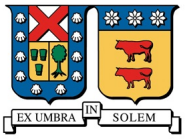

UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

Proyecto 1

Dinámica de Fluidos Computacional

IPM-468 - 2022

Departamento Ing. Mecánica UTFSM

Martín Achondo Mercado, 201860005-9

Eduardo Hasbun Contreras, 201841031-4

Profesor: Dr. Olivier Skurtys

14 de Noviembre de 2022

Resumen

Para este problema, se trabajó un sistema lineal. En este se pudo evidenciar el gran poder que tiene el método de Gradiente conjugado frente a métodos iterativos clásicos. Este método logró residuales de 10^{-9} para matrices de $ntimesn$ con $n = 100000$ en sólo 18 iteraciones, frente al método de Jacobi el cual necesitó 91 iteraciones. además, se pudo evidenciar la gran utilidad de utilizar matrices dispersas para la programación de los métodos y la sensibilidad respecto al punto de partida.

Índice

1. Introducción **3**
1.1. Presentación del problema 3

2. Metodología **3**
2.1. Métodos iterativos 4
2.1.1. Método de gradiente conjugado 5
2.1.2. Método de Jacobi 6
2.2. Simulaciones 6

3. Resultados **7**
3.1. Resultados para vector de partida: $x_0 = 1$ 7
3.2. Resultados para vector de partida: $x_0 = 0$ 7

4. Análisis de resultados **8**

5. Conclusiones **10**

6. Referencias **10**

7. Anexos **11**
7.1. Códigos elaborados 11

1. Introducción

En esta pregunta se utilizará el método de Gradiente Conjugado para resolver un sistema de ecuaciones en específico, intentando llegar a una resolver para $n = 100000$ ecuaciones. El propósito de esto es visualizar el poder de este método y compararlo con el método iterativo de Jacobi. Ambos métodos se compararán respecto al número de iteraciones y sensibilidad al punto de partida inicial.

1.1. Presentación del problema

Se considera una matriz A de tamaño $n \times n$. Los elementos de su diagonal principal son igual a 3, es decir, $a_{ii} = 3$ y los elementos de las diagonales adyacentes, por encima y por debajo, son igual a -1. Adicionalmente, la matriz tiene como elementos $1/2$ para $a_{i,n+1-i}$ para todo $i = 1, \cdots, n$ excepto si $i = \frac{n}{2}$ y $n = \frac{n}{2} + 1$.

Se define el vector b con entradas de la forma: $b = (2, 1.5, \cdots, 1.5, 1, 1, 1.5, \cdots, 1.5, 2)$. Esto lleva a $n - 4$ repeticiones de 1.5 y 2 repeticiones de 1 y de 2.

El objetivo entonces es poder resolver el sistema lineal $Ax = b$ con la matriz A y vector b presentado anteriormente para alcanzar una cantidad de ecuaciones de $n = 100000$ mediante el uso del método de Gradiente Conjugado y el método de Jacobi.

2. Metodología

Para una mejor resolución del sistema lineal a resolver, es importante revisar sus características. Este, escrito con sus entradas, se puede simplificar de la siguiente manera:

$$\begin{pmatrix} 3 & -1 & 0 & \cdots & \cdot & \cdot & \cdot & \cdots & 0 & 0.5 \\ -1 & 3 & -1 & 0 & & & & & \iddots & 0 \\ 0 & \ddots & \ddots & \ddots & & & & \iddots & 0 & \vdots \\ \vdots & & \ddots & \ddots & \ddots & & 0.5 & & & \cdot \\ \cdot & & & -1 & 3 & -1 & & & & \cdot \\ \cdot & & & & -1 & 3 & -1 & & & \cdot \\ \cdot & & & & 0.5 & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & \iddots & & & & \ddots & \ddots & \ddots & 0 \\ 0 & \iddots & & & & & 0 & -1 & 3 & -1 \\ 0.5 & 0 & \cdots & \cdot & \cdot & \cdot & \cdots & 0 & -1 & 3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_{\frac{n}{2}} \\ x_{\frac{n}{2}+1} \\ \vdots \\ \vdots \\ x_{n-1} \\ x_n \end{pmatrix} = \begin{pmatrix} 2 \\ 1.5 \\ \vdots \\ 1.5 \\ 1 \\ 1 \\ 1.5 \\ \vdots \\ 1.5 \\ 2 \end{pmatrix}$$

$$Ax = b \tag{2.1}$$

Puntos importantes a identificar respecto a la matriz A son los siguientes:

