U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:51.2
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
70210 ALDO HENRIQUE XAVIER 19HEL111 - 1 apos 01/10, 50% _____ / _____ / _____

Sistemas Lineares - Jacobi

Este método é iterativo, ou seja, não exato, já que parte-se de um valor atribuído inicial e mediante buscas locais sucessivas, o mesmo converge para a resposta correta. A diferença entre o resultado sugerido e aquele real, é denominada erro, e pode ser tornada tão pequena quanto se queira. O método entretanto tem um senão, que é um critério de convergência bastante rígido. Se não obedecido o algoritmo não converge para a resposta correta, e portanto o método é inútil.

Considere $A.x = b$ um sistema de quações lineares de ordem n e cujo $det(A) \neq 0$. A maneira extensa de escrever tal sistema é

$$\begin{array}{ccccccc} a_{11}x_1 & + & a_{12}x_2 & +...+ & a_{1n}x_n & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & +...+ & a_{2n}x_n & = & b_2 \\ ... & & & & & & \\ a_{n1}x_1 & + & a_{n2}x_2 & +...+ & a_{nn}x_n & = & b_n \end{array}$$

A matriz A dos coeficientes, pode ser decomposta em $A = L + D + R$, onde

L é a matriz inferior de A (o resto é zero)

D é a matriz contendo apenas a diagonal principal de A, o resto é zero.

R é a matriz superior de A (o resto é zero)

Veja por exemplo, se

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \therefore L = \begin{pmatrix} 0 & 0 & 0 \\ 4 & 0 & 0 \\ 7 & 8 & 0 \end{pmatrix},$$

e

$$D = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 9 \end{pmatrix}, R = \begin{pmatrix} 0 & 2 & 3 \\ 0 & 0 & 6 \\ 0 & 0 & 0 \end{pmatrix}$$

Supondo $det(D) \neq 0$, pode-se escrever: $(L + D + R)x = b$ e daqui $Dx = -(L + R)x + b$. e $x = -D^{-1}(L + R)x + D^{-1}b$. Chamando $B = -D^{-1}(L + R)$ e $g = D^{-1}.b$ obtem-se a equação geral deste método que é

$$x = Bx + g$$

. Note que é uma equação recursiva (ou iterativa) já que o x aparece nos 2 lados da mesma. A idéia aqui é atribuir um valor a x e com ele calcular um novo x, que é jogado no lugar do x velho e gera um novo x e assim sucessivamente até haver a convergência esperada (de antemão). Como supusemos $det(D) \neq 0$ pode-se dividir cada equação do sistema pelo coeficiente da diagonal principal, resultando

$$A^* = L^* + I + R^*$$

Agora o processo iterativo pode ser definido como

$$x^{k+1} = -(L^* + R^*)x^k + b^*$$

onde os elementos de L^*, R^* e b^* são:

$l_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i > j$ e zero senão

$d_{ij} = a_{ij}$ se $i = j$ e zero senão

$r_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i < j$ e zero senão

$b_ij^* = \frac{b_i}{a_{ii}}$

Critérios de convergência

Depois de dividida a matriz pelo elemento da diagonal principal, deve-se ter o **critério das linhas** que diz

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}^*| < 1$$

OU então o **critério das colunas** que diz

$$max_{1 \leq j \leq n} \sum_{i=1, i \neq j}^{n} |a_{ij}^*| < 1$$

Note que qualquer um dos dois critérios sendo satisfeitos, isso garante que o sistema tem convergência.

Definindo uma matriz como estritamente diagonalmente dominante se

$$\sum_{j=1, j \neq i}^{n} |a_{ij}| < |a_{ii}|$$

e se A é estritamente diagonalmente dominante, então A satisfaz o critério das linhas, e o sistema converge. Então, um bom critério de convergência pode ser

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}|* < 1$$

Exemplo

Seja resolver o sistema abaixo, usando o método de Jacobi

$$\begin{array}{ccccccc} 10x & + & 2y & + & z & = & 7 \\ x & + & 5y & + & z & = & -8 \\ 2x & + & 3y & + & 10z & = & 6 \end{array}$$

com $x^0 = (0.7, -1.6\ e\ 0.6)^t$ e $\epsilon < 0.11$.

Convergência ?

$|a_{12}| + |a_{13}| < |a_{11}|$? ou $|2| + |1| < |10|$? R: sim
$|a_{21}| + |a_{23}| < |a_{22}|$? ou $|1| + |1| < |5|$? R: sim
$|a_{31}| + |a_{32}| < |a_{33}|$? ou $|2| + |3| < |10|$? R: sim
logo, o sistema convergirá!

Solução

Dividindo cada equação pelo elemento da diagonal principal, fica

$$\begin{array}{ccccccc} x & + & 0.2y & + & 0.1z & = & 0.7 \\ 0.2x & + & y & + & 0.2z & = & -1.6 \\ 0.2x & + & 0.3y & + & z & = & 0.6 \end{array}$$

Apenas para ilustrar o critério das linhas para testar convergência (desnecessário no caso, já que já vimos que o sistema converge, mas em tese...)
$|a_{12}^*| + |a_{13}^*| = |0.2| + |0.1| = 0.3$ e
$|a_{21}^*| + |a_{23}^*| = |0.2| + |0.2| = 0.4$ e
$|a_{31}^*| + |a_{32}^*| = |0.2| + |0.3| = 0.5$ e
$\Rightarrow\ max_{1 \leq i \leq n}(0.3, 0.4\ 0.5) = 0.5 < 1$

Iterações

$x^{k+1} = -0.2y^k - 0.1z^k + 0.7$
$y^{k+1} = -0.2x^k - 0.2z^k - 1.6$
$z^{k+1} = -0.2x^k - 0.3y^k + 0.6$
Começando com $(0.7,\ -1.6,\ 0.6)$ obtemos uma primeira aproximação que é $(0.96,\ -1.86,\ 0.94)$ e jogando estes novos valores na fórmula iterativa, obtem-se:

$x \rightarrow 0.7,\ 0.96,\ 0.978,\ 0.9994,\ 0.9979$
$y \rightarrow -1.6,\ -1.86,\ -1.98,\ -1.9888,\ -1.9996$
$z \rightarrow 0.6,\ 0.94,\ 0.966,\ 0.9984,\ 0.9968$

O critério de parada pode ser
$max(x^{k+1} - x^k) < \epsilon$. No exemplo acima $0.0108 < 0.011$. Ou seja, neste caso, a solução procurada é $x = 0.9979$, $y = -1.9996$ e $z = 0.9978$. Salta aos olhos neste caso que se a tolerância for um pouco menor, a resposta exata será encontrada que é $(1, -2, 1)$. Basta lançar este resultado no sistema original que a exatidão será confirmada.

Solução Python

```
#jacobi
def jacobi(a,b):
 import numpy
 n=len(a) #numero de linhas de a
          # len(a[0])=colunas de a
 xvelho=numpy.zeros((n),float)
 erro=numpy.zeros((n),float)
 xnovo=numpy.zeros((n),float)
 f=numpy.zeros((n,len(a[0])),float)
 d=numpy.zeros((len(a[0])),float)
 tol=0.1
 for i in range(0,n):
  for j in range(0,n):
   if i==j:
    f[i,j]=0
   else:
    f[i,j]=-a[i,j]/a[i,i]
 for i in range(0,n):
  d[i]=b[i]/a[i,i]
 erromax=tol+1
 while erromax>tol:
  for i in range(0,n):
   xnovo[i]=0
   for k in range(0,n):
    xnovo[i]=xnovo[i]+f[i,k]*xvelho[k]
  for i in range(0,n):
   xnovo[i]=xnovo[i]+d[i]
  for i in range(0,n):
   erro[i]=abs(xnovo[i]-xvelho[i])
  erromax=erro[0]
  for i in range(1,n):
   if erro[i]>erromax:
    erromax=erro[i]
  for i in range(0,n):
   xvelho[i]=xnovo[i]
 return(xnovo)
import numpy
# para usar faça:
# a=numpy.array([[10,7,3],[1,5,4],[3,5,9]],float)
# b=numpy.array([22,33,44],float)
# print(jacobi(a,b))
```

☞ Para você fazer

Resolva usando este método e anexe a listagem do programa (ou copie no verso desta folha):

$$A = \begin{pmatrix} 89 & 20 & 4 & 24 & 35 \\ 6 & 83 & 29 & 15 & 25 \\ 11 & 34 & 71 & 22 & 3 \\ 26 & 27 & 30 & 117 & 31 \\ 12 & 10 & 19 & 33 & 79 \end{pmatrix}$$

e

$$B = (2141, 1638, 1540, 2741, 1182)$$

As respostas encontradas, com $\epsilon < 0.0001$ e sugestão inicial igual à coluna B são:

x_1	x_2	x_3	x_4	x_5

111-70210 - 01/10

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:51.2
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
70227 ALVARO JOSE BARANOSKI 19HEL111 - 2 apos 01/10, 50% _____ / _____ / _____

Sistemas Lineares - Jacobi

Este método é iterativo, ou seja, não exato, já que parte-se de um valor atribuído inicial e mediante buscas locais sucessivas, o mesmo converge para a resposta correta. A diferença entre o resultado sugerido e aquele real, é denominada erro, e pode ser tornada tão pequena quanto se queira. O método entretanto tem um senão, que é um critério de convergência bastante rígido. Se não obedecido o algoritmo não converge para a resposta correta, e portanto o método é inútil.

Considere $A.x = b$ um sistema de quações lineares de ordem n e cujo $det(A) \neq 0$. A maneira extensa de escrever tal sistema é

$$\begin{array}{ccccccc} a_{11}x_1 & + & a_{12}x_2 & +...+ & a_{1n}x_n & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & +...+ & a_{2n}x_n & = & b_2 \\ ... & & & & & & \\ a_{n1}x_1 & + & a_{n2}x_2 & +...+ & a_{nn}x_n & = & b_n \end{array}$$

A matriz A dos coeficientes, pode ser decomposta em $A = L + D + R$, onde

L é a matriz inferior de A (o resto é zero)

D é a matriz contendo apenas a diagonal principal de A, o resto é zero.

R é a matriz superior de A (o resto é zero)

Veja por exemplo, se

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \therefore L = \begin{pmatrix} 0 & 0 & 0 \\ 4 & 0 & 0 \\ 7 & 8 & 0 \end{pmatrix},$$

e

$$D = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 9 \end{pmatrix}, R = \begin{pmatrix} 0 & 2 & 3 \\ 0 & 0 & 6 \\ 0 & 0 & 0 \end{pmatrix}$$

Supondo $det(D) \neq 0$, pode-se escrever: $(L + D + R)x = b$ e daqui $Dx = -(L + R)x + b$. e $x = -D^{-1}(L + R)x + D^{-1}b$. Chamando $B = -D^{-1}(L + R)$ e $g = D^{-1}.b$ obtem-se a equação geral deste método que é

$$x = Bx + g$$

. Note que é uma equação recursiva (ou iterativa) já que o x aparece nos 2 lados da mesma. A idéia aqui é atribuir um valor a x e com ele calcular um novo x, que é jogado no lugar do x velho e gera um novo x e assim sucessivamente até haver a convergência esperada (de antemão). Como supusemos $det(D) \neq 0$ pode-se dividir cada equação do sistema pelo coeficiente da diagonal principal, resultando

$$A^* = L^* + I + R^*$$

Agora o processo iterativo pode ser definido como

$$x^{k+1} = -(L^* + R^*)x^k + b^*$$

onde os elementos de L^*, R^* e b^* são:

$l_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i > j$ e zero senão

$d_{ij} = a_{ij}$ se $i = j$ e zero senão

$r_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i < j$ e zero senão

$b_ij^* = \frac{b_i}{a_{ii}}$

Critérios de convergência

Depois de dividida a matriz pelo elemento da diagonal principal, deve-se ter o **critério das linhas** que diz

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}^*| < 1$$

OU então o **critério das colunas** que diz

$$max_{1 \leq j \leq n} \sum_{i=1, i \neq j}^{n} |a_{ij}^*| < 1$$

Note que qualquer um dos dois critérios sendo satisfeitos, isso garante que o sistema tem convergência.

Definindo uma matriz como estritamente diagonalmente dominante se

$$\sum_{j=1, j \neq i}^{n} |a_{ij}| < |a_{ii}|$$

e se A é estritamente diagonalmente dominante, então A satisfaz o critério das linhas, e o sistema converge. Então, um bom critério de convergência pode ser

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}|* < 1$$

Exemplo

Seja resolver o sistema abaixo, usando o método de Jacobi

$$\begin{array}{ccccccc} 10x & + & 2y & + & z & = & 7 \\ x & + & 5y & + & z & = & -8 \\ 2x & + & 3y & + & 10z & = & 6 \end{array}$$

com $x^0 = (0.7, -1.6\ e\ 0.6)^t$ e $\epsilon < 0.11$.

Convergência ?

$|a_{12}| + |a_{13}| < |a_{11}|$? ou $|2| + |1| < |10|$? R: sim
$|a_{21}| + |a_{23}| < |a_{22}|$? ou $|1| + |1| < |5|$? R: sim
$|a_{31}| + |a_{32}| < |a_{33}|$? ou $|2| + |3| < |10|$? R: sim
logo, o sistema convergirá!

Solução

Dividindo cada equação pelo elemento da diagonal principal, fica

$$\begin{array}{ccccccc} x & + & 0.2y & + & 0.1z & = & 0.7 \\ 0.2x & + & y & + & 0.2z & = & -1.6 \\ 0.2x & + & 0.3y & + & z & = & 0.6 \end{array}$$

Apenas para ilustrar o critério das linhas para testar convergência (desnecessário no caso, já que já vimos que o sistema converge, mas em tese...)
$|a_{12}^*| + |a_{13}^*| = |0.2| + |0.1| = 0.3$ e
$|a_{21}^*| + |a_{23}^*| = |0.2| + |0.2| = 0.4$ e
$|a_{31}^*| + |a_{32}^*| = |0.2| + |0.3| = 0.5$ e
$\Rightarrow\ max_{1 \leq i \leq n}(0.3, 0.4\ 0.5) = 0.5 < 1$

Iterações

$x^{k+1} = -0.2y^k - 0.1z^k + 0.7$
$y^{k+1} = -0.2x^k - 0.2z^k - 1.6$
$z^{k+1} = -0.2x^k - 0.3y^k + 0.6$
Começando com $(0.7,\ -1.6,\ 0.6)$ obtemos uma primeira aproximação que é $(0.96,\ -1.86,\ 0.94)$ e jogando estes novos valores na fórmula iterativa, obtem-se:

$x \rightarrow 0.7,\ 0.96,\ 0.978,\ 0.9994,\ 0.9979$
$y \rightarrow -1.6,\ -1.86,\ -1.98,\ -1.9888,\ -1.9996$
$z \rightarrow 0.6,\ 0.94,\ 0.966,\ 0.9984,\ 0.9968$

O critério de parada pode ser
$max(x^{k+1} - x^k) < \epsilon$. No exemplo acima $0.0108 < 0.011$. Ou seja, neste caso, a solução procurada é $x = 0.9979$, $y = -1.9996$ e $z = 0.9978$. Salta aos olhos neste caso que se a tolerância for um pouco menor, a resposta exata será encontrada que é $(1, -2, 1)$. Basta lançar este resultado no sistema original que a exatidão será confirmada.

Solução Python

```
#jacobi
def jacobi(a,b):
 import numpy
 n=len(a) #numero de linhas de a
          # len(a[0])=colunas de a
 xvelho=numpy.zeros((n),float)
 erro=numpy.zeros((n),float)
 xnovo=numpy.zeros((n),float)
 f=numpy.zeros((n,len(a[0])),float)
 d=numpy.zeros((len(a[0])),float)
 tol=0.1
 for i in range(0,n):
  for j in range(0,n):
   if i==j:
    f[i,j]=0
   else:
    f[i,j]=-a[i,j]/a[i,i]
 for i in range(0,n):
  d[i]=b[i]/a[i,i]
 erromax=tol+1
 while erromax>tol:
  for i in range(0,n):
   xnovo[i]=0
   for k in range(0,n):
    xnovo[i]=xnovo[i]+f[i,k]*xvelho[k]
  for i in range(0,n):
   xnovo[i]=xnovo[i]+d[i]
  for i in range(0,n):
   erro[i]=abs(xnovo[i]-xvelho[i])
  erromax=erro[0]
  for i in range(1,n):
   if erro[i]>erromax:
    erromax=erro[i]
  for i in range(0,n):
   xvelho[i]=xnovo[i]
 return(xnovo)
import numpy
# para usar faça:
# a=numpy.array([[10,7,3],[1,5,4],[3,5,9]],float)
# b=numpy.array([22,33,44],float)
# print(jacobi(a,b))
```

☞ Para você fazer

Resolva usando este método e anexe a listagem do programa (ou copie no verso desta folha):

$$A = \begin{pmatrix} 53 & 12 & 13 & 16 & 11 \\ 23 & 89 & 33 & 26 & 4 \\ 30 & 20 & 77 & 15 & 8 \\ 1 & 17 & 19 & 53 & 10 \\ 27 & 24 & 5 & 35 & 95 \end{pmatrix}$$

e

$$B = (1003, 2197, 1111, 1245, 2527)$$

As respostas encontradas, com $\epsilon < 0.0001$ e sugestão inicial igual à coluna B são:

x_1	x_2	x_3	x_4	x_5

111-70227 - 01/10

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:51.2
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
70234 ANAJULIA DE COUTO FLORIANO
19HEL111 - 3 apos 01/10, 50% _____ / _____ /

Sistemas Lineares - Jacobi

Este método é iterativo, ou seja, não exato, já que parte-se de um valor atribuído inicial e mediante buscas locais sucessivas, o mesmo converge para a resposta correta. A diferença entre o resultado sugerido e aquele real, é denominada erro, e pode ser tornada tão pequena quanto se queira. O método entretanto tem um senão, que é um critério de convergência bastante rígido. Se não obedecido o algoritmo não converge para a resposta correta, e portanto o método é inútil.

Considere $A.x = b$ um sistema de quações lineares de ordem n e cujo $det(A) \neq 0$. A maneira extensa de escrever tal sistema é

$$\begin{array}{ccccccc} a_{11}x_1 & + & a_{12}x_2 & +...+ & a_{1n}x_n & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & +...+ & a_{2n}x_n & = & b_2 \\ ... & & & & & & \\ a_{n1}x_1 & + & a_{n2}x_2 & +...+ & a_{nn}x_n & = & b_n \end{array}$$

A matriz A dos coeficientes, pode ser decomposta em $A = L + D + R$, onde

L é a matriz inferior de A (o resto é zero)

D é a matriz contendo apenas a diagonal principal de A, o resto é zero.

R é a matriz superior de A (o resto é zero)

Veja por exemplo, se

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \therefore L = \begin{pmatrix} 0 & 0 & 0 \\ 4 & 0 & 0 \\ 7 & 8 & 0 \end{pmatrix},$$

e

$$D = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 9 \end{pmatrix}, R = \begin{pmatrix} 0 & 2 & 3 \\ 0 & 0 & 6 \\ 0 & 0 & 0 \end{pmatrix}$$

Supondo $det(D) \neq 0$, pode-se escrever: $(L + D + R)x = b$ e daqui $Dx = -(L + R)x + b$. e $x = -D^{-1}(L + R)x + D^{-1}b$. Chamando $B = -D^{-1}(L + R)$ e $g = D^{-1}.b$ obtem-se a equação geral deste método que é

$$x = Bx + g$$

. Note que é uma equação recursiva (ou iterativa) já que o x aparece nos 2 lados da mesma. A idéia aqui é atribuir um valor a x e com ele calcular um novo x, que é jogado no lugar do x velho e gera um novo x e assim sucessivamente até haver a convergência esperada (de antemão). Como supusemos $det(D) \neq 0$ pode-se dividir cada equação do sistema pelo coeficiente da diagonal principal, resultando

$$A^* = L^* + I + R^*$$

Agora o processo iterativo pode ser definido como

$$x^{k+1} = -(L^* + R^*)x^k + b^*$$

onde os elementos de L^*, R^* e b^* são:

$l_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i > j$ e zero senão

$d_{ij} = a_{ij}$ se $i = j$ e zero senão

$r_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i < j$ e zero senão

$b_ij^* = \frac{b_i}{a_{ii}}$

Critérios de convergência

Depois de dividida a matriz pelo elemento da diagonal principal, deve-se ter o **critério das linhas** que diz

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}^*| < 1$$

OU então o **critério das colunas** que diz

$$max_{1 \leq j \leq n} \sum_{i=1, i \neq j}^{n} |a_{ij}^*| < 1$$

Note que qualquer um dos dois critérios sendo satisfeitos, isso garante que o sistema tem convergência.

Definindo uma matriz como estritamente diagonalmente dominante se

$$\sum_{j=1, j \neq i}^{n} |a_{ij}| < |a_{ii}|$$

e se A é estritamente diagonalmente dominante, então A satisfaz o critério das linhas, e o sistema converge. Então, um bom critério de convergência pode ser

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}^|* < 1$$

Exemplo

Seja resolver o sistema abaixo, usando o método de Jacobi

$$\begin{array}{ccccccc} 10x & + & 2y & + & z & = & 7 \\ x & + & 5y & + & z & = & -8 \\ 2x & + & 3y & + & 10z & = & 6 \end{array}$$

com $x^0 = (0.7, -1.6\ e\ 0.6)^t$ e $\epsilon < 0.11$.

Convergência ?

$|a_{12}| + |a_{13}| < |a_{11}|$? ou $|2| + |1| < |10|$? R: sim
$|a_{21}| + |a_{23}| < |a_{22}|$? ou $|1| + |1| < |5|$? R: sim
$|a_{31}| + |a_{32}| < |a_{33}|$? ou $|2| + |3| < |10|$? R: sim
logo, o sistema convergirá!

Solução

Dividindo cada equação pelo elemento da diagonal principal, fica

$$\begin{array}{ccccccc} x & + & 0.2y & + & 0.1z & = & 0.7 \\ 0.2x & + & y & + & 0.2z & = & -1.6 \\ 0.2x & + & 0.3y & + & z & = & 0.6 \end{array}$$

Apenas para ilustrar o critério das linhas para testar convergência (desnecessário no caso, já que já vimos que o sistema converge, mas em tese...)
$|a_{12}^*| + |a_{13}^*| = |0.2| + |0.1| = 0.3$ e
$|a_{21}^*| + |a_{23}^*| = |0.2| + |0.2| = 0.4$ e
$|a_{31}^*| + |a_{32}^*| = |0.2| + |0.3| = 0.5$ e
$\Rightarrow\ max_{1 \leq i \leq n}(0.3, 0.4\ 0.5) = 0.5 < 1$

Iterações

$x^{k+1} = -0.2y^k - 0.1z^k + 0.7$
$y^{k+1} = -0.2x^k - 0.2z^k - 1.6$
$z^{k+1} = -0.2x^k - 0.3y^k + 0.6$
Começando com $(0.7,\ -1.6,\ 0.6)$ obtemos uma primeira aproximação que é $(0.96,\ -1.86,\ 0.94)$ e jogando estes novos valores na fórmula iterativa, obtem-se:

$x \rightarrow 0.7,\ 0.96,\ 0.978,\ 0.9994,\ 0.9979$
$y \rightarrow -1.6,\ -1.86,\ -1.98,\ -1.9888,\ -1.9996$
$z \rightarrow 0.6,\ 0.94,\ 0.966,\ 0.9984,\ 0.9968$

O critério de parada pode ser
$max(x^{k+1} - x^k) < \epsilon$. No exemplo acima $0.0108 < 0.011$. Ou seja, neste caso, a solução procurada é $x = 0.9979$, $y = -1.9996$ e $z = 0.9978$. Salta aos olhos neste caso que se a tolerância for um pouco menor, a resposta exata será encontrada que é $(1, -2, 1)$. Basta lançar este resultado no sistema original que a exatidão será confirmada.

Solução Python

```
#jacobi
def jacobi(a,b):
 import numpy
 n=len(a) #numero de linhas de a
          # len(a[0])=colunas de a
 xvelho=numpy.zeros((n),float)
 erro=numpy.zeros((n),float)
 xnovo=numpy.zeros((n),float)
 f=numpy.zeros((n,len(a[0])),float)
 d=numpy.zeros((len(a[0])),float)
 tol=0.1
 for i in range(0,n):
  for j in range(0,n):
   if i==j:
    f[i,j]=0
   else:
    f[i,j]=-a[i,j]/a[i,i]
 for i in range(0,n):
  d[i]=b[i]/a[i,i]
 erromax=tol+1
 while erromax>tol:
  for i in range(0,n):
   xnovo[i]=0
   for k in range(0,n):
    xnovo[i]=xnovo[i]+f[i,k]*xvelho[k]
  for i in range(0,n):
   xnovo[i]=xnovo[i]+d[i]
  for i in range(0,n):
   erro[i]=abs(xnovo[i]-xvelho[i])
  erromax=erro[0]
  for i in range(1,n):
   if erro[i]>erromax:
    erromax=erro[i]
  for i in range(0,n):
   xvelho[i]=xnovo[i]
 return(xnovo)
import numpy
# para usar faça:
# a=numpy.array([[10,7,3],[1,5,4],[3,5,9]],float)
# b=numpy.array([22,33,44],float)
# print(jacobi(a,b))
```

☞ Para você fazer

Resolva usando este método e anexe a listagem do programa (ou copie no verso desta folha):

$$A = \begin{pmatrix} 89 & 34 & 26 & 9 & 19 \\ 10 & 37 & 4 & 3 & 13 \\ 1 & 6 & 59 & 15 & 32 \\ 31 & 25 & 33 & 105 & 12 \\ 18 & 17 & 23 & 20 & 79 \end{pmatrix}$$

e

$$B = (1930, 522, 1343, 1498, 1984)$$

As respostas encontradas, com $\epsilon < 0.0001$ e sugestão inicial igual à coluna B são:

x_1	x_2	x_3	x_4	x_5

111-70234 - 01/10

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:51.2
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
70827 ANDRE LUIS O. ANDRADE 19HEL111 - 4 apos 01/10, 50% _____ / _____ / _____

Sistemas Lineares - Jacobi

Este método é iterativo, ou seja, não exato, já que parte-se de um valor atribuído inicial e mediante buscas locais sucessivas, o mesmo converge para a resposta correta. A diferença entre o resultado sugerido e aquele real, é denominada erro, e pode ser tornada tão pequena quanto se queira. O método entretanto tem um senão, que é um critério de convergência bastante rígido. Se não obedecido o algoritmo não converge para a resposta correta, e portanto o método é inútil.

Considere $A.x = b$ um sistema de quações lineares de ordem n e cujo $det(A) \neq 0$. A maneira extensa de escrever tal sistema é

$$\begin{array}{ccccccc} a_{11}x_1 & + & a_{12}x_2 & +...+ & a_{1n}x_n & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & +...+ & a_{2n}x_n & = & b_2 \\ ... & & & & & & \\ a_{n1}x_1 & + & a_{n2}x_2 & +...+ & a_{nn}x_n & = & b_n \end{array}$$

A matriz A dos coeficientes, pode ser decomposta em $A = L + D + R$, onde

L é a matriz inferior de A (o resto é zero)

D é a matriz contendo apenas a diagonal principal de A, o resto é zero.

