Algebra matricial - El método de Gauss-Seidel

Curso de Física Computacional

M. en C. Gustavo Contreras Mayén

14 de mayo de 2017

Contenido

Contenido

Método Gauss-Seidel

Las ecuaciones $\mathbf{A}\ \mathbf{x} = \mathbf{b}$ en su forma escalar, se escriben de la siguiente manera:

$$\sum_{j=1}^{n} A_{ij}\ x_j = b_i \qquad i = 1, 2, \ldots, n$$

Despejando el término que contiene a x_i de la suma, obtenemos

$$A_{ii}\ x_i + \sum_{\substack{j=1 \\ j \neq i}}^{n} A_{ij}\ x_j = b_i \qquad i = 1, 2, \ldots, n$$

Resolviendo para x_i, resulta

$$x_i = \frac{1}{A_{ii}} \left(b_i - \sum_{\substack{j=1 \\ j \neq i}}^{n} A_{ij}\, x_j \right) \qquad i = 1, 2, \ldots, n$$

La ecuación anterior sugiere el siguiente esquema iterativo:

$$x_i \leftarrow \frac{1}{A_{ii}} \left(b_i - \sum_{\substack{j=1 \\ j \neq i}}^{n} A_{ij}\, x_j \right) \qquad i = 1, 2, \ldots, n$$

Iniciamos eligiendo un vector $\mathbf{x}$. Si la suposición inicial no fue buena, podemos elegir de manera aleatoria a $\mathbf{x}$.

La ecuación anterior se utiliza nuevamente para calcular cada uno de los elementos de x, utilizando siempre los últimos valores disponibles de x_j.

Esto completa un ciclo de iteración.

El procedimiento se repite hasta que los cambios en $\mathbf{x}$ con cada iteración sucesiva se vuelven lo suficientemente pequeños.

Es posible mejorar la convergencia del método de Gauss-Seidel con una técnica conocida como *relajación*.

La relajación

La idea es tomar un nuevo valor de x_i como un promedio ponderado de su valor anterior y el valor predicho por la ecuación anterior. La correspondiente fórmula iterativa, es

$$x_i \leftarrow \frac{\omega}{A_{ii}} \left(b_i - \sum_{\substack{j=1 \\ j \neq i}}^{n} A_{ij} x_j \right) + (1-\omega) x_i \qquad i = 1, 2, \ldots, n$$

donde ω es el *factor de relajación*.

Nótese que

1. Si $\omega = 1$, no se presenta la relajación.

Nótese que

1. Si $\omega = 1$, no se presenta la relajación.
2. Si $\omega < 1$, la ecuación de relajación, representa una interpolación entre los valores anteriores de x_i y el valor dado por la ecuación inicial. A esto se le llama *subrelajación*.

Nótese que

1. Si $\omega = 1$, no se presenta la relajación.
2. Si $\omega < 1$, la ecuación de relajación, representa una interpolación entre los valores anteriores de x_i y el valor dado por la ecuación inicial. A esto se le llama *subrelajación*.
3. Si $\omega > 1$, tenemos una extrapolación o sobrerelajación.

No hay ningún método práctico para determinar el valor óptimo de ω de antemano, sin embargo, una buena estimación se puede calcular en tiempo de ejecución.

Sea

$$\Delta x^k = |\mathbf{x}^{(k-1)} - \mathbf{x}^{(k)}|$$

la magnitud del cambio de $\mathbf{x}$ durante la k-ésima iteración (sin relajación, i.e. $\omega = 1$).

Si k es lo suficientemente grande ($k \geqslant 5$), se puede demostrar que una aproximación para el valor óptimo de ω es

$$\omega_{\text{opt}} \simeq \frac{2}{1 + \sqrt{1 - \left(\Delta x^{(k+p)} / \Delta x^{(k)}\right)^{1/p}}}$$

donde p es un entero positivo.

Método de GS con relajación

1. Realizar k iteraciones con $\omega = 1$ ($k = 10$ es razonable).

Método de GS con relajación

1. Realizar k iteraciones con $\omega = 1$ ($k = 10$ es razonable).
2. Luego de la k-ésima iteración, guardar $\Delta x^{(k)}$.

Método de GS con relajación

1. Realizar k iteraciones con $\omega = 1$ ($k = 10$ es razonable).
2. Luego de la k-ésima iteración, guardar $\Delta x^{(k)}$.
3. Ejecutar p iteraciones adicionales.

Método de GS con relajación

1. Realizar k iteraciones con $\omega = 1$ ($k = 10$ es razonable).
2. Luego de la k-ésima iteración, guardar $\Delta x^{(k)}$.
3. Ejecutar p iteraciones adicionales.
4. Guardar $\Delta x^{(k+p)}$ en la última iteración.

Método de GS con relajación

1. Realizar k iteraciones con $\omega = 1$ ($k = 10$ es razonable).
2. Luego de la k-ésima iteración, guardar $\Delta x^{(k)}$.
3. Ejecutar p iteraciones adicionales.
4. Guardar $\Delta x^{(k+p)}$ en la última iteración.
5. Ejecutar las demás iteraciones con $\omega = \omega_{\text{opt}}$, donde ω_{opt} se calcula como se indicó anteriormente.

Algorimto para Gauss-Seidel

La función `gaussSeidel` es una implementación del método de Gauss-Seidel con relajación. Se calcula automáticamente el valor deω_{opt} utilizando $k = 10$ y $p = 1$.

