1. Flujo de Stokes

La hipótesis básica de un flujo reptante, desarrollada por Stokes (1851), es que los términos de densidad (inercia) son despreciables en la ecuación de movimiento. Este tipo de flujo es el que se tiene en el caso de fluidos de alta viscosidad a baja velocidad, dando lugar a un número de Reynolds bajo $Re << 1$. Sumando la condición de incompresibilidad el sistema de ecuaciones diferenciales resulta

$$\nabla \cdot \boldsymbol{v} = 0, \tag{1}$$

$$\rho \frac{\partial \boldsymbol{v}}{\partial t} = \nabla \cdot \Sigma + \rho \boldsymbol{f}, \tag{2}$$

donde

$$\boldsymbol{v} = \boldsymbol{V}_\Gamma \qquad \text{sobre } \Gamma_D, \text{ para } t > 0, \tag{3}$$

$$\Sigma \boldsymbol{n} = 0 \qquad \text{sobre } \Gamma_o, \text{ para } t > 0, \tag{4}$$

$$\Sigma := -p\mathbf{I} + \mu \left( \nabla \boldsymbol{v} + \nabla \boldsymbol{v}^T \right), \tag{5}$$

con $\Gamma := \partial\Omega$ y, $\Gamma = \Gamma_D \cup \Gamma_o$.

Siendo $\boldsymbol{v}$ la velocidad, $p$ la presión, $\rho$ la densidad, $\mu$ la viscosidad dinámica y $\Sigma$ el tensor de tensiones. La condición impuesta por (4) es del tipo *do-nothing* o de *tensión nula*. La misma se incluye en la formulación débil.

### 1.1.1. Formulación débil

La formulación débil se define multiplicando la ecuación de movimiento (2) con una función suave $\boldsymbol{\xi} : \Omega \to \mathbb{R}^d$ que satisface $\boldsymbol{\xi}(x) = 0$ para todo $x \in \Gamma_D$, e integrando sobre $\Omega$. Análogamente, se multiplica la ecuación de continuidad (1) con una función suave $\varphi : \Omega \to \mathbb{R}$. Esto conduce a

$$\int_\Omega \boldsymbol{\xi} \cdot \frac{\partial \boldsymbol{v}}{\partial t} + \frac{\nu}{2} D(\boldsymbol{\xi}) : D(\boldsymbol{v}) - p \nabla \cdot \boldsymbol{\xi} = \int_\Omega \boldsymbol{\xi} \cdot \boldsymbol{f}_v,$$

$$- \int_\Omega \varphi \nabla \cdot \boldsymbol{v} = 0$$

Aquí se usó la expresión del tensor rapidez de deformación $D(\boldsymbol{\xi}) := \nabla \boldsymbol{\xi} + \nabla \boldsymbol{\xi}^T$. Usando esta formulación se incorpora la condición de tensión libre a la salida (4) en la formulación débil.

### 1.1.2. Discretización

Para discretizar la formulación débil del problema se procede de acuerdo al método de las líneas. Así, primero se discretiza el problema en el espacio, produciendo un sistema acoplado de ODEs, el que luego se discretiza y resuelve usando un procedimiento *time-stepping*.

La discretización espacial consiste de elementos triangulares conformes con refinamiento por bisección. Sean los espacios abstractos de elementos finitos:

$$\begin{aligned}
\mathcal{V}_h &:= \operatorname{span}\{\boldsymbol{\xi}^1, \ldots, \boldsymbol{\xi}^{M_v}\} \subset \boldsymbol{H}^1(\Omega) && \text{para la velocidad,} \\
\mathcal{V}_{0,h} &:= \mathcal{V}_h \cap \boldsymbol{H}^1_{0,\Gamma_w}(\Omega), && \\
\mathcal{P}_h &:= \operatorname{span}\{\varphi_p^1, \ldots, \varphi_p^{M_p}\} \subset H^1(\Omega) && \text{para la presión,} \\
\mathcal{U}_h &:= \mathcal{V}_h \times \mathcal{P}_h. &&
\end{aligned}$$

Se debe tener en cuenta que los espacios $\mathcal{V}_h$ y $\mathcal{P}_h$ deben cumplir con la condición *LBB* para que la discretización este bien planteada.

Para la discretización temporal se elige una secuencia de pasos de tiempo variables $\tau_n > 0$, $n = 1, \ldots, N$ y se fija

$$t_n := \sum_{m=1}^{n} \tau_m, \quad T = t_N, \quad \tau = \max_{m=1,\ldots,N} \tau_m.$$

Se busca una aproximación $u_h^n := (\boldsymbol{v}_h^n, p_h^n) \in \mathcal{U}_h$ de la solución exacta en los tiempos discretos $t_n$. Como es usual, esto se alcanza reemplazando la derivada en el tiempo $u_h'(t_n)$ con el cociente de una diferencia (*Backward Euler*)

$$\frac{du_h}{dt}(t_n) \approx \frac{u_h^n - u_h^{n-1}}{\tau_n}.$$

Una vez discretizada la formulación débil se puede reordenar formando el siguiente sistema matricial.