R é a matriz superior de A (o resto é zero)

Veja por exemplo, se

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \therefore L = \begin{pmatrix} 0 & 0 & 0 \\ 4 & 0 & 0 \\ 7 & 8 & 0 \end{pmatrix},$$

e

$$D = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 9 \end{pmatrix}, R = \begin{pmatrix} 0 & 2 & 3 \\ 0 & 0 & 6 \\ 0 & 0 & 0 \end{pmatrix}$$

Supondo $det(D) \neq 0$, pode-se escrever: $(L + D + R)x = b$ e daqui $Dx = -(L + R)x + b$. e $x = -D^{-1}(L + R)x + D^{-1}b$. Chamando $B = -D^{-1}(L + R)$ e $g = D^{-1}.b$ obtem-se a equação geral deste método que é

$$x = Bx + g$$

. Note que é uma equação recursiva (ou iterativa) já que o x aparece nos 2 lados da mesma. A idéia aqui é atribuir um valor a x e com ele calcular um novo x, que é jogado no lugar do x velho e gera um novo x e assim sucessivamente até haver a convergência esperada (de antemão). Como supusemos $det(D) \neq 0$ pode-se dividir cada equação do sistema pelo coeficiente da diagonal principal, resultando

$$A^* = L^* + I + R^*$$

Agora o processo iterativo pode ser definido como

$$x^{k+1} = -(L^* + R^*)x^k + b^*$$

onde os elementos de L^*, R^* e b^* são:

$l_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i > j$ e zero senão

$d_{ij} = a_{ij}$ se $i = j$ e zero senão

$r_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i < j$ e zero senão

$b_ij^* = \frac{b_i}{a_{ii}}$

Critérios de convergência

Depois de dividida a matriz pelo elemento da diagonal principal, deve-se ter o **critério das linhas** que diz

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}^*| < 1$$

OU então o **critério das colunas** que diz

$$max_{1 \leq j \leq n} \sum_{i=1, i \neq j}^{n} |a_{ij}^*| < 1$$

Note que qualquer um dos dois critérios sendo satisfeitos, isso garante que o sistema tem convergência.

Definindo uma matriz como estritamente diagonalmente dominante se

$$\sum_{j=1, j \neq i}^{n} |a_{ij}| < |a_{ii}|$$

e se A é estritamente diagonalmente dominante, então A satisfaz o critério das linhas, e o sistema converge. Então, um bom critério de convergência pode ser

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}|* < 1$$

Exemplo

Seja resolver o sistema abaixo, usando o método de Jacobi

$$\begin{array}{ccccccc} 10x & + & 2y & + & z & = & 7 \\ x & + & 5y & + & z & = & -8 \\ 2x & + & 3y & + & 10z & = & 6 \end{array}$$

com $x^0 = (0.7, -1.6\ e\ 0.6)^t$ e $\epsilon < 0.11$.

Convergência ?

$|a_{12}| + |a_{13}| < |a_{11}|$? ou $|2| + |1| < |10|$? R: sim
$|a_{21}| + |a_{23}| < |a_{22}|$? ou $|1| + |1| < |5|$? R: sim
$|a_{31}| + |a_{32}| < |a_{33}|$? ou $|2| + |3| < |10|$? R: sim
logo, o sistema convergirá!

Solução

Dividindo cada equação pelo elemento da diagonal principal, fica

$$\begin{array}{ccccccc} x & + & 0.2y & + & 0.1z & = & 0.7 \\ 0.2x & + & y & + & 0.2z & = & -1.6 \\ 0.2x & + & 0.3y & + & z & = & 0.6 \end{array}$$

Apenas para ilustrar o critério das linhas para testar convergência (desnecessário no caso, já que já vimos que o sistema converge, mas em tese...)
$|a_{12}^*| + |a_{13}^*| = |0.2| + |0.1| = 0.3$ e
$|a_{21}^*| + |a_{23}^*| = |0.2| + |0.2| = 0.4$ e
$|a_{31}^*| + |a_{32}^*| = |0.2| + |0.3| = 0.5$ e
$\Rightarrow\ max_{1 \leq i \leq n}(0.3, 0.4\ 0.5) = 0.5 < 1$

Iterações

$x^{k+1} = -0.2y^k - 0.1z^k + 0.7$
$y^{k+1} = -0.2x^k - 0.2z^k - 1.6$
$z^{k+1} = -0.2x^k - 0.3y^k + 0.6$
Começando com $(0.7,\ -1.6,\ 0.6)$ obtemos uma primeira aproximação que é $(0.96,\ -1.86,\ 0.94)$ e jogando estes novos valores na fórmula iterativa, obtem-se:

$x \rightarrow 0.7,\ 0.96,\ 0.978,\ 0.9994,\ 0.9979$
$y \rightarrow -1.6,\ -1.86,\ -1.98,\ -1.9888,\ -1.9996$
$z \rightarrow 0.6,\ 0.94,\ 0.966,\ 0.9984,\ 0.9968$

O critério de parada pode ser
$max(x^{k+1} - x^k) < \epsilon$. No exemplo acima $0.0108 < 0.011$. Ou seja, neste caso, a solução procurada é $x = 0.9979$, $y = -1.9996$ e $z = 0.9978$. Salta aos olhos neste caso que se a tolerância for um pouco menor, a resposta exata será encontrada que é $(1, -2, 1)$. Basta lançar este resultado no sistema original que a exatidão será confirmada.

Solução Python

```
#jacobi
def jacobi(a,b):
 import numpy
 n=len(a) #numero de linhas de a
          # len(a[0])=colunas de a
 xvelho=numpy.zeros((n),float)
 erro=numpy.zeros((n),float)
 xnovo=numpy.zeros((n),float)
 f=numpy.zeros((n,len(a[0])),float)
 d=numpy.zeros((len(a[0])),float)
 tol=0.1
 for i in range(0,n):
  for j in range(0,n):
   if i==j:
    f[i,j]=0
   else:
    f[i,j]=-a[i,j]/a[i,i]
 for i in range(0,n):
  d[i]=b[i]/a[i,i]
 erromax=tol+1
 while erromax>tol:
  for i in range(0,n):
   xnovo[i]=0
   for k in range(0,n):
    xnovo[i]=xnovo[i]+f[i,k]*xvelho[k]
  for i in range(0,n):
   xnovo[i]=xnovo[i]+d[i]
  for i in range(0,n):
   erro[i]=abs(xnovo[i]-xvelho[i])
  erromax=erro[0]
  for i in range(1,n):
   if erro[i]>erromax:
    erromax=erro[i]
  for i in range(0,n):
   xvelho[i]=xnovo[i]
 return(xnovo)
import numpy
# para usar faça:
# a=numpy.array([[10,7,3],[1,5,4],[3,5,9]],float)
# b=numpy.array([22,33,44],float)
# print(jacobi(a,b))
```

☞ Para você fazer

Resolva usando este método e anexe a listagem do programa (ou copie no verso desta folha):

$$A = \begin{pmatrix} 67 & 23 & 4 & 1 & 35 \\ 25 & 73 & 3 & 27 & 17 \\ 33 & 24 & 120 & 29 & 30 \\ 15 & 21 & 16 & 87 & 32 \\ 12 & 13 & 18 & 7 & 51 \end{pmatrix}$$

e

$$B = (1547, 1541, 3295, 2096, 1469)$$

As respostas encontradas, com $\epsilon < 0.0001$ e sugestão inicial igual à coluna B são:

x_1	x_2	x_3	x_4	x_5

111-70827 - 01/10

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:51.2
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
70834 ANDRE LUIZ DOMINGOS 19HEL111 - 5
apos 01/10, 50% _____ / _____ / _____

Sistemas Lineares - Jacobi

Este método é iterativo, ou seja, não exato, já que parte-se de um valor atribuído inicial e mediante buscas locais sucessivas, o mesmo converge para a resposta correta. A diferença entre o resultado sugerido e aquele real, é denominada erro, e pode ser tornada tão pequena quanto se queira. O método entretanto tem um senão, que é um critério de convergência bastante rígido. Se não obedecido o algoritmo não converge para a resposta correta, e portanto o método é inútil.

Considere $A.x = b$ um sistema de quações lineares de ordem n e cujo $det(A) \neq 0$. A maneira extensa de escrever tal sistema é

$$\begin{array}{ccccccc} a_{11}x_1 & + & a_{12}x_2 & +...+ & a_{1n}x_n & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & +...+ & a_{2n}x_n & = & b_2 \\ ... & & & & & & \\ a_{n1}x_1 & + & a_{n2}x_2 & +...+ & a_{nn}x_n & = & b_n \end{array}$$

A matriz A dos coeficientes, pode ser decomposta em $A = L + D + R$, onde

L é a matriz inferior de A (o resto é zero)

D é a matriz contendo apenas a diagonal principal de A, o resto é zero.

R é a matriz superior de A (o resto é zero)

Veja por exemplo, se

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \therefore L = \begin{pmatrix} 0 & 0 & 0 \\ 4 & 0 & 0 \\ 7 & 8 & 0 \end{pmatrix},$$

e

$$D = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 9 \end{pmatrix}, R = \begin{pmatrix} 0 & 2 & 3 \\ 0 & 0 & 6 \\ 0 & 0 & 0 \end{pmatrix}$$

Supondo $det(D) \neq 0$, pode-se escrever: $(L + D + R)x = b$ e daqui $Dx = -(L + R)x + b$. e $x = -D^{-1}(L + R)x + D^{-1}b$. Chamando $B = -D^{-1}(L + R)$ e $g = D^{-1}.b$ obtem-se a equação geral deste método que é

$$x = Bx + g$$

. Note que é uma equação recursiva (ou iterativa) já que o x aparece nos 2 lados da mesma. A idéia aqui é atribuir um valor a x e com ele calcular um novo x, que é jogado no lugar do x velho e gera um novo x e assim sucessivamente até haver a convergência esperada (de antemão). Como supusemos $det(D) \neq 0$ pode-se dividir cada equação do sistema pelo coeficiente da diagonal principal, resultando

$$A^* = L^* + I + R^*$$

Agora o processo iterativo pode ser definido como

$$x^{k+1} = -(L^* + R^*)x^k + b^*$$

onde os elementos de L^*, R^* e b^* são:

$l_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i > j$ e zero senão

$d_{ij} = a_{ij}$ se $i = j$ e zero senão

$r_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i < j$ e zero senão

$b_ij^* = \frac{b_i}{a_{ii}}$

Critérios de convergência

Depois de dividida a matriz pelo elemento da diagonal principal, deve-se ter o **critério das linhas** que diz

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}^*| < 1$$

OU então o **critério das colunas** que diz

$$max_{1 \leq j \leq n} \sum_{i=1, i \neq j}^{n} |a_{ij}^*| < 1$$

Note que qualquer um dos dois critérios sendo satisfeitos, isso garante que o sistema tem convergência.

Definindo uma matriz como estritamente diagonalmente dominante se

$$\sum_{j=1, j \neq i}^{n} |a_{ij}| < |a_{ii}|$$

e se A é estritamente diagonalmente dominante, então A satisfaz o critério das linhas, e o sistema converge. Então, um bom critério de convergência pode ser

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}|* < 1$$

Exemplo

Seja resolver o sistema abaixo, usando o método de Jacobi

$$\begin{array}{ccccccc} 10x & + & 2y & + & z & = & 7 \\ x & + & 5y & + & z & = & -8 \\ 2x & + & 3y & + & 10z & = & 6 \end{array}$$

com $x^0 = (0.7, -1.6\ e\ 0.6)^t$ e $\epsilon < 0.11$.

Convergência ?

$|a_{12}| + |a_{13}| < |a_{11}|$? ou $|2| + |1| < |10|$? R: sim
$|a_{21}| + |a_{23}| < |a_{22}|$? ou $|1| + |1| < |5|$? R: sim
$|a_{31}| + |a_{32}| < |a_{33}|$? ou $|2| + |3| < |10|$? R: sim
logo, o sistema convergirá!

Solução

Dividindo cada equação pelo elemento da diagonal principal, fica

$$\begin{array}{ccccccc} x & + & 0.2y & + & 0.1z & = & 0.7 \\ 0.2x & + & y & + & 0.2z & = & -1.6 \\ 0.2x & + & 0.3y & + & z & = & 0.6 \end{array}$$

Apenas para ilustrar o critério das linhas para testar convergência (desnecessário no caso, já que já vimos que o sistema converge, mas em tese...)
$|a_{12}^*| + |a_{13}^*| = |0.2| + |0.1| = 0.3$ e
$|a_{21}^*| + |a_{23}^*| = |0.2| + |0.2| = 0.4$ e
$|a_{31}^*| + |a_{32}^*| = |0.2| + |0.3| = 0.5$ e
$\Rightarrow\ max_{1 \leq i \leq n}(0.3, 0.4\ 0.5) = 0.5 < 1$

Iterações

$x^{k+1} = -0.2y^k - 0.1z^k + 0.7$
$y^{k+1} = -0.2x^k - 0.2z^k - 1.6$
$z^{k+1} = -0.2x^k - 0.3y^k + 0.6$
Começando com $(0.7,\ -1.6,\ 0.6)$ obtemos uma primeira aproximação que é $(0.96,\ -1.86,\ 0.94)$ e jogando estes novos valores na fórmula iterativa, obtem-se:

$x \rightarrow 0.7,\ 0.96,\ 0.978,\ 0.9994,\ 0.9979$
$y \rightarrow -1.6,\ -1.86,\ -1.98,\ -1.9888,\ -1.9996$
$z \rightarrow 0.6,\ 0.94,\ 0.966,\ 0.9984,\ 0.9968$

O critério de parada pode ser
$max(x^{k+1} - x^k) < \epsilon$. No exemplo acima $0.0108 < 0.011$. Ou seja, neste caso, a solução procurada é $x = 0.9979$, $y = -1.9996$ e $z = 0.9978$. Salta aos olhos neste caso que se a tolerância for um pouco menor, a resposta exata será encontrada que é $(1, -2, 1)$. Basta lançar este resultado no sistema original que a exatidão será confirmada.

Solução Python

```
#jacobi
def jacobi(a,b):
 import numpy
 n=len(a) #numero de linhas de a
          # len(a[0])=colunas de a
 xvelho=numpy.zeros((n),float)
 erro=numpy.zeros((n),float)
 xnovo=numpy.zeros((n),float)
 f=numpy.zeros((n,len(a[0])),float)
 d=numpy.zeros((len(a[0])),float)
 tol=0.1
 for i in range(0,n):
  for j in range(0,n):
   if i==j:
    f[i,j]=0
   else:
    f[i,j]=-a[i,j]/a[i,i]
 for i in range(0,n):
  d[i]=b[i]/a[i,i]
 erromax=tol+1
 while erromax>tol:
  for i in range(0,n):
   xnovo[i]=0
   for k in range(0,n):
    xnovo[i]=xnovo[i]+f[i,k]*xvelho[k]
  for i in range(0,n):
   xnovo[i]=xnovo[i]+d[i]
  for i in range(0,n):
   erro[i]=abs(xnovo[i]-xvelho[i])
  erromax=erro[0]
  for i in range(1,n):
   if erro[i]>erromax:
    erromax=erro[i]
  for i in range(0,n):
   xvelho[i]=xnovo[i]
 return(xnovo)
import numpy
# para usar faça:
# a=numpy.array([[10,7,3],[1,5,4],[3,5,9]],float)
# b=numpy.array([22,33,44],float)
# print(jacobi(a,b))
```

☞ Para você fazer

Resolva usando este método e anexe a listagem do programa (ou copie no verso desta folha):

$$A = \begin{pmatrix} 90 & 32 & 10 & 30 & 11 \\ 21 & 56 & 9 & 12 & 13 \\ 4 & 14 & 49 & 25 & 2 \\ 5 & 34 & 19 & 82 & 17 \\ 1 & 20 & 3 & 35 & 62 \end{pmatrix}$$

e

$$B = (1332, 1016, 1149, 1674, 1552)$$

As respostas encontradas, com $\epsilon < 0.0001$ e sugestão inicial igual à coluna B são:

x_1	x_2	x_3	x_4	x_5

111-70834 - 01/10

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:51.2
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
70241 BIANCA LIMA DE OLIVEIRA 19HEL111 - 6 apos 01/10, 50% _____ / _____ / _____

Sistemas Lineares - Jacobi

Este método é iterativo, ou seja, não exato, já que parte-se de um valor atribuído inicial e mediante buscas locais sucessivas, o mesmo converge para a resposta correta. A diferença entre o resultado sugerido e aquele real, é denominada erro, e pode ser tornada tão pequena quanto se queira. O método entretanto tem um senão, que é um critério de convergência bastante rígido. Se não obedecido o algoritmo não converge para a resposta correta, e portanto o método é inútil.

Considere $A.x = b$ um sistema de quações lineares de ordem n e cujo $det(A) \neq 0$. A maneira extensa de escrever tal sistema é

$$\begin{array}{ccccccc} a_{11}x_1 & + & a_{12}x_2 & +...+ & a_{1n}x_n & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & +...+ & a_{2n}x_n & = & b_2 \\ ... & & & & & & \\ a_{n1}x_1 & + & a_{n2}x_2 & +...+ & a_{nn}x_n & = & b_n \end{array}$$

A matriz A dos coeficientes, pode ser decomposta em $A = L + D + R$, onde

L é a matriz inferior de A (o resto é zero)

D é a matriz contendo apenas a diagonal principal de A, o resto é zero.

R é a matriz superior de A (o resto é zero)

Veja por exemplo, se

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \therefore L = \begin{pmatrix} 0 & 0 & 0 \\ 4 & 0 & 0 \\ 7 & 8 & 0 \end{pmatrix},$$

e

$$D = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 9 \end{pmatrix}, R = \begin{pmatrix} 0 & 2 & 3 \\ 0 & 0 & 6 \\ 0 & 0 & 0 \end{pmatrix}$$

Supondo $det(D) \neq 0$, pode-se escrever: $(L + D + R)x = b$ e daqui $Dx = -(L + R)x + b$. e $x = -D^{-1}(L + R)x + D^{-1}b$. Chamando $B = -D^{-1}(L + R)$ e $g = D^{-1}.b$ obtem-se a equação geral deste método que é

$$x = Bx + g$$

. Note que é uma equação recursiva (ou iterativa) já que o x aparece nos 2 lados da mesma. A idéia aqui é atribuir um valor a x e com ele calcular um novo x, que é jogado no lugar do x velho e gera um novo x e assim sucessivamente até haver a convergência esperada (de antemão). Como supusemos $det(D) \neq 0$ pode-se dividir cada equação do sistema pelo coeficiente da diagonal principal, resultando

$$A^* = L^* + I + R^*$$

Agora o processo iterativo pode ser definido como

$$x^{k+1} = -(L^* + R^*)x^k + b^*$$

onde os elementos de L^*, R^* e b^* são:

$l_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i > j$ e zero senão

$d_{ij} = a_{ij}$ se $i = j$ e zero senão

$r_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i < j$ e zero senão

$b_ij^* = \frac{b_i}{a_{ii}}$

Critérios de convergência

Depois de dividida a matriz pelo elemento da diagonal principal, deve-se ter o **critério das linhas** que diz

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}^*| < 1$$

OU então o **critério das colunas** que diz

$$max_{1 \leq j \leq n} \sum_{i=1, i \neq j}^{n} |a_{ij}^*| < 1$$

Note que qualquer um dos dois critérios sendo satisfeitos, isso garante que o sistema tem convergência.

Definindo uma matriz como estritamente diagonalmente dominante se

$$\sum_{j=1, j \neq i}^{n} |a_{ij}| < |a_{ii}|$$

e se A é estritamente diagonalmente dominante, então A satisfaz o critério das linhas, e o sistema converge. Então, um bom critério de convergência pode ser

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}|* < 1$$

Exemplo

Seja resolver o sistema abaixo, usando o método de Jacobi

$$\begin{array}{ccccccc} 10x & + & 2y & + & z & = & 7 \\ x & + & 5y & + & z & = & -8 \\ 2x & + & 3y & + & 10z & = & 6 \end{array}$$

com $x^0 = (0.7, -1.6\ e\ 0.6)^t$ e $\epsilon < 0.11$.

Convergência ?

$|a_{12}| + |a_{13}| < |a_{11}|$? ou $|2| + |1| < |10|$? R: sim
$|a_{21}| + |a_{23}| < |a_{22}|$? ou $|1| + |1| < |5|$? R: sim
$|a_{31}| + |a_{32}| < |a_{33}|$? ou $|2| + |3| < |10|$? R: sim
logo, o sistema convergirá!

Solução

Dividindo cada equação pelo elemento da diagonal principal, fica

$$\begin{array}{ccccccc} x & + & 0.2y & + & 0.1z & = & 0.7 \\ 0.2x & + & y & + & 0.2z & = & -1.6 \\ 0.2x & + & 0.3y & + & z & = & 0.6 \end{array}$$

Apenas para ilustrar o critério das linhas para testar convergência (desnecessário no caso, já que já vimos que o sistema converge, mas em tese...)
$|a_{12}^*| + |a_{13}^*| = |0.2| + |0.1| = 0.3$ e
$|a_{21}^*| + |a_{23}^*| = |0.2| + |0.2| = 0.4$ e
$|a_{31}^*| + |a_{32}^*| = |0.2| + |0.3| = 0.5$ e
$\Rightarrow\ max_{1 \leq i \leq n}(0.3, 0.4\ 0.5) = 0.5 < 1$

Iterações

$x^{k+1} = -0.2y^k - 0.1z^k + 0.7$
$y^{k+1} = -0.2x^k - 0.2z^k - 1.6$
$z^{k+1} = -0.2x^k - 0.3y^k + 0.6$
Começando com $(0.7,\ -1.6,\ 0.6)$ obtemos uma primeira aproximação que é $(0.96,\ -1.86,\ 0.94)$ e jogando estes novos valores na fórmula iterativa, obtem-se:

$x \rightarrow 0.7,\ 0.96,\ 0.978,\ 0.9994,\ 0.9979$
$y \rightarrow -1.6,\ -1.86,\ -1.98,\ -1.9888,\ -1.9996$
$z \rightarrow 0.6,\ 0.94,\ 0.966,\ 0.9984,\ 0.9968$

O critério de parada pode ser
$max(x^{k+1} - x^k) < \epsilon$. No exemplo acima $0.0108 < 0.011$. Ou seja, neste caso, a solução procurada é $x = 0.9979$, $y = -1.9996$ e $z = 0.9978$. Salta aos olhos neste caso que se a tolerância for um pouco menor, a resposta exata será encontrada que é $(1, -2, 1)$. Basta lançar este resultado no sistema original que a exatidão será confirmada.

Solução Python

```
#jacobi
def jacobi(a,b):
 import numpy
 n=len(a) #numero de linhas de a
          # len(a[0])=colunas de a
 xvelho=numpy.zeros((n),float)
 erro=numpy.zeros((n),float)
 xnovo=numpy.zeros((n),float)
 f=numpy.zeros((n,len(a[0])),float)
 d=numpy.zeros((len(a[0])),float)
 tol=0.1
 for i in range(0,n):
  for j in range(0,n):
   if i==j:
    f[i,j]=0
   else:
    f[i,j]=-a[i,j]/a[i,i]
 for i in range(0,n):
  d[i]=b[i]/a[i,i]
 erromax=tol+1
 while erromax>tol:
  for i in range(0,n):
   xnovo[i]=0
   for k in range(0,n):
    xnovo[i]=xnovo[i]+f[i,k]*xvelho[k]
  for i in range(0,n):
   xnovo[i]=xnovo[i]+d[i]
  for i in range(0,n):
   erro[i]=abs(xnovo[i]-xvelho[i])
  erromax=erro[0]
  for i in range(1,n):
   if erro[i]>erromax:
    erromax=erro[i]
  for i in range(0,n):
   xvelho[i]=xnovo[i]
 return(xnovo)
import numpy
# para usar faça:
# a=numpy.array([[10,7,3],[1,5,4],[3,5,9]],float)
# b=numpy.array([22,33,44],float)
# print(jacobi(a,b))
```

☞ Para você fazer

Resolva usando este método e anexe a listagem do programa (ou copie no verso desta folha):

$$A = \begin{pmatrix} 90 & 11 & 13 & 34 & 29 \\ 8 & 101 & 30 & 21 & 33 \\ 7 & 35 & 77 & 25 & 9 \\ 1 & 28 & 16 & 65 & 18 \\ 17 & 4 & 22 & 5 & 50 \end{pmatrix}$$

e

$$B = (2664, 2681, 2181, 1663, 1385)$$

As respostas encontradas, com $\epsilon < 0.0001$ e sugestão inicial igual à coluna B são:

x_1	x_2	x_3	x_4	x_5

111-70241 - 01/10

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:51.2
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
70258 BRUNO MACHADO CARNEIRO
19HEL111 - 7 apos 01/10, 50% _____ / _____ /

Sistemas Lineares - Jacobi

Este método é iterativo, ou seja, não exato, já que parte-se de um valor atribuído inicial e mediante buscas locais sucessivas, o mesmo converge para a resposta correta. A diferença entre o resultado sugerido e aquele real, é denominada erro, e pode ser tornada tão pequena quanto se queira. O método entretanto tem um senão, que é um critério de convergência bastante rígido. Se não obedecido o algoritmo não converge para a resposta correta, e portanto o método é inútil.

Considere $A.x = b$ um sistema de quações lineares de ordem n e cujo $det(A) \neq 0$. A maneira extensa de escrever tal sistema é

$$\begin{array}{ccccccc} a_{11}x_1 & + & a_{12}x_2 & +...+ & a_{1n}x_n & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & +...+ & a_{2n}x_n & = & b_2 \\ ... & & & & & & \\ a_{n1}x_1 & + & a_{n2}x_2 & +...+ & a_{nn}x_n & = & b_n \end{array}$$

A matriz A dos coeficientes, pode ser decomposta em $A = L + D + R$, onde

L é a matriz inferior de A (o resto é zero)

D é a matriz contendo apenas a diagonal principal de A, o resto é zero.

