Pontificia Universidad Javeriana

Taller Analisis Numerico

Profesora

Eddy Herrera Daza

Integrantes

Diego Arroyo
arroyodiego@javeriana.edu.co
git: DiegoArroyoG

Santiago Caro
santiago.caro@javeriana.edu.co
git: SantiagoCaroDuque

Nicolas Lopez
lopezn.i@javeriana.edu.co
git: NicolasLopezFer

26 de agosto de 2019

1. Tabla de Contenidos

2. Introducción

En este taller se van a tratar temas referentes a la descomposición, operación y solución de matrices y los sistemas de ecuaciones que puedan estar asociados.

Adicionalmente, en la resolución del taller anterior hicimos uso de Python, para esta segunda parte del taller hemos concluido mas factible el uso de R.

3. Punto 1

En este punto se hará un reconocimiento del paquete "pracmaz se evaluara la matriz de transición por el método de Gauss-Seidel por relajación.

Se debe crear una matriz cuadrada y verificar que su convergencia sea menor a 1000, finalmente se compararan los resultados de la matriz de transición según Jacobi ($-D^{-1} * [E_i + E_s]$) y Gauss-Seidel ($[-D^{-1} * E_s] * [I + D^{-1} * E_i]$)

3.1. Diseño

3.1.1. Valores de entrada

M : Matriz cuadrada con convergencia menor a 1000 n : Tamaño de la matriz

3.1.2. Valores de salida

T_j : Matriz de transición según Jacobi. T_g : Matriz de transición según Gauss-Seidel.

3.2. Algoritmo

Algorithm 1 Punto 1.

procedure PUNTO 1(M)
  $I \leftarrow eye(n, n)$
  $M_1 \leftarrow ones(n, n)$
  $M_0 \leftarrow zeros(n, n)$
  $D \leftarrow Diag(M)$
  $E_i \leftarrow tril(A)$
  $E_s \leftarrow triu(A)$
  $T_j \leftarrow -D^{-1} * [E_i + E_s]$
  $T_g \leftarrow [-D^{-1} * E_s] * [I + D^{-1} * E_i]$
end procedure

4. Punto 2

Este punto se dividirá en 3 secciones, la primera se tratara de descomponer la matriz en su diagonal (D), esquina inferior (E_i) y esquina superior (E_s), seguido a esto se utilizará la función ïtersolveçon el método Gauss-Seidel para resolver un sistema asociado a la matriz (M) con el vector (b) y finalmente se calculara el error relativo en cada iteración del método de Jacobi

4.1. Diseño

4.1.1. Valores de entrada

M : Matriz cuadrada con convergencia menor a 1000 n : Tamaño de la matriz b : Vector de soluciones

4.1.2. Valores de salida

D : Matriz diagonal del vector M. E_i : Matriz con la esquina inferior del vector M. E_s : Matriz con la esquina superior del vector M. iS : Solución mediante ïtersolve". E_r : Error relativo.

4.2. Algoritmo

Algorithm 2 Punto 2.

procedure PUNTO 2(M)
    $D \leftarrow diag(M)$
    $E_i \leftarrow tril(M)$
    $E_s \leftarrow triu(M)$

    $iS \leftarrow itersolve(M, b, tol = 1e-9, method = \text{"}Gauss - Seidel\text{"})$

    $x_{n-1} = b$
    while $i \neq 5$ **do**
        $M_j = matrix(b - (E_i + E_s) * x_{n-1})$
        $x_n = D^{-1} * M_j$
        $E_r \leftarrow \frac{norm(x_n - x_{n-1})}{x_{n-1}}$
        $x_{n-1} = x_n$
        $i++$
    end while
end procedure

5. Punto 3

Se implementaran dos funciones, una que retorne la esquina inferior de la matriz y otra su diagonal.

5.1. Diseño

5.1.1. Valores de entrada

M : Matriz cuadrada con convergencia menor a 1000 n : Tamaño de la matriz

5.1.2. Valores de salida

D : Matriz diagonal del vector M. E_i : Matriz con la esquina inferior del vector M.

5.2. Algoritmo

Algorithm 3 Esquina inferior.

procedure ESQUINA INFERIOR(M)
  $k = 0$
  if $k == 0$ **then**
    $M[upper.tri(M, diag = TRUE] \leftarrow 0$
  else
    $M[col(M) >= row(M) + k + 1] \leftarrow 0$
  end if
  return $E_i = M$
end procedure

5.3. Algoritmo

Algorithm 4 Diagonal.

procedure DIAGONAL(M)
  $k = 0$
  if $k == 0$ **then**
    $M[upper.tri(M, diag = TRUE] \leftarrow 0$
    $M[lower.tri(M, diag = TRUE] \leftarrow 0$
  else
    $M[col(M) >= row(M) + k + 1] \leftarrow 0$
  end if
  return $D = M$
end procedure

6. Punto 4

Se realiza un conteo de las multiplicaciones que se deben hacer en el método directo de Gauss Jordan para resolver un sistema de ecuaciones.

6.1. Diseño

6.1.1. Valores de entrada

M : Matriz cuadrada con convergencia menor a 1000 n : Tamaño de la matriz b : Vector de soluciones

6.1.2. Valores de salida

C : Cantidad de multiplicaciones que se realizaron.

6.2. Algoritmo

Algorithm 5 Punto 4.

```
procedure PUNTO 4(M, b)
    return C
end procedure
```

7. Punto 5

Dado un sistema de ecuaciones se debe hallar los valores de las incógnitas para asegurar la convergencia por el método de Jacobi.

7.1. Diseño

7.1.1. Valores de entrada

7.1.2. Valores de salida

7.2. Algoritmo

Algorithm 6 Punto 5.

procedure PUNTO 5
    return
end procedure

8. Punto 6

Con la ayuda del paquete "matrix"se descompondrá una matriz (M) de la forma "LU", es decir, en términos de su esquina inferior y superior.

8.1. Diseño

8.1.1. Valores de Entrada

M : Matriz cuadrada con convergencia menor a 1000.

8.1.2. Valores de Salida

LU : Matriz en términos de u esquina inferior y superior.

8.2. Algoritmo

Algorithm 7 Punto 6

procedure PUNTO 6(M)
    $LU \leftarrow lu(M)$
end procedure

9. Punto 8

Se demostrará que la matriz de transición por el método de Gauss-Seidel esta dada por $T = (-D^{-1} * E_s) * (I + E_i * D^{-1})^{-1}$.

9.1. Diseño

9.1.1. Valores de Entrada

M : Matriz cuadrada con convergencia menor a 1000. b : Vector de soluciones

9.1.2. Valores de Salida

iS : Solución de la matriz por el método Gauss-Seidel

9.2. Algoritmo

Algorithm 8 Punto 8

procedure PUNTO 8(M, b)
    $D \leftarrow Diag(M)$
    $E_i \leftarrow tril(A)$
    $E_s \leftarrow triu(A)$
    $N \leftarrow 3$
    $M \leftarrow Diag(rep(3, N)) + Diag(rep(-2, N-1), k = -1) + Diag(rep(-2, N-1), k = 1)$
    $iS \leftarrow itersolve(M, b, tol = 1e-9, method = "Gauss-Seidel")$
end procedure