$$\mathcal{A} \cdot \boldsymbol{u} \;=\; \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^T & \mathbf{0} \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{v} \\ \boldsymbol{p} \end{bmatrix} \;=\; \begin{bmatrix} \mathbf{F} \\ \mathbf{0} \end{bmatrix} \tag{6}$$

donde

$$\mathbf{A}_{ij} \;:=\; \int_\Omega \frac{1}{\tau_n} \boldsymbol{\xi}_i \, \boldsymbol{\xi}_j + \frac{\nu}{2} D(\boldsymbol{\xi}_i) : D(\boldsymbol{\xi}_j) \tag{7}$$

$$\mathbf{B}_{ik} \;:=\; \int_\Omega \phi_k \nabla \cdot \xi_j \tag{8}$$

$$\mathbf{F}_i \;:=\; \int_\Omega \boldsymbol{\xi}_i \, \boldsymbol{f}_v + \frac{1}{\tau_n} \boldsymbol{\xi}_i \, \boldsymbol{v}^{n-1} \tag{9}$$

$$\mathbf{A} \in \mathbb{R}^{n\times n} \tag{10}$$
$$\mathbf{B} \in \mathbb{R}^{n\times m} \tag{11}$$
$$\mathbf{F} \in \mathbb{R}^{n} \tag{12}$$
$$\mathcal{A} \in \mathbb{R}^{(n+m)\times(n+m)} \tag{13}$$

El sistema lineal (6) se conoce como un problema del tipo *saddle-point*, un sistema con esta estructura no solo aparece para el flujo de Stokes. Otros ejemplos donde se lo puede encontrar son: problemas de optimización no-lineal, elasticidad incompresible, análisis de circuitos y análisis estructural entre otros. Para una descripción más detallada sobre los casos donde aparecen este tipo de sistemas, incluyendo pequeñas variaciones de los mismos, ver Benzi et al. (2005).

## 2. SOLVERS

### 2.1. Complemento de Schur, factorización en bloque

Si la submatriz $\mathbf{A}$ es invertible, la matriz $\mathcal{A}$ admite la siguiente factorización triangular en bloques,

$$\mathcal{A} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^T & \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{B}^T \cdot \mathbf{A}^{-1} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0} & -\mathbf{S} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{A}^{-1} \cdot \mathbf{B} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}, \tag{14}$$

donde $\mathbf{S} = \mathbf{B}^T \cdot \mathbf{A}^{-1} \cdot \mathbf{B}$ se denomina el complemento de Schur de $\mathbf{A}$ en $\mathcal{A}$. Un importante número de propiedades se pueden derivar en base a la ecuación (14), ver Benzi et al. (2005).

Otras factorizaciones tipo *LU* son:

$$\mathcal{A} = \begin{bmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{B}^T & -\mathbf{S} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{A}^{-1} \cdot \mathbf{B} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}, \tag{15}$$

y

$$\mathcal{A} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{B}^T \cdot \mathbf{A}^{-1} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{0} & -\mathbf{S} \end{bmatrix}. \tag{16}$$

La primera con $U$ teniendo *unos* en la diagonal y la segunda con $L$ teniendo *unos* en la diagonal. Si se utiliza la factorización $LU$ en bloque (16), premultiplicando por $L^{-1}$, se puede obtener el siguiente sistema transformado

$$\begin{bmatrix} \mathbf{I} & \mathbf{0} \\ -\mathbf{B}^T \cdot \mathbf{A}^{-1} & \mathbf{I} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^T & \mathbf{0} \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{v} \\ \boldsymbol{p} \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{B}^T \cdot \mathbf{A}^{-1} & \mathbf{I} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{F} \\ \mathbf{0} \end{bmatrix} \tag{17}$$

esto es,

$$\begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{0} & \mathbf{S} \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{v} \\ \boldsymbol{p} \end{bmatrix} = \begin{bmatrix} \mathbf{F} \\ \mathbf{B}^T \cdot \mathbf{A}^{-1} \cdot \mathbf{F} \end{bmatrix}. \tag{18}$$

Utilizando sustitución hacia atrás en bloques se puede resolver el sistema de forma segregada. Entonces se obtiene para la presión

$$\mathbf{S} \cdot \boldsymbol{p} \;=\; -\mathbf{B}^T \cdot \mathbf{A}^{-1} \cdot \mathbf{F}, \tag{19}$$

una vez que se obtuvo $\boldsymbol{p}^*$ de (19), se puede obtener $\boldsymbol{v}^*$ de la ecuación

$$\mathbf{A} \cdot \boldsymbol{v} \;=\; \mathbf{F} - \mathbf{B} \cdot \boldsymbol{p}^* \tag{20}$$

Como puede verse, en el caso segregado se resuelve de forma separada dos sistemas lineales, uno para la presión y otro para la velocidad de tamaño *m* y *n* respectivamente. La forma de resolver este sistema da lugar a una clasificación más, la resolución iterativa o la directa del sistema segregado.

## 2.2. Gradientes conjugados

El método de gradientes conjugados es el método iterativo más destacado para resolver sistemas lineales de ecuaciones *sparse*, cuando la matriz de los coeficientes es cuadrada, simétrica y definida positiva. El método es efectivo para resolver sistemas de ecuaciones de la forma,

$$\mathcal{M} \cdot \boldsymbol{x} = \boldsymbol{b} \tag{21}$$

donde $\boldsymbol{x}$ es la incógnita, $\boldsymbol{b}$ es un vector conocido, y $\mathcal{M}$ es la matriz de los coeficientes. El algoritmo (2.1) muestra los pasos para resolver el sistema de ecuaciones. Para más detalles ver Shewchuk (1994).