R é a matriz superior de A (o resto é zero)

Veja por exemplo, se

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \therefore L = \begin{pmatrix} 0 & 0 & 0 \\ 4 & 0 & 0 \\ 7 & 8 & 0 \end{pmatrix},$$

e

$$D = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 9 \end{pmatrix}, R = \begin{pmatrix} 0 & 2 & 3 \\ 0 & 0 & 6 \\ 0 & 0 & 0 \end{pmatrix}$$

Supondo $det(D) \neq 0$, pode-se escrever: $(L + D + R)x = b$ e daqui $Dx = -(L + R)x + b$. e $x = -D^{-1}(L + R)x + D^{-1}b$. Chamando $B = -D^{-1}(L + R)$ e $g = D^{-1}.b$ obtem-se a equação geral deste método que é

$$x = Bx + g$$

. Note que é uma equação recursiva (ou iterativa) já que o x aparece nos 2 lados da mesma. A idéia aqui é atribuir um valor a x e com ele calcular um novo x, que é jogado no lugar do x velho e gera um novo x e assim sucessivamente até haver a convergência esperada (de antemão). Como supusemos $det(D) \neq 0$ pode-se dividir cada equação do sistema pelo coeficiente da diagonal principal, resultando

$$A^* = L^* + I + R^*$$

Agora o processo iterativo pode ser definido como

$$x^{k+1} = -(L^* + R^*)x^k + b^*$$

onde os elementos de L^*, R^* e b^* são:

$l_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i > j$ e zero senão

$d_{ij} = a_{ij}$ se $i = j$ e zero senão

$r_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i < j$ e zero senão

$b_ij^* = \frac{b_i}{a_{ii}}$

Critérios de convergência

Depois de dividida a matriz pelo elemento da diagonal principal, deve-se ter o **critério das linhas** que diz

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}^*| < 1$$

OU então o **critério das colunas** que diz

$$max_{1 \leq j \leq n} \sum_{i=1, i \neq j}^{n} |a_{ij}^*| < 1$$

Note que qualquer um dos dois critérios sendo satisfeitos, isso garante que o sistema tem convergência.

Definindo uma matriz como estritamente diagonalmente dominante se

$$\sum_{j=1, j \neq i}^{n} |a_{ij}| < |a_{ii}|$$

e se A é estritamente diagonalmente dominante, então A satisfaz o critério das linhas, e o sistema converge. Então, um bom critério de convergência pode ser

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}^|* < 1$$

Exemplo

Seja resolver o sistema abaixo, usando o método de Jacobi

$$\begin{array}{ccccccc} 10x & + & 2y & + & z & = & 7 \\ x & + & 5y & + & z & = & -8 \\ 2x & + & 3y & + & 10z & = & 6 \end{array}$$

com $x^0 = (0.7, -1.6\ e\ 0.6)^t$ e $\epsilon < 0.11$.

Convergência ?

$|a_{12}| + |a_{13}| < |a_{11}|$? ou $|2| + |1| < |10|$? R: sim
$|a_{21}| + |a_{23}| < |a_{22}|$? ou $|1| + |1| < |5|$? R: sim
$|a_{31}| + |a_{32}| < |a_{33}|$? ou $|2| + |3| < |10|$? R: sim
logo, o sistema convergirá!

Solução

Dividindo cada equação pelo elemento da diagonal principal, fica

$$\begin{array}{ccccccc} x & + & 0.2y & + & 0.1z & = & 0.7 \\ 0.2x & + & y & + & 0.2z & = & -1.6 \\ 0.2x & + & 0.3y & + & z & = & 0.6 \end{array}$$

Apenas para ilustrar o critério das linhas para testar convergência (desnecessário no caso, já que já vimos que o sistema converge, mas em tese...)
$|a_{12}^*| + |a_{13}^*| = |0.2| + |0.1| = 0.3$ e
$|a_{21}^*| + |a_{23}^*| = |0.2| + |0.2| = 0.4$ e
$|a_{31}^*| + |a_{32}^*| = |0.2| + |0.3| = 0.5$ e
$\Rightarrow\ max_{1 \leq i \leq n}(0.3, 0.4\ 0.5) = 0.5 < 1$

Iterações

$x^{k+1} = -0.2y^k - 0.1z^k + 0.7$
$y^{k+1} = -0.2x^k - 0.2z^k - 1.6$
$z^{k+1} = -0.2x^k - 0.3y^k + 0.6$
Começando com $(0.7,\ -1.6,\ 0.6)$ obtemos uma primeira aproximação que é $(0.96,\ -1.86,\ 0.94)$ e jogando estes novos valores na fórmula iterativa, obtem-se:

$x \rightarrow 0.7,\ 0.96,\ 0.978,\ 0.9994,\ 0.9979$
$y \rightarrow -1.6,\ -1.86,\ -1.98,\ -1.9888,\ -1.9996$
$z \rightarrow 0.6,\ 0.94,\ 0.966,\ 0.9984,\ 0.9968$

O critério de parada pode ser
$max(x^{k+1} - x^k) < \epsilon$. No exemplo acima $0.0108 < 0.011$. Ou seja, neste caso, a solução procurada é $x = 0.9979$, $y = -1.9996$ e $z = 0.9978$. Salta aos olhos neste caso que se a tolerância for um pouco menor, a resposta exata será encontrada que é $(1, -2, 1)$. Basta lançar este resultado no sistema original que a exatidão será confirmada.

Solução Python

```
#jacobi
def jacobi(a,b):
 import numpy
 n=len(a) #numero de linhas de a
          # len(a[0])=colunas de a
 xvelho=numpy.zeros((n),float)
 erro=numpy.zeros((n),float)
 xnovo=numpy.zeros((n),float)
 f=numpy.zeros((n,len(a[0])),float)
 d=numpy.zeros((len(a[0])),float)
 tol=0.1
 for i in range(0,n):
  for j in range(0,n):
   if i==j:
    f[i,j]=0
   else:
    f[i,j]=-a[i,j]/a[i,i]
 for i in range(0,n):
  d[i]=b[i]/a[i,i]
 erromax=tol+1
 while erromax>tol:
  for i in range(0,n):
   xnovo[i]=0
   for k in range(0,n):
    xnovo[i]=xnovo[i]+f[i,k]*xvelho[k]
  for i in range(0,n):
   xnovo[i]=xnovo[i]+d[i]
  for i in range(0,n):
   erro[i]=abs(xnovo[i]-xvelho[i])
  erromax=erro[0]
  for i in range(1,n):
   if erro[i]>erromax:
    erromax=erro[i]
  for i in range(0,n):
   xvelho[i]=xnovo[i]
 return(xnovo)
import numpy
# para usar faça:
# a=numpy.array([[10,7,3],[1,5,4],[3,5,9]],float)
# b=numpy.array([22,33,44],float)
# print(jacobi(a,b))
```

☞ Para você fazer

Resolva usando este método e anexe a listagem do programa (ou copie no verso desta folha):

$$A = \begin{pmatrix} 65 & 35 & 4 & 19 & 5 \\ 7 & 74 & 15 & 18 & 26 \\ 3 & 30 & 83 & 29 & 20 \\ 27 & 11 & 1 & 59 & 13 \\ 28 & 6 & 14 & 9 & 61 \end{pmatrix}$$

e

$$B = (1056, 999, 1593, 1268, 1268)$$

As respostas encontradas, com $\epsilon < 0.0001$ e sugestão inicial igual à coluna B são:

x_1	x_2	x_3	x_4	x_5

111-70258 - 01/10

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:51.2
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
70265 CAIO HEITOR BARZ MOLINARI
19HEL111 - 8 apos 01/10, 50% _____ / _____ /

Sistemas Lineares - Jacobi

Este método é iterativo, ou seja, não exato, já que parte-se de um valor atribuído inicial e mediante buscas locais sucessivas, o mesmo converge para a resposta correta. A diferença entre o resultado sugerido e aquele real, é denominada erro, e pode ser tornada tão pequena quanto se queira. O método entretanto tem um senão, que é um critério de convergência bastante rígido. Se não obedecido o algoritmo não converge para a resposta correta, e portanto o método é inútil.

Considere $A.x = b$ um sistema de quações lineares de ordem n e cujo $det(A) \neq 0$. A maneira extensa de escrever tal sistema é

$$\begin{array}{ccccccc} a_{11}x_1 & + & a_{12}x_2 & +...+ & a_{1n}x_n & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & +...+ & a_{2n}x_n & = & b_2 \\ ... & & & & & & \\ a_{n1}x_1 & + & a_{n2}x_2 & +...+ & a_{nn}x_n & = & b_n \end{array}$$

A matriz A dos coeficientes, pode ser decomposta em $A = L + D + R$, onde

L é a matriz inferior de A (o resto é zero)

D é a matriz contendo apenas a diagonal principal de A, o resto é zero.

R é a matriz superior de A (o resto é zero)

Veja por exemplo, se

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \therefore L = \begin{pmatrix} 0 & 0 & 0 \\ 4 & 0 & 0 \\ 7 & 8 & 0 \end{pmatrix},$$

e

$$D = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 9 \end{pmatrix}, R = \begin{pmatrix} 0 & 2 & 3 \\ 0 & 0 & 6 \\ 0 & 0 & 0 \end{pmatrix}$$

Supondo $det(D) \neq 0$, pode-se escrever: $(L + D + R)x = b$ e daqui $Dx = -(L + R)x + b$. e $x = -D^{-1}(L + R)x + D^{-1}b$. Chamando $B = -D^{-1}(L + R)$ e $g = D^{-1}.b$ obtem-se a equação geral deste método que é

$$x = Bx + g$$

. Note que é uma equação recursiva (ou iterativa) já que o x aparece nos 2 lados da mesma. A idéia aqui é atribuir um valor a x e com ele calcular um novo x, que é jogado no lugar do x velho e gera um novo x e assim sucessivamente até haver a convergência esperada (de antemão). Como supusemos $det(D) \neq 0$ pode-se dividir cada equação do sistema pelo coeficiente da diagonal principal, resultando

$$A^* = L^* + I + R^*$$

Agora o processo iterativo pode ser definido como

$$x^{k+1} = -(L^* + R^*)x^k + b^*$$

onde os elementos de L^*, R^* e b^* são:

$l_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i > j$ e zero senão

$d_{ij} = a_{ij}$ se $i = j$ e zero senão

$r_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i < j$ e zero senão

$b_ij^* = \frac{b_i}{a_{ii}}$

Critérios de convergência

Depois de dividida a matriz pelo elemento da diagonal principal, deve-se ter o **critério das linhas** que diz

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}^*| < 1$$

OU então o **critério das colunas** que diz

$$max_{1 \leq j \leq n} \sum_{i=1, i \neq j}^{n} |a_{ij}^*| < 1$$

Note que qualquer um dos dois critérios sendo satisfeitos, isso garante que o sistema tem convergência.

Definindo uma matriz como estritamente diagonalmente dominante se

$$\sum_{j=1, j \neq i}^{n} |a_{ij}| < |a_{ii}|$$

e se A é estritamente diagonalmente dominante, então A satisfaz o critério das linhas, e o sistema converge. Então, um bom critério de convergência pode ser

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}^|* < 1$$

Exemplo

Seja resolver o sistema abaixo, usando o método de Jacobi

$$\begin{array}{ccccccc} 10x & + & 2y & + & z & = & 7 \\ x & + & 5y & + & z & = & -8 \\ 2x & + & 3y & + & 10z & = & 6 \end{array}$$

com $x^0 = (0.7, -1.6\ e\ 0.6)^t$ e $\epsilon < 0.11$.

Convergência ?

$|a_{12}| + |a_{13}| < |a_{11}|$? ou $|2| + |1| < |10|$? R: sim
$|a_{21}| + |a_{23}| < |a_{22}|$? ou $|1| + |1| < |5|$? R: sim
$|a_{31}| + |a_{32}| < |a_{33}|$? ou $|2| + |3| < |10|$? R: sim
logo, o sistema convergirá!

Solução

Dividindo cada equação pelo elemento da diagonal principal, fica

$$\begin{array}{ccccccc} x & + & 0.2y & + & 0.1z & = & 0.7 \\ 0.2x & + & y & + & 0.2z & = & -1.6 \\ 0.2x & + & 0.3y & + & z & = & 0.6 \end{array}$$

Apenas para ilustrar o critério das linhas para testar convergência (desnecessário no caso, já que já vimos que o sistema converge, mas em tese...)
$|a_{12}^*| + |a_{13}^*| = |0.2| + |0.1| = 0.3$ e
$|a_{21}^*| + |a_{23}^*| = |0.2| + |0.2| = 0.4$ e
$|a_{31}^*| + |a_{32}^*| = |0.2| + |0.3| = 0.5$ e
$\Rightarrow\ max_{1 \leq i \leq n}(0.3, 0.4\ 0.5) = 0.5 < 1$

Iterações

$x^{k+1} = -0.2y^k - 0.1z^k + 0.7$
$y^{k+1} = -0.2x^k - 0.2z^k - 1.6$
$z^{k+1} = -0.2x^k - 0.3y^k + 0.6$
Começando com $(0.7,\ -1.6,\ 0.6)$ obtemos uma primeira aproximação que é $(0.96,\ -1.86,\ 0.94)$ e jogando estes novos valores na fórmula iterativa, obtem-se:

$x \rightarrow 0.7,\ 0.96,\ 0.978,\ 0.9994,\ 0.9979$
$y \rightarrow -1.6,\ -1.86,\ -1.98,\ -1.9888,\ -1.9996$
$z \rightarrow 0.6,\ 0.94,\ 0.966,\ 0.9984,\ 0.9968$

O critério de parada pode ser
$max(x^{k+1} - x^k) < \epsilon$. No exemplo acima $0.0108 < 0.011$. Ou seja, neste caso, a solução procurada é $x = 0.9979$, $y = -1.9996$ e $z = 0.9978$. Salta aos olhos neste caso que se a tolerância for um pouco menor, a resposta exata será encontrada que é $(1, -2, 1)$. Basta lançar este resultado no sistema original que a exatidão será confirmada.

Solução Python

```
#jacobi
def jacobi(a,b):
 import numpy
 n=len(a) #numero de linhas de a
          # len(a[0])=colunas de a
 xvelho=numpy.zeros((n),float)
 erro=numpy.zeros((n),float)
 xnovo=numpy.zeros((n),float)
 f=numpy.zeros((n,len(a[0])),float)
 d=numpy.zeros((len(a[0])),float)
 tol=0.1
 for i in range(0,n):
  for j in range(0,n):
   if i==j:
    f[i,j]=0
   else:
    f[i,j]=-a[i,j]/a[i,i]
 for i in range(0,n):
  d[i]=b[i]/a[i,i]
 erromax=tol+1
 while erromax>tol:
  for i in range(0,n):
   xnovo[i]=0
   for k in range(0,n):
    xnovo[i]=xnovo[i]+f[i,k]*xvelho[k]
  for i in range(0,n):
   xnovo[i]=xnovo[i]+d[i]
  for i in range(0,n):
   erro[i]=abs(xnovo[i]-xvelho[i])
  erromax=erro[0]
  for i in range(1,n):
   if erro[i]>erromax:
    erromax=erro[i]
  for i in range(0,n):
   xvelho[i]=xnovo[i]
 return(xnovo)
import numpy
# para usar faça:
# a=numpy.array([[10,7,3],[1,5,4],[3,5,9]],float)
# b=numpy.array([22,33,44],float)
# print(jacobi(a,b))
```

☞ Para você fazer

Resolva usando este método e anexe a listagem do programa (ou copie no verso desta folha):

$$A = \begin{pmatrix} 62 & 18 & 10 & 25 & 1 \\ 4 & 41 & 2 & 3 & 29 \\ 24 & 22 & 78 & 16 & 14 \\ 8 & 23 & 15 & 86 & 34 \\ 26 & 9 & 30 & 35 & 107 \end{pmatrix}$$

e

$$B = (1106, 513, 1846, 1425, 2143)$$

As respostas encontradas, com $\epsilon < 0.0001$ e sugestão inicial igual à coluna B são:

x_1	x_2	x_3	x_4	x_5

111-70265 - 01/10

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:51.2
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
71156 CAIO MAGANHINI LOPES 19HEL111 - 9 apos 01/10, 50% _____ / _____ / _____

Sistemas Lineares - Jacobi

Este método é iterativo, ou seja, não exato, já que parte-se de um valor atribuído inicial e mediante buscas locais sucessivas, o mesmo converge para a resposta correta. A diferença entre o resultado sugerido e aquele real, é denominada erro, e pode ser tornada tão pequena quanto se queira. O método entretanto tem um senão, que é um critério de convergência bastante rígido. Se não obedecido o algoritmo não converge para a resposta correta, e portanto o método é inútil.

Considere $A.x = b$ um sistema de quações lineares de ordem n e cujo $det(A) \neq 0$. A maneira extensa de escrever tal sistema é

$$\begin{array}{ccccccccc} a_{11}x_1 & + & a_{12}x_2 & +...+ & a_{1n}x_n & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & +...+ & a_{2n}x_n & = & b_2 \\ ... & & & & & & \\ a_{n1}x_1 & + & a_{n2}x_2 & +...+ & a_{nn}x_n & = & b_n \end{array}$$

A matriz A dos coeficientes, pode ser decomposta em $A = L + D + R$, onde

L é a matriz inferior de A (o resto é zero)

D é a matriz contendo apenas a diagonal principal de A, o resto é zero.

R é a matriz superior de A (o resto é zero)

Veja por exemplo, se

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \therefore L = \begin{pmatrix} 0 & 0 & 0 \\ 4 & 0 & 0 \\ 7 & 8 & 0 \end{pmatrix},$$

e

$$D = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 9 \end{pmatrix}, R = \begin{pmatrix} 0 & 2 & 3 \\ 0 & 0 & 6 \\ 0 & 0 & 0 \end{pmatrix}$$

Supondo $det(D) \neq 0$, pode-se escrever: $(L + D + R)x = b$ e daqui $Dx = -(L + R)x + b$. e $x = -D^{-1}(L + R)x + D^{-1}b$. Chamando $B = -D^{-1}(L + R)$ e $g = D^{-1}.b$ obtem-se a equação geral deste método que é

$$x = Bx + g$$

. Note que é uma equação recursiva (ou iterativa) já que o x aparece nos 2 lados da mesma. A idéia aqui é atribuir um valor a x e com ele calcular um novo x, que é jogado no lugar do x velho e gera um novo x e assim sucessivamente até haver a convergência esperada (de antemão). Como supusemos $det(D) \neq 0$ pode-se dividir cada equação do sistema pelo coeficiente da diagonal principal, resultando

$$A^* = L^* + I + R^*$$

Agora o processo iterativo pode ser definido como

$$x^{k+1} = -(L^* + R^*)x^k + b^*$$

onde os elementos de L^*, R^* e b^* são:

$l_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i > j$ e zero senão

$d_{ij} = a_{ij}$ se $i = j$ e zero senão

$r_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i < j$ e zero senão

$b_ij^* = \frac{b_i}{a_{ii}}$

Critérios de convergência

Depois de dividida a matriz pelo elemento da diagonal principal, deve-se ter o **critério das linhas** que diz

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}^*| < 1$$

OU então o **critério das colunas** que diz

$$max_{1 \leq j \leq n} \sum_{i=1, i \neq j}^{n} |a_{ij}^*| < 1$$

Note que qualquer um dos dois critérios sendo satisfeitos, isso garante que o sistema tem convergência.

Definindo uma matriz como estritamente diagonalmente dominante se

$$\sum_{j=1, j \neq i}^{n} |a_{ij}| < |a_{ii}|$$

e se A é estritamente diagonalmente dominante, então A satisfaz o critério das linhas, e o sistema converge. Então, um bom critério de convergência pode ser

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}|* < 1$$

Exemplo

Seja resolver o sistema abaixo, usando o método de Jacobi

$$\begin{array}{ccccccc} 10x & + & 2y & + & z & = & 7 \\ x & + & 5y & + & z & = & -8 \\ 2x & + & 3y & + & 10z & = & 6 \end{array}$$

com $x^0 = (0.7, -1.6\ e\ 0.6)^t$ e $\epsilon < 0.11$.

Convergência ?

$|a_{12}| + |a_{13}| < |a_{11}|$? ou $|2| + |1| < |10|$? R: sim
$|a_{21}| + |a_{23}| < |a_{22}|$? ou $|1| + |1| < |5|$? R: sim
$|a_{31}| + |a_{32}| < |a_{33}|$? ou $|2| + |3| < |10|$? R: sim
logo, o sistema convergirá!

Solução

Dividindo cada equação pelo elemento da diagonal principal, fica

$$\begin{array}{ccccccc} x & + & 0.2y & + & 0.1z & = & 0.7 \\ 0.2x & + & y & + & 0.2z & = & -1.6 \\ 0.2x & + & 0.3y & + & z & = & 0.6 \end{array}$$

Apenas para ilustrar o critério das linhas para testar convergência (desnecessário no caso, já que já vimos que o sistema converge, mas em tese...)
$|a_{12}^*| + |a_{13}^*| = |0.2| + |0.1| = 0.3$ e
$|a_{21}^*| + |a_{23}^*| = |0.2| + |0.2| = 0.4$ e
$|a_{31}^*| + |a_{32}^*| = |0.2| + |0.3| = 0.5$ e
$\Rightarrow\ max_{1 \leq i \leq n}(0.3, 0.4\ 0.5) = 0.5 < 1$

Iterações

$x^{k+1} = -0.2y^k - 0.1z^k + 0.7$
$y^{k+1} = -0.2x^k - 0.2z^k - 1.6$
$z^{k+1} = -0.2x^k - 0.3y^k + 0.6$
Começando com $(0.7,\ -1.6,\ 0.6)$ obtemos uma primeira aproximação que é $(0.96,\ -1.86,\ 0.94)$ e jogando estes novos valores na fórmula iterativa, obtem-se:

$x \rightarrow 0.7,\ 0.96,\ 0.978,\ 0.9994,\ 0.9979$
$y \rightarrow -1.6,\ -1.86,\ -1.98,\ -1.9888,\ -1.9996$
$z \rightarrow 0.6,\ 0.94,\ 0.966,\ 0.9984,\ 0.9968$

O critério de parada pode ser
$max(x^{k+1} - x^k) < \epsilon$. No exemplo acima $0.0108 < 0.011$. Ou seja, neste caso, a solução procurada é $x = 0.9979$, $y = -1.9996$ e $z = 0.9978$. Salta aos olhos neste caso que se a tolerância for um pouco menor, a resposta exata será encontrada que é $(1, -2, 1)$. Basta lançar este resultado no sistema original que a exatidão será confirmada.

Solução Python

```
#jacobi
def jacobi(a,b):
 import numpy
 n=len(a) #numero de linhas de a
          # len(a[0])=colunas de a
 xvelho=numpy.zeros((n),float)
 erro=numpy.zeros((n),float)
 xnovo=numpy.zeros((n),float)
 f=numpy.zeros((n,len(a[0])),float)
 d=numpy.zeros((len(a[0])),float)
 tol=0.1
 for i in range(0,n):
  for j in range(0,n):
   if i==j:
    f[i,j]=0
   else:
    f[i,j]=-a[i,j]/a[i,i]
 for i in range(0,n):
  d[i]=b[i]/a[i,i]
 erromax=tol+1
 while erromax>tol:
  for i in range(0,n):
   xnovo[i]=0
   for k in range(0,n):
    xnovo[i]=xnovo[i]+f[i,k]*xvelho[k]
  for i in range(0,n):
   xnovo[i]=xnovo[i]+d[i]
  for i in range(0,n):
   erro[i]=abs(xnovo[i]-xvelho[i])
  erromax=erro[0]
  for i in range(1,n):
   if erro[i]>erromax:
    erromax=erro[i]
  for i in range(0,n):
   xvelho[i]=xnovo[i]
 return(xnovo)
import numpy
# para usar faça:
# a=numpy.array([[10,7,3],[1,5,4],[3,5,9]],float)
# b=numpy.array([22,33,44],float)
# print(jacobi(a,b))
```

☞ Para você fazer

Resolva usando este método e anexe a listagem do programa (ou copie no verso desta folha):

$$A = \begin{pmatrix} 59 & 20 & 5 & 12 & 16 \\ 22 & 93 & 24 & 13 & 30 \\ 6 & 26 & 61 & 9 & 15 \\ 8 & 35 & 31 & 81 & 4 \\ 17 & 34 & 28 & 32 & 112 \end{pmatrix}$$

e

$$B = (998, 2477, 1630, 2108, 2475)$$

As respostas encontradas, com $\epsilon < 0.0001$ e sugestão inicial igual à coluna B são:

x_1	x_2	x_3	x_4	x_5

111-71156 - 01/10

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:51.2
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
70272 DANIEL SANTOS DA SILVA 19HEL111 - 10 apos 01/10, 50% _____ / _____ / _____

Sistemas Lineares - Jacobi

Este método é iterativo, ou seja, não exato, já que parte-se de um valor atribuído inicial e mediante buscas locais sucessivas, o mesmo converge para a resposta correta. A diferença entre o resultado sugerido e aquele real, é denominada erro, e pode ser tornada tão pequena quanto se queira. O método entretanto tem um senão, que é um critério de convergência bastante rígido. Se não obedecido o algoritmo não converge para a resposta correta, e portanto o método é inútil.

Considere $A.x = b$ um sistema de quações lineares de ordem n e cujo $det(A) \neq 0$. A maneira extensa de escrever tal sistema é

$$\begin{array}{ccccccc} a_{11}x_1 & + & a_{12}x_2 & +...+ & a_{1n}x_n & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & +...+ & a_{2n}x_n & = & b_2 \\ ... & & & & & & \\ a_{n1}x_1 & + & a_{n2}x_2 & +...+ & a_{nn}x_n & = & b_n \end{array}$$

A matriz A dos coeficientes, pode ser decomposta em $A = L + D + R$, onde

L é a matriz inferior de A (o resto é zero)

D é a matriz contendo apenas a diagonal principal de A, o resto é zero.

R é a matriz superior de A (o resto é zero)

Veja por exemplo, se

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \therefore L = \begin{pmatrix} 0 & 0 & 0 \\ 4 & 0 & 0 \\ 7 & 8 & 0 \end{pmatrix},$$

e

$$D = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 9 \end{pmatrix}, R = \begin{pmatrix} 0 & 2 & 3 \\ 0 & 0 & 6 \\ 0 & 0 & 0 \end{pmatrix}$$

Supondo $det(D) \neq 0$, pode-se escrever: $(L + D + R)x = b$ e daqui $Dx = -(L + R)x + b$. e $x = -D^{-1}(L + R)x + D^{-1}b$. Chamando $B = -D^{-1}(L + R)$ e $g = D^{-1}.b$ obtem-se a equação geral deste método que é

$$x = Bx + g$$

. Note que é uma equação recursiva (ou iterativa) já que o x aparece nos 2 lados da mesma. A idéia aqui é atribuir um valor a x e com ele calcular um novo x, que é jogado no lugar do x velho e gera um novo x e assim sucessivamente até haver a convergência esperada (de antemão). Como supusemos $det(D) \neq 0$ pode-se dividir cada equação do sistema pelo coeficiente da diagonal principal, resultando

$$A^* = L^* + I + R^*$$

Agora o processo iterativo pode ser definido como

$$x^{k+1} = -(L^* + R^*)x^k + b^*$$

onde os elementos de L^*, R^* e b^* são:

$l_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i > j$ e zero senão

$d_{ij} = a_{ij}$ se $i = j$ e zero senão

$r_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i < j$ e zero senão

$b_ij^* = \frac{b_i}{a_{ii}}$

Critérios de convergência

Depois de dividida a matriz pelo elemento da diagonal principal, deve-se ter o **critério das linhas** que diz

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}^*| < 1$$

OU então o **critério das colunas** que diz

$$max_{1 \leq j \leq n} \sum_{i=1, i \neq j}^{n} |a_{ij}^*| < 1$$

Note que qualquer um dos dois critérios sendo satisfeitos, isso garante que o sistema tem convergência.

