Métodos iterativos de solución para matrices

Curso de Física Computacional

M. en C. Gustavo Contreras Mayén

Facultad de Ciencias - UNAM

23 de noviembre de 2017

1. Métodos iterativos

1. Métodos iterativos

Métodos iterativos

Hasta ahora, hemos discutido sólo los métodos directos de solución. La característica común de estos métodos es que calculan la solución con un número finito de operaciones.

Por otra parte, sabemos que si la computadora tuviera una precisión infinita (no hay errores de redondeo), la solución sería exacta.

Definición

Los métodos indirectos o iterativos, comienzan con una estimación inicial de la solución de x y luego mejoran repetidamente la solución hasta que el cambio en x se hace insignificante.

Definición

Dado que el número requerido de iteraciones puede ser grande, los métodos indirectos son, en general, más lentos que sus homólogos directos.

Sin embargo, los métodos iterativos tienen las siguientes ventajas que los hacen atractivos para ciertos problemas:

Consideraciones

1. Es factible para almacenar sólo los elementos distintos de cero de la matriz de coeficientes.

   Esto hace que sea posible para hacer frente a matrices muy grandes que son escasas, pero no necesariamente en bandas. En muchos problemas, no hay necesidad de almacenar la matriz de coeficientes en absoluto.

Consideraciones

2. Hay procedimientos iterativos de auto-corrección, es decir, que los errores de redondeo (o incluso errores aritméticos) en un ciclo iterativo se corrigen en los ciclos posteriores.

Un serio inconveniente de los métodos iterativos es que no siempre convergen a la solución.

Se puede demostrar que la convergencia está garantizada sólo si la matriz de coeficientes es diagonal dominante.

La estimación inicial de x no juega ningún papel en la determinación de si la convergencia se produce, si el procedimiento converge para un vector de partida, lo haría para cualquier vector de partida.

La estimación inicial afecta sólo el número de iteraciones que son necesarias para la convergencia.

Definición de dominancia diagonal

Una matriz $\mathbf{A}$ de $n \times n$ se dice que es *diagonal dominante* si cada elemento de la diagonal es más grande que la suma de los otros elementos de la misma fila (estamos hablando aquí de valor absoluto.

Por lo tanto dominancia diagonal que requiere

$$|A_{ii}| > \sum_{\substack{j=1 \\ j \neq i}}^{n} |A_{ij}| \qquad (i = 1, 2, \ldots, n)$$

Ejemplo de dominancia diagonal

La matriz

$$\begin{bmatrix} -2 & 4 & -1 \\ 1 & -1 & 3 \\ 4 & 2 & 1 \end{bmatrix}$$

No es dominante diagonal.

Ejemplo de dominancia diagonal

Pero si reordenamos los renglones de la siguiente manera

$$\begin{bmatrix} 4 & -2 & 1 \\ -2 & 4 & -1 \\ 1 & -1 & 3 \end{bmatrix}$$

resulta ser diagonal dominante.

Método Gauss-Seidel

Las ecuaciones $\mathbf{A}x = \mathbf{b}$ en su forma escalar, son:

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

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

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

Operación del método G-S

Resolviendo para x_i, resulta

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

Operación del método G-S

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 x. Si la suposición inicial no fue buena, podemos elegir de manera aleatoria a x.

Operación del método G-S

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 x con cada iteración sucesiva vuelven lo suficientemente pequeños.

Operación del método G-S

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

Operación del método G-S

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*.

Factor de relajación

Nótese que

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

Factor de 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*.

Factor de 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*.
3. Si $\omega > 1$, tenemos una extrapolación o sobrerelajación.

Factor de relajación

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.

Estimación del factor de relajación

Sea

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

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

Estimación del factor de relajación

Sea

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

la magnitud del cambio de 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.

Elementos esenciales para el método de GS con relajación

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

Elementos esenciales para el método de GS con relajación

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

Elementos esenciales para el método de GS con relajación

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

Algorimto para Gauss-Seidel

El usuario debe proporcionar la función `iterEqs` que calcula la mejora de x a partir de las fórmulas iterativas.

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

Código 1: Función Gauss-Seidel

```
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 = sqrt(dot(x - xVieja, x
     - xVieja))
10         if dx < tol: return x, i,
     omega
11
```

```
12            if i == k: dx1 = dx
13            if i == k + p:
14                dx2 = dx
15                omega = 2.0/(1.0 + sqrt(
    1.0 - (dx2/dx1)**(1.0/p)))
16    print ('No converge Gauss-Seidel')
```

Ejercicio

Resuelve 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

Ejecuta el programa con $n = 20$, la solución exacta es $x_i = -\frac{n}{4} + \frac{i}{2}$, con $i = 1, 2, \ldots, n$.

¿Qué necesitamos?

Necesitamos desarrollar las fórmulas iterativas a partir de:

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

¿Qué necesitamos?

Para x_1, tenemos

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

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

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

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

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

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 $\mathbf{b}$.

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

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

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

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 $\mathbf{b}$.

La función `iterEqs`

Código 2: Código para el ejercicio GS

```
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
```

Código principal I

Código 3: Código principal

```
1 n = eval(input('Numero de ecuaciones
      '))
2
3 x = zeros((n),dtype='float6 4')
4
5 x,numIter,omega = gaussSeidel(
     iterEqs, x)
6
7 print ('\nNumero de iteraciones =',
     numIter)
8
```

Código principal II

```
9  print ('\nFactor de relajacion =',
       omega)
10
11 print ('\nLa solucion es :',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.

Explorando la solución II.

¿Y si aumentamos el tamaño del arreglo?

El algoritmo que se propuso para el ejercicio, considera la solución para un sistema de n ecuaciones, ya lo tenemos para $n = 20$, completa la siguiente tabla:

n	iteraciones	ω_{opt}
20		
25		
50		
75		