**Algoritmo 2.1**
Dado $\boldsymbol{x}_{(0)}$, $\epsilon << 1$ y $N_{max}$. Siendo $\|\cdot\|$ una norma apropiada.

1. Calcular $\boldsymbol{d}_{(0)} = \boldsymbol{r}_{(0)} = \boldsymbol{b} - \mathcal{M} \cdot \boldsymbol{x}_{(0)}$ residuo inicial y primera dirección de búsqueda,
2. Longitud del paso: $\alpha_{(i)} = \boldsymbol{r}^T_{(i)} \cdot \boldsymbol{r}_{(i)} / \boldsymbol{d}^T_{(i)} \cdot \mathcal{M} \cdot \boldsymbol{d}_{(i)}$, distancia a recorrer sobre la dirección de búsqueda,
3. Solución aproximada: $\boldsymbol{x}_{(i+1)} = \boldsymbol{x}_{(i)} + \alpha_{(i)}\, \boldsymbol{d}_{(i)}$
4. Residuo: $\boldsymbol{r}_{(i+1)} = \boldsymbol{r}_{(i)} - \alpha_{(i)}\, \mathcal{M} \cdot \boldsymbol{d}_{(i)}$
5. Mejora del paso: $\beta_{(i+1)} = \boldsymbol{r}^T_{(i+1)} \cdot \boldsymbol{r}_{(i+1)} / \boldsymbol{r}^T_{(i)} \cdot \boldsymbol{r}_{(i)}$
6. Nueva dirección de búsqueda: $\boldsymbol{d}_{(i+1)} = \boldsymbol{r}_{(i+1)} + \beta_{(i+1)}\, \boldsymbol{d}_{(i)}$

Si $\left\|\boldsymbol{r}_{(i+1)}\right\| > \epsilon$ volver al paso 2 y $i \leftarrow i + 1$. Si no, la solución del problema es $\boldsymbol{x}_{(i+1)}$.

### 2.3. Gradientes conjugados aplicados al sistema de Stokes

Para resolver el sistema de Stokes utilizando el método de gradientes conjugados se resuelve el sistema (19), utilizando implícitamente el sistema (20).

El residuo para el paso *i* correspondiente al sistema (19) es

$$\boldsymbol{r}_{(i)} = \mathbf{B}^T \cdot \mathbf{A}^{-1} \cdot \mathbf{F} - \mathbf{S} \cdot \boldsymbol{p}_{(i)}, \tag{22}$$

recordando que $\mathbf{S} = \mathbf{B}^T \cdot \mathbf{A}^{-1} \cdot \mathbf{B}$, se obtiene

$$\boldsymbol{r}_{(i)} = \mathbf{B}^T \cdot \mathbf{A}^{-1} \cdot \mathbf{F} + \mathbf{B}^T \cdot \mathbf{A}^{-1} \cdot \mathbf{B} \cdot \boldsymbol{p}_{(i)}, \tag{23}$$

Se utiliza la (20) en la anterior para eliminar $\boldsymbol{p}_{(i)}$. La expresión del residuo resulta

$$\boldsymbol{r}_{(i)} = -\mathbf{B}^T \cdot \boldsymbol{v}_{(i)} \tag{24}$$

Se introducen las siguientes incógnitas intermedias $\boldsymbol{u}$ y $\boldsymbol{z}$. Donde $\boldsymbol{u}$ se obtiene a partir de (20),

$$\begin{aligned} \mathbf{A} \cdot \boldsymbol{v}_{(i+1)} &= \mathbf{F} - \mathbf{B} \cdot \boldsymbol{p}_{(i+1)} \qquad (25)\\ &= \mathbf{F} - \mathbf{B} \cdot \left(\boldsymbol{p}_{(i)} + \rho\, \boldsymbol{d}_{(i)}\right) \\ &= \mathbf{F} - \mathbf{B} \cdot \boldsymbol{p}_{(i)} - \rho\, \mathbf{B} \cdot \boldsymbol{d}_{(i)} \\ &= \mathbf{A} \cdot \boldsymbol{v}_{(i)} - \rho\, \mathbf{B} \cdot \boldsymbol{d}_{(i)} \\ \boldsymbol{v}_{(i+1)} &= \boldsymbol{v}_{(i)} - \rho\, \mathbf{A}^{-1} \cdot \mathbf{B} \cdot \boldsymbol{d}_{(i)} \\ \boldsymbol{v}_{(i+1)} &= \boldsymbol{v}_{(i)} - \rho\, \boldsymbol{u} \qquad (26) \end{aligned}$$

donde

$$\mathbf{A} \cdot \boldsymbol{u} = \mathbf{B} \cdot \boldsymbol{d}_{(i)}. \tag{27}$$

La incógnita $\boldsymbol{z}$ se obtiene de (24),

$$\begin{aligned} \boldsymbol{r}_{(i+1)} &= -\mathbf{B}^T \cdot \boldsymbol{v}_{(i+1)} \\ &= -\mathbf{B}^T \cdot \left(\boldsymbol{v}_{(i)} - \rho \boldsymbol{u}\right) \\ &= -\mathbf{B}^T \cdot \boldsymbol{v}_{(i)} + \rho \mathbf{B}^T \cdot \boldsymbol{u} \\ \boldsymbol{r}_{(i+1)} &= \boldsymbol{r}_{(i)} + \rho \boldsymbol{z} \qquad (28) \end{aligned}$$

donde

$$\boldsymbol{z} = \mathbf{B}^T \cdot \boldsymbol{u}. \tag{29}$$

Con lo anterior en mente se presenta el algoritmo de resolución del sistema de Stokes por el método de gradientes conjugados.