Definindo uma matriz como estritamente diagonalmente dominante se

$$\sum_{j=1, j \neq i}^{n} |a_{ij}| < |a_{ii}|$$

e se A é estritamente diagonalmente dominante, então A satisfaz o critério das linhas, e o sistema converge. Então, um bom critério de convergência pode ser

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}|* < 1$$

Exemplo

Seja resolver o sistema abaixo, usando o método de Jacobi

$$\begin{array}{ccccccc} 10x & + & 2y & + & z & = & 7 \\ x & + & 5y & + & z & = & -8 \\ 2x & + & 3y & + & 10z & = & 6 \end{array}$$

com $x^0 = (0.7, -1.6\ e\ 0.6)^t$ e $\epsilon < 0.11$.

Convergência ?

$|a_{12}| + |a_{13}| < |a_{11}|$? ou $|2| + |1| < |10|$? R: sim
$|a_{21}| + |a_{23}| < |a_{22}|$? ou $|1| + |1| < |5|$? R: sim
$|a_{31}| + |a_{32}| < |a_{33}|$? ou $|2| + |3| < |10|$? R: sim
logo, o sistema convergirá!

Solução

Dividindo cada equação pelo elemento da diagonal principal, fica

$$\begin{array}{ccccccc} x & + & 0.2y & + & 0.1z & = & 0.7 \\ 0.2x & + & y & + & 0.2z & = & -1.6 \\ 0.2x & + & 0.3y & + & z & = & 0.6 \end{array}$$

Apenas para ilustrar o critério das linhas para testar convergência (desnecessário no caso, já que já vimos que o sistema converge, mas em tese...)
$|a_{12}^*| + |a_{13}^*| = |0.2| + |0.1| = 0.3$ e
$|a_{21}^*| + |a_{23}^*| = |0.2| + |0.2| = 0.4$ e
$|a_{31}^*| + |a_{32}^*| = |0.2| + |0.3| = 0.5$ e
$\Rightarrow\ max_{1 \leq i \leq n}(0.3, 0.4\ 0.5) = 0.5 < 1$

Iterações

$x^{k+1} = -0.2y^k - 0.1z^k + 0.7$
$y^{k+1} = -0.2x^k - 0.2z^k - 1.6$
$z^{k+1} = -0.2x^k - 0.3y^k + 0.6$
Começando com $(0.7,\ -1.6,\ 0.6)$ obtemos uma primeira aproximação que é $(0.96,\ -1.86,\ 0.94)$ e jogando estes novos valores na fórmula iterativa, obtem-se:

$x \rightarrow 0.7,\ 0.96,\ 0.978,\ 0.9994,\ 0.9979$
$y \rightarrow -1.6,\ -1.86,\ -1.98,\ -1.9888,\ -1.9996$
$z \rightarrow 0.6,\ 0.94,\ 0.966,\ 0.9984,\ 0.9968$

O critério de parada pode ser
$max(x^{k+1} - x^k) < \epsilon$. No exemplo acima $0.0108 < 0.011$. Ou seja, neste caso, a solução procurada é $x = 0.9979$, $y = -1.9996$ e $z = 0.9978$. Salta aos olhos neste caso que se a tolerância for um pouco menor, a resposta exata será encontrada que é $(1, -2, 1)$. Basta lançar este resultado no sistema original que a exatidão será confirmada.

Solução Python

```
#jacobi
def jacobi(a,b):
 import numpy
 n=len(a) #numero de linhas de a
          # len(a[0])=colunas de a
 xvelho=numpy.zeros((n),float)
 erro=numpy.zeros((n),float)
 xnovo=numpy.zeros((n),float)
 f=numpy.zeros((n,len(a[0])),float)
 d=numpy.zeros((len(a[0])),float)
 tol=0.1
 for i in range(0,n):
  for j in range(0,n):
   if i==j:
    f[i,j]=0
   else:
    f[i,j]=-a[i,j]/a[i,i]
 for i in range(0,n):
  d[i]=b[i]/a[i,i]
 erromax=tol+1
 while erromax>tol:
  for i in range(0,n):
   xnovo[i]=0
   for k in range(0,n):
    xnovo[i]=xnovo[i]+f[i,k]*xvelho[k]
  for i in range(0,n):
   xnovo[i]=xnovo[i]+d[i]
  for i in range(0,n):
   erro[i]=abs(xnovo[i]-xvelho[i])
  erromax=erro[0]
  for i in range(1,n):
   if erro[i]>erromax:
    erromax=erro[i]
  for i in range(0,n):
   xvelho[i]=xnovo[i]
 return(xnovo)
import numpy
# para usar faça:
# a=numpy.array([[10,7,3],[1,5,4],[3,5,9]],float)
# b=numpy.array([22,33,44],float)
# print(jacobi(a,b))
```

☞ Para você fazer

Resolva usando este método e anexe a listagem do programa (ou copie no verso desta folha):

$$A = \begin{pmatrix} 106 & 31 & 29 & 14 & 30 \\ 34 & 85 & 17 & 21 & 6 \\ 5 & 22 & 75 & 26 & 16 \\ 18 & 3 & 1 & 55 & 27 \\ 35 & 4 & 8 & 24 & 72 \end{pmatrix}$$

e

$$B = (1899, 1464, 1092, 510, 857)$$

As respostas encontradas, com $\epsilon < 0.0001$ e sugestão inicial igual à coluna B são:

x_1	x_2	x_3	x_4	x_5

111-70272 - 01/10

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:51.2
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
70289 ELEN DE JESUS SANTANA 19HEL111 - 11 apos 01/10, 50% _____ / _____ / _____

Sistemas Lineares - Jacobi

Este método é iterativo, ou seja, não exato, já que parte-se de um valor atribuído inicial e mediante buscas locais sucessivas, o mesmo converge para a resposta correta. A diferença entre o resultado sugerido e aquele real, é denominada erro, e pode ser tornada tão pequena quanto se queira. O método entretanto tem um senão, que é um critério de convergência bastante rígido. Se não obedecido o algoritmo não converge para a resposta correta, e portanto o método é inútil.

Considere $A.x = b$ um sistema de quações lineares de ordem n e cujo $det(A) \neq 0$. A maneira extensa de escrever tal sistema é

$$\begin{array}{lllllll} a_{11}x_1 & + & a_{12}x_2 & +...+ & a_{1n}x_n & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & +...+ & a_{2n}x_n & = & b_2 \\ ... & & & & & & \\ a_{n1}x_1 & + & a_{n2}x_2 & +...+ & a_{nn}x_n & = & b_n \end{array}$$

A matriz A dos coeficientes, pode ser decomposta em $A = L + D + R$, onde

L é a matriz inferior de A (o resto é zero)

D é a matriz contendo apenas a diagonal principal de A, o resto é zero.

R é a matriz superior de A (o resto é zero)

Veja por exemplo, se

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \therefore L = \begin{pmatrix} 0 & 0 & 0 \\ 4 & 0 & 0 \\ 7 & 8 & 0 \end{pmatrix},$$

e

$$D = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 9 \end{pmatrix}, R = \begin{pmatrix} 0 & 2 & 3 \\ 0 & 0 & 6 \\ 0 & 0 & 0 \end{pmatrix}$$

Supondo $det(D) \neq 0$, pode-se escrever: $(L + D + R)x = b$ e daqui $Dx = -(L + R)x + b$. e $x = -D^{-1}(L + R)x + D^{-1}b$. Chamando $B = -D^{-1}(L + R)$ e $g = D^{-1}.b$ obtem-se a equação geral deste método que é

$$x = Bx + g$$

. Note que é uma equação recursiva (ou iterativa) já que o x aparece nos 2 lados da mesma. A idéia aqui é atribuir um valor a x e com ele calcular um novo x, que é jogado no lugar do x velho e gera um novo x e assim sucessivamente até haver a convergência esperada (de antemão). Como supusemos $det(D) \neq 0$ pode-se dividir cada equação do sistema pelo coeficiente da diagonal principal, resultando

$$A^* = L^* + I + R^*$$

Agora o processo iterativo pode ser definido como

$$x^{k+1} = -(L^* + R^*)x^k + b^*$$

onde os elementos de L^*, R^* e b^* são:

$l_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i > j$ e zero senão

$d_{ij} = a_{ij}$ se $i = j$ e zero senão

$r_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i < j$ e zero senão

$b_ij^* = \frac{b_i}{a_{ii}}$

Critérios de convergência

Depois de dividida a matriz pelo elemento da diagonal principal, deve-se ter o **critério das linhas** que diz

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}^*| < 1$$

OU então o **critério das colunas** que diz

$$max_{1 \leq j \leq n} \sum_{i=1, i \neq j}^{n} |a_{ij}^*| < 1$$

Note que qualquer um dos dois critérios sendo satisfeitos, isso garante que o sistema tem convergência.

Definindo uma matriz como estritamente diagonalmente dominante se

$$\sum_{j=1, j \neq i}^{n} |a_{ij}| < |a_{ii}|$$

e se A é estritamente diagonalmente dominante, então A satisfaz o critério das linhas, e o sistema converge. Então, um bom critério de convergência pode ser

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}|* < 1$$

Exemplo

Seja resolver o sistema abaixo, usando o método de Jacobi

$$\begin{array}{lllllll} 10x & + & 2y & + & z & = & 7 \\ x & + & 5y & + & z & = & -8 \\ 2x & + & 3y & + & 10z & = & 6 \end{array}$$

com $x^0 = (0.7, -1.6\ e\ 0.6)^t$ e $\epsilon < 0.11$.

Convergência ?

$|a_{12}| + |a_{13}| < |a_{11}|$? ou $|2| + |1| < |10|$? R: sim
$|a_{21}| + |a_{23}| < |a_{22}|$? ou $|1| + |1| < |5|$? R: sim
$|a_{31}| + |a_{32}| < |a_{33}|$? ou $|2| + |3| < |10|$? R: sim
logo, o sistema convergirá!

Solução

Dividindo cada equação pelo elemento da diagonal principal, fica

$$\begin{array}{lllllll} x & + & 0.2y & + & 0.1z & = & 0.7 \\ 0.2x & + & y & + & 0.2z & = & -1.6 \\ 0.2x & + & 0.3y & + & z & = & 0.6 \end{array}$$

Apenas para ilustrar o critério das linhas para testar convergência (desnecessário no caso, já que já vimos que o sistema converge, mas em tese...)
$|a_{12}^*| + |a_{13}^*| = |0.2| + |0.1| = 0.3$ e
$|a_{21}^*| + |a_{23}^*| = |0.2| + |0.2| = 0.4$ e
$|a_{31}^*| + |a_{32}^*| = |0.2| + |0.3| = 0.5$ e
$\Rightarrow\ max_{1 \leq i \leq n}(0.3, 0.4\ 0.5) = 0.5 < 1$

Iterações

$x^{k+1} = -0.2y^k - 0.1z^k + 0.7$
$y^{k+1} = -0.2x^k - 0.2z^k - 1.6$
$z^{k+1} = -0.2x^k - 0.3y^k + 0.6$
Começando com $(0.7,\ -1.6,\ 0.6)$ obtemos uma primeira aproximação que é $(0.96,\ -1.86,\ 0.94)$ e jogando estes novos valores na fórmula iterativa, obtem-se:

$x \rightarrow 0.7,\ 0.96,\ 0.978,\ 0.9994,\ 0.9979$
$y \rightarrow -1.6,\ -1.86,\ -1.98,\ -1.9888,\ -1.9996$
$z \rightarrow 0.6,\ 0.94,\ 0.966,\ 0.9984,\ 0.9968$

O critério de parada pode ser
$max(x^{k+1} - x^k) < \epsilon$. No exemplo acima $0.0108 < 0.011$. Ou seja, neste caso, a solução procurada é $x = 0.9979$, $y = -1.9996$ e $z = 0.9978$. Salta aos olhos neste caso que se a tolerância for um pouco menor, a resposta exata será encontrada que é $(1, -2, 1)$. Basta lançar este resultado no sistema original que a exatidão será confirmada.

Solução Python

```
#jacobi
def jacobi(a,b):
 import numpy
 n=len(a) #numero de linhas de a
          # len(a[0])=colunas de a
 xvelho=numpy.zeros((n),float)
 erro=numpy.zeros((n),float)
 xnovo=numpy.zeros((n),float)
 f=numpy.zeros((n,len(a[0])),float)
 d=numpy.zeros((len(a[0])),float)
 tol=0.1
 for i in range(0,n):
  for j in range(0,n):
   if i==j:
    f[i,j]=0
   else:
    f[i,j]=-a[i,j]/a[i,i]
 for i in range(0,n):
  d[i]=b[i]/a[i,i]
 erromax=tol+1
 while erromax>tol:
  for i in range(0,n):
   xnovo[i]=0
   for k in range(0,n):
    xnovo[i]=xnovo[i]+f[i,k]*xvelho[k]
  for i in range(0,n):
   xnovo[i]=xnovo[i]+d[i]
  for i in range(0,n):
   erro[i]=abs(xnovo[i]-xvelho[i])
  erromax=erro[0]
  for i in range(1,n):
   if erro[i]>erromax:
    erromax=erro[i]
  for i in range(0,n):
   xvelho[i]=xnovo[i]
 return(xnovo)
import numpy
# para usar faça:
# a=numpy.array([[10,7,3],[1,5,4],[3,5,9]],float)
# b=numpy.array([22,33,44],float)
# print(jacobi(a,b))
```

☞ Para você fazer

Resolva usando este método e anexe a listagem do programa (ou copie no verso desta folha):

$$A = \begin{pmatrix} 85 & 29 & 17 & 3 & 35 \\ 20 & 50 & 12 & 5 & 10 \\ 22 & 21 & 83 & 32 & 4 \\ 7 & 13 & 18 & 43 & 1 \\ 9 & 34 & 33 & 24 & 101 \end{pmatrix}$$

e

$$B = (1102, 855, 2076, 1147, 2044)$$

As respostas encontradas, com $\epsilon < 0.0001$ e sugestão inicial igual à coluna B são:

x_1	x_2	x_3	x_4	x_5

111-70289 - 01/10

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:51.2
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
70841 FELIPE NAVES MENDONCA 19HEL111 - 12 apos 01/10, 50% _____ / _____ / _____

Sistemas Lineares - Jacobi

Este método é iterativo, ou seja, não exato, já que parte-se de um valor atribuído inicial e mediante buscas locais sucessivas, o mesmo converge para a resposta correta. A diferença entre o resultado sugerido e aquele real, é denominada erro, e pode ser tornada tão pequena quanto se queira. O método entretanto tem um senão, que é um critério de convergência bastante rígido. Se não obedecido o algoritmo não converge para a resposta correta, e portanto o método é inútil.

Considere $A.x = b$ um sistema de quações lineares de ordem n e cujo $det(A) \neq 0$. A maneira extensa de escrever tal sistema é

$$\begin{array}{ccccccc} a_{11}x_1 & + & a_{12}x_2 & +...+ & a_{1n}x_n & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & +...+ & a_{2n}x_n & = & b_2 \\ ... & & & & & & \\ a_{n1}x_1 & + & a_{n2}x_2 & +...+ & a_{nn}x_n & = & b_n \end{array}$$

A matriz A dos coeficientes, pode ser decomposta em $A = L + D + R$, onde

L é a matriz inferior de A (o resto é zero)

D é a matriz contendo apenas a diagonal principal de A, o resto é zero.

R é a matriz superior de A (o resto é zero)

Veja por exemplo, se

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \therefore L = \begin{pmatrix} 0 & 0 & 0 \\ 4 & 0 & 0 \\ 7 & 8 & 0 \end{pmatrix},$$

e

$$D = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 9 \end{pmatrix}, R = \begin{pmatrix} 0 & 2 & 3 \\ 0 & 0 & 6 \\ 0 & 0 & 0 \end{pmatrix}$$

Supondo $det(D) \neq 0$, pode-se escrever: $(L + D + R)x = b$ e daqui $Dx = -(L + R)x + b$. e $x = -D^{-1}(L + R)x + D^{-1}b$. Chamando $B = -D^{-1}(L + R)$ e $g = D^{-1}.b$ obtem-se a equação geral deste método que é

$$x = Bx + g$$

. Note que é uma equação recursiva (ou iterativa) já que o x aparece nos 2 lados da mesma. A idéia aqui é atribuir um valor a x e com ele calcular um novo x, que é jogado no lugar do x velho e gera um novo x e assim sucessivamente até haver a convergência esperada (de antemão). Como supusemos $det(D) \neq 0$ pode-se dividir cada equação do sistema pelo coeficiente da diagonal principal, resultando

$$A^* = L^* + I + R^*$$

Agora o processo iterativo pode ser definido como

$$x^{k+1} = -(L^* + R^*)x^k + b^*$$

onde os elementos de L^*, R^* e b^* são:

$l_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i > j$ e zero senão

$d_{ij} = a_{ij}$ se $i = j$ e zero senão

$r_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i < j$ e zero senão

$b_ij^* = \frac{b_i}{a_{ii}}$

Critérios de convergência

Depois de dividida a matriz pelo elemento da diagonal principal, deve-se ter o **critério das linhas** que diz

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}^*| < 1$$

OU então o **critério das colunas** que diz

$$max_{1 \leq j \leq n} \sum_{i=1, i \neq j}^{n} |a_{ij}^*| < 1$$

Note que qualquer um dos dois critérios sendo satisfeitos, isso garante que o sistema tem convergência.

Definindo uma matriz como estritamente diagonalmente dominante se

$$\sum_{j=1, j \neq i}^{n} |a_{ij}| < |a_{ii}|$$

e se A é estritamente diagonalmente dominante, então A satisfaz o critério das linhas, e o sistema converge. Então, um bom critério de convergência pode ser

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}|* < 1$$

Exemplo

Seja resolver o sistema abaixo, usando o método de Jacobi

$$\begin{array}{ccccccc} 10x & + & 2y & + & z & = & 7 \\ x & + & 5y & + & z & = & -8 \\ 2x & + & 3y & + & 10z & = & 6 \end{array}$$

com $x^0 = (0.7, -1.6\ e\ 0.6)^t$ e $\epsilon < 0.11$.

Convergência ?

$|a_{12}| + |a_{13}| < |a_{11}|$? ou $|2| + |1| < |10|$? R: sim
$|a_{21}| + |a_{23}| < |a_{22}|$? ou $|1| + |1| < |5|$? R: sim
$|a_{31}| + |a_{32}| < |a_{33}|$? ou $|2| + |3| < |10|$? R: sim
logo, o sistema convergirá!

Solução

Dividindo cada equação pelo elemento da diagonal principal, fica

$$\begin{array}{ccccccc} x & + & 0.2y & + & 0.1z & = & 0.7 \\ 0.2x & + & y & + & 0.2z & = & -1.6 \\ 0.2x & + & 0.3y & + & z & = & 0.6 \end{array}$$

Apenas para ilustrar o critério das linhas para testar convergência (desnecessário no caso, já que já vimos que o sistema converge, mas em tese...)
$|a_{12}^*| + |a_{13}^*| = |0.2| + |0.1| = 0.3$ e
$|a_{21}^*| + |a_{23}^*| = |0.2| + |0.2| = 0.4$ e
$|a_{31}^*| + |a_{32}^*| = |0.2| + |0.3| = 0.5$ e
$\Rightarrow\ max_{1 \leq i \leq n}(0.3, 0.4\ 0.5) = 0.5 < 1$

Iterações

$x^{k+1} = -0.2y^k - 0.1z^k + 0.7$
$y^{k+1} = -0.2x^k - 0.2z^k - 1.6$
$z^{k+1} = -0.2x^k - 0.3y^k + 0.6$
Começando com $(0.7,\ -1.6,\ 0.6)$ obtemos uma primeira aproximação que é $(0.96,\ -1.86,\ 0.94)$ e jogando estes novos valores na fórmula iterativa, obtem-se:

$x \rightarrow 0.7,\ 0.96,\ 0.978,\ 0.9994,\ 0.9979$
$y \rightarrow -1.6,\ -1.86,\ -1.98,\ -1.9888,\ -1.9996$
$z \rightarrow 0.6,\ 0.94,\ 0.966,\ 0.9984,\ 0.9968$

O critério de parada pode ser
$max(x^{k+1} - x^k) < \epsilon$. No exemplo acima $0.0108 < 0.011$. Ou seja, neste caso, a solução procurada é $x = 0.9979$, $y = -1.9996$ e $z = 0.9978$. Salta aos olhos neste caso que se a tolerância for um pouco menor, a resposta exata será encontrada que é $(1, -2, 1)$. Basta lançar este resultado no sistema original que a exatidão será confirmada.

Solução Python

```
#jacobi
def jacobi(a,b):
 import numpy
 n=len(a) #numero de linhas de a
          # len(a[0])=colunas de a
 xvelho=numpy.zeros((n),float)
 erro=numpy.zeros((n),float)
 xnovo=numpy.zeros((n),float)
 f=numpy.zeros((n,len(a[0])),float)
 d=numpy.zeros((len(a[0])),float)
 tol=0.1
 for i in range(0,n):
  for j in range(0,n):
   if i==j:
    f[i,j]=0
   else:
    f[i,j]=-a[i,j]/a[i,i]
 for i in range(0,n):
  d[i]=b[i]/a[i,i]
 erromax=tol+1
 while erromax>tol:
  for i in range(0,n):
   xnovo[i]=0
   for k in range(0,n):
    xnovo[i]=xnovo[i]+f[i,k]*xvelho[k]
  for i in range(0,n):
   xnovo[i]=xnovo[i]+d[i]
  for i in range(0,n):
   erro[i]=abs(xnovo[i]-xvelho[i])
  erromax=erro[0]
  for i in range(1,n):
   if erro[i]>erromax:
    erromax=erro[i]
  for i in range(0,n):
   xvelho[i]=xnovo[i]
 return(xnovo)
import numpy
# para usar faça:
# a=numpy.array([[10,7,3],[1,5,4],[3,5,9]],float)
# b=numpy.array([22,33,44],float)
# print(jacobi(a,b))
```

☞ Para você fazer

Resolva usando este método e anexe a listagem do programa (ou copie no verso desta folha):

$$A = \begin{pmatrix} 66 & 5 & 9 & 25 & 21 \\ 4 & 77 & 11 & 31 & 29 \\ 17 & 18 & 90 & 30 & 24 \\ 1 & 12 & 14 & 48 & 20 \\ 27 & 19 & 3 & 2 & 54 \end{pmatrix}$$

e

$$B = (1828, 1422, 2163, 1313, 1304)$$

As respostas encontradas, com $\epsilon < 0.0001$ e sugestão inicial igual à coluna B são:

x_1	x_2	x_3	x_4	x_5

111-70841 - 01/10

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:51.2
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
71251 GIOVANA ASSIS JIENTARA 19HEL111 - 13 apos 01/10, 50% _____ / _____ / _____

Sistemas Lineares - Jacobi

Este método é iterativo, ou seja, não exato, já que parte-se de um valor atribuído inicial e mediante buscas locais sucessivas, o mesmo converge para a resposta correta. A diferença entre o resultado sugerido e aquele real, é denominada erro, e pode ser tornada tão pequena quanto se queira. O método entretanto tem um senão, que é um critério de convergência bastante rígido. Se não obedecido o algoritmo não converge para a resposta correta, e portanto o método é inútil.

Considere $A.x = b$ um sistema de quações lineares de ordem n e cujo $det(A) \neq 0$. A maneira extensa de escrever tal sistema é

$$\begin{array}{ccccccc} a_{11}x_1 & + & a_{12}x_2 & +...+ & a_{1n}x_n & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & +...+ & a_{2n}x_n & = & b_2 \\ ... & & & & & & \\ a_{n1}x_1 & + & a_{n2}x_2 & +...+ & a_{nn}x_n & = & b_n \end{array}$$

A matriz A dos coeficientes, pode ser decomposta em $A = L + D + R$, onde

L é a matriz inferior de A (o resto é zero)

D é a matriz contendo apenas a diagonal principal de A, o resto é zero.

R é a matriz superior de A (o resto é zero)

Veja por exemplo, se

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \therefore L = \begin{pmatrix} 0 & 0 & 0 \\ 4 & 0 & 0 \\ 7 & 8 & 0 \end{pmatrix},$$

e

$$D = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 9 \end{pmatrix}, R = \begin{pmatrix} 0 & 2 & 3 \\ 0 & 0 & 6 \\ 0 & 0 & 0 \end{pmatrix}$$

Supondo $det(D) \neq 0$, pode-se escrever: $(L + D + R)x = b$ e daqui $Dx = -(L + R)x + b$. e $x = -D^{-1}(L + R)x + D^{-1}b$. Chamando $B = -D^{-1}(L + R)$ e $g = D^{-1}.b$ obtem-se a equação geral deste método que é

$$x = Bx + g$$

. Note que é uma equação recursiva (ou iterativa) já que o x aparece nos 2 lados da mesma. A idéia aqui é atribuir um valor a x e com ele calcular um novo x, que é jogado no lugar do x velho e gera um novo x e assim sucessivamente até haver a convergência esperada (de antemão). Como supusemos $det(D) \neq 0$ pode-se dividir cada equação do sistema pelo coeficiente da diagonal principal, resultando

$$A^* = L^* + I + R^*$$

Agora o processo iterativo pode ser definido como

$$x^{k+1} = -(L^* + R^*)x^k + b^*$$

onde os elementos de L^*, R^* e b^* são:

$l_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i > j$ e zero senão

$d_{ij} = a_{ij}$ se $i = j$ e zero senão

$r_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i < j$ e zero senão

$b_ij^* = \frac{b_i}{a_{ii}}$

Critérios de convergência

Depois de dividida a matriz pelo elemento da diagonal principal, deve-se ter o **critério das linhas** que diz

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}^*| < 1$$

OU então o **critério das colunas** que diz

$$max_{1 \leq j \leq n} \sum_{i=1, i \neq j}^{n} |a_{ij}^*| < 1$$

Note que qualquer um dos dois critérios sendo satisfeitos, isso garante que o sistema tem convergência.

Definindo uma matriz como estritamente diagonalmente dominante se

$$\sum_{j=1, j \neq i}^{n} |a_{ij}| < |a_{ii}|$$

e se A é estritamente diagonalmente dominante, então A satisfaz o critério das linhas, e o sistema converge. Então, um bom critério de convergência pode ser

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}|* < 1$$

Exemplo

Seja resolver o sistema abaixo, usando o método de Jacobi

$$\begin{array}{ccccccc} 10x & + & 2y & + & z & = & 7 \\ x & + & 5y & + & z & = & -8 \\ 2x & + & 3y & + & 10z & = & 6 \end{array}$$

com $x^0 = (0.7, -1.6\ e\ 0.6)^t$ e $\epsilon < 0.11$.