El usuario debe proporcionar la función `iterEqs` que calcula la mejora de $\mathbf{x}$ a partir de las fórmulas iterativas.

La función devuelve el vector solución $\mathbf{x}$, el número de iteraciones llevadas a cabo y el valor de ω_{opt} utilizado.

```
1  def gaussSeidel(iterEqs, x, tol = 1.0e-9):
2      omega = 1.0
3      k = 10
4      p = 1
5
6      for i in range(1,501):
7          xVieja = x.copy()
8          x = iterEqs(x, omega)
9          dx = np.sqrt(np.dot(x - xVieja, x - xVieja))
10         if dx < tol: return x, i, omega
11
12 # se calcula el relajamiento luego de k +p
       iteraciones
13         if i == k: dx1 = dx
14         if i == k + p:
15             dx2 = dx
16             omega = 2.0/(1.0 + np.sqrt(1.0 - (dx2/dx1)
                   **(1.0/p)))
17 print 'No converge Gauss-Seidel'
```

La función `iterEqs`

```
1  def iterEqs(x, omega):
2      n = len(x)
3      x[0] = omega*(x[1] - x[n-1])/2.0 + (1.0 - omega)*
           x[0]
4
5      for i in range(1, n - 1):
6          x[i] = omega*(x[i-1] + x[i+1])/2.0 + (1.0 -
               omega)*x[i]
7
8      x[n-1] = omega*(1.0 - x[0] + x[n-2])/2.0 + (1.0 -
           omega)*x[n-1]
9
10     return x
```

Ejercicio

Resolver el siguiente sistema de n ecuaciones simultáneas por el método de Gauss-Seidel con relajación (el programa deberá resolver para cualquier valor de n)

$$\begin{bmatrix} 2 & -1 & 0 & 0 & \dots & 0 & 0 & 0 & 1 \\ -1 & 2 & -1 & 0 & \dots & 0 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & \dots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & -1 & 2 & -1 \\ 1 & 0 & 0 & 0 & \dots & 0 & 0 & -1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{n-2} \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Ejercicio

Ejecutar el programa con $n = 20$, la solución exacta es

$$x_i = -\frac{n}{4} + \frac{i}{2} \qquad \text{con } i = 1, 2, \ldots, n$$

¿Qué necesitamos?

Necesitamos desarrollar las fórmulas iterativas a partir de:

$$x_i \leftarrow \frac{\omega}{A_{ii}} \left(b_i - \sum_{\substack{j=1 \\ j \neq i}}^{n} A_{ij} x_j \right) + (1-\omega) x_i \qquad i = 1, 2, \ldots, n$$

Para x_1, tenemos

$$x_1 = \frac{\omega}{2} \left((1)(x_2) - (1)(x_n) \right) + (1-\omega) x_1$$

$$x_1 = \frac{\omega (x_2 - x_n)}{2} + (1-\omega) x_1$$

$$x_1 = \frac{\omega(x_2 - x_n)}{2} + (1 - \omega)x_1$$

$$x_i = \frac{\omega(x_{i-1} + x_{i+1})}{2} + (1 - \omega)x_i \qquad i = 2, 3, \ldots, n - 1$$

$$x_n = \frac{\omega(1 - x_1 + x_{n-1})}{2} + (1 - \omega)x_n$$

Estas expresiones son las que serán evaluadas en la función `iterEqs`.

$$x_1 = \frac{\omega(x_2 - x_n)}{2} + (1 - \omega)x_1$$

$$x_i = \frac{\omega(x_{i-1} + x_{i+1})}{2} + (1 - \omega)x_i \qquad i = 2, 3, \ldots, n - 1$$

$$x_n = \frac{\omega(1 - x_1 + x_{n-1})}{2} + (1 - \omega)x_n$$

Estas expresiones son las que serán evaluadas en la función `iterEqs`.

Nota: Cada matriz $\mathbf{A}$ requiere de la construcción de sus ecuaciones de iteración, revisa que se necesita conocer los valores de la diagonal principal $\mathbf{d}$, así como del vector de coeficientes $\mathbf{b}$.

Código principal

```
1  n = eval(input('Numero de ecuaciones ==> '))
2
3  x = np.zeros((n), dtype='float64')
4
5  x, numIter, omega = gaussSeidel(iterEqs, x)
6
7  print ('\nNumero de iteraciones =', numIter)
8
9  print ('\nFactor de relajacion =', omega)
10
11 print ('\nLa solucion es :\n', x)
```

Solución al problema

La solución que nos devuelve el algoritmo, con $n = 20$ es:

```
 Número de ecuaciones = 20

Número de iteraciones = 259

Factor de relajación = 1.70545231071
```

Explorando la solución I.

Como podemos ver en la solución, el número de iteraciones es elevado, ¿a qué se debe?

Revisando la configuración del arreglo, notamos que no es dominante diagonal, por lo que en gran medida, el procedimiento de iteración es elevado, ahora: ¿qué podemos hacer?

Podemos reconfigurar el arreglo de tal manera en que sea dominante diagonal y podríamos revisar cuántas iteraciones requiere y compararlas contra la manera inicial.

La convergencia es muy lenta, porque la matriz de coeficientes carece de dominancia diagonal.

Si cambiamos cada término diagonal del coeficiente de 2 a 4, la matriz $\mathbf{A}$ sería diagonalmente dominante y la solución convergería en...¿?