**Algoritmo 2.2**

Dado $\boldsymbol{p}_{(0)}$ (si no hay una estimación inicial tomar $\boldsymbol{p}_{(0)} = 0$), $\epsilon << 1$ y $N_{max} >> 1$. Estando $\boldsymbol{p}_{(i)}$, $\boldsymbol{d}_{(i)}$, $\boldsymbol{r}_{(i)}$, y $\boldsymbol{z}$, en el espacio de la presión $\mathcal{P}_h$ y $\boldsymbol{v}_{(i)}$, $\boldsymbol{u}$, en el espacio de la velocidad $\mathcal{V}_h$.

Resolver: $\mathbf{A} \cdot \boldsymbol{v}_{(0)} = \boldsymbol{F} - \mathbf{B} \cdot \boldsymbol{p}_{(0)}$

1. Residuo y dirección de búsqueda inicial: $\boldsymbol{d}_{(0)} = \boldsymbol{r}_{(0)} = -\mathbf{B}^T \cdot \boldsymbol{v}_{(0)}$.

2. Obtener $\boldsymbol{u}$
$$\mathbf{A} \cdot \boldsymbol{u} = \mathbf{B} \cdot \boldsymbol{d}_{(i)}$$

3. Obtener $\boldsymbol{z}$ donde
$$\boldsymbol{z} = \mathbf{B}^T \cdot \boldsymbol{u}$$

4. Longitud del paso:
$$\rho = \frac{\boldsymbol{r}_{(i)}^T \cdot \boldsymbol{r}_{(i)}}{\boldsymbol{d}_{(i)}^T \cdot \mathbf{B}^T \cdot \boldsymbol{u}} \tag{30}$$

5. Solución aproximada
$$\boldsymbol{p}_{(i+1)} = \boldsymbol{p}_{(i)} + \rho\, \boldsymbol{d}_{(i)} \tag{31}$$
$$\boldsymbol{v}_{(i+1)} = \boldsymbol{v}_{(i)} - \rho\, \boldsymbol{u} \tag{32}$$

6. Residuo
$$\boldsymbol{r}_{(i+1)} = \boldsymbol{r}_{(i)} + \rho\, \boldsymbol{z} \tag{33}$$

7. Si $\left\|\boldsymbol{r}_{(i+1)}\right\| < \epsilon$ o $n > N_{max}$, salir, si no hacer:
$$\beta_{(i+1)} = \frac{\boldsymbol{r}_{(i+1)}^T \cdot \boldsymbol{r}_{(i+1)}}{\boldsymbol{r}_{(i)}^T \cdot \boldsymbol{r}_{(i)}} \tag{34}$$
$$\boldsymbol{d}_{(i+1)} = \boldsymbol{r}_{(i)} + \beta_{(i+1)}\, \boldsymbol{d}_{(i)} \tag{35}$$

   volver al paso 2 con $i \leftarrow i + 1$.

Para más detalles ver Pironneau (1989).

### 2.4. Solver directo

Un solver directo (en el presente trabajo implementado con UMFPACK) factoriza la matriz $\mathbf{P}\mathcal{A}\mathbf{Q}$ o $\mathbf{PR}\mathcal{A}\mathbf{Q}$ en el producto $\mathbf{LU}$, donde $\mathbf{L}$ y $\mathbf{U}$ son matrices triangular inferior y superior, respectivamente, $\mathbf{P}$ y $\mathbf{Q}$ son matrices de permutación, y $\mathbf{R}$ es una matriz diagonal de factores para escalar las filas. Tanto $\mathbf{P}$ como $\mathbf{Q}$ se elijen para reducir el *fill-in* (nuevas entradas no nulas en $\mathbf{L}$ y $\mathbf{U}$ correspondientes a entradas nulas en $\mathcal{A}$). La matriz de permutación $\mathbf{P}$ tiene el doble rol de reducir el *fill-in* y de mantener la exactitud numérica.

Un solver directo para matrices rala consta de cuatro pasos diferentes,

1. Un paso de ordenamiento que reordena filas y columnas, tal que los factores sufren poco *fill-in*, o que la matriz tenga una estructura especial tal como una forma triangular de a bloques,

2. Un paso de análisis o de factorización simbólica que determina la estructura de elementos no nulos de los factores y crea una estructura adecuada de los factores,

3. Una factorización numérica que calcula los factores **L** y **U**,

4. Un paso de resolución que realiza una sustitución hacia atrás y hacia adelante utilizando los factores del punto anterior.