Convergência ?

$|a_{12}| + |a_{13}| < |a_{11}|$? ou $|2| + |1| < |10|$? R: sim
$|a_{21}| + |a_{23}| < |a_{22}|$? ou $|1| + |1| < |5|$? R: sim
$|a_{31}| + |a_{32}| < |a_{33}|$? ou $|2| + |3| < |10|$? R: sim
logo, o sistema convergirá!

Solução

Dividindo cada equação pelo elemento da diagonal principal, fica

$$\begin{array}{ccccccc} x & + & 0.2y & + & 0.1z & = & 0.7 \\ 0.2x & + & y & + & 0.2z & = & -1.6 \\ 0.2x & + & 0.3y & + & z & = & 0.6 \end{array}$$

Apenas para ilustrar o critério das linhas para testar convergência (desnecessário no caso, já que já vimos que o sistema converge, mas em tese...)
$|a_{12}^*| + |a_{13}^*| = |0.2| + |0.1| = 0.3$ e
$|a_{21}^*| + |a_{23}^*| = |0.2| + |0.2| = 0.4$ e
$|a_{31}^*| + |a_{32}^*| = |0.2| + |0.3| = 0.5$ e
$\Rightarrow\ max_{1 \leq i \leq n}(0.3, 0.4\ 0.5) = 0.5 < 1$

Iterações

$x^{k+1} = -0.2y^k - 0.1z^k + 0.7$
$y^{k+1} = -0.2x^k - 0.2z^k - 1.6$
$z^{k+1} = -0.2x^k - 0.3y^k + 0.6$
Começando com $(0.7,\ -1.6,\ 0.6)$ obtemos uma primeira aproximação que é $(0.96,\ -1.86,\ 0.94)$ e jogando estes novos valores na fórmula iterativa, obtem-se:

$x \rightarrow 0.7,\ 0.96,\ 0.978,\ 0.9994,\ 0.9979$
$y \rightarrow -1.6,\ -1.86,\ -1.98,\ -1.9888,\ -1.9996$
$z \rightarrow 0.6,\ 0.94,\ 0.966,\ 0.9984,\ 0.9968$

O critério de parada pode ser
$max(x^{k+1} - x^k) < \epsilon$. No exemplo acima $0.0108 < 0.011$. Ou seja, neste caso, a solução procurada é $x = 0.9979$, $y = -1.9996$ e $z = 0.9978$. Salta aos olhos neste caso que se a tolerância for um pouco menor, a resposta exata será encontrada que é $(1, -2, 1)$. Basta lançar este resultado no sistema original que a exatidão será confirmada.

Solução Python

```
#jacobi
def jacobi(a,b):
 import numpy
 n=len(a) #numero de linhas de a
          # len(a[0])=colunas de a
 xvelho=numpy.zeros((n),float)
 erro=numpy.zeros((n),float)
 xnovo=numpy.zeros((n),float)
 f=numpy.zeros((n,len(a[0])),float)
 d=numpy.zeros((len(a[0])),float)
 tol=0.1
 for i in range(0,n):
  for j in range(0,n):
   if i==j:
    f[i,j]=0
   else:
    f[i,j]=-a[i,j]/a[i,i]
 for i in range(0,n):
  d[i]=b[i]/a[i,i]
 erromax=tol+1
 while erromax>tol:
  for i in range(0,n):
   xnovo[i]=0
   for k in range(0,n):
    xnovo[i]=xnovo[i]+f[i,k]*xvelho[k]
  for i in range(0,n):
   xnovo[i]=xnovo[i]+d[i]
  for i in range(0,n):
   erro[i]=abs(xnovo[i]-xvelho[i])
  erromax=erro[0]
  for i in range(1,n):
   if erro[i]>erromax:
    erromax=erro[i]
  for i in range(0,n):
   xvelho[i]=xnovo[i]
 return(xnovo)
import numpy
# para usar faça:
# a=numpy.array([[10,7,3],[1,5,4],[3,5,9]],float)
# b=numpy.array([22,33,44],float)
# print(jacobi(a,b))
```

☞ Para você fazer

Resolva usando este método e anexe a listagem do programa (ou copie no verso desta folha):

$$A = \begin{pmatrix} 92 & 17 & 35 & 25 & 13 \\ 28 & 86 & 10 & 31 & 14 \\ 2 & 9 & 31 & 4 & 12 \\ 33 & 3 & 7 & 76 & 32 \\ 6 & 16 & 1 & 24 & 50 \end{pmatrix}$$

e

$$B = (1267, 913, 484, 743, 505)$$

As respostas encontradas, com $\epsilon < 0.0001$ e sugestão inicial igual à coluna B são:

x_1	x_2	x_3	x_4	x_5

111-71251 - 01/10

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:51.2
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
70296 GIOVANNI MOTA BRIGATI 19HEL111 - 14 apos 01/10, 50% _____ / _____ / _____

Sistemas Lineares - Jacobi

Este método é iterativo, ou seja, não exato, já que parte-se de um valor atribuído inicial e mediante buscas locais sucessivas, o mesmo converge para a resposta correta. A diferença entre o resultado sugerido e aquele real, é denominada erro, e pode ser tornada tão pequena quanto se queira. O método entretanto tem um senão, que é um critério de convergência bastante rígido. Se não obedecido o algoritmo não converge para a resposta correta, e portanto o método é inútil.

Considere $A.x = b$ um sistema de quações lineares de ordem n e cujo $det(A) \neq 0$. A maneira extensa de escrever tal sistema é

$$\begin{array}{ccccccccc} a_{11}x_1 & + & a_{12}x_2 & +...+ & a_{1n}x_n & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & +...+ & a_{2n}x_n & = & b_2 \\ ... & & & & & & \\ a_{n1}x_1 & + & a_{n2}x_2 & +...+ & a_{nn}x_n & = & b_n \end{array}$$

A matriz A dos coeficientes, pode ser decomposta em $A = L + D + R$, onde

L é a matriz inferior de A (o resto é zero)

D é a matriz contendo apenas a diagonal principal de A, o resto é zero.

R é a matriz superior de A (o resto é zero)

Veja por exemplo, se

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \therefore L = \begin{pmatrix} 0 & 0 & 0 \\ 4 & 0 & 0 \\ 7 & 8 & 0 \end{pmatrix},$$

e

$$D = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 9 \end{pmatrix}, R = \begin{pmatrix} 0 & 2 & 3 \\ 0 & 0 & 6 \\ 0 & 0 & 0 \end{pmatrix}$$

Supondo $det(D) \neq 0$, pode-se escrever: $(L + D + R)x = b$ e daqui $Dx = -(L + R)x + b$. e $x = -D^{-1}(L + R)x + D^{-1}b$. Chamando $B = -D^{-1}(L + R)$ e $g = D^{-1}.b$ obtem-se a equação geral deste método que é

$$x = Bx + g$$

. Note que é uma equação recursiva (ou iterativa) já que o x aparece nos 2 lados da mesma. A idéia aqui é atribuir um valor a x e com ele calcular um novo x, que é jogado no lugar do x velho e gera um novo x e assim sucessivamente até haver a convergência esperada (de antemão). Como supusemos $det(D) \neq 0$ pode-se dividir cada equação do sistema pelo coeficiente da diagonal principal, resultando

$$A^* = L^* + I + R^*$$

Agora o processo iterativo pode ser definido como

$$x^{k+1} = -(L^* + R^*)x^k + b^*$$

onde os elementos de L^*, R^* e b^* são:

$l_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i > j$ e zero senão

$d_{ij} = a_{ij}$ se $i = j$ e zero senão

$r_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i < j$ e zero senão

$b_ij^* = \frac{b_i}{a_{ii}}$

Critérios de convergência

Depois de dividida a matriz pelo elemento da diagonal principal, deve-se ter o **critério das linhas** que diz

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}^*| < 1$$

OU então o **critério das colunas** que diz

$$max_{1 \leq j \leq n} \sum_{i=1, i \neq j}^{n} |a_{ij}^*| < 1$$

Note que qualquer um dos dois critérios sendo satisfeitos, isso garante que o sistema tem convergência.

Definindo uma matriz como estritamente diagonalmente dominante se

$$\sum_{j=1, j \neq i}^{n} |a_{ij}| < |a_{ii}|$$

e se A é estritamente diagonalmente dominante, então A satisfaz o critério das linhas, e o sistema converge. Então, um bom critério de convergência pode ser

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}|* < 1$$

Exemplo

Seja resolver o sistema abaixo, usando o método de Jacobi

$$\begin{array}{ccccccc} 10x & + & 2y & + & z & = & 7 \\ x & + & 5y & + & z & = & -8 \\ 2x & + & 3y & + & 10z & = & 6 \end{array}$$

com $x^0 = (0.7, -1.6\ e\ 0.6)^t$ e $\epsilon < 0.11$.

Convergência ?

$|a_{12}| + |a_{13}| < |a_{11}|$? ou $|2| + |1| < |10|$? R: sim
$|a_{21}| + |a_{23}| < |a_{22}|$? ou $|1| + |1| < |5|$? R: sim
$|a_{31}| + |a_{32}| < |a_{33}|$? ou $|2| + |3| < |10|$? R: sim
logo, o sistema convergirá!

Solução

Dividindo cada equação pelo elemento da diagonal principal, fica

$$\begin{array}{ccccccc} x & + & 0.2y & + & 0.1z & = & 0.7 \\ 0.2x & + & y & + & 0.2z & = & -1.6 \\ 0.2x & + & 0.3y & + & z & = & 0.6 \end{array}$$

Apenas para ilustrar o critério das linhas para testar convergência (desnecessário no caso, já que já vimos que o sistema converge, mas em tese...)
$|a_{12}^*| + |a_{13}^*| = |0.2| + |0.1| = 0.3$ e
$|a_{21}^*| + |a_{23}^*| = |0.2| + |0.2| = 0.4$ e
$|a_{31}^*| + |a_{32}^*| = |0.2| + |0.3| = 0.5$ e
$\Rightarrow\ max_{1 \leq i \leq n}(0.3, 0.4\ 0.5) = 0.5 < 1$

Iterações

$x^{k+1} = -0.2y^k - 0.1z^k + 0.7$
$y^{k+1} = -0.2x^k - 0.2z^k - 1.6$
$z^{k+1} = -0.2x^k - 0.3y^k + 0.6$
Começando com $(0.7,\ -1.6,\ 0.6)$ obtemos uma primeira aproximação que é $(0.96,\ -1.86,\ 0.94)$ e jogando estes novos valores na fórmula iterativa, obtem-se:

$x \rightarrow 0.7,\ 0.96,\ 0.978,\ 0.9994,\ 0.9979$
$y \rightarrow -1.6,\ -1.86,\ -1.98,\ -1.9888,\ -1.9996$
$z \rightarrow 0.6,\ 0.94,\ 0.966,\ 0.9984,\ 0.9968$

O critério de parada pode ser
$max(x^{k+1} - x^k) < \epsilon$. No exemplo acima $0.0108 < 0.011$. Ou seja, neste caso, a solução procurada é $x = 0.9979$, $y = -1.9996$ e $z = 0.9978$. Salta aos olhos neste caso que se a tolerância for um pouco menor, a resposta exata será encontrada que é $(1, -2, 1)$. Basta lançar este resultado no sistema original que a exatidão será confirmada.

Solução Python

```
#jacobi
def jacobi(a,b):
 import numpy
 n=len(a) #numero de linhas de a
          # len(a[0])=colunas de a
 xvelho=numpy.zeros((n),float)
 erro=numpy.zeros((n),float)
 xnovo=numpy.zeros((n),float)
 f=numpy.zeros((n,len(a[0])),float)
 d=numpy.zeros((len(a[0])),float)
 tol=0.1
 for i in range(0,n):
  for j in range(0,n):
   if i==j:
    f[i,j]=0
   else:
    f[i,j]=-a[i,j]/a[i,i]
 for i in range(0,n):
  d[i]=b[i]/a[i,i]
 erromax=tol+1
 while erromax>tol:
  for i in range(0,n):
   xnovo[i]=0
   for k in range(0,n):
    xnovo[i]=xnovo[i]+f[i,k]*xvelho[k]
  for i in range(0,n):
   xnovo[i]=xnovo[i]+d[i]
  for i in range(0,n):
   erro[i]=abs(xnovo[i]-xvelho[i])
  erromax=erro[0]
  for i in range(1,n):
   if erro[i]>erromax:
    erromax=erro[i]
  for i in range(0,n):
   xvelho[i]=xnovo[i]
 return(xnovo)
import numpy
# para usar faça:
# a=numpy.array([[10,7,3],[1,5,4],[3,5,9]],float)
# b=numpy.array([22,33,44],float)
# print(jacobi(a,b))
```

☞ Para você fazer

Resolva usando este método e anexe a listagem do programa (ou copie no verso desta folha):

$$A = \begin{pmatrix} 58 & 23 & 18 & 6 & 7 \\ 13 & 84 & 33 & 20 & 17 \\ 28 & 10 & 51 & 8 & 1 \\ 25 & 26 & 15 & 103 & 35 \\ 32 & 2 & 21 & 34 & 91 \end{pmatrix}$$

e

$$B = (804, 1808, 720, 2300, 1596)$$

As respostas encontradas, com $\epsilon < 0.0001$ e sugestão inicial igual à coluna B são:

x_1	x_2	x_3	x_4	x_5

111-70296 - 01/10

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:51.2
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
70308 GUILHERME FONTES DUARTE
19HEL111 - 15 apos 01/10, 50% _____ / / _____

Sistemas Lineares - Jacobi

Este método é iterativo, ou seja, não exato, já que parte-se de um valor atribuído inicial e mediante buscas locais sucessivas, o mesmo converge para a resposta correta. A diferença entre o resultado sugerido e aquele real, é denominada erro, e pode ser tornada tão pequena quanto se queira. O método entretanto tem um senão, que é um critério de convergência bastante rígido. Se não obedecido o algoritmo não converge para a resposta correta, e portanto o método é inútil.

Considere $A.x = b$ um sistema de quações lineares de ordem n e cujo $det(A) \neq 0$. A maneira extensa de escrever tal sistema é

$$\begin{array}{ccccccc} a_{11}x_1 & + & a_{12}x_2 & +...+ & a_{1n}x_n & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & +...+ & a_{2n}x_n & = & b_2 \\ ... & & & & & & \\ a_{n1}x_1 & + & a_{n2}x_2 & +...+ & a_{nn}x_n & = & b_n \end{array}$$

A matriz A dos coeficientes, pode ser decomposta em $A = L + D + R$, onde

L é a matriz inferior de A (o resto é zero)

D é a matriz contendo apenas a diagonal principal de A, o resto é zero.

R é a matriz superior de A (o resto é zero)

Veja por exemplo, se

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \therefore L = \begin{pmatrix} 0 & 0 & 0 \\ 4 & 0 & 0 \\ 7 & 8 & 0 \end{pmatrix},$$

e

$$D = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 9 \end{pmatrix}, R = \begin{pmatrix} 0 & 2 & 3 \\ 0 & 0 & 6 \\ 0 & 0 & 0 \end{pmatrix}$$

Supondo $det(D) \neq 0$, pode-se escrever: $(L + D + R)x = b$ e daqui $Dx = -(L + R)x + b$. e $x = -D^{-1}(L + R)x + D^{-1}b$. Chamando $B = -D^{-1}(L + R)$ e $g = D^{-1}.b$ obtem-se a equação geral deste método que é

$$x = Bx + g$$

. Note que é uma equação recursiva (ou iterativa) já que o x aparece nos 2 lados da mesma. A idéia aqui é atribuir um valor a x e com ele calcular um novo x, que é jogado no lugar do x velho e gera um novo x e assim sucessivamente até haver a convergência esperada (de antemão). Como supusemos $det(D) \neq 0$ pode-se dividir cada equação do sistema pelo coeficiente da diagonal principal, resultando

$$A^* = L^* + I + R^*$$

Agora o processo iterativo pode ser definido como

$$x^{k+1} = -(L^* + R^*)x^k + b^*$$

onde os elementos de L^*, R^* e b^* são:

$l_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i > j$ e zero senão

$d_{ij} = a_{ij}$ se $i = j$ e zero senão

$r_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i < j$ e zero senão

$b_ij^* = \frac{b_i}{a_{ii}}$

Critérios de convergência

Depois de dividida a matriz pelo elemento da diagonal principal, deve-se ter o **critério das linhas** que diz

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}^*| < 1$$

OU então o **critério das colunas** que diz

$$max_{1 \leq j \leq n} \sum_{i=1, i \neq j}^{n} |a_{ij}^*| < 1$$

Note que qualquer um dos dois critérios sendo satisfeitos, isso garante que o sistema tem convergência.

Definindo uma matriz como estritamente diagonalmente dominante se

$$\sum_{j=1, j \neq i}^{n} |a_{ij}| < |a_{ii}|$$

e se A é estritamente diagonalmente dominante, então A satisfaz o critério das linhas, e o sistema converge. Então, um bom critério de convergência pode ser

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}^|* < 1$$

Exemplo

Seja resolver o sistema abaixo, usando o método de Jacobi

$$\begin{array}{ccccccc} 10x & + & 2y & + & z & = & 7 \\ x & + & 5y & + & z & = & -8 \\ 2x & + & 3y & + & 10z & = & 6 \end{array}$$

com $x^0 = (0.7, -1.6\ e\ 0.6)^t$ e $\epsilon < 0.11$.

Convergência ?

$|a_{12}| + |a_{13}| < |a_{11}|$? ou $|2| + |1| < |10|$? R: sim
$|a_{21}| + |a_{23}| < |a_{22}|$? ou $|1| + |1| < |5|$? R: sim
$|a_{31}| + |a_{32}| < |a_{33}|$? ou $|2| + |3| < |10|$? R: sim
logo, o sistema convergirá!

Solução

Dividindo cada equação pelo elemento da diagonal principal, fica

$$\begin{array}{ccccccc} x & + & 0.2y & + & 0.1z & = & 0.7 \\ 0.2x & + & y & + & 0.2z & = & -1.6 \\ 0.2x & + & 0.3y & + & z & = & 0.6 \end{array}$$

Apenas para ilustrar o critério das linhas para testar convergência (desnecessário no caso, já que já vimos que o sistema converge, mas em tese...)
$|a_{12}^*| + |a_{13}^*| = |0.2| + |0.1| = 0.3$ e
$|a_{21}^*| + |a_{23}^*| = |0.2| + |0.2| = 0.4$ e
$|a_{31}^*| + |a_{32}^*| = |0.2| + |0.3| = 0.5$ e
$\Rightarrow\ max_{1 \leq i \leq n}(0.3, 0.4\ 0.5) = 0.5 < 1$

Iterações

$x^{k+1} = -0.2y^k - 0.1z^k + 0.7$
$y^{k+1} = -0.2x^k - 0.2z^k - 1.6$
$z^{k+1} = -0.2x^k - 0.3y^k + 0.6$
Começando com $(0.7,\ -1.6,\ 0.6)$ obtemos uma primeira aproximação que é $(0.96,\ -1.86,\ 0.94)$ e jogando estes novos valores na fórmula iterativa, obtem-se:

$x \rightarrow 0.7,\ 0.96,\ 0.978,\ 0.9994,\ 0.9979$
$y \rightarrow -1.6,\ -1.86,\ -1.98,\ -1.9888,\ -1.9996$
$z \rightarrow 0.6,\ 0.94,\ 0.966,\ 0.9984,\ 0.9968$

O critério de parada pode ser
$max(x^{k+1} - x^k) < \epsilon$. No exemplo acima $0.0108 < 0.011$. Ou seja, neste caso, a solução procurada é $x = 0.9979$, $y = -1.9996$ e $z = 0.9978$. Salta aos olhos neste caso que se a tolerância for um pouco menor, a resposta exata será encontrada que é $(1, -2, 1)$. Basta lançar este resultado no sistema original que a exatidão será confirmada.

Solução Python

```
#jacobi
def jacobi(a,b):
 import numpy
 n=len(a) #numero de linhas de a
          # len(a[0])=colunas de a
 xvelho=numpy.zeros((n),float)
 erro=numpy.zeros((n),float)
 xnovo=numpy.zeros((n),float)
 f=numpy.zeros((n,len(a[0])),float)
 d=numpy.zeros((len(a[0])),float)
 tol=0.1
 for i in range(0,n):
  for j in range(0,n):
   if i==j:
    f[i,j]=0
   else:
    f[i,j]=-a[i,j]/a[i,i]
 for i in range(0,n):
  d[i]=b[i]/a[i,i]
 erromax=tol+1
 while erromax>tol:
  for i in range(0,n):
   xnovo[i]=0
   for k in range(0,n):
    xnovo[i]=xnovo[i]+f[i,k]*xvelho[k]
  for i in range(0,n):
   xnovo[i]=xnovo[i]+d[i]
  for i in range(0,n):
   erro[i]=abs(xnovo[i]-xvelho[i])
  erromax=erro[0]
  for i in range(1,n):
   if erro[i]>erromax:
    erromax=erro[i]
  for i in range(0,n):
   xvelho[i]=xnovo[i]
 return(xnovo)
import numpy
# para usar faça:
# a=numpy.array([[10,7,3],[1,5,4],[3,5,9]],float)
# b=numpy.array([22,33,44],float)
# print(jacobi(a,b))
```

☞ Para você fazer

Resolva usando este método e anexe a listagem do programa (ou copie no verso desta folha):

$$A = \begin{pmatrix} 85 & 20 & 31 & 16 & 15 \\ 19 & 57 & 1 & 11 & 18 \\ 26 & 17 & 92 & 32 & 14 \\ 35 & 29 & 33 & 101 & 2 \\ 9 & 12 & 8 & 4 & 34 \end{pmatrix}$$

e

$$B = (1495, 1449, 2134, 2347, 954)$$

As respostas encontradas, com $\epsilon < 0.0001$ e sugestão inicial igual à coluna B são:

x_1	x_2	x_3	x_4	x_5

111-70308 - 01/10

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:51.2
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
70315 HENRIQUE DOMICIANO OSINSKI
19HEL111 - 16 apos 01/10, 50% _____ / / _____

Sistemas Lineares - Jacobi

Este método é iterativo, ou seja, não exato, já que parte-se de um valor atribuído inicial e mediante buscas locais sucessivas, o mesmo converge para a resposta correta. A diferença entre o resultado sugerido e aquele real, é denominada erro, e pode ser tornada tão pequena quanto se queira. O método entretanto tem um senão, que é um critério de convergência bastante rígido. Se não obedecido o algoritmo não converge para a resposta correta, e portanto o método é inútil.

Considere $A.x = b$ um sistema de quações lineares de ordem n e cujo $det(A) \neq 0$. A maneira extensa de escrever tal sistema é

$$\begin{array}{ccccccc} a_{11}x_1 & + & a_{12}x_2 & +...+ & a_{1n}x_n & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & +...+ & a_{2n}x_n & = & b_2 \\ ... & & & & & & \\ a_{n1}x_1 & + & a_{n2}x_2 & +...+ & a_{nn}x_n & = & b_n \end{array}$$

A matriz A dos coeficientes, pode ser decomposta em $A = L + D + R$, onde

L é a matriz inferior de A (o resto é zero)

D é a matriz contendo apenas a diagonal principal de A, o resto é zero.

R é a matriz superior de A (o resto é zero)

Veja por exemplo, se

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \therefore L = \begin{pmatrix} 0 & 0 & 0 \\ 4 & 0 & 0 \\ 7 & 8 & 0 \end{pmatrix},$$

e

$$D = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 9 \end{pmatrix}, R = \begin{pmatrix} 0 & 2 & 3 \\ 0 & 0 & 6 \\ 0 & 0 & 0 \end{pmatrix}$$

Supondo $det(D) \neq 0$, pode-se escrever: $(L + D + R)x = b$ e daqui $Dx = -(L + R)x + b$. e $x = -D^{-1}(L + R)x + D^{-1}b$. Chamando $B = -D^{-1}(L + R)$ e $g = D^{-1}.b$ obtem-se a equação geral deste método que é

$$x = Bx + g$$

. Note que é uma equação recursiva (ou iterativa) já que o x aparece nos 2 lados da mesma. A idéia aqui é atribuir um valor a x e com ele calcular um novo x, que é jogado no lugar do x velho e gera um novo x e assim sucessivamente até haver a convergência esperada (de antemão). Como supusemos $det(D) \neq 0$ pode-se dividir cada equação do sistema pelo coeficiente da diagonal principal, resultando

$$A^* = L^* + I + R^*$$

Agora o processo iterativo pode ser definido como

$$x^{k+1} = -(L^* + R^*)x^k + b^*$$

onde os elementos de L^*, R^* e b^* são:

$l_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i > j$ e zero senão

$d_{ij} = a_{ij}$ se $i = j$ e zero senão

$r_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i < j$ e zero senão

$b_ij^* = \frac{b_i}{a_{ii}}$

Critérios de convergência

Depois de dividida a matriz pelo elemento da diagonal principal, deve-se ter o **critério das linhas** que diz

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}^*| < 1$$

OU então o **critério das colunas** que diz

$$max_{1 \leq j \leq n} \sum_{i=1, i \neq j}^{n} |a_{ij}^*| < 1$$

Note que qualquer um dos dois critérios sendo satisfeitos, isso garante que o sistema tem convergência.

Definindo uma matriz como estritamente diagonalmente dominante se

$$\sum_{j=1, j \neq i}^{n} |a_{ij}| < |a_{ii}|$$

e se A é estritamente diagonalmente dominante, então A satisfaz o critério das linhas, e o sistema converge. Então, um bom critério de convergência pode ser

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}^|* < 1$$

Exemplo

Seja resolver o sistema abaixo, usando o método de Jacobi

$$\begin{array}{ccccccc} 10x & + & 2y & + & z & = & 7 \\ x & + & 5y & + & z & = & -8 \\ 2x & + & 3y & + & 10z & = & 6 \end{array}$$

com $x^0 = (0.7, -1.6\ e\ 0.6)^t$ e $\epsilon < 0.11$.

Convergência ?

$|a_{12}| + |a_{13}| < |a_{11}|$? ou $|2| + |1| < |10|$? R: sim
$|a_{21}| + |a_{23}| < |a_{22}|$? ou $|1| + |1| < |5|$? R: sim
$|a_{31}| + |a_{32}| < |a_{33}|$? ou $|2| + |3| < |10|$? R: sim
logo, o sistema convergirá!