1. La matriz A es dispersa, la mayoría de sus entradas están compuestas por ceros. Es por esto que para n elevados se preferirá programar subrutinas que permitan trabajar esta matriz como dispersa y así ahorrar memoria. Una técnica usada el representar la matriz como 3 vectores, uno con todas las entradas no nulas de la matriz, otro con el número de la fila en la que se encuentran las entradas del vector anterior en la matriz y el último con las columnas también de las entradas de la matriz.
2. La matriz A es simétrica: $A^T = A$
3. La matriz A es estrictamente diagonal dominante. Esto es debido a que en cada fila, el elemento de la diagonal es mayor a la suma absoluta de los elementos que no están en la diagonal.

$$|a_{ii}| = \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}| \quad \forall i = 1, \cdots, n \tag{2.2}$$

4. La matriz A es definida positiva. Es decir, para cualquier vector v, se tiene:

$$v^T A v > 0 \tag{2.3}$$

2.1. Métodos iterativos

Los métodos iterativos son métodos para aproximar soluciones de sistemas lineales. De esta forma se genera una sucesión para encontrar la solución x^* del sistema $Ax = b$. En general entonces, se espera que a partir de una punto de partida inicial x_0, se cumpla que:

$$\lim_{x \to \infty} x_k = x^* \tag{2.4}$$

Dentro de las formas en los que se puede aproximar esta solución se encuentran:

- Métodos iterativos lineales: Se suele aproximar la solución por:

$$M x_{k+1} = N x_k + b \tag{2.5}$$

  Para estos métodos se pide que el radio espectral de la matriz iteración: $M^{-1}N$ sea menor a 1. La convergencia de estos métodos suele ser lenta para grandes sistemas.

- Métodos de proyección: Se suele aproximar la solución como:

$$x_{k+1} = x_k + v_k \tag{2.6}$$

En donde v_k se calcula de tal forma para lograr minimizar un funcional cuyo mínimo es equivalente a resolver el sistema lineal. Estos métodos suelen converger bastante rápido.

2.1.1. Método de gradiente conjugado

El método de gradiente conjugado es parte de los métodos de proyección que funciona para resolver sistemas lineales de la forma $Ax = b$ dado que plantea minimizar el siguiente funcional:

$$J(x) = \frac{1}{2}x^T Ax - b^T x \tag{2.7}$$

Notar que si A es simétrica, lo cual se cumple para el problema, se tiene que:

$$\nabla J(x) = Ax - b \tag{2.8}$$

Por esta razón, minimizar el funcional, lo cual equivale a anular su derivada, resuelve el sistema lineal $Ax = b$.

Este método al ser un método de proyección, lo que hace es ir bajar hacia este "mínimo".La dirección de bajada en este método es de: $p_k = -\nabla J(x_k)$.

Con lo dicho anteriormente, se muestra las iteraciones realizadas en este método: Se parte con un vector de partida x_0, se calcula el residual r_0 y la primera dirección de bajada p_0.

$$\begin{aligned} r_0 &= b - Ax_0 \\ p_0 &= r_0 \end{aligned} \tag{2.9}$$

Luego a esto, para cada iteración k, se calculan los parámetros α y β que controlan el paso hacia la solución y sobre la siguiente dirección de bajada respectivamente, seleccionadas de tal forma para hacer eficiente el método. De esta manera se tiene:

$$\begin{aligned} \alpha_k &= \frac{(r_k, r_k)}{(p_k, Ap_k)} \\ x_{k+1} &= x_k + \alpha_k p_k \\ r_{k+1} &= r_k - \alpha_k A p_k \\ \beta_{k+1} &= \frac{(r_{k+1}, r_{k+1})}{(r_k, r_k)} \\ p_{k+1} &= r_{k+1} + \beta_{k+1} p_k \end{aligned} \tag{2.10}$$

De esta manera se converge a la solución deseada. Notar que el algoritmo anterior funciona si la matriz es simétrica y definida positiva.