Usualmente los pasos 1 y 2 involucran únicamente los *grafos* de la matriz, y por lo tanto operaciones con enteros. Los pasos 3 y 4 involucran operaciones de punto flotante. El paso 3 es usualmente el que consume más tiempo, mientrás que el paso 4 es alrededor de un orden de magnitud más rápido. El algoritmo usado en el paso 1 es bastante independiente del usado en el 3. Pero el algoritmo en el paso 2 esta frecuentemente muy relacionado con el del paso 3. Para los sistemas no-simétricos más generales el solver combina los pasos 1, 2, y 3 (tal es el caso de UMFPACK) tal que los valores numéricos también cumplen un rol en la determinación del orden de la eliminación. Ver Duff et al. (1986) y Davis (2004b).

## 3. IMPLEMENTACIÓN

El sistema de ecuaciones (6) es producto de la aproximación de la solución del problema de Stokes utilizando el método de los elementos finitos. Para ensamblar las matrices **A**, **B** y el vector **F** se utiliza el paquete de herramientas ALBERTA (Schmidt and Siebert (2005)).

El desarrollo de las estructuras de datos de ALBERTA se basa en el concepto abstracto de un espacio de elementos finitos como un triplo formado por *i)* un conjunto de elementos de malla, *ii)* un conjunto de funciones base locales sobre un elemento y *iii)* una conexión de las funciones base locales y globales dando los grados de libertad globales (Schmidt and Siebert (2005)). Usando estas estructuras de datos ALBERTA provee toda la estructura abstracta como espacio de elementos finitos, junto con mallas jerárquicas, rutinas para adaptar la malla y la administración completa de los espacios de elementos finitos y los correspondientes grados de libertad durante las modificaciones de la malla.

Estas características hacen que ALBERTA sea una buena elección al momento de ensamblar las matrices necesarias para armar el sistema (6).

### 3.1. Implementación solver CG

Para resolver el sistema (6) utilizando el método de gradientes conjugados se implementa el algoritmo (2.2). Las operaciones entre matrices y vectores se realizan utilizando funciones de ALBERTA basadas en las librerías *BLAS*. Por otro lado la resolución de los subsistemas de ecuaciones (27) y (29) se realizan utilizando solvers provistos por ALBERTA. Entre estos se elijen los solvers estándar CG y el GMRes.

### 3.2. Implementación solver directo

El solver a utilizar es UMFPACK (Davis and Duff (1997), Davis and Duff (1999), Davis (2004b), y Davis (2004a), http://www.cise.ufl.edu/research/sparse/umfpack/).

UMFPACK es un conjunto de rutinas para resolver sistemas lineales *sparse* no-simétricos, $\mathcal{A}\boldsymbol{x} = \boldsymbol{b}$, el cual utiliza el método *Unsymmetric MultiFrontal*. Está escrito en ANSI/ISO C. Incluye una interface para MATLAB, una llamable desde C, y una llamable desde Fortran.

Una de las rutinas que posee UMFPACK es un conversor de matrices, el argumento de la misma es la matriz en formato de triplo $(i, j, a_{ij})$ y el resultado es la matriz en un formato óptimo para trabajar dentro de UMFPACK. Esto motiva la implementación de un conversor de matrices desde la estructura de datos utilizada en ALBERTA a formato triplo. Luego se utiliza la rutina mencionada anteriormente poder trabajar con UMFPACK. Se implementa también un conversor de vectores desde y hacia ALBERTA, primero para exportar los vectores correspondientes al término forzante desde ALBERTA, luego para importar la solución del problema.

Debido a que se desea resolver el problema (6) también se implementa una función para ensamblar la matriz $\mathcal{A}$ teniendo en cuenta las condiciones de borde.

## 4. RESULTADOS NUMÉRICOS

### 4.1. Hardware y sistema operativo

Los cálculos se realizaron en una computadora de escritorio con las siguientes características: posee un procesador Intel Pentium 4 (3.00 GHz), 2Ghz de RAM (400 MHz) y como sistema operativo se utiliza Ubuntu 7.10, Kernel 2.6.22-15. Para compilar las librerías y los programas se utiliza el compilador de *GNU*, *gcc* versión 4.1.

### 4.2. Parámetros a determinar

En los experimentos numéricos se plantea aumentar la cantidad de grados de libertad, observando en cada caso la performance de cada método a través de los siguientes parámetros

- Tiempo de cálculo,
- Cantidad de memoria *RAM* pedida al sistema operativo (SO).

### 4.3. Caso de estudio 1

El primer experimento numérico utiliza un ejemplo ya tratado en Caron et al. (2007) Este ejemplo consiste en analizar el flujo de Stokes producido cuando se deforma un dominio en el tiempo como el que se muestra en la figura (1). Este se puede considerar como un modelo simplificado del llenado de un molde metálico con caucho cuando se utiliza moldeo por compresión/transferencia. Para más detalles ver Caron et al. (2007).