Solução

Dividindo cada equação pelo elemento da diagonal principal, fica

$$\begin{array}{ccccccc} x & + & 0.2y & + & 0.1z & = & 0.7 \\ 0.2x & + & y & + & 0.2z & = & -1.6 \\ 0.2x & + & 0.3y & + & z & = & 0.6 \end{array}$$

Apenas para ilustrar o critério das linhas para testar convergência (desnecessário no caso, já que já vimos que o sistema converge, mas em tese...)
$|a_{12}^*| + |a_{13}^*| = |0.2| + |0.1| = 0.3$ e
$|a_{21}^*| + |a_{23}^*| = |0.2| + |0.2| = 0.4$ e
$|a_{31}^*| + |a_{32}^*| = |0.2| + |0.3| = 0.5$ e
$\Rightarrow\ max_{1 \leq i \leq n}(0.3, 0.4\ 0.5) = 0.5 < 1$

Iterações

$x^{k+1} = -0.2y^k - 0.1z^k + 0.7$
$y^{k+1} = -0.2x^k - 0.2z^k - 1.6$
$z^{k+1} = -0.2x^k - 0.3y^k + 0.6$
Começando com $(0.7,\ -1.6,\ 0.6)$ obtemos uma primeira aproximação que é $(0.96,\ -1.86,\ 0.94)$ e jogando estes novos valores na fórmula iterativa, obtem-se:

$x \rightarrow 0.7,\ 0.96,\ 0.978,\ 0.9994,\ 0.9979$
$y \rightarrow -1.6,\ -1.86,\ -1.98,\ -1.9888,\ -1.9996$
$z \rightarrow 0.6,\ 0.94,\ 0.966,\ 0.9984,\ 0.9968$

O critério de parada pode ser
$max(x^{k+1} - x^k) < \epsilon$. No exemplo acima $0.0108 < 0.011$. Ou seja, neste caso, a solução procurada é $x = 0.9979$, $y = -1.9996$ e $z = 0.9978$. Salta aos olhos neste caso que se a tolerância for um pouco menor, a resposta exata será encontrada que é $(1, -2, 1)$. Basta lançar este resultado no sistema original que a exatidão será confirmada.

Solução Python

```
#jacobi
def jacobi(a,b):
 import numpy
 n=len(a) #numero de linhas de a
          # len(a[0])=colunas de a
 xvelho=numpy.zeros((n),float)
 erro=numpy.zeros((n),float)
 xnovo=numpy.zeros((n),float)
 f=numpy.zeros((n,len(a[0])),float)
 d=numpy.zeros((len(a[0])),float)
 tol=0.1
 for i in range(0,n):
  for j in range(0,n):
   if i==j:
    f[i,j]=0
   else:
    f[i,j]=-a[i,j]/a[i,i]
 for i in range(0,n):
  d[i]=b[i]/a[i,i]
 erromax=tol+1
 while erromax>tol:
  for i in range(0,n):
   xnovo[i]=0
   for k in range(0,n):
    xnovo[i]=xnovo[i]+f[i,k]*xvelho[k]
  for i in range(0,n):
   xnovo[i]=xnovo[i]+d[i]
  for i in range(0,n):
   erro[i]=abs(xnovo[i]-xvelho[i])
  erromax=erro[0]
  for i in range(1,n):
   if erro[i]>erromax:
    erromax=erro[i]
  for i in range(0,n):
   xvelho[i]=xnovo[i]
 return(xnovo)
import numpy
# para usar faça:
# a=numpy.array([[10,7,3],[1,5,4],[3,5,9]],float)
# b=numpy.array([22,33,44],float)
# print(jacobi(a,b))
```

☞ Para você fazer

Resolva usando este método e anexe a listagem do programa (ou copie no verso desta folha):

$$A = \begin{pmatrix} 64 & 22 & 5 & 28 & 3 \\ 16 & 65 & 1 & 33 & 9 \\ 4 & 27 & 94 & 35 & 25 \\ 10 & 31 & 19 & 74 & 11 \\ 13 & 15 & 23 & 20 & 72 \end{pmatrix}$$

e

$$B = (1049, 1387, 2018, 1733, 1210)$$

As respostas encontradas, com $\epsilon < 0.0001$ e sugestão inicial igual à coluna B são:

x_1	x_2	x_3	x_4	x_5

111-70315 - 01/10

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:51.2
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
70322 HENRIQUE MARTINS DOMANSKI
19HEL111 - 17 apos 01/10, 50% _____ / / _____

Sistemas Lineares - Jacobi

Este método é iterativo, ou seja, não exato, já que parte-se de um valor atribuído inicial e mediante buscas locais sucessivas, o mesmo converge para a resposta correta. A diferença entre o resultado sugerido e aquele real, é denominada erro, e pode ser tornada tão pequena quanto se queira. O método entretanto tem um senão, que é um critério de convergência bastante rígido. Se não obedecido o algoritmo não converge para a resposta correta, e portanto o método é inútil.

Considere $A.x = b$ um sistema de quações lineares de ordem n e cujo $det(A) \neq 0$. A maneira extensa de escrever tal sistema é

$$\begin{array}{lcllcl} a_{11}x_1 & + & a_{12}x_2 & +...+ & a_{1n}x_n & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & +...+ & a_{2n}x_n & = & b_2 \\ ... & & & & & & \\ a_{n1}x_1 & + & a_{n2}x_2 & +...+ & a_{nn}x_n & = & b_n \end{array}$$

A matriz A dos coeficientes, pode ser decomposta em $A = L + D + R$, onde

L é a matriz inferior de A (o resto é zero)

D é a matriz contendo apenas a diagonal principal de A, o resto é zero.

R é a matriz superior de A (o resto é zero)

Veja por exemplo, se

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \therefore L = \begin{pmatrix} 0 & 0 & 0 \\ 4 & 0 & 0 \\ 7 & 8 & 0 \end{pmatrix},$$

e

$$D = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 9 \end{pmatrix}, R = \begin{pmatrix} 0 & 2 & 3 \\ 0 & 0 & 6 \\ 0 & 0 & 0 \end{pmatrix}$$

Supondo $det(D) \neq 0$, pode-se escrever: $(L + D + R)x = b$ e daqui $Dx = -(L + R)x + b$. e $x = -D^{-1}(L + R)x + D^{-1}b$. Chamando $B = -D^{-1}(L + R)$ e $g = D^{-1}.b$ obtem-se a equação geral deste método que é

$$x = Bx + g$$

. Note que é uma equação recursiva (ou iterativa) já que o x aparece nos 2 lados da mesma. A idéia aqui é atribuir um valor a x e com ele calcular um novo x, que é jogado no lugar do x velho e gera um novo x e assim sucessivamente até haver a convergência esperada (de antemão). Como supusemos $det(D) \neq 0$ pode-se dividir cada equação do sistema pelo coeficiente da diagonal principal, resultando

$$A^* = L^* + I + R^*$$

Agora o processo iterativo pode ser definido como

$$x^{k+1} = -(L^* + R^*)x^k + b^*$$

onde os elementos de L^*, R^* e b^* são:

$l_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i > j$ e zero senão

$d_{ij} = a_{ij}$ se $i = j$ e zero senão

$r_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i < j$ e zero senão

$b_ij^* = \frac{b_i}{a_{ii}}$

Critérios de convergência

Depois de dividida a matriz pelo elemento da diagonal principal, deve-se ter o **critério das linhas** que diz

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}^*| < 1$$

OU então o **critério das colunas** que diz

$$max_{1 \leq j \leq n} \sum_{i=1, i \neq j}^{n} |a_{ij}^*| < 1$$

Note que qualquer um dos dois critérios sendo satisfeitos, isso garante que o sistema tem convergência.

Definindo uma matriz como estritamente diagonalmente dominante se

$$\sum_{j=1, j \neq i}^{n} |a_{ij}| < |a_{ii}|$$

e se A é estritamente diagonalmente dominante, então A satisfaz o critério das linhas, e o sistema converge. Então, um bom critério de convergência pode ser

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}^|* < 1$$

Exemplo

Seja resolver o sistema abaixo, usando o método de Jacobi

$$\begin{array}{rcrcrcr} 10x & + & 2y & + & z & = & 7 \\ x & + & 5y & + & z & = & -8 \\ 2x & + & 3y & + & 10z & = & 6 \end{array}$$

com $x^0 = (0.7, -1.6\ e\ 0.6)^t$ e $\epsilon < 0.11$.

Convergência ?

$|a_{12}| + |a_{13}| < |a_{11}|$? ou $|2| + |1| < |10|$? R: sim
$|a_{21}| + |a_{23}| < |a_{22}|$? ou $|1| + |1| < |5|$? R: sim
$|a_{31}| + |a_{32}| < |a_{33}|$? ou $|2| + |3| < |10|$? R: sim
logo, o sistema convergirá!

Solução

Dividindo cada equação pelo elemento da diagonal principal, fica

$$\begin{array}{rcrcrcr} x & + & 0.2y & + & 0.1z & = & 0.7 \\ 0.2x & + & y & + & 0.2z & = & -1.6 \\ 0.2x & + & 0.3y & + & z & = & 0.6 \end{array}$$

Apenas para ilustrar o critério das linhas para testar convergência (desnecessário no caso, já que já vimos que o sistema converge, mas em tese...)
$|a_{12}^*| + |a_{13}^*| = |0.2| + |0.1| = 0.3$ e
$|a_{21}^*| + |a_{23}^*| = |0.2| + |0.2| = 0.4$ e
$|a_{31}^*| + |a_{32}^*| = |0.2| + |0.3| = 0.5$ e
$\Rightarrow\ max_{1 \leq i \leq n}(0.3, 0.4\ 0.5) = 0.5 < 1$

Iterações

$x^{k+1} = -0.2y^k - 0.1z^k + 0.7$
$y^{k+1} = -0.2x^k - 0.2z^k - 1.6$
$z^{k+1} = -0.2x^k - 0.3y^k + 0.6$
Começando com $(0.7,\ -1.6,\ 0.6)$ obtemos uma primeira aproximação que é $(0.96,\ -1.86,\ 0.94)$ e jogando estes novos valores na fórmula iterativa, obtem-se:

$x \rightarrow 0.7,\ 0.96,\ 0.978,\ 0.9994,\ 0.9979$
$y \rightarrow -1.6,\ -1.86,\ -1.98,\ -1.9888,\ -1.9996$
$z \rightarrow 0.6,\ 0.94,\ 0.966,\ 0.9984,\ 0.9968$

O critério de parada pode ser
$max(x^{k+1} - x^k) < \epsilon$. No exemplo acima $0.0108 < 0.011$. Ou seja, neste caso, a solução procurada é $x = 0.9979$, $y = -1.9996$ e $z = 0.9978$. Salta aos olhos neste caso que se a tolerância for um pouco menor, a resposta exata será encontrada que é $(1, -2, 1)$. Basta lançar este resultado no sistema original que a exatidão será confirmada.

Solução Python

```
#jacobi
def jacobi(a,b):
 import numpy
 n=len(a) #numero de linhas de a
          # len(a[0])=colunas de a
 xvelho=numpy.zeros((n),float)
 erro=numpy.zeros((n),float)
 xnovo=numpy.zeros((n),float)
 f=numpy.zeros((n,len(a[0])),float)
 d=numpy.zeros((len(a[0])),float)
 tol=0.1
 for i in range(0,n):
  for j in range(0,n):
   if i==j:
    f[i,j]=0
   else:
    f[i,j]=-a[i,j]/a[i,i]
 for i in range(0,n):
  d[i]=b[i]/a[i,i]
 erromax=tol+1
 while erromax>tol:
  for i in range(0,n):
   xnovo[i]=0
   for k in range(0,n):
    xnovo[i]=xnovo[i]+f[i,k]*xvelho[k]
  for i in range(0,n):
   xnovo[i]=xnovo[i]+d[i]
  for i in range(0,n):
   erro[i]=abs(xnovo[i]-xvelho[i])
  erromax=erro[0]
  for i in range(1,n):
   if erro[i]>erromax:
    erromax=erro[i]
  for i in range(0,n):
   xvelho[i]=xnovo[i]
 return(xnovo)
import numpy
# para usar faça:
# a=numpy.array([[10,7,3],[1,5,4],[3,5,9]],float)
# b=numpy.array([22,33,44],float)
# print(jacobi(a,b))
```

☞ Para você fazer

Resolva usando este método e anexe a listagem do programa (ou copie no verso desta folha):

$$A = \begin{pmatrix} 112 & 23 & 31 & 24 & 29 \\ 12 & 52 & 11 & 21 & 7 \\ 34 & 32 & 100 & 16 & 14 \\ 13 & 28 & 30 & 82 & 6 \\ 15 & 8 & 35 & 33 & 94 \end{pmatrix}$$

e

$$B = (1512, 814, 1262, 1608, 2155)$$

As respostas encontradas, com $\epsilon < 0.0001$ e sugestão inicial igual à coluna B são:

x_1	x_2	x_3	x_4	x_5

111-70322 - 01/10

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:51.2
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
70339 HERICK DENKE 19HEL111 - 18 apos 01/10, 50% _____ / _____ / _____

Sistemas Lineares - Jacobi

Este método é iterativo, ou seja, não exato, já que parte-se de um valor atribuído inicial e mediante buscas locais sucessivas, o mesmo converge para a resposta correta. A diferença entre o resultado sugerido e aquele real, é denominada erro, e pode ser tornada tão pequena quanto se queira. O método entretanto tem um senão, que é um critério de convergência bastante rígido. Se não obedecido o algoritmo não converge para a resposta correta, e portanto o método é inútil.

Considere $A.x = b$ um sistema de quações lineares de ordem n e cujo $det(A) \neq 0$. A maneira extensa de escrever tal sistema é

$$\begin{array}{ccccccccc} a_{11}x_1 & + & a_{12}x_2 & +...+ & a_{1n}x_n & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & +...+ & a_{2n}x_n & = & b_2 \\ ... & & & & & & \\ a_{n1}x_1 & + & a_{n2}x_2 & +...+ & a_{nn}x_n & = & b_n \end{array}$$

A matriz A dos coeficientes, pode ser decomposta em $A = L + D + R$, onde

L é a matriz inferior de A (o resto é zero)

D é a matriz contendo apenas a diagonal principal de A, o resto é zero.

R é a matriz superior de A (o resto é zero)

Veja por exemplo, se

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \therefore L = \begin{pmatrix} 0 & 0 & 0 \\ 4 & 0 & 0 \\ 7 & 8 & 0 \end{pmatrix},$$

e

$$D = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 9 \end{pmatrix}, R = \begin{pmatrix} 0 & 2 & 3 \\ 0 & 0 & 6 \\ 0 & 0 & 0 \end{pmatrix}$$

Supondo $det(D) \neq 0$, pode-se escrever: $(L + D + R)x = b$ e daqui $Dx = -(L + R)x + b$. e $x = -D^{-1}(L + R)x + D^{-1}b$. Chamando $B = -D^{-1}(L + R)$ e $g = D^{-1}.b$ obtem-se a equação geral deste método que é

$$x = Bx + g$$

. Note que é uma equação recursiva (ou iterativa) já que o x aparece nos 2 lados da mesma. A idéia aqui é atribuir um valor a x e com ele calcular um novo x, que é jogado no lugar do x velho e gera um novo x e assim sucessivamente até haver a convergência esperada (de antemão). Como supusemos $det(D) \neq 0$ pode-se dividir cada equação do sistema pelo coeficiente da diagonal principal, resultando

$$A^* = L^* + I + R^*$$

Agora o processo iterativo pode ser definido como

$$x^{k+1} = -(L^* + R^*)x^k + b^*$$

onde os elementos de L^*, R^* e b^* são:

$l_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i > j$ e zero senão

$d_{ij} = a_{ij}$ se $i = j$ e zero senão

$r_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i < j$ e zero senão

$b_ij^* = \frac{b_i}{a_{ii}}$

Critérios de convergência

Depois de dividida a matriz pelo elemento da diagonal principal, deve-se ter o **critério das linhas** que diz

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}^*| < 1$$

OU então o **critério das colunas** que diz

$$max_{1 \leq j \leq n} \sum_{i=1, i \neq j}^{n} |a_{ij}^*| < 1$$

Note que qualquer um dos dois critérios sendo satisfeitos, isso garante que o sistema tem convergência.

Definindo uma matriz como estritamente diagonalmente dominante se

$$\sum_{j=1, j \neq i}^{n} |a_{ij}| < |a_{ii}|$$

e se A é estritamente diagonalmente dominante, então A satisfaz o critério das linhas, e o sistema converge. Então, um bom critério de convergência pode ser

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}|* < 1$$

Exemplo

Seja resolver o sistema abaixo, usando o método de Jacobi

$$\begin{array}{ccccccc} 10x & + & 2y & + & z & = & 7 \\ x & + & 5y & + & z & = & -8 \\ 2x & + & 3y & + & 10z & = & 6 \end{array}$$

com $x^0 = (0.7, -1.6\ e\ 0.6)^t$ e $\epsilon < 0.11$.

Convergência ?

$|a_{12}| + |a_{13}| < |a_{11}|$? ou $|2| + |1| < |10|$? R: sim
$|a_{21}| + |a_{23}| < |a_{22}|$? ou $|1| + |1| < |5|$? R: sim
$|a_{31}| + |a_{32}| < |a_{33}|$? ou $|2| + |3| < |10|$? R: sim
logo, o sistema convergirá!

Solução

Dividindo cada equação pelo elemento da diagonal principal, fica

$$\begin{array}{ccccccc} x & + & 0.2y & + & 0.1z & = & 0.7 \\ 0.2x & + & y & + & 0.2z & = & -1.6 \\ 0.2x & + & 0.3y & + & z & = & 0.6 \end{array}$$

Apenas para ilustrar o critério das linhas para testar convergência (desnecessário no caso, já que já vimos que o sistema converge, mas em tese...)
$|a_{12}^*| + |a_{13}^*| = |0.2| + |0.1| = 0.3$ e
$|a_{21}^*| + |a_{23}^*| = |0.2| + |0.2| = 0.4$ e
$|a_{31}^*| + |a_{32}^*| = |0.2| + |0.3| = 0.5$ e
$\Rightarrow\ max_{1 \leq i \leq n}(0.3, 0.4\ 0.5) = 0.5 < 1$

Iterações

$x^{k+1} = -0.2y^k - 0.1z^k + 0.7$
$y^{k+1} = -0.2x^k - 0.2z^k - 1.6$
$z^{k+1} = -0.2x^k - 0.3y^k + 0.6$
Começando com $(0.7,\ -1.6,\ 0.6)$ obtemos uma primeira aproximação que é $(0.96,\ -1.86,\ 0.94)$ e jogando estes novos valores na fórmula iterativa, obtem-se:

$x \rightarrow 0.7,\ 0.96,\ 0.978,\ 0.9994,\ 0.9979$
$y \rightarrow -1.6,\ -1.86,\ -1.98,\ -1.9888,\ -1.9996$
$z \rightarrow 0.6,\ 0.94,\ 0.966,\ 0.9984,\ 0.9968$

O critério de parada pode ser
$max(x^{k+1} - x^k) < \epsilon$. No exemplo acima $0.0108 < 0.011$. Ou seja, neste caso, a solução procurada é $x = 0.9979$, $y = -1.9996$ e $z = 0.9978$. Salta aos olhos neste caso que se a tolerância for um pouco menor, a resposta exata será encontrada que é $(1, -2, 1)$. Basta lançar este resultado no sistema original que a exatidão será confirmada.

Solução Python

```
#jacobi
def jacobi(a,b):
 import numpy
 n=len(a) #numero de linhas de a
          # len(a[0])=colunas de a
 xvelho=numpy.zeros((n),float)
 erro=numpy.zeros((n),float)
 xnovo=numpy.zeros((n),float)
 f=numpy.zeros((n,len(a[0])),float)
 d=numpy.zeros((len(a[0])),float)
 tol=0.1
 for i in range(0,n):
  for j in range(0,n):
   if i==j:
    f[i,j]=0
   else:
    f[i,j]=-a[i,j]/a[i,i]
 for i in range(0,n):
  d[i]=b[i]/a[i,i]
 erromax=tol+1
 while erromax>tol:
  for i in range(0,n):
   xnovo[i]=0
   for k in range(0,n):
    xnovo[i]=xnovo[i]+f[i,k]*xvelho[k]
  for i in range(0,n):
   xnovo[i]=xnovo[i]+d[i]
  for i in range(0,n):
   erro[i]=abs(xnovo[i]-xvelho[i])
  erromax=erro[0]
  for i in range(1,n):
   if erro[i]>erromax:
    erromax=erro[i]
  for i in range(0,n):
   xvelho[i]=xnovo[i]
 return(xnovo)
import numpy
# para usar faça:
# a=numpy.array([[10,7,3],[1,5,4],[3,5,9]],float)
# b=numpy.array([22,33,44],float)
# print(jacobi(a,b))
```

☞ Para você fazer

Resolva usando este método e anexe a listagem do programa (ou copie no verso desta folha):

$$A = \begin{pmatrix} 74 & 10 & 29 & 2 & 28 \\ 14 & 95 & 33 & 13 & 31 \\ 25 & 5 & 71 & 18 & 22 \\ 4 & 26 & 3 & 71 & 30 \\ 1 & 6 & 34 & 20 & 62 \end{pmatrix}$$

e

$$B = (1095, 1049, 956, 1105, 899)$$

As respostas encontradas, com $\epsilon < 0.0001$ e sugestão inicial igual à coluna B são:

x_1	x_2	x_3	x_4	x_5

111-70339 - 01/10

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:51.2
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
70346 IURI PAVANI DE SOUZA 19HEL111 - 19 apos 01/10, 50% _____ / _____ / _____

Sistemas Lineares - Jacobi

Este método é iterativo, ou seja, não exato, já que parte-se de um valor atribuído inicial e mediante buscas locais sucessivas, o mesmo converge para a resposta correta. A diferença entre o resultado sugerido e aquele real, é denominada erro, e pode ser tornada tão pequena quanto se queira. O método entretanto tem um senão, que é um critério de convergência bastante rígido. Se não obedecido o algoritmo não converge para a resposta correta, e portanto o método é inútil.

Considere $A.x = b$ um sistema de quações lineares de ordem n e cujo $det(A) \neq 0$. A maneira extensa de escrever tal sistema é

$$\begin{array}{ccccccc} a_{11}x_1 & + & a_{12}x_2 & +...+ & a_{1n}x_n & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & +...+ & a_{2n}x_n & = & b_2 \\ ... & & & & & & \\ a_{n1}x_1 & + & a_{n2}x_2 & +...+ & a_{nn}x_n & = & b_n \end{array}$$

A matriz A dos coeficientes, pode ser decomposta em $A = L + D + R$, onde

L é a matriz inferior de A (o resto é zero)

D é a matriz contendo apenas a diagonal principal de A, o resto é zero.

R é a matriz superior de A (o resto é zero)

Veja por exemplo, se

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \therefore L = \begin{pmatrix} 0 & 0 & 0 \\ 4 & 0 & 0 \\ 7 & 8 & 0 \end{pmatrix},$$

e

$$D = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 9 \end{pmatrix}, R = \begin{pmatrix} 0 & 2 & 3 \\ 0 & 0 & 6 \\ 0 & 0 & 0 \end{pmatrix}$$

Supondo $det(D) \neq 0$, pode-se escrever: $(L + D + R)x = b$ e daqui $Dx = -(L + R)x + b$. e $x = -D^{-1}(L + R)x + D^{-1}b$. Chamando $B = -D^{-1}(L + R)$ e $g = D^{-1}.b$ obtem-se a equação geral deste método que é

$$x = Bx + g$$

. Note que é uma equação recursiva (ou iterativa) já que o x aparece nos 2 lados da mesma. A idéia aqui é atribuir um valor a x e com ele calcular um novo x, que é jogado no lugar do x velho e gera um novo x e assim sucessivamente até haver a convergência esperada (de antemão). Como supusemos $det(D) \neq 0$ pode-se dividir cada equação do sistema pelo coeficiente da diagonal principal, resultando

$$A^* = L^* + I + R^*$$

Agora o processo iterativo pode ser definido como

$$x^{k+1} = -(L^* + R^*)x^k + b^*$$

onde os elementos de L^*, R^* e b^* são:

$l_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i > j$ e zero senão

$d_{ij} = a_{ij}$ se $i = j$ e zero senão

$r_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i < j$ e zero senão

$b_ij^* = \frac{b_i}{a_{ii}}$

Critérios de convergência

Depois de dividida a matriz pelo elemento da diagonal principal, deve-se ter o **critério das linhas** que diz

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}^*| < 1$$

OU então o **critério das colunas** que diz

$$max_{1 \leq j \leq n} \sum_{i=1, i \neq j}^{n} |a_{ij}^*| < 1$$

Note que qualquer um dos dois critérios sendo satisfeitos, isso garante que o sistema tem convergência.

Definindo uma matriz como estritamente diagonalmente dominante se

$$\sum_{j=1, j \neq i}^{n} |a_{ij}| < |a_{ii}|$$

e se A é estritamente diagonalmente dominante, então A satisfaz o critério das linhas, e o sistema converge. Então, um bom critério de convergência pode ser

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}|* < 1$$

Exemplo

Seja resolver o sistema abaixo, usando o método de Jacobi

$$\begin{array}{ccccccc} 10x & + & 2y & + & z & = & 7 \\ x & + & 5y & + & z & = & -8 \\ 2x & + & 3y & + & 10z & = & 6 \end{array}$$

com $x^0 = (0.7, -1.6\ e\ 0.6)^t$ e $\epsilon < 0.11$.

Convergência ?

$|a_{12}| + |a_{13}| < |a_{11}|$? ou $|2| + |1| < |10|$? R: sim
$|a_{21}| + |a_{23}| < |a_{22}|$? ou $|1| + |1| < |5|$? R: sim
$|a_{31}| + |a_{32}| < |a_{33}|$? ou $|2| + |3| < |10|$? R: sim
logo, o sistema convergirá!

Solução

Dividindo cada equação pelo elemento da diagonal principal, fica

$$\begin{array}{ccccccc} x & + & 0.2y & + & 0.1z & = & 0.7 \\ 0.2x & + & y & + & 0.2z & = & -1.6 \\ 0.2x & + & 0.3y & + & z & = & 0.6 \end{array}$$

Apenas para ilustrar o critério das linhas para testar convergência (desnecessário no caso, já que já vimos que o sistema converge, mas em tese...)
$|a_{12}^*| + |a_{13}^*| = |0.2| + |0.1| = 0.3$ e
$|a_{21}^*| + |a_{23}^*| = |0.2| + |0.2| = 0.4$ e
$|a_{31}^*| + |a_{32}^*| = |0.2| + |0.3| = 0.5$ e
$\Rightarrow\ max_{1 \leq i \leq n}(0.3, 0.4\ 0.5) = 0.5 < 1$

Iterações

$x^{k+1} = -0.2y^k - 0.1z^k + 0.7$
$y^{k+1} = -0.2x^k - 0.2z^k - 1.6$
$z^{k+1} = -0.2x^k - 0.3y^k + 0.6$
Começando com $(0.7,\ -1.6,\ 0.6)$ obtemos uma primeira aproximação que é $(0.96,\ -1.86,\ 0.94)$ e jogando estes novos valores na fórmula iterativa, obtem-se:

$x \rightarrow 0.7,\ 0.96,\ 0.978,\ 0.9994,\ 0.9979$
$y \rightarrow -1.6,\ -1.86,\ -1.98,\ -1.9888,\ -1.9996$
$z \rightarrow 0.6,\ 0.94,\ 0.966,\ 0.9984,\ 0.9968$

O critério de parada pode ser
$max(x^{k+1} - x^k) < \epsilon$. No exemplo acima $0.0108 < 0.011$. Ou seja, neste caso, a solução procurada é $x = 0.9979$, $y = -1.9996$ e $z = 0.9978$. Salta aos olhos neste caso que se a tolerância for um pouco menor, a resposta exata será encontrada que é $(1, -2, 1)$. Basta lançar este resultado no sistema original que a exatidão será confirmada.