Es importante destacar que este método se plantea como exacto, pero es más rápido iterar para

alcanzar una buena aproximación dado el bajo número de iteraciones requerido.

2.1.2. Método de Jacobi

El método de Jacobi es parte de los métodos iterativos lineales, en donde $M = D$ (diagonal))y $N = -(L + U)$, los elementos fuera de la diagonal de A. De esta manera, a partir de un punto de partida x_0, la iteración k queda como:

$$Dx_{k+1} = -(L + U)x_k + b \tag{2.11}$$

Notar que en cada iteración aparece un sistema lineal a resolver, el cual es trivial dado que D es una matriz diagonal. Notar que la restricción para el radio espectral se aplica a la siguiente matriz iteración: $D^{-1}(L + U)$

De todas formas, se puede demostrar que este método converge si la matriz A es estrictamente diagonal dominante, condición la cual se cumple.

2.2. Simulaciones

Para ambos métodos se calculará el residuo de la solución y se verificará que sea menor a la tolerancia de 10^{-9} como punto de parada:

$$\|Ax_k - b\| < \text{tol} \tag{2.12}$$

Se prefiere comparar este residual sin dividir por el número de puntos dado que al dividir por n, se obtiene que a mayor n se converge más rápido, lo cual no es necesariamente cierto. Además, se presentará el máximo valor en el residual:

$$\text{máx}(Ax_k - b) \tag{2.13}$$

Con esto, se verificará el número de iteraciones para ambos métodos bajo la tolerancia estipulada. Las simulaciones se realizarán con dos vectores de partida. El primero compuesto por todos los elementos igual a 1 y el segundo con todos los elementos igual a 0. De esta manera se podrá evaluar la sensibilidad respecto al punto de partida.

3. Resultados

Se presentan los resultados obtenidos para cada vector de partida.

3.1. Resultados para vector de partida: $x_0 = 1$

Tabla 3.1: Resultados para método de gradiente conjugado

n	Residual	máx(r_k)	Iteraciones
10	7E-17	4E-16	5
100	5E-10	4E-16	18
1000	5E-10	4E-16	18
10000	5E-10	4E-16	18
100000	5E-10	4E-16	18

Tabla 3.2: Resultados para método de Jacobi

n	Residual	máx(r_k)	Iteraciones
10	8E-10	3E-10	60
100	9E-10	2E-10	91
1000	9E-10	2E-10	91
10000	9E-10	2E-10	91
100000	9E-10	2E-10	91

3.2. Resultados para vector de partida: $x_0 = 0$

Tabla 3.3: Resultados para método de gradiente conjugado

n	Residual	máx(r_k)	Iteraciones
10	3E-16	2E-16	5
100	3E-10	6E-11	21
1000	4E-10	9E-11	21
10000	4E-10	9E-11	21
100000	4E-10	9E-11	21

Tabla 3.4: Resultados para método de Jacobi

n	Residual	máx(r_k)	Iteraciones
10	8E-10	3E-10	61
100	9E-10	2E-10	94
1000	9E-10	2E-10	94
10000	9E-10	2E-10	94
100000	9E-10	2E-10	94

4. Análisis de resultados

A primera instancia se nota claramente que ambos métodos convergen a la solución, obteniendo un residual menor a 10^{-9}. Esto se debe a que la matriz A cumple con todos los criterios de convergencia para los métodos antes planteados, es decir, la matriz es simétrica, definida positiva y estrictamente diagonal dominante.

Respecto a los métodos, se nota que el método de gradiente conjugado tiene una mayor convergencia que el método de Jacobi. En los resultados para $n \geq 100$ y con e primer vector de partida, el método de gradiente conjugado convergió a la solución en solo 18 iteraciones, en comparación con el método de Jacobi el cual necesitaba 91. Esto se evidencia debido a que los métodos de proyección suelen convergen más rápido que los métodos iterativos lineales. Además, notar que en todos los casos, el valor máximo del residual siempre fue menor (por 6 ordenes de magnitud) que el método de Jacobi.