El ejemplo planteado es no-estacionario. Para evaluar la performance del método se toman valores críticos del ensayo numérico. Para el caso de la memoria RAM se

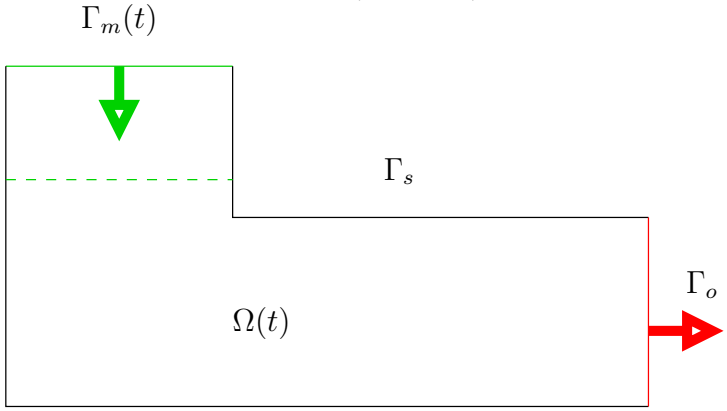


Figura 1: La figura ilustra el dominio $\Omega(t)$, la frontera $\Gamma = \Gamma_s \cup \Gamma_o \cup \Gamma_m(t)$, donde $\Gamma_s$ es la porción de la frontera fija e impermeable, $\Gamma_o$ representa una ventana de salida fija y $\Gamma_m(t)$ es una frontera móvil

toma el valor máximo requerido y para el tiempo de cálculo por paso se toma el valor medio.

### 4.3.1. Resultados

En la tabla (1) se presentan los resultados obtenidos. La estrategia es la siguiente: se resuelve el problema con ambos solvers incrementándose la cantidad de grados de libertad (GDL), se deja de calcular cuando hay algún limitante. En el caso del solver CG el tiempo de cálculo es la principal restricción, mientras que para el solver UMFPACK el limitante es la cantidad de memoria RAM necesaria para realizar el cálculo. Así se detiene el cálculo cuando el solver CG toma 12796 segundos (aproximadamente 3:30 horas) y cuando UMFPACK requiere 1662 Mb (casi el el total de memoria disponible. En los cálculos correspondientes al solver CG de la tabla (1) se utiliza uno tolerancia de $10^{-10}$ para el lazo exterior y $10^{-12}$ para la resolución iterativa de los sistemas (27) y (29).

En la figura (2) se muestra el consumo de memoria RAM del solver UMFPACK. Además se muestra la estimación de la cantidad de memoria requerida cuando se vuelve a aumentar la cantidad de grados de libertad del problema.

En la figura (3) se muestra la evolución de consumo de memoria durante el cálculo. Se indica también el momento en donde empieza a funcionar UMFPACK, cabe aclarar que en el caso de utilizar el solver CG en este punto no aumenta el uso de memoria RAM. Se observa entonces la cantidad de memoria extra para resolver utilizando el solver directo UMFPACK.

| Sistema | | UMFPACK | | CG | | Relación tiempos |
|---|---|---|---|---|---|---|
| GDL | Entradas no nulas[ %] | Wallclock time [seg] | RAM [Mb] | Wallclock time [seg] | N° Iter. | $T_{CG}/T_{\text{UMFPACK}}$ |
| 470 | 5.24 | 0.03 | 3.312 | 0.6 | 43 | 20 |
| 704 | 3.68 | 0.04 | 3.416 | 1.4 | 48 | 35 |
| 1139 | 2.24 | 0.08 | 5.340 | 4.06 | 71 | 51 |
| 1670 | 1.58 | 0.09 | 6.008 | 8.86 | 81 | 98 |
| 2751 | 0.95 | 0.14 | 8.948 | 28.02 | 105 | 200 |
| 4250 | 0.64 | 0.23 | 12.768 | 90.67 | 128 | 394 |
| 7271 | 0.37 | 0.36 | 18.712 | 368.89 | 162 | 1025 |
| 12002 | 0.23 | 0.80 | 32.008 | 888.13 | 171 | 1110 |
| 21495 | 0.13 | 1.83 | 55.132 | 1989.7 | 186 | 1087 |
| 70679 | 0.07 | 3.35 | 196.028 | 4812.74 | 187 | 1436 |
| 131090 | 0.04 | 7.78 | 389.840 | 12796 | 191 | 1644 |
| 251991 | 0.01 | 21.03 | 784.272 | - | - | - |
| 483410 | 0.01 | 63.07 | 1662.21 | - | - | - |

Tabla 1: Resultados comparativos de ambos solvers con el tamaño del problema creciente

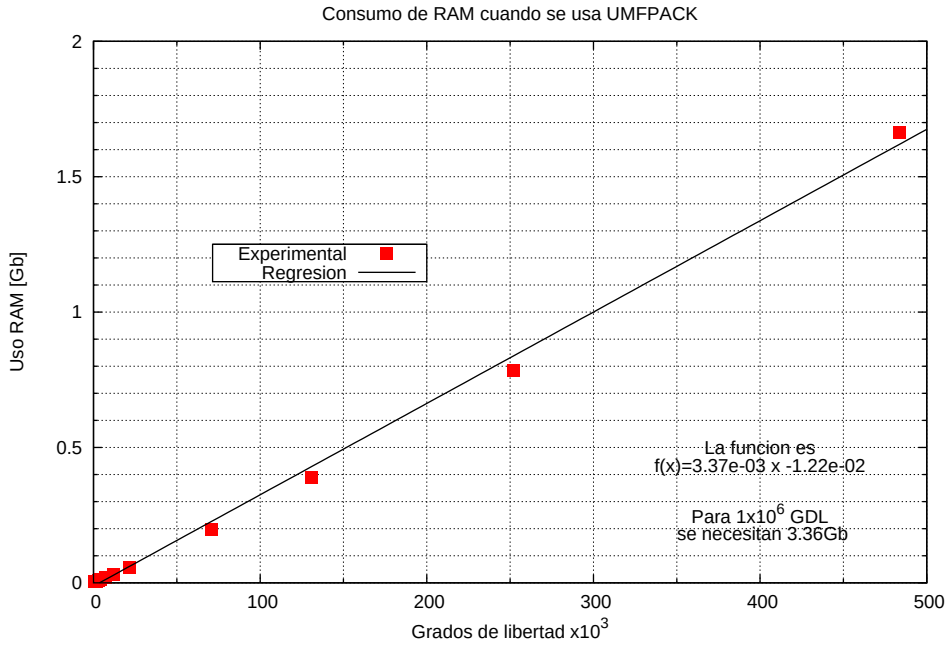


Figura 2: Cantidad de memoria RAM necesaria cuando se aumenta el tamaño del problema