Solução Python

```
#jacobi
def jacobi(a,b):
 import numpy
 n=len(a) #numero de linhas de a
          # len(a[0])=colunas de a
 xvelho=numpy.zeros((n),float)
 erro=numpy.zeros((n),float)
 xnovo=numpy.zeros((n),float)
 f=numpy.zeros((n,len(a[0])),float)
 d=numpy.zeros((len(a[0])),float)
 tol=0.1
 for i in range(0,n):
  for j in range(0,n):
   if i==j:
    f[i,j]=0
   else:
    f[i,j]=-a[i,j]/a[i,i]
 for i in range(0,n):
  d[i]=b[i]/a[i,i]
 erromax=tol+1
 while erromax>tol:
  for i in range(0,n):
   xnovo[i]=0
   for k in range(0,n):
    xnovo[i]=xnovo[i]+f[i,k]*xvelho[k]
  for i in range(0,n):
   xnovo[i]=xnovo[i]+d[i]
  for i in range(0,n):
   erro[i]=abs(xnovo[i]-xvelho[i])
  erromax=erro[0]
  for i in range(1,n):
   if erro[i]>erromax:
    erromax=erro[i]
  for i in range(0,n):
   xvelho[i]=xnovo[i]
 return(xnovo)
import numpy
# para usar faça:
# a=numpy.array([[10,7,3],[1,5,4],[3,5,9]],float)
# b=numpy.array([22,33,44],float)
# print(jacobi(a,b))
```

☞ Para você fazer

Resolva usando este método e anexe a listagem do programa (ou copie no verso desta folha):

$$A = \begin{pmatrix} 84 & 2 & 29 & 15 & 30 \\ 35 & 96 & 27 & 12 & 21 \\ 13 & 6 & 40 & 3 & 10 \\ 24 & 25 & 31 & 96 & 9 \\ 5 & 34 & 33 & 4 & 78 \end{pmatrix}$$

e

$$B = (1254, 1760, 779, 1473, 1251)$$

As respostas encontradas, com $\epsilon < 0.0001$ e sugestão inicial igual à coluna B são:

x_1	x_2	x_3	x_4	x_5

111-70346 - 01/10

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:51.2
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
70858 JHONATAN K. RIBEIRO 19HEL111 - 20 apos 01/10, 50% _____ / _____ / _____

Sistemas Lineares - Jacobi

Este método é iterativo, ou seja, não exato, já que parte-se de um valor atribuído inicial e mediante buscas locais sucessivas, o mesmo converge para a resposta correta. A diferença entre o resultado sugerido e aquele real, é denominada erro, e pode ser tornada tão pequena quanto se queira. O método entretanto tem um senão, que é um critério de convergência bastante rígido. Se não obedecido o algoritmo não converge para a resposta correta, e portanto o método é inútil.

Considere $A.x = b$ um sistema de quações lineares de ordem n e cujo $det(A) \neq 0$. A maneira extensa de escrever tal sistema é

$$\begin{array}{ccccccc} a_{11}x_1 & + & a_{12}x_2 & +...+ & a_{1n}x_n & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & +...+ & a_{2n}x_n & = & b_2 \\ ... & & & & & & \\ a_{n1}x_1 & + & a_{n2}x_2 & +...+ & a_{nn}x_n & = & b_n \end{array}$$

A matriz A dos coeficientes, pode ser decomposta em $A = L + D + R$, onde

L é a matriz inferior de A (o resto é zero)

D é a matriz contendo apenas a diagonal principal de A, o resto é zero.

R é a matriz superior de A (o resto é zero)

Veja por exemplo, se

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \therefore L = \begin{pmatrix} 0 & 0 & 0 \\ 4 & 0 & 0 \\ 7 & 8 & 0 \end{pmatrix},$$

e

$$D = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 9 \end{pmatrix}, R = \begin{pmatrix} 0 & 2 & 3 \\ 0 & 0 & 6 \\ 0 & 0 & 0 \end{pmatrix}$$

Supondo $det(D) \neq 0$, pode-se escrever: $(L + D + R)x = b$ e daqui $Dx = -(L + R)x + b$. e $x = -D^{-1}(L + R)x + D^{-1}b$. Chamando $B = -D^{-1}(L + R)$ e $g = D^{-1}.b$ obtem-se a equação geral deste método que é

$$x = Bx + g$$

. Note que é uma equação recursiva (ou iterativa) já que o x aparece nos 2 lados da mesma. A idéia aqui é atribuir um valor a x e com ele calcular um novo x, que é jogado no lugar do x velho e gera um novo x e assim sucessivamente até haver a convergência esperada (de antemão). Como supusemos $det(D) \neq 0$ pode-se dividir cada equação do sistema pelo coeficiente da diagonal principal, resultando

$$A^* = L^* + I + R^*$$

Agora o processo iterativo pode ser definido como

$$x^{k+1} = -(L^* + R^*)x^k + b^*$$

onde os elementos de L^*, R^* e b^* são:

$l_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i > j$ e zero senão

$d_{ij} = a_{ij}$ se $i = j$ e zero senão

$r_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i < j$ e zero senão

$b_ij^* = \frac{b_i}{a_{ii}}$

Critérios de convergência

Depois de dividida a matriz pelo elemento da diagonal principal, deve-se ter o **critério das linhas** que diz

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}^*| < 1$$

OU então o **critério das colunas** que diz

$$max_{1 \leq j \leq n} \sum_{i=1, i \neq j}^{n} |a_{ij}^*| < 1$$

Note que qualquer um dos dois critérios sendo satisfeitos, isso garante que o sistema tem convergência.

Definindo uma matriz como estritamente diagonalmente dominante se

$$\sum_{j=1, j \neq i}^{n} |a_{ij}| < |a_{ii}|$$

e se A é estritamente diagonalmente dominante, então A satisfaz o critério das linhas, e o sistema converge. Então, um bom critério de convergência pode ser

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}|* < 1$$

Exemplo

Seja resolver o sistema abaixo, usando o método de Jacobi

$$\begin{array}{ccccccc} 10x & + & 2y & + & z & = & 7 \\ x & + & 5y & + & z & = & -8 \\ 2x & + & 3y & + & 10z & = & 6 \end{array}$$

com $x^0 = (0.7, -1.6\ e\ 0.6)^t$ e $\epsilon < 0.11$.

Convergência ?

$|a_{12}| + |a_{13}| < |a_{11}|$? ou $|2| + |1| < |10|$? R: sim
$|a_{21}| + |a_{23}| < |a_{22}|$? ou $|1| + |1| < |5|$? R: sim
$|a_{31}| + |a_{32}| < |a_{33}|$? ou $|2| + |3| < |10|$? R: sim
logo, o sistema convergirá!

Solução

Dividindo cada equação pelo elemento da diagonal principal, fica

$$\begin{array}{ccccccc} x & + & 0.2y & + & 0.1z & = & 0.7 \\ 0.2x & + & y & + & 0.2z & = & -1.6 \\ 0.2x & + & 0.3y & + & z & = & 0.6 \end{array}$$

Apenas para ilustrar o critério das linhas para testar convergência (desnecessário no caso, já que já vimos que o sistema converge, mas em tese...)
$|a_{12}^*| + |a_{13}^*| = |0.2| + |0.1| = 0.3$ e
$|a_{21}^*| + |a_{23}^*| = |0.2| + |0.2| = 0.4$ e
$|a_{31}^*| + |a_{32}^*| = |0.2| + |0.3| = 0.5$ e
$\Rightarrow\ max_{1 \leq i \leq n}(0.3, 0.4\ 0.5) = 0.5 < 1$

Iterações

$x^{k+1} = -0.2y^k - 0.1z^k + 0.7$
$y^{k+1} = -0.2x^k - 0.2z^k - 1.6$
$z^{k+1} = -0.2x^k - 0.3y^k + 0.6$
Começando com $(0.7,\ -1.6,\ 0.6)$ obtemos uma primeira aproximação que é $(0.96,\ -1.86,\ 0.94)$ e jogando estes novos valores na fórmula iterativa, obtem-se:

$x \rightarrow 0.7,\ 0.96,\ 0.978,\ 0.9994,\ 0.9979$
$y \rightarrow -1.6,\ -1.86,\ -1.98,\ -1.9888,\ -1.9996$
$z \rightarrow 0.6,\ 0.94,\ 0.966,\ 0.9984,\ 0.9968$

O critério de parada pode ser
$max(x^{k+1} - x^k) < \epsilon$. No exemplo acima $0.0108 < 0.011$. Ou seja, neste caso, a solução procurada é $x = 0.9979$, $y = -1.9996$ e $z = 0.9978$. Salta aos olhos neste caso que se a tolerância for um pouco menor, a resposta exata será encontrada que é $(1, -2, 1)$. Basta lançar este resultado no sistema original que a exatidão será confirmada.

Solução Python

```
#jacobi
def jacobi(a,b):
 import numpy
 n=len(a) #numero de linhas de a
          # len(a[0])=colunas de a
 xvelho=numpy.zeros((n),float)
 erro=numpy.zeros((n),float)
 xnovo=numpy.zeros((n),float)
 f=numpy.zeros((n,len(a[0])),float)
 d=numpy.zeros((len(a[0])),float)
 tol=0.1
 for i in range(0,n):
  for j in range(0,n):
   if i==j:
    f[i,j]=0
   else:
    f[i,j]=-a[i,j]/a[i,i]
 for i in range(0,n):
  d[i]=b[i]/a[i,i]
 erromax=tol+1
 while erromax>tol:
  for i in range(0,n):
   xnovo[i]=0
   for k in range(0,n):
    xnovo[i]=xnovo[i]+f[i,k]*xvelho[k]
  for i in range(0,n):
   xnovo[i]=xnovo[i]+d[i]
  for i in range(0,n):
   erro[i]=abs(xnovo[i]-xvelho[i])
  erromax=erro[0]
  for i in range(1,n):
   if erro[i]>erromax:
    erromax=erro[i]
  for i in range(0,n):
   xvelho[i]=xnovo[i]
 return(xnovo)
import numpy
# para usar faça:
# a=numpy.array([[10,7,3],[1,5,4],[3,5,9]],float)
# b=numpy.array([22,33,44],float)
# print(jacobi(a,b))
```

☞ Para você fazer

Resolva usando este método e anexe a listagem do programa (ou copie no verso desta folha):

$$A = \begin{pmatrix} 96 & 33 & 30 & 11 & 20 \\ 5 & 41 & 13 & 16 & 1 \\ 2 & 3 & 62 & 32 & 24 \\ 27 & 28 & 18 & 94 & 12 \\ 6 & 7 & 14 & 25 & 55 \end{pmatrix}$$

e

$$B = (2524, 771, 1515, 2568, 1521)$$

As respostas encontradas, com $\epsilon < 0.0001$ e sugestão inicial igual à coluna B são:

x_1	x_2	x_3	x_4	x_5

111-70858 - 01/10

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:51.2
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
70353 JODERSON GOMES DA SILVA 19HEL111 - 21 apos 01/10, 50% _____ / _____ / _____

Sistemas Lineares - Jacobi

Este método é iterativo, ou seja, não exato, já que parte-se de um valor atribuído inicial e mediante buscas locais sucessivas, o mesmo converge para a resposta correta. A diferença entre o resultado sugerido e aquele real, é denominada erro, e pode ser tornada tão pequena quanto se queira. O método entretanto tem um senão, que é um critério de convergência bastante rígido. Se não obedecido o algoritmo não converge para a resposta correta, e portanto o método é inútil.

Considere $A.x = b$ um sistema de quações lineares de ordem n e cujo $det(A) \neq 0$. A maneira extensa de escrever tal sistema é

$$\begin{array}{ccccccccc} a_{11}x_1 & + & a_{12}x_2 & +...+ & a_{1n}x_n & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & +...+ & a_{2n}x_n & = & b_2 \\ ... & & & & & & \\ a_{n1}x_1 & + & a_{n2}x_2 & +...+ & a_{nn}x_n & = & b_n \end{array}$$

A matriz A dos coeficientes, pode ser decomposta em $A = L + D + R$, onde

L é a matriz inferior de A (o resto é zero)

D é a matriz contendo apenas a diagonal principal de A, o resto é zero.

R é a matriz superior de A (o resto é zero)

Veja por exemplo, se

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \therefore L = \begin{pmatrix} 0 & 0 & 0 \\ 4 & 0 & 0 \\ 7 & 8 & 0 \end{pmatrix},$$

e

$$D = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 9 \end{pmatrix}, R = \begin{pmatrix} 0 & 2 & 3 \\ 0 & 0 & 6 \\ 0 & 0 & 0 \end{pmatrix}$$

Supondo $det(D) \neq 0$, pode-se escrever: $(L + D + R)x = b$ e daqui $Dx = -(L + R)x + b.$ e $x = -D^{-1}(L + R)x + D^{-1}b$. Chamando $B = -D^{-1}(L + R)$ e $g = D^{-1}.b$ obtem-se a equação geral deste método que é

$$x = Bx + g$$

. Note que é uma equação recursiva (ou iterativa) já que o x aparece nos 2 lados da mesma. A idéia aqui é atribuir um valor a x e com ele calcular um novo x, que é jogado no lugar do x velho e gera um novo x e assim sucessivamente até haver a convergência esperada (de antemão). Como supusemos $det(D) \neq 0$ pode-se dividir cada equação do sistema pelo coeficiente da diagonal principal, resultando

$$A^* = L^* + I + R^*$$

Agora o processo iterativo pode ser definido como

$$x^{k+1} = -(L^* + R^*)x^k + b^*$$

onde os elementos de L^*, R^* e b^* são:

$l_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i > j$ e zero senão

$d_{ij} = a_{ij}$ se $i = j$ e zero senão

$r_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i < j$ e zero senão

$b_ij^* = \frac{b_i}{a_{ii}}$

Critérios de convergência

Depois de dividida a matriz pelo elemento da diagonal principal, deve-se ter o **critério das linhas** que diz

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}^*| < 1$$

OU então o **critério das colunas** que diz

$$max_{1 \leq j \leq n} \sum_{i=1, i \neq j}^{n} |a_{ij}^*| < 1$$

Note que qualquer um dos dois critérios sendo satisfeitos, isso garante que o sistema tem convergência.

Definindo uma matriz como estritamente diagonalmente dominante se

$$\sum_{j=1, j \neq i}^{n} |a_{ij}| < |a_{ii}|$$

e se A é estritamente diagonalmente dominante, então A satisfaz o critério das linhas, e o sistema converge. Então, um bom critério de convergência pode ser

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}|* < 1$$

Exemplo

Seja resolver o sistema abaixo, usando o método de Jacobi

$$\begin{array}{ccccccc} 10x & + & 2y & + & z & = & 7 \\ x & + & 5y & + & z & = & -8 \\ 2x & + & 3y & + & 10z & = & 6 \end{array}$$

com $x^0 = (0.7, -1.6\ e\ 0.6)^t$ e $\epsilon < 0.11$.

Convergência ?

$|a_{12}| + |a_{13}| < |a_{11}|$? ou $|2| + |1| < |10|$? R: sim
$|a_{21}| + |a_{23}| < |a_{22}|$? ou $|1| + |1| < |5|$? R: sim
$|a_{31}| + |a_{32}| < |a_{33}|$? ou $|2| + |3| < |10|$? R: sim
logo, o sistema convergirá!

Solução

Dividindo cada equação pelo elemento da diagonal principal, fica

$$\begin{array}{ccccccc} x & + & 0.2y & + & 0.1z & = & 0.7 \\ 0.2x & + & y & + & 0.2z & = & -1.6 \\ 0.2x & + & 0.3y & + & z & = & 0.6 \end{array}$$

Apenas para ilustrar o critério das linhas para testar convergência (desnecessário no caso, já que já vimos que o sistema converge, mas em tese...)
$|a_{12}^*| + |a_{13}^*| = |0.2| + |0.1| = 0.3$ e
$|a_{21}^*| + |a_{23}^*| = |0.2| + |0.2| = 0.4$ e
$|a_{31}^*| + |a_{32}^*| = |0.2| + |0.3| = 0.5$ e
$\Rightarrow\ max_{1 \leq i \leq n}(0.3, 0.4\ 0.5) = 0.5 < 1$

Iterações

$x^{k+1} = -0.2y^k - 0.1z^k + 0.7$
$y^{k+1} = -0.2x^k - 0.2z^k - 1.6$
$z^{k+1} = -0.2x^k - 0.3y^k + 0.6$
Começando com $(0.7,\ -1.6,\ 0.6)$ obtemos uma primeira aproximação que é $(0.96,\ -1.86,\ 0.94)$ e jogando estes novos valores na fórmula iterativa, obtem-se:

$x \rightarrow 0.7,\ 0.96,\ 0.978,\ 0.9994,\ 0.9979$
$y \rightarrow -1.6,\ -1.86,\ -1.98,\ -1.9888,\ -1.9996$
$z \rightarrow 0.6,\ 0.94,\ 0.966,\ 0.9984,\ 0.9968$

O critério de parada pode ser
$max(x^{k+1} - x^k) < \epsilon$. No exemplo acima $0.0108 < 0.011$. Ou seja, neste caso, a solução procurada é $x = 0.9979$, $y = -1.9996$ e $z = 0.9978$. Salta aos olhos neste caso que se a tolerância for um pouco menor, a resposta exata será encontrada que é $(1, -2, 1)$. Basta lançar este resultado no sistema original que a exatidão será confirmada.

Solução Python

```
#jacobi
def jacobi(a,b):
 import numpy
 n=len(a) #numero de linhas de a
          # len(a[0])=colunas de a
 xvelho=numpy.zeros((n),float)
 erro=numpy.zeros((n),float)
 xnovo=numpy.zeros((n),float)
 f=numpy.zeros((n,len(a[0])),float)
 d=numpy.zeros((len(a[0])),float)
 tol=0.1
 for i in range(0,n):
  for j in range(0,n):
   if i==j:
    f[i,j]=0
   else:
    f[i,j]=-a[i,j]/a[i,i]
 for i in range(0,n):
  d[i]=b[i]/a[i,i]
 erromax=tol+1
 while erromax>tol:
  for i in range(0,n):
   xnovo[i]=0
   for k in range(0,n):
    xnovo[i]=xnovo[i]+f[i,k]*xvelho[k]
  for i in range(0,n):
   xnovo[i]=xnovo[i]+d[i]
  for i in range(0,n):
   erro[i]=abs(xnovo[i]-xvelho[i])
  erromax=erro[0]
  for i in range(1,n):
   if erro[i]>erromax:
    erromax=erro[i]
  for i in range(0,n):
   xvelho[i]=xnovo[i]
 return(xnovo)
import numpy
# para usar faça:
# a=numpy.array([[10,7,3],[1,5,4],[3,5,9]],float)
# b=numpy.array([22,33,44],float)
# print(jacobi(a,b))
```

☞ Para você fazer

Resolva usando este método e anexe a listagem do programa (ou copie no verso desta folha):

$$A = \begin{pmatrix} 107 & 30 & 9 & 33 & 28 \\ 10 & 73 & 12 & 26 & 21 \\ 17 & 20 & 62 & 4 & 19 \\ 6 & 31 & 1 & 46 & 5 \\ 11 & 8 & 14 & 15 & 50 \end{pmatrix}$$

e

$$B = (2406, 1204, 1003, 835, 777)$$

As respostas encontradas, com $\epsilon < 0.0001$ e sugestão inicial igual à coluna B são:

x_1	x_2	x_3	x_4	x_5

111-70353 - 01/10

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:51.2
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
70360 LEONARDO CORNELIO KUIPERS
19HEL111 - 22 apos 01/10, 50% _____ / / _____

Sistemas Lineares - Jacobi

Este método é iterativo, ou seja, não exato, já que parte-se de um valor atribuído inicial e mediante buscas locais sucessivas, o mesmo converge para a resposta correta. A diferença entre o resultado sugerido e aquele real, é denominada erro, e pode ser tornada tão pequena quanto se queira. O método entretanto tem um senão, que é um critério de convergência bastante rígido. Se não obedecido o algoritmo não converge para a resposta correta, e portanto o método é inútil.

Considere $A.x = b$ um sistema de quações lineares de ordem n e cujo $det(A) \neq 0$. A maneira extensa de escrever tal sistema é

$$\begin{array}{lcllcl} a_{11}x_1 & + & a_{12}x_2 & +...+ & a_{1n}x_n & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & +...+ & a_{2n}x_n & = & b_2 \\ ... & & & & & & \\ a_{n1}x_1 & + & a_{n2}x_2 & +...+ & a_{nn}x_n & = & b_n \end{array}$$

A matriz A dos coeficientes, pode ser decomposta em $A = L + D + R$, onde

L é a matriz inferior de A (o resto é zero)

D é a matriz contendo apenas a diagonal principal de A, o resto é zero.

R é a matriz superior de A (o resto é zero)

Veja por exemplo, se

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \therefore L = \begin{pmatrix} 0 & 0 & 0 \\ 4 & 0 & 0 \\ 7 & 8 & 0 \end{pmatrix},$$

e

$$D = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 9 \end{pmatrix}, R = \begin{pmatrix} 0 & 2 & 3 \\ 0 & 0 & 6 \\ 0 & 0 & 0 \end{pmatrix}$$

Supondo $det(D) \neq 0$, pode-se escrever: $(L + D + R)x = b$ e daqui $Dx = -(L + R)x + b$. e $x = -D^{-1}(L + R)x + D^{-1}b$. Chamando $B = -D^{-1}(L + R)$ e $g = D^{-1}.b$ obtem-se a equação geral deste método que é

$$x = Bx + g$$

. Note que é uma equação recursiva (ou iterativa) já que o x aparece nos 2 lados da mesma. A idéia aqui é atribuir um valor a x e com ele calcular um novo x, que é jogado no lugar do x velho e gera um novo x e assim sucessivamente até haver a convergência esperada (de antemão). Como supusemos $det(D) \neq 0$ pode-se dividir cada equação do sistema pelo coeficiente da diagonal principal, resultando

$$A^* = L^* + I + R^*$$

Agora o processo iterativo pode ser definido como

$$x^{k+1} = -(L^* + R^*)x^k + b^*$$

onde os elementos de L^*, R^* e b^* são:

$l_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i > j$ e zero senão

$d_{ij} = a_{ij}$ se $i = j$ e zero senão

$r_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i < j$ e zero senão

$b_ij^* = \frac{b_i}{a_{ii}}$

Critérios de convergência

Depois de dividida a matriz pelo elemento da diagonal principal, deve-se ter o **critério das linhas** que diz

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}^*| < 1$$

OU então o **critério das colunas** que diz

$$max_{1 \leq j \leq n} \sum_{i=1, i \neq j}^{n} |a_{ij}^*| < 1$$

Note que qualquer um dos dois critérios sendo satisfeitos, isso garante que o sistema tem convergência.

Definindo uma matriz como estritamente diagonalmente dominante se

$$\sum_{j=1, j \neq i}^{n} |a_{ij}| < |a_{ii}|$$

e se A é estritamente diagonalmente dominante, então A satisfaz o critério das linhas, e o sistema converge. Então, um bom critério de convergência pode ser

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}^|* < 1$$

Exemplo

Seja resolver o sistema abaixo, usando o método de Jacobi

$$\begin{array}{rcrcrcr} 10x & + & 2y & + & z & = & 7 \\ x & + & 5y & + & z & = & -8 \\ 2x & + & 3y & + & 10z & = & 6 \end{array}$$

com $x^0 = (0.7, -1.6\ e\ 0.6)^t$ e $\epsilon < 0.11$.

Convergência ?

$|a_{12}| + |a_{13}| < |a_{11}|$? ou $|2| + |1| < |10|$? R: sim
$|a_{21}| + |a_{23}| < |a_{22}|$? ou $|1| + |1| < |5|$? R: sim
$|a_{31}| + |a_{32}| < |a_{33}|$? ou $|2| + |3| < |10|$? R: sim
logo, o sistema convergirá!

Solução

Dividindo cada equação pelo elemento da diagonal principal, fica

$$\begin{array}{rcrcrcr} x & + & 0.2y & + & 0.1z & = & 0.7 \\ 0.2x & + & y & + & 0.2z & = & -1.6 \\ 0.2x & + & 0.3y & + & z & = & 0.6 \end{array}$$

Apenas para ilustrar o critério das linhas para testar convergência (desnecessário no caso, já que já vimos que o sistema converge, mas em tese...)
$|a_{12}^*| + |a_{13}^*| = |0.2| + |0.1| = 0.3$ e
$|a_{21}^*| + |a_{23}^*| = |0.2| + |0.2| = 0.4$ e
$|a_{31}^*| + |a_{32}^*| = |0.2| + |0.3| = 0.5$ e
$\Rightarrow\ max_{1 \leq i \leq n}(0.3, 0.4\ 0.5) = 0.5 < 1$

Iterações

$x^{k+1} = -0.2y^k - 0.1z^k + 0.7$
$y^{k+1} = -0.2x^k - 0.2z^k - 1.6$
$z^{k+1} = -0.2x^k - 0.3y^k + 0.6$
Começando com $(0.7,\ -1.6,\ 0.6)$ obtemos uma primeira aproximação que é $(0.96,\ -1.86,\ 0.94)$ e jogando estes novos valores na fórmula iterativa, obtem-se:

$x \rightarrow 0.7,\ 0.96,\ 0.978,\ 0.9994,\ 0.9979$
$y \rightarrow -1.6,\ -1.86,\ -1.98,\ -1.9888,\ -1.9996$
$z \rightarrow 0.6,\ 0.94,\ 0.966,\ 0.9984,\ 0.9968$

O critério de parada pode ser
$max(x^{k+1} - x^k) < \epsilon$. No exemplo acima $0.0108 < 0.011$. Ou seja, neste caso, a solução procurada é $x = 0.9979$, $y = -1.9996$ e $z = 0.9978$. Salta aos olhos neste caso que se a tolerância for um pouco menor, a resposta exata será encontrada que é $(1, -2, 1)$. Basta lançar este resultado no sistema original que a exatidão será confirmada.