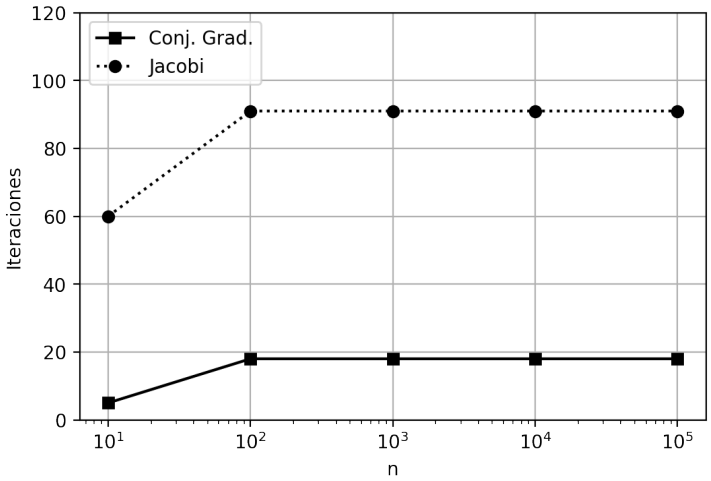

Figura 4.1: Iteraciones para ambos métodos con el primer vector de partida

En la figura anterior se nota como para ambos métodos se estanca el número de iteraciones en un punto fijo.

Respecto a la variación del punto de partida, se nota que no genera grandes cambios en el residual de la solución. Lo que si vale destacar es que el valor máximo del residual para el caso de gradiente

conjugado aumentó en 5 ordenes de magnitud. De todas formas el residual termina quedando menor a 10^{-9}, lo cual era condición. Por otra parte, en ambos métodos se nota que las iteraciones totales aumentaron en 3 unidades. Esto evidencia el desplazamiento del punto de partida inicial. De todas formas, para $x_0 = 1$, era esperable que los métodos converjan más rápido dado que la solución en su mayoría está compuesta por el valor 1.

Otro punto importante de destacar es que todos los algoritmos fueron programados tratando a la matriz como dispersa. Esto permitió correr todas las simulaciones en tiempos menores a 1 segundo, incluso para el caso $n = 100000$. Esto evidencia el ahorro de memoria al tratar sistemas de este estilo en la práctica y su gran aplicación en cálculos de alto nivel. A primera instancia se programó los métodos considerando la matriz como densa, y se pudo como máximo simular hasta $n = 20000$ y con tiempos de cálculo bastante elevados.

Por último, es importante destacar que los métodos comentados no se pueden aplicar en cualquier problema, dado que dependen de la matriz a trabajar. Por ejemplo, si la matriz no fuera simétrica, no se podría aplicar el método de gradiente conjugado. Formas para poder aplicar de todas formas este método existen y funcionan preacondicionando la matriz para que sea simétrica, modificando el sistema como:

$$A^T Ax = A^T b \tag{4.1}$$

De esta forma, la nueva matriz del sistema $A^T A$ es simétrica y se puede aplicar el método. Esta aplicación y solución termina siendo equivalente a minimizar el funcional: $\|Ax - b\|^2$. Para complementar, existen también otros métodos para iterar sobre sistemas lineales si es que el requerido no se puede aplicar, los cuales pueden ser Gauss-Seidel, ADI o GMRES, pero hay que tener cuidado para no perder eficiencia en este tipo de sistemas que son dispersos. Sería interesante comparar las iteraciones de estos métodos con los ya estipulados.

5. Conclusiones

Para finalizar, se nota como se pudo obtener convergencia para los métodos de gradiente conjugado y de Jacobi. Esto claramente se pudo hacer debido a que la matriz es simétrica, definida positiva y estrictamente diagonal dominante, requerimientos básicos de ambos métodos.