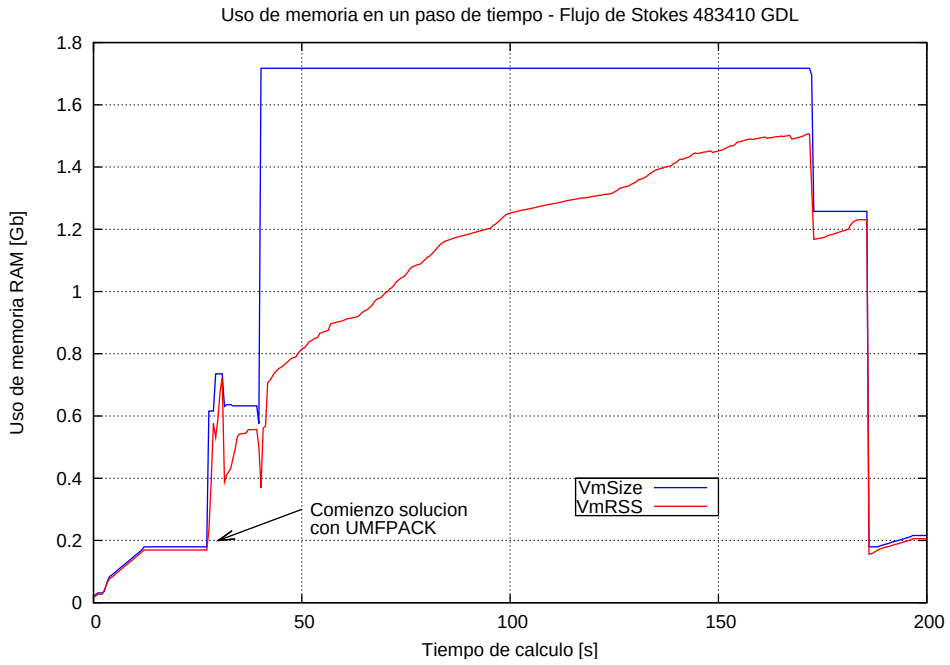


Figura 3: Uso de memoria del programa, donde VmSize es la memoria pedida al sistema y VmRSS es la memoria utilizada por el programa

### 4.4. Caso de estudio 2

Como segundo ejemplo se plantea lo siguiente: analizar el flujo transitorio generado cuando se tiene un fluido en reposo entre dos paredes y se mueve la pared inferior bruscamente desde cero a velocidad constante en *x*, manteniendo la pared superior en reposo. El objetivo es alcanzar con la simulación la solución de estado estacionario verificando el tiempo requerido por cada solver.

El dominio de estudio es $[0, 1]^2$, en el mismo se imponen condiciones del tipo *do-nothing* en los lados verticales, al lado superior se le impone velocidad nula y al lado inferior se le impone en $t = 0$ una velocidad constante unitaria en la dirección $x$.

Además de los solvers utilizados anteriormente se considera una tercera posibilidad: utilizar la misma factorización simbólica en todos los pasos de tiempo. Esto permite un ahorro en tiempo de CPU a costo de mantener la factorización en memoria. Esto se puede hacer debido a que la distribución de entradas no nulas de la matriz está vinculada con la conectividad de la malla y a que, en este caso, la misma no cambia en diferentes pasos.

Se plantea resolver el problema en $(\Omega, t) = [0; 1]^2 \times [0; 0,25)$. La cantidad de pasos de tiempo utilizados son $400$ y se toman densidades de malla desde $6903$ grados de libertad, $6098$ para la velocidad y $805$ para la presión, hasta $82839$, $73490$ y $9349$, respectivamente. Las tolerancias utilizadas para el solver CG coinciden con las del caso de estudio 1.

| GDL | GDL vel. | GDL pres. | CG | UMFPACK 1 | UMFPACK 2 |
|---|---|---|---|---|---|
| 6903 | 6098 | 805 | 00:13:10 | 00:08:00 | 00:06:57 |
| 22999 | 20370 | 2629 | 00:54:21 | 00:39:33 | 00:38:28 |
| 43021 | 38154 | 4867 | 02:33:39 | 01:11:43 | 01:05:40 |
| 82839 | 73490 | 9349 | 05:57:06 | 02:50:56 | 02:51:22 |

Tabla 2: Comparación de tiempos de cálculo para el problema de flujo entre placas. UMFPACK 1 se refiere al caso cuando no se reutiliza la factorización simbólica, UMFPACK 2 para el caso en que si se reutiliza. El tiempo se encuentra en formato HH:mm:ss

En la figura (4) se observa que la solución en $t = 0{,}25$ no es una recta como se espera en el caso estacionario (ver Batchelor (1967) página 191). Esto se debe a que el dominio no es el adecuado para representar la solución exacta. Como el objetivo es comparar la performance de cada solver no se considera necesario representar exactamente la solución exacta del problema, si no asegurar que se llega al estado estacionario.