Solução Python

```
#jacobi
def jacobi(a,b):
 import numpy
 n=len(a) #numero de linhas de a
          # len(a[0])=colunas de a
 xvelho=numpy.zeros((n),float)
 erro=numpy.zeros((n),float)
 xnovo=numpy.zeros((n),float)
 f=numpy.zeros((n,len(a[0])),float)
 d=numpy.zeros((len(a[0])),float)
 tol=0.1
 for i in range(0,n):
  for j in range(0,n):
   if i==j:
    f[i,j]=0
   else:
    f[i,j]=-a[i,j]/a[i,i]
 for i in range(0,n):
  d[i]=b[i]/a[i,i]
 erromax=tol+1
 while erromax>tol:
  for i in range(0,n):
   xnovo[i]=0
   for k in range(0,n):
    xnovo[i]=xnovo[i]+f[i,k]*xvelho[k]
  for i in range(0,n):
   xnovo[i]=xnovo[i]+d[i]
  for i in range(0,n):
   erro[i]=abs(xnovo[i]-xvelho[i])
  erromax=erro[0]
  for i in range(1,n):
   if erro[i]>erromax:
    erromax=erro[i]
  for i in range(0,n):
   xvelho[i]=xnovo[i]
 return(xnovo)
import numpy
# para usar faça:
# a=numpy.array([[10,7,3],[1,5,4],[3,5,9]],float)
# b=numpy.array([22,33,44],float)
# print(jacobi(a,b))
```

☞ Para você fazer

Resolva usando este método e anexe a listagem do programa (ou copie no verso desta folha):

$$A = \begin{pmatrix} 36 & 6 & 10 & 8 & 11 \\ 3 & 60 & 17 & 30 & 7 \\ 29 & 27 & 109 & 25 & 21 \\ 1 & 24 & 20 & 62 & 16 \\ 9 & 33 & 28 & 19 & 91 \end{pmatrix}$$

e

$$B = (917, 1150, 2871, 1193, 2523)$$

As respostas encontradas, com $\epsilon < 0.0001$ e sugestão inicial igual à coluna B são:

x_1	x_2	x_3	x_4	x_5

111-70360 - 01/10

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:51.2
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
70377 LUCAS ANTONIO THEODORO SANTOS
19HEL111 - 23 apos 01/10, 50% _____ / / _____

Sistemas Lineares - Jacobi

Este método é iterativo, ou seja, não exato, já que parte-se de um valor atribuído inicial e mediante buscas locais sucessivas, o mesmo converge para a resposta correta. A diferença entre o resultado sugerido e aquele real, é denominada erro, e pode ser tornada tão pequena quanto se queira. O método entretanto tem um senão, que é um critério de convergência bastante rígido. Se não obedecido o algoritmo não converge para a resposta correta, e portanto o método é inútil.

Considere $A.x = b$ um sistema de quações lineares de ordem n e cujo $det(A) \neq 0$. A maneira extensa de escrever tal sistema é

$$\begin{array}{ccccccc} a_{11}x_1 & + & a_{12}x_2 & +...+ & a_{1n}x_n & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & +...+ & a_{2n}x_n & = & b_2 \\ ... & & & & & & \\ a_{n1}x_1 & + & a_{n2}x_2 & +...+ & a_{nn}x_n & = & b_n \end{array}$$

A matriz A dos coeficientes, pode ser decomposta em $A = L + D + R$, onde

L é a matriz inferior de A (o resto é zero)

D é a matriz contendo apenas a diagonal principal de A, o resto é zero.

R é a matriz superior de A (o resto é zero)

Veja por exemplo, se

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \therefore L = \begin{pmatrix} 0 & 0 & 0 \\ 4 & 0 & 0 \\ 7 & 8 & 0 \end{pmatrix},$$

e

$$D = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 9 \end{pmatrix}, R = \begin{pmatrix} 0 & 2 & 3 \\ 0 & 0 & 6 \\ 0 & 0 & 0 \end{pmatrix}$$

Supondo $det(D) \neq 0$, pode-se escrever: $(L + D + R)x = b$ e daqui $Dx = -(L + R)x + b$. e $x = -D^{-1}(L + R)x + D^{-1}b$. Chamando $B = -D^{-1}(L + R)$ e $g = D^{-1}.b$ obtem-se a equação geral deste método que é

$$x = Bx + g$$

. Note que é uma equação recursiva (ou iterativa) já que o x aparece nos 2 lados da mesma. A idéia aqui é atribuir um valor a x e com ele calcular um novo x, que é jogado no lugar do x velho e gera um novo x e assim sucessivamente até haver a convergência esperada (de antemão). Como supusemos $det(D) \neq 0$ pode-se dividir cada equação do sistema pelo coeficiente da diagonal principal, resultando

$$A^* = L^* + I + R^*$$

Agora o processo iterativo pode ser definido como

$$x^{k+1} = -(L^* + R^*)x^k + b^*$$

onde os elementos de L^*, R^* e b^* são:

$l_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i > j$ e zero senão

$d_{ij} = a_{ij}$ se $i = j$ e zero senão

$r_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i < j$ e zero senão

$b_ij^* = \frac{b_i}{a_{ii}}$

Critérios de convergência

Depois de dividida a matriz pelo elemento da diagonal principal, deve-se ter o **critério das linhas** que diz

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}^*| < 1$$

OU então o **critério das colunas** que diz

$$max_{1 \leq j \leq n} \sum_{i=1, i \neq j}^{n} |a_{ij}^*| < 1$$

Note que qualquer um dos dois critérios sendo satisfeitos, isso garante que o sistema tem convergência.

Definindo uma matriz como estritamente diagonalmente dominante se

$$\sum_{j=1, j \neq i}^{n} |a_{ij}| < |a_{ii}|$$

e se A é estritamente diagonalmente dominante, então A satisfaz o critério das linhas, e o sistema converge. Então, um bom critério de convergência pode ser

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}^|* < 1$$

Exemplo

Seja resolver o sistema abaixo, usando o método de Jacobi

$$\begin{array}{ccccccc} 10x & + & 2y & + & z & = & 7 \\ x & + & 5y & + & z & = & -8 \\ 2x & + & 3y & + & 10z & = & 6 \end{array}$$

com $x^0 = (0.7, -1.6\ e\ 0.6)^t$ e $\epsilon < 0.11$.

Convergência ?

$|a_{12}| + |a_{13}| < |a_{11}|$? ou $|2| + |1| < |10|$? R: sim
$|a_{21}| + |a_{23}| < |a_{22}|$? ou $|1| + |1| < |5|$? R: sim
$|a_{31}| + |a_{32}| < |a_{33}|$? ou $|2| + |3| < |10|$? R: sim
logo, o sistema convergirá!

Solução

Dividindo cada equação pelo elemento da diagonal principal, fica

$$\begin{array}{ccccccc} x & + & 0.2y & + & 0.1z & = & 0.7 \\ 0.2x & + & y & + & 0.2z & = & -1.6 \\ 0.2x & + & 0.3y & + & z & = & 0.6 \end{array}$$

Apenas para ilustrar o critério das linhas para testar convergência (desnecessário no caso, já que já vimos que o sistema converge, mas em tese...)
$|a_{12}^*| + |a_{13}^*| = |0.2| + |0.1| = 0.3$ e
$|a_{21}^*| + |a_{23}^*| = |0.2| + |0.2| = 0.4$ e
$|a_{31}^*| + |a_{32}^*| = |0.2| + |0.3| = 0.5$ e
$\Rightarrow\ max_{1 \leq i \leq n}(0.3, 0.4\ 0.5) = 0.5 < 1$

Iterações

$x^{k+1} = -0.2y^k - 0.1z^k + 0.7$
$y^{k+1} = -0.2x^k - 0.2z^k - 1.6$
$z^{k+1} = -0.2x^k - 0.3y^k + 0.6$
Começando com $(0.7,\ -1.6,\ 0.6)$ obtemos uma primeira aproximação que é $(0.96,\ -1.86,\ 0.94)$ e jogando estes novos valores na fórmula iterativa, obtem-se:

$x \rightarrow 0.7,\ 0.96,\ 0.978,\ 0.9994,\ 0.9979$
$y \rightarrow -1.6,\ -1.86,\ -1.98,\ -1.9888,\ -1.9996$
$z \rightarrow 0.6,\ 0.94,\ 0.966,\ 0.9984,\ 0.9968$

O critério de parada pode ser
$max(x^{k+1} - x^k) < \epsilon$. No exemplo acima $0.0108 < 0.011$. Ou seja, neste caso, a solução procurada é $x = 0.9979$, $y = -1.9996$ e $z = 0.9978$. Salta aos olhos neste caso que se a tolerância for um pouco menor, a resposta exata será encontrada que é $(1, -2, 1)$. Basta lançar este resultado no sistema original que a exatidão será confirmada.

Solução Python

```
#jacobi
def jacobi(a,b):
 import numpy
 n=len(a) #numero de linhas de a
          # len(a[0])=colunas de a
 xvelho=numpy.zeros((n),float)
 erro=numpy.zeros((n),float)
 xnovo=numpy.zeros((n),float)
 f=numpy.zeros((n,len(a[0])),float)
 d=numpy.zeros((len(a[0])),float)
 tol=0.1
 for i in range(0,n):
  for j in range(0,n):
   if i==j:
    f[i,j]=0
   else:
    f[i,j]=-a[i,j]/a[i,i]
 for i in range(0,n):
  d[i]=b[i]/a[i,i]
 erromax=tol+1
 while erromax>tol:
  for i in range(0,n):
   xnovo[i]=0
   for k in range(0,n):
    xnovo[i]=xnovo[i]+f[i,k]*xvelho[k]
  for i in range(0,n):
   xnovo[i]=xnovo[i]+d[i]
  for i in range(0,n):
   erro[i]=abs(xnovo[i]-xvelho[i])
  erromax=erro[0]
  for i in range(1,n):
   if erro[i]>erromax:
    erromax=erro[i]
  for i in range(0,n):
   xvelho[i]=xnovo[i]
 return(xnovo)
import numpy
# para usar faça:
# a=numpy.array([[10,7,3],[1,5,4],[3,5,9]],float)
# b=numpy.array([22,33,44],float)
# print(jacobi(a,b))
```

☞ Para você fazer

Resolva usando este método e anexe a listagem do programa (ou copie no verso desta folha):

$$A = \begin{pmatrix} 88 & 16 & 13 & 17 & 33 \\ 22 & 66 & 1 & 32 & 9 \\ 20 & 24 & 83 & 3 & 35 \\ 7 & 8 & 27 & 57 & 12 \\ 4 & 19 & 6 & 28 & 61 \end{pmatrix}$$

e

$$B = (1814, 1123, 1650, 1055, 1164)$$

As respostas encontradas, com $\epsilon < 0.0001$ e sugestão inicial igual à coluna B são:

x_1	x_2	x_3	x_4	x_5

111-70377 - 01/10

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:51.2
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
70384 LUCAS BOHN DUTRA 19HEL111 - 24
apos 01/10, 50% _____ / _____ / _____

Sistemas Lineares - Jacobi

Este método é iterativo, ou seja, não exato, já que parte-se de um valor atribuído inicial e mediante buscas locais sucessivas, o mesmo converge para a resposta correta. A diferença entre o resultado sugerido e aquele real, é denominada erro, e pode ser tornada tão pequena quanto se queira. O método entretanto tem um senão, que é um critério de convergência bastante rígido. Se não obedecido o algoritmo não converge para a resposta correta, e portanto o método é inútil.

Considere $A.x = b$ um sistema de quações lineares de ordem n e cujo $det(A) \neq 0$. A maneira extensa de escrever tal sistema é

$$\begin{array}{ccccccc} a_{11}x_1 & + & a_{12}x_2 & +...+ & a_{1n}x_n & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & +...+ & a_{2n}x_n & = & b_2 \\ ... & & & & & & \\ a_{n1}x_1 & + & a_{n2}x_2 & +...+ & a_{nn}x_n & = & b_n \end{array}$$

A matriz A dos coeficientes, pode ser decomposta em $A = L + D + R$, onde

L é a matriz inferior de A (o resto é zero)

D é a matriz contendo apenas a diagonal principal de A, o resto é zero.

R é a matriz superior de A (o resto é zero)

Veja por exemplo, se

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \therefore L = \begin{pmatrix} 0 & 0 & 0 \\ 4 & 0 & 0 \\ 7 & 8 & 0 \end{pmatrix},$$

e

$$D = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 9 \end{pmatrix}, R = \begin{pmatrix} 0 & 2 & 3 \\ 0 & 0 & 6 \\ 0 & 0 & 0 \end{pmatrix}$$

Supondo $det(D) \neq 0$, pode-se escrever: $(L + D + R)x = b$ e daqui $Dx = -(L + R)x + b$. e $x = -D^{-1}(L + R)x + D^{-1}b$. Chamando $B = -D^{-1}(L + R)$ e $g = D^{-1}.b$ obtem-se a equação geral deste método que é

$$x = Bx + g$$

. Note que é uma equação recursiva (ou iterativa) já que o x aparece nos 2 lados da mesma. A idéia aqui é atribuir um valor a x e com ele calcular um novo x, que é jogado no lugar do x velho e gera um novo x e assim sucessivamente até haver a convergência esperada (de antemão). Como supusemos $det(D) \neq 0$ pode-se dividir cada equação do sistema pelo coeficiente da diagonal principal, resultando

$$A^* = L^* + I + R^*$$

Agora o processo iterativo pode ser definido como

$$x^{k+1} = -(L^* + R^*)x^k + b^*$$

onde os elementos de L^*, R^* e b^* são:

$l_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i > j$ e zero senão

$d_{ij} = a_{ij}$ se $i = j$ e zero senão

$r_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i < j$ e zero senão

$b_ij^* = \frac{b_i}{a_{ii}}$

Critérios de convergência

Depois de dividida a matriz pelo elemento da diagonal principal, deve-se ter o **critério das linhas** que diz

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}^*| < 1$$

OU então o **critério das colunas** que diz

$$max_{1 \leq j \leq n} \sum_{i=1, i \neq j}^{n} |a_{ij}^*| < 1$$

Note que qualquer um dos dois critérios sendo satisfeitos, isso garante que o sistema tem convergência.

Definindo uma matriz como estritamente diagonalmente dominante se

$$\sum_{j=1, j \neq i}^{n} |a_{ij}| < |a_{ii}|$$

e se A é estritamente diagonalmente dominante, então A satisfaz o critério das linhas, e o sistema converge. Então, um bom critério de convergência pode ser

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}|* < 1$$

Exemplo

Seja resolver o sistema abaixo, usando o método de Jacobi

$$\begin{array}{ccccccc} 10x & + & 2y & + & z & = & 7 \\ x & + & 5y & + & z & = & -8 \\ 2x & + & 3y & + & 10z & = & 6 \end{array}$$

com $x^0 = (0.7, -1.6\ e\ 0.6)^t$ e $\epsilon < 0.11$.

Convergência ?

$|a_{12}| + |a_{13}| < |a_{11}|$? ou $|2| + |1| < |10|$? R: sim
$|a_{21}| + |a_{23}| < |a_{22}|$? ou $|1| + |1| < |5|$? R: sim
$|a_{31}| + |a_{32}| < |a_{33}|$? ou $|2| + |3| < |10|$? R: sim
logo, o sistema convergirá!

Solução

Dividindo cada equação pelo elemento da diagonal principal, fica

$$\begin{array}{ccccccc} x & + & 0.2y & + & 0.1z & = & 0.7 \\ 0.2x & + & y & + & 0.2z & = & -1.6 \\ 0.2x & + & 0.3y & + & z & = & 0.6 \end{array}$$

Apenas para ilustrar o critério das linhas para testar convergência (desnecessário no caso, já que já vimos que o sistema converge, mas em tese...)
$|a_{12}^*| + |a_{13}^*| = |0.2| + |0.1| = 0.3$ e
$|a_{21}^*| + |a_{23}^*| = |0.2| + |0.2| = 0.4$ e
$|a_{31}^*| + |a_{32}^*| = |0.2| + |0.3| = 0.5$ e
$\Rightarrow\ max_{1 \leq i \leq n}(0.3, 0.4\ 0.5) = 0.5 < 1$

Iterações

$x^{k+1} = -0.2y^k - 0.1z^k + 0.7$
$y^{k+1} = -0.2x^k - 0.2z^k - 1.6$
$z^{k+1} = -0.2x^k - 0.3y^k + 0.6$
Começando com $(0.7,\ -1.6,\ 0.6)$ obtemos uma primeira aproximação que é $(0.96,\ -1.86,\ 0.94)$ e jogando estes novos valores na fórmula iterativa, obtem-se:

$x \rightarrow 0.7,\ 0.96,\ 0.978,\ 0.9994,\ 0.9979$
$y \rightarrow -1.6,\ -1.86,\ -1.98,\ -1.9888,\ -1.9996$
$z \rightarrow 0.6,\ 0.94,\ 0.966,\ 0.9984,\ 0.9968$

O critério de parada pode ser
$max(x^{k+1} - x^k) < \epsilon$. No exemplo acima $0.0108 < 0.011$. Ou seja, neste caso, a solução procurada é $x = 0.9979$, $y = -1.9996$ e $z = 0.9978$. Salta aos olhos neste caso que se a tolerância for um pouco menor, a resposta exata será encontrada que é $(1, -2, 1)$. Basta lançar este resultado no sistema original que a exatidão será confirmada.

Solução Python

```
#jacobi
def jacobi(a,b):
 import numpy
 n=len(a) #numero de linhas de a
          # len(a[0])=colunas de a
 xvelho=numpy.zeros((n),float)
 erro=numpy.zeros((n),float)
 xnovo=numpy.zeros((n),float)
 f=numpy.zeros((n,len(a[0])),float)
 d=numpy.zeros((len(a[0])),float)
 tol=0.1
 for i in range(0,n):
  for j in range(0,n):
   if i==j:
    f[i,j]=0
   else:
    f[i,j]=-a[i,j]/a[i,i]
 for i in range(0,n):
  d[i]=b[i]/a[i,i]
 erromax=tol+1
 while erromax>tol:
  for i in range(0,n):
   xnovo[i]=0
   for k in range(0,n):
    xnovo[i]=xnovo[i]+f[i,k]*xvelho[k]
  for i in range(0,n):
   xnovo[i]=xnovo[i]+d[i]
  for i in range(0,n):
   erro[i]=abs(xnovo[i]-xvelho[i])
  erromax=erro[0]
  for i in range(1,n):
   if erro[i]>erromax:
    erromax=erro[i]
  for i in range(0,n):
   xvelho[i]=xnovo[i]
 return(xnovo)
import numpy
# para usar faça:
# a=numpy.array([[10,7,3],[1,5,4],[3,5,9]],float)
# b=numpy.array([22,33,44],float)
# print(jacobi(a,b))
```

☞ Para você fazer

Resolva usando este método e anexe a listagem do programa (ou copie no verso desta folha):

$$A = \begin{pmatrix} 69 & 8 & 25 & 17 & 16 \\ 20 & 65 & 9 & 24 & 11 \\ 7 & 1 & 44 & 19 & 12 \\ 22 & 21 & 27 & 89 & 13 \\ 18 & 2 & 6 & 30 & 57 \end{pmatrix}$$

e

$$B = (1469, 985, 658, 1506, 1138)$$

As respostas encontradas, com $\epsilon < 0.0001$ e sugestão inicial igual à coluna B são:

x_1	x_2	x_3	x_4	x_5

111-70384 - 01/10

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:51.2
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
70391 LUCAS CORDEIRO NEVES 19HEL111 - 25 apos 01/10, 50% _____ / _____ / _____

Sistemas Lineares - Jacobi

Este método é iterativo, ou seja, não exato, já que parte-se de um valor atribuído inicial e mediante buscas locais sucessivas, o mesmo converge para a resposta correta. A diferença entre o resultado sugerido e aquele real, é denominada erro, e pode ser tornada tão pequena quanto se queira. O método entretanto tem um senão, que é um critério de convergência bastante rígido. Se não obedecido o algoritmo não converge para a resposta correta, e portanto o método é inútil.

Considere $A.x = b$ um sistema de quações lineares de ordem n e cujo $det(A) \neq 0$. A maneira extensa de escrever tal sistema é

$$\begin{array}{ccccccc} a_{11}x_1 & + & a_{12}x_2 & +...+ & a_{1n}x_n & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & +...+ & a_{2n}x_n & = & b_2 \\ ... & & & & & & \\ a_{n1}x_1 & + & a_{n2}x_2 & +...+ & a_{nn}x_n & = & b_n \end{array}$$

A matriz A dos coeficientes, pode ser decomposta em $A = L + D + R$, onde

L é a matriz inferior de A (o resto é zero)

D é a matriz contendo apenas a diagonal principal de A, o resto é zero.

R é a matriz superior de A (o resto é zero)

Veja por exemplo, se

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \therefore L = \begin{pmatrix} 0 & 0 & 0 \\ 4 & 0 & 0 \\ 7 & 8 & 0 \end{pmatrix},$$

e

$$D = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 9 \end{pmatrix}, R = \begin{pmatrix} 0 & 2 & 3 \\ 0 & 0 & 6 \\ 0 & 0 & 0 \end{pmatrix}$$

Supondo $det(D) \neq 0$, pode-se escrever: $(L + D + R)x = b$ e daqui $Dx = -(L + R)x + b$. e $x = -D^{-1}(L + R)x + D^{-1}b$. Chamando $B = -D^{-1}(L + R)$ e $g = D^{-1}.b$ obtem-se a equação geral deste método que é

$$x = Bx + g$$

. Note que é uma equação recursiva (ou iterativa) já que o x aparece nos 2 lados da mesma. A idéia aqui é atribuir um valor a x e com ele calcular um novo x, que é jogado no lugar do x velho e gera um novo x e assim sucessivamente até haver a convergência esperada (de antemão). Como supusemos $det(D) \neq 0$ pode-se dividir cada equação do sistema pelo coeficiente da diagonal principal, resultando

$$A^* = L^* + I + R^*$$

Agora o processo iterativo pode ser definido como

$$x^{k+1} = -(L^* + R^*)x^k + b^*$$

onde os elementos de L^*, R^* e b^* são:

$l_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i > j$ e zero senão

$d_{ij} = a_{ij}$ se $i = j$ e zero senão

$r_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i < j$ e zero senão

$b_ij^* = \frac{b_i}{a_{ii}}$

Critérios de convergência

Depois de dividida a matriz pelo elemento da diagonal principal, deve-se ter o **critério das linhas** que diz

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}^*| < 1$$

OU então o **critério das colunas** que diz

$$max_{1 \leq j \leq n} \sum_{i=1, i \neq j}^{n} |a_{ij}^*| < 1$$

Note que qualquer um dos dois critérios sendo satisfeitos, isso garante que o sistema tem convergência.

Definindo uma matriz como estritamente diagonalmente dominante se

$$\sum_{j=1, j \neq i}^{n} |a_{ij}| < |a_{ii}|$$

e se A é estritamente diagonalmente dominante, então A satisfaz o critério das linhas, e o sistema converge. Então, um bom critério de convergência pode ser

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}|* < 1$$

Exemplo

Seja resolver o sistema abaixo, usando o método de Jacobi

$$\begin{array}{ccccccc} 10x & + & 2y & + & z & = & 7 \\ x & + & 5y & + & z & = & -8 \\ 2x & + & 3y & + & 10z & = & 6 \end{array}$$

com $x^0 = (0.7, -1.6\ e\ 0.6)^t$ e $\epsilon < 0.11$.

Convergência ?

$|a_{12}| + |a_{13}| < |a_{11}|$? ou $|2| + |1| < |10|$? R: sim
$|a_{21}| + |a_{23}| < |a_{22}|$? ou $|1| + |1| < |5|$? R: sim
$|a_{31}| + |a_{32}| < |a_{33}|$? ou $|2| + |3| < |10|$? R: sim
logo, o sistema convergirá!

Solução

Dividindo cada equação pelo elemento da diagonal principal, fica

$$\begin{array}{ccccccc} x & + & 0.2y & + & 0.1z & = & 0.7 \\ 0.2x & + & y & + & 0.2z & = & -1.6 \\ 0.2x & + & 0.3y & + & z & = & 0.6 \end{array}$$

Apenas para ilustrar o critério das linhas para testar convergência (desnecessário no caso, já que já vimos que o sistema converge, mas em tese...)
$|a_{12}^*| + |a_{13}^*| = |0.2| + |0.1| = 0.3$ e
$|a_{21}^*| + |a_{23}^*| = |0.2| + |0.2| = 0.4$ e
$|a_{31}^*| + |a_{32}^*| = |0.2| + |0.3| = 0.5$ e
$\Rightarrow\ max_{1 \leq i \leq n}(0.3, 0.4\ 0.5) = 0.5 < 1$

Iterações

$x^{k+1} = -0.2y^k - 0.1z^k + 0.7$
$y^{k+1} = -0.2x^k - 0.2z^k - 1.6$
$z^{k+1} = -0.2x^k - 0.3y^k + 0.6$
Começando com $(0.7,\ -1.6,\ 0.6)$ obtemos uma primeira aproximação que é $(0.96,\ -1.86,\ 0.94)$ e jogando estes novos valores na fórmula iterativa, obtem-se:

$x \rightarrow 0.7,\ 0.96,\ 0.978,\ 0.9994,\ 0.9979$
$y \rightarrow -1.6,\ -1.86,\ -1.98,\ -1.9888,\ -1.9996$
$z \rightarrow 0.6,\ 0.94,\ 0.966,\ 0.9984,\ 0.9968$

O critério de parada pode ser
$max(x^{k+1} - x^k) < \epsilon$. No exemplo acima $0.0108 < 0.011$. Ou seja, neste caso, a solução procurada é $x = 0.9979$, $y = -1.9996$ e $z = 0.9978$. Salta aos olhos neste caso que se a tolerância for um pouco menor, a resposta exata será encontrada que é $(1, -2, 1)$. Basta lançar este resultado no sistema original que a exatidão será confirmada.

Solução Python

```
#jacobi
def jacobi(a,b):
 import numpy
 n=len(a) #numero de linhas de a
          # len(a[0])=colunas de a
 xvelho=numpy.zeros((n),float)
 erro=numpy.zeros((n),float)
 xnovo=numpy.zeros((n),float)
 f=numpy.zeros((n,len(a[0])),float)
 d=numpy.zeros((len(a[0])),float)
 tol=0.1
 for i in range(0,n):
  for j in range(0,n):
   if i==j:
    f[i,j]=0
   else:
    f[i,j]=-a[i,j]/a[i,i]
 for i in range(0,n):
  d[i]=b[i]/a[i,i]
 erromax=tol+1
 while erromax>tol:
  for i in range(0,n):
   xnovo[i]=0
   for k in range(0,n):
    xnovo[i]=xnovo[i]+f[i,k]*xvelho[k]
  for i in range(0,n):
   xnovo[i]=xnovo[i]+d[i]
  for i in range(0,n):
   erro[i]=abs(xnovo[i]-xvelho[i])
  erromax=erro[0]
  for i in range(1,n):
   if erro[i]>erromax:
    erromax=erro[i]
  for i in range(0,n):
   xvelho[i]=xnovo[i]
 return(xnovo)
import numpy
# para usar faça:
# a=numpy.array([[10,7,3],[1,5,4],[3,5,9]],float)
# b=numpy.array([22,33,44],float)
# print(jacobi(a,b))
```

☞ Para você fazer

Resolva usando este método e anexe a listagem do programa (ou copie no verso desta folha):

$$A = \begin{pmatrix} 59 & 3 & 