De lo obtenido, se nota el gran poder que tiene el método de gradiente conjugado frente al método de Jacobi, dada sus bajas iteraciones para alcanzar la tolerancia deseada. Esto se evidencia con las 18 iteraciones necesarias frente a las 91 del segundo método. El método de gradiente conjugado termina siendo muy aplicado en sistemas de este tipo, dispersos y simétricos, característicos de matrices que aparecen en solucionadores de la ecuación de Poisson, o métodos implícitos por ejemplo.

Por último, es válido destacar el hecho de programar eficientemente los algoritmos. Con esto se hace alusión a no crear matrices densas y preferir formulaciones dispersas si el problema lo permite. Esto logra hacer cálculos de $n = 100000$ como se pudo evidenciar, en menos de 1 segundo, hecho que con matrices densas no se habría podido lograr.

6. Referencias

[1] Chapra, Steven. Canale, Raymond. (2007). *Métodos numéricos para ingenieros*. McGraw-Hill.

[2] Quarteroni A. Sacco, R. Saleri, F. (2000). *Numerical Mathematics*. Springer

[3] Skurtys, Olivier. (2022). *Computación Científica*. UTFSM.

7. Anexos

7.1. Códigos elaborados

Código 7.1: Subrutina del método de gradiente conjugado

```
1
2  subroutine conj_grad_method(n,aij,rows,cols,b,x,tol)
3      implicit none
4      integer, intent(in) :: n,rows(n+1),cols(4*n-4)
5      real(8), intent(in) :: b(n),tol,aij(4*n-4)
6      real(8), intent(inout) :: x(n)
7
8      integer :: iter,imax
9      real(8) :: r(n),p(n),q(n),vv(n)
10     real(8) :: er,alpha,beta,rr
11
12     call matmul_sparse(n,rows,cols,aij,x,vv)
13     r = b - vv
14     p = r
15     call matmul_sparse(n,rows,cols,aij,p,q)
16
17     er = 1.0d0
18     imax = 100
19
20     do while (er.gt.tol)
21         alpha = dot_product(r,r)/dot_product(p,q)
22         x = x + alpha*p
23         rr = dot_product(r,r)
24         r = r - alpha*q
25         beta = dot_product(r,r)/rr
26         p = r + beta*p
27         call matmul_sparse(n,rows,cols,aij,p,q)
28
29         call matmul_sparse(n,rows,cols,aij,x,vv)
30         er = norm2(b - vv)
31
32         if (iter.gt.imax) then
33             exit
34         end if
35         iter = iter + 1
36
37     end do
38
39     write(*,*) 'Iterations', iter
40     write(*,*) 'Residual',  er
41     write(*,*) 'Max value', maxval(b-vv)
42
43 end subroutine conj_grad_method
```

Código 7.2: Subrutina del método de Jacobi

```
1
2  subroutine jacobi_method(n,aij,rows,cols,b,x,tol)
3      implicit none
4      integer, intent(in) :: n,rows(n+1),cols(4*n-4)
5      real(8), intent(in) :: b(n),tol,aij(4*n-4)
6      real(8), intent(inout) :: x(n)
7
8      integer :: iter,imax, rowsUL(n+1),colsUL(4*n-4)
9      real(8) :: x_old(n),vv(n),d,aijUL(4*n-4)
10     real(8) :: er
11
12     x_old(:) = x(:)
13     call create_UL_sparse(n,aijUL,rowsUL,colsUL,d)
14
15     er = 0.9d0
16     imax = 10000
17
18     do while (er.gt.tol)
19
20         call matmul_sparse(n,rowsUL,colsUL,aijUL,x_old,vv)
21
22         x(:) = (1.0d0/d)*(vv(:) + b(:))
23
24         call matmul_sparse(n,rows,cols,aij,x,vv)
25         er = norm2(b - vv)
26
27         if (iter.gt.imax) then
28             exit
29         end if
30         iter = iter + 1
31         x_old(:) = x
32
33     end do
34
35     write(*,*) 'Iterations', iter
36     write(*,*) 'Residual',  er
37     write(*,*) 'Max value', maxval(b-vv)
38
39 end subroutine jacobi_method
```