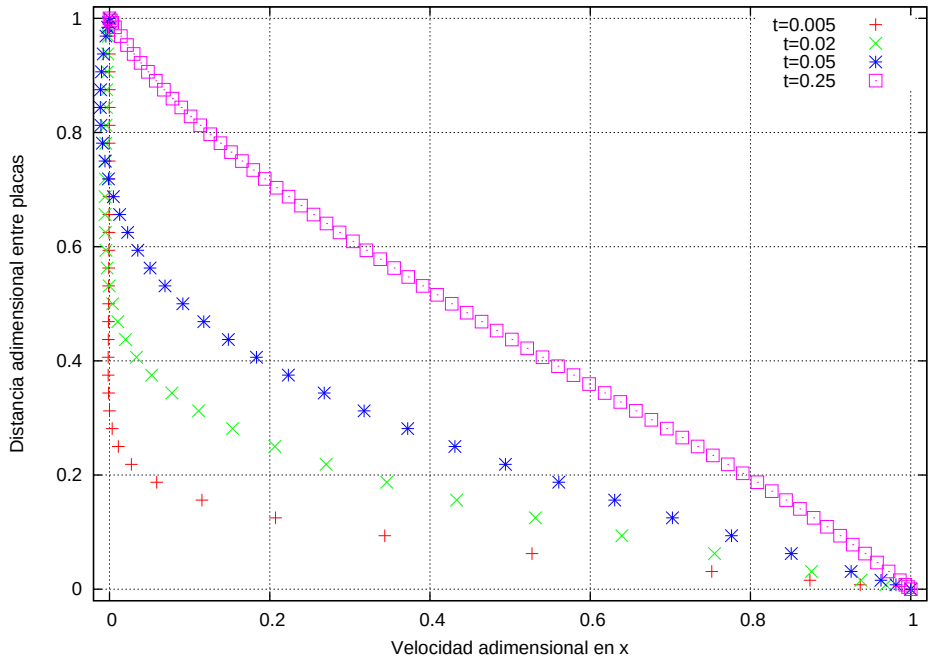


Figura 4: Velocidad en la dirección *x* del flujo entre dos placas

## 5. CONCLUSIONES

Se resuelve el problema de Stokes utilizando dos estrategias. La primera consiste en implementar el algoritmo (2.2), la segunda resuelve el sistema (6) directamente utilizando un solver directo (UMFPACK).

Los resultados muestran que el solver directo es mucho más rápido que el iterativo, esto se logra a expensas del uso de memoria, lo cual limita el tamaño del problema a unos $500000$ grados de libertad. Cabe aclarar que es una limitación no salvable, pasado el límite no se llega a una solución. En el caso del solver CG la limitación es el tiempo que se tarda en resolver, pero se puede llegar a una solución, aunque no resulta práctico.

En los casos en que la factorización simbólica consuma una cantidad de tiempo significativa, y que no se modifique la malla en el tiempo, se puede reutilizar la factorización simbólica. Como se dijo, la distribución de entradas no nulas está vinculada con la conectividad de la malla, mientras más compleja sea la geometría del problema más tiempo consumirá la factorización simbólica.

## REFERENCIAS

Batchelor G.K. *Fluid Dynamics*. 1967.

Benzi M., Golub G.H., and Liesen J. Numerical solution of saddle point problems. *Acta Numerica*, 14:1–137, 2005.

Caron P.A., Porta P.F., and Köster D.T. Flujo de un fluido newtoniano reactivo en un dominio movil. *Asociacion Argentina de Mecanica Computacional*, Mecanica Computacional(26):974, 2007.

Davis T.A. Algorithm 832: Umfpack v4.3—an unsymmetric-pattern multifrontal method. *ACM Transactions on Mathematical Software (TOMS)*, 30(2):196–199, 2004a.

Davis T.A. A column pre-ordering strategy for the unsymmetric-pattern multifrontal method. *ACM Transactions on Mathematical Software (TOMS)*, 30(2):165–195, 2004b.

Davis T.A. and Duff I.S. An unsymmetric-pattern multifrontal method for sparse lu factorization. *SIAM J. Matrix Anal. Appl.*, 18(1):140–158, 1997. doi:http://dx.doi.org/10.1137/S0895479894246905.

Davis T.A. and Duff I.S. A combined unifrontal/multifrontal method for unsymmetric sparse matrices. *ACM Transactions on Mathematical Software (TOMS)*, 25(1):1–20, 1999.

Duff I.S., Erisman A.M., and Reid J.K. *Direct methods for sparse matrices*. 1986.

Pironneau O. *The Finite Element Methods for Fluids*. 1989.

Schmidt A. and Siebert K.G. *Design of Adaptive Finite Element Software: The Finite Element Toolbox ALBERTA*. Lecture Notes in Computational Science and Engineering. 1 edition, 2005.

Shewchuk J.R. An introduction to the conjugate gradient method without the agonizing pain. *Preprint*, 1994.