Análisis Numérico: Taller 2

Diana Sofía Carrillo Nelson Alejandro Mosquera Camilo Narvaez

Pontificia Universidad Javeriana
{ds-carrillog, nelson.mosquera, camilonarvaez}@javeriana.edu.co
21 de septiembre de 2021

Índice

Índice de figuras

1. Punto 1

$$\begin{aligned} u + 3v - w &= 18 \\ 4u - v + w &= 27{,}34 \\ u + v + 7w &= 16{,}2 \end{aligned}$$

1.1. Punto a

Para saber si la matriz es de coeficientes diagonal dominante, es necesario calcular la matriz de transición. Para el método de Jacobi, tenemos:

$$T = -D^{-1}(L+U) = -\begin{bmatrix} 0 & a_{1,2}/a_{1,1} & a_{1,3}/a_{1,1} & \dots & a_{1,n}/a_{1,1} \\ a_{2,1}/a_{2,2} & 0 & a_{2,3}/a_{2,2} & \dots & a_{2,n}/a_{2,2} \\ \vdots & \vdots & \vdots & \dots & \vdots \\ a_{n,1}/a_{n,n} & a_{n,2}/a_{n,n} & a_{n,3}/a_{n,n} & \dots & 0 \end{bmatrix}$$

Figura 1: Matriz de transición para el método iterativo de Jacobi

Por lo que la matriz de transición para el sistema dado es:

$$\begin{pmatrix} 0 & 3/1 & -1/1 \\ 4/-1 & 0 & 1/-1 \\ 1/7 & 1/7 & 0 \end{pmatrix} \tag{1}$$

La norma de fila de la matriz de transición (T) es $\parallel T \parallel = 5$, esta norma es mayor a 1 por lo que la matriz original no cumple la propiedad de ser diagonal dominante.

Para reorganizar la matriz de tal forma que sea diagonal dominante, debemos empezar por ver cuáles son las filas problemáticas de la matriz. Para la fila 1, la operación nos da 2 (problemática); para la fila 2, la operación nos da 5 (problemática); y para la fila 3, la operación nos da 2/7 (no problemática). De acuerdo a lo anterior, las filas a cambiar deben ser la 1 y la 2.

Empezando con la fila 1, una de las operaciones posibles es sumarle la fila 2. Al hacer lo anterior, tendríamos que la operación de suma para la matriz de transición con la nueva fila sería 2/5, cumpliendo la condición de ser menor que 1. Continuando con la fila 2, una de las operaciones posibles es restarle la fila 1 multiplicada por 4. Al hacer lo anterior. Tendríamos que la operación de suma para la matriz de transición con la nueva fila sería 5/13, cumpliendo la condición de ser menor que 1.

La matriz resultante sería la siguiente:

$$\begin{pmatrix} 5 & 2 & 0 & 45{,}34 \\ 0 & -13 & 5 & -44{,}66 \\ 1 & 1 & 7 & 16{,}2 \end{pmatrix} \tag{2}$$

Siendo la matriz de transición:

$$\begin{pmatrix} 0 & 2/5 & 0 \\ 0 & 0 & 5/-13 \\ 1/7 & 1/7 & 0 \end{pmatrix} \tag{3}$$

Siendo la norma de fila $\parallel T \parallel = 2/7$, comprobando la que la matriz cumple con la propiedad de ser una matriz diagonal dominante. Este resultado es, por ejemplo, una condición suficiente para asegurar la convergencia del método de Jacobi.

1.2. Punto b

La justificación a este punto, se encuentre en el punto a. La matriz de transición por el método de Jacobi, usando la matriz que ya sabemos que es diagonal dominante, es:

$$\begin{pmatrix} 0 & 2/5 & 0 \\ 0 & 0 & 5/-13 \\ 1/7 & 1/7 & 0 \end{pmatrix} \tag{4}$$

La norma de fila es $\parallel T \parallel = 2/7 < 1$, por lo que es diagonal dominante. En el caso del método de Jacobi, esta condición ya implica la convergencia del método para cualquier vector inicial $X(0) \in R^n$.

1.3. Punto c

Para este punto, desarrollamos dos programas en Python para comparar las soluciones al anterior sistema de ecuaciones utilizando los métodos de Jacobi y de Gauss Seidel. Para ambos programas, se usó una tolerancia de 10 6 , un número máximo de iteraciones de 20 y el vector inicial unitario. La solución dada por Jacobi es:

$$u = 7{,}585955729288199$$

$$v = 3{,}7051051506654042$$

$$w = 0{,}7012761603269835$$

, en 8 iteraciones.

Con el objetivo de comparar, en ambos programas calculamos el error relativo entre el resultado dado por cada método de solución versus el resultado dado por funciones contenidas en la librería numpy. El error relativo para el método de Jacobi fue de 2.547663911161327e-07, mientras que el error relativo para el método de Gauss Seidel fue de 8.21768574543654e-08.

Con base a estos resultados, podemos decir que la solución dada por el método de Gauss Seidel es la más aproximada a la solución real, sin contar que este método necesitó un menor número de iteraciones para llegar a esta solución. En conclusión, el método de Gauss Seidel, para este sistema de ecuaciones, es más eficiente y más fiable.

1.4. Punto d

A continuación se muestra las ultimas 5 iteraciones del método SOR y sus soluciones aproximadas con una tolerancia de 10^{-16} y 10 ω diferentes.

```
Iteración:283
[[ 1.          11.11531915 10.41404255]
 [30.34382979  1.          31.04510638]
 [ 7.58595745 11.29106383  1.        ]]
[7.58595745 3.70510638 0.7012766 ]
Error: 5.02429587e-15

Iteración:284
[[ 1.          11.11531915 10.41404255]
 [30.34382979  1.          31.04510638]
 [ 7.58595745 11.29106383  1.        ]]
[7.58595745 3.70510638 0.7012766 ]
Error: 5.02429587e-15

Iteración:285
[[ 1.          11.11531915 10.41404255]
 [30.34382979  1.          31.04510638]
 [ 7.58595745 11.29106383  1.        ]]
[7.58595745 3.70510638 0.7012766 ]
Error: 3.55271368e-15

Iteración:286
[[ 1.          11.11531915 10.41404255]
 [30.34382979  1.          31.04510638]
 [ 7.58595745 11.29106383  1.        ]]
[7.58595745 3.70510638 0.7012766 ]
Error: 3.55271368e-15

Iteración:287
[[ 1.          11.11531915 10.41404255]
 [30.34382979  1.          31.04510638]
 [ 7.58595745 11.29106383  1.        ]]
[7.58595745 3.70510638 0.7012766 ]
Error:          0

Respuesta:  [7.58595745 3.70510638 0.7012766 ]
```

Figura 2: Método SOR con $\omega = 0{,}12$

```
Iteración:190
[[ 1.         11.11531915 10.41404255]
 [30.34382979  1.         31.04510638]
 [ 7.58595745 11.29106383  1.        ]]
[7.58595745 3.70510638 0.7012766 ]
Error: 1.28094913e-14

Iteración:191
[[ 1.         11.11531915 10.41404255]
 [30.34382979  1.         31.04510638]
 [ 7.58595745 11.29106383  1.        ]]
[7.58595745 3.70510638 0.7012766 ]
Error: 7.10542736e-15

Iteración:192
[[ 1.         11.11531915 10.41404255]
 [30.34382979  1.         31.04510638]
 [ 7.58595745 11.29106383  1.        ]]
[7.58595745 3.70510638 0.7012766 ]
Error: 7.94410929e-15

Iteración:193
[[ 1.         11.11531915 10.41404255]
 [30.34382979  1.         31.04510638]
 [ 7.58595745 11.29106383  1.        ]]
[7.58595745 3.70510638 0.7012766 ]
Error: 5.02429587e-15

Iteración:194
[[ 1.         11.11531915 10.41404255]
 [30.34382979  1.         31.04510638]
 [ 7.58595745 11.29106383  1.        ]]
[7.58595745 3.70510638 0.7012766 ]
Error:          0

Respuesta:  [7.58595745 3.70510638 0.7012766 ]
```

Figura 3: Método SOR con $\omega = 0{,}17$

```
Iteración:171
[[ 1.         11.11531915 10.41404255]
 [30.34382979  1.         31.04510638]
 [ 7.58595745 11.29106383  1.        ]]
[7.58595745 3.70510638 0.7012766 ]
Error: 8.70233572e-15

Iteración:172
[[ 1.         11.11531915 10.41404255]
 [30.34382979  1.         31.04510638]
 [ 7.58595745 11.29106383  1.        ]]
[7.58595745 3.70510638 0.7012766 ]
Error: 5.02429587e-15

Iteración:173
[[ 1.         11.11531915 10.41404255]
 [30.34382979  1.         31.04510638]
 [ 7.58595745 11.29106383  1.        ]]
[7.58595745 3.70510638 0.7012766 ]
Error: 7.94410929e-15

Iteración:174
[[ 1.         11.11531915 10.41404255]
 [30.34382979  1.         31.04510638]
 [ 7.58595745 11.29106383  1.        ]]
[7.58595745 3.70510638 0.7012766 ]
Error: 6.1534806e-15

Iteración:175
[[ 1.         11.11531915 10.41404255]
 [30.34382979  1.         31.04510638]
 [ 7.58595745 11.29106383  1.        ]]
[7.58595745 3.70510638 0.7012766 ]
Error:          0

Respuesta:  [7.58595745 3.70510638 0.7012766 ]
```

Figura 4: Método SOR con $\omega = 0{,}19$

```
Iteración:169
[[ 1.         11.11531915 10.41404255]
 [30.34382979  1.         31.04510638]
 [ 7.58595745 11.29106383  1.        ]]
[7.58595745 3.70510638 0.7012766 ]
Error: 8.70233572e-15

Iteración:170
[[ 1.         11.11531915 10.41404255]
 [30.34382979  1.         31.04510638]
 [ 7.58595745 11.29106383  1.        ]]
[7.58595745 3.70510638 0.7012766 ]
Error: 5.02429587e-15

Iteración:171
[[ 1.         11.11531915 10.41404255]
 [30.34382979  1.         31.04510638]
 [ 7.58595745 11.29106383  1.        ]]
[7.58595745 3.70510638 0.7012766 ]
Error: 3.55271368e-15

Iteración:172
[[ 1.         11.11531915 10.41404255]
 [30.34382979  1.         31.04510638]
 [ 7.58595745 11.29106383  1.        ]]
[7.58595745 3.70510638 0.7012766 ]
Error: 3.55271368e-15

Iteración:173
[[ 1.         11.11531915 10.41404255]
 [30.34382979  1.         31.04510638]
 [ 7.58595745 11.29106383  1.        ]]
[7.58595745 3.70510638 0.7012766 ]
Error:          0

Respuesta:  [7.58595745 3.70510638 0.7012766 ]
```

Figura 5: Método SOR con $\omega = 0{,}2$

```
Iteración:150
[[ 1.         11.11531915 10.41404255]
 [30.34382979  1.         31.04510638]
 [ 7.58595745 11.29106383  1.        ]]
[7.58595745 3.70510638 0.7012766 ]
Error: 8.70233572e-15

Iteración:151
[[ 1.         11.11531915 10.41404255]
 [30.34382979  1.         31.04510638]
 [ 7.58595745 11.29106383  1.        ]]
[7.58595745 3.70510638 0.7012766 ]
Error: 7.94410929e-15

Iteración:152
[[ 1.         11.11531915 10.41404255]
 [30.34382979  1.         31.04510638]
 [ 7.58595745 11.29106383  1.        ]]
[7.58595745 3.70510638 0.7012766 ]
Error: 3.55271368e-15

Iteración:153
[[ 1.         11.11531915 10.41404255]
 [30.34382979  1.         31.04510638]
 [ 7.58595745 11.29106383  1.        ]]
[7.58595745 3.70510638 0.7012766 ]
Error: 6.1534806e-15

Iteración:154
[[ 1.         11.11531915 10.41404255]
 [30.34382979  1.         31.04510638]
 [ 7.58595745 11.29106383  1.        ]]
[7.58595745 3.70510638 0.7012766 ]
Error:          0

Respuesta:  [7.58595745 3.70510638 0.7012766 ]
```

Figura 6: Método SOR con $\omega = 0{,}212$

```
Iteración:146
[[ 1.         11.11531915 10.41404255]
 [30.34382979  1.         31.04510638]
 [ 7.58595745 11.29106383  1.        ]]
[7.58595745 3.70510638 0.7012766 ]
Error: 1.94590142e-14

Iteración:147
[[ 1.         11.11531915 10.41404255]
 [30.34382979  1.         31.04510638]
 [ 7.58595745 11.29106383  1.        ]]
[7.58595745 3.70510638 0.7012766 ]
Error: 1.00485917e-14

Iteración:148
[[ 1.         11.11531915 10.41404255]
 [30.34382979  1.         31.04510638]
 [ 7.58595745 11.29106383  1.        ]]
[7.58595745 3.70510638 0.7012766 ]
Error: 7.94410929e-15

Iteración:149
[[ 1.         11.11531915 10.41404255]
 [30.34382979  1.         31.04510638]
 [ 7.58595745 11.29106383  1.        ]]
[7.58595745 3.70510638 0.7012766 ]
Error: 1.0658141e-14

Iteración:150
[[ 1.         11.11531915 10.41404255]
 [30.34382979  1.         31.04510638]
 [ 7.58595745 11.29106383  1.        ]]
[7.58595745 3.70510638 0.7012766 ]
Error:          0

Respuesta:  [7.58595745 3.70510638 0.7012766 ]
```

Figura 7: Método SOR con $\omega = 0{,}215$

```
Iteración:70
[[ 1.         11.11531915 10.41404255]
 [30.34382979  1.         31.04510638]
 [ 7.58595745 11.29106383  1.        ]]
[7.58595745 3.70510638 0.7012766 ]
Error: 1.32930374e-14

Iteración:71
[[ 1.         11.11531915 10.41404255]
 [30.34382979  1.         31.04510638]
 [ 7.58595745 11.29106383  1.        ]]
[7.58595745 3.70510638 0.7012766 ]
Error: 3.55271368e-15

Iteración:72
[[ 1.         11.11531915 10.41404255]
 [30.34382979  1.         31.04510638]
 [ 7.58595745 11.29106383  1.        ]]
[7.58595745 3.70510638 0.7012766 ]
Error: 8.70233572e-15

Iteración:73
[[ 1.         11.11531915 10.41404255]
 [30.34382979  1.         31.04510638]
 [ 7.58595745 11.29106383  1.        ]]
[7.58595745 3.70510638 0.7012766 ]
Error: 1.0658141e-14

Iteración:74
[[ 1.         11.11531915 10.41404255]
 [30.34382979  1.         31.04510638]
 [ 7.58595745 11.29106383  1.        ]]
[7.58595745 3.70510638 0.7012766 ]
Error:          0

Respuesta:  [7.58595745 3.70510638 0.7012766 ]
```

Figura 8: Método SOR con $\omega = 0{,}4$

```
Iteración:996
[[ 1.         11.11531915 10.41404255]
 [30.34382979  1.         31.04510638]
 [ 7.58595745 11.29106383  1.        ]]
[7.58595745 3.70510638 0.7012766 ]
Error: 1.0658141e-14

Iteración:997
[[ 1.         11.11531915 10.41404255]
 [30.34382979  1.         31.04510638]
 [ 7.58595745 11.29106383  1.        ]]
[7.58595745 3.70510638 0.7012766 ]
Error: 8.70233572e-15

Iteración:998
[[ 1.         11.11531915 10.41404255]
 [30.34382979  1.         31.04510638]
 [ 7.58595745 11.29106383  1.        ]]
[7.58595745 3.70510638 0.7012766 ]
Error: 1.0658141e-14

Iteración:999
[[ 1.         11.11531915 10.41404255]
 [30.34382979  1.         31.04510638]
 [ 7.58595745 11.29106383  1.        ]]
[7.58595745 3.70510638 0.7012766 ]
Error: 8.70233572e-15

Iteración:1000
[[ 1.         11.11531915 10.41404255]
 [30.34382979  1.         31.04510638]
 [ 7.58595745 11.29106383  1.        ]]
[7.58595745 3.70510638 0.7012766 ]
Error: 1.0658141e-14

Respuesta:  [7.58595745 3.70510638 0.7012766 ]
```

Figura 9: Método SOR con $\omega = 0{,}41$

```
Iteración:64
[[ 1.         11.11531915 10.41404255]
 [30.34382979  1.         31.04510638]
 [ 7.58595745 11.29106383  1.        ]]
[7.58595745 3.70510638 0.7012766 ]
Error: 2.51214793e-14

Iteración:65
[[ 1.         11.11531915 10.41404255]
 [30.34382979  1.         31.04510638]
 [ 7.58595745 11.29106383  1.        ]]
[7.58595745 3.70510638 0.7012766 ]
Error: 5.02429587e-15

Iteración:66
[[ 1.         11.11531915 10.41404255]
 [30.34382979  1.         31.04510638]
 [ 7.58595745 11.29106383  1.        ]]
[7.58595745 3.70510638 0.7012766 ]
Error: 8.70233572e-15

Iteración:67
[[ 1.         11.11531915 10.41404255]
 [30.34382979  1.         31.04510638]
 [ 7.58595745 11.29106383  1.        ]]
[7.58595745 3.70510638 0.7012766 ]
Error: 1.0658141e-14

Iteración:68
[[ 1.         11.11531915 10.41404255]
 [30.34382979  1.         31.04510638]
 [ 7.58595745 11.29106383  1.        ]]
[7.58595745 3.70510638 0.7012766 ]
Error:          0

Respuesta:  [7.58595745 3.70510638 0.7012766 ]
```

Figura 10: Método SOR con $\omega = 0{,}42$

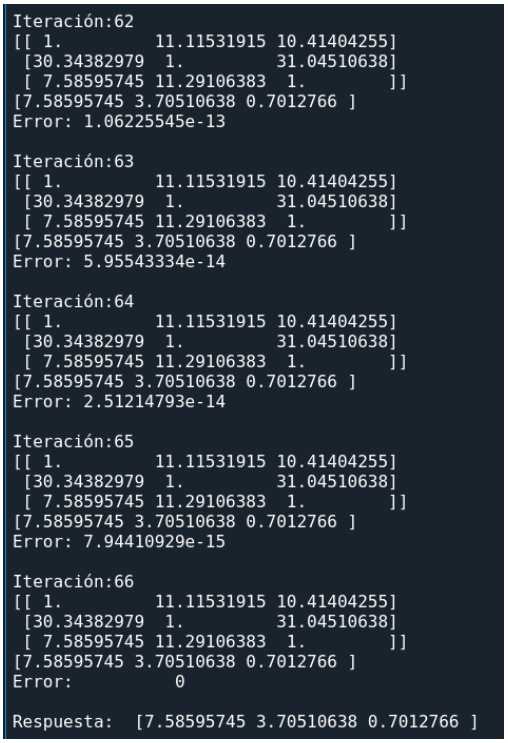

Figura 11: Método SOR con $\omega = 0{,}423$

Al realizar las pruebas se pudo llegar a la conclusión de que el intervalo de ω debe estar entre $0 < \omega < 1$ con esto aclarando que es un método de sub-relajación.

1.5. Punto e

$$w_{óptimo} = 1 + (\frac{\mu}{1 + \sqrt{1 - \mu^2}})^2$$

Para que se pueda utilizar esta función el parámetro de relajación cumple la condición necesaria de convergencia $w \in (0, 2)$; Los valores propios de la matriz de iteración Jacobi, $C_{Jac} = I - D^{-1}A$ son todos reales y El sistema lineal tiene solución y es única $detA \neq 0$. Con esto presente μ se puede representar como $\mu = p(C_{jac}) < 1$

2. Punto 3

Siguiendo el modelo f(x) entregado, se hizo la solución de este por medio de un programa en R. El resultado fue: para 1500 personas infectadas, el día 23 aproximadamente; para 1800 personas infectadas, el día 24 aproximadamente; y para 2000 personas afectadas, el día 25 aproximadamente.

3. Punto 9

A partir del enunciado, se realizó una implementación en Python del método gradiente conjugado, calculando el número de iteraciones y el tiempo necesitado para llegar a la solución. También, se comparó el tiempo con el tiempo tomado por dos métodos ofrecidos por librerías, el primero siendo la función general de solución de sistemas de ecuaciones (no usa gradiente conjugado) y el segundo usando el método de gradiente conjugado junto con la característica de usar una matriz dispersa.

En la implementación, no fue posible construir una matriz dispersa debido al tamaño requerido, sin embargo, se aseguró que se cumplieran los requisitos de ser una matriz simétrica y definida positiva.

Al correr el programa, se requiere un gran esfuerzo para lograr a la solución de una matriz con n = 10000, desde la creación de la matriz y el vector hasta la implementación de los métodos en sí. Las pruebas realizadas fueron hechas con un valor provisional de n = 1000.

Referencias

[1] https://sophiamyang.medium.com/descent-method-steepest-descent-and-conjugate-gradient-in-python-85aa4c4aac7b

[2] https://docs.scipy.org/doc/scipy/reference/generated/scipy.sparse.linalg.cg.html.

[3] [1] L. Ojeda, "Escuela Superior Politécnica del Litoral Facultad de Ciencias Naturales y Matemáticas Departamento de Matemáticas ANÁLISIS NUMÉRICO BÁSICO Un enfoque algorítmico con el soporte de Python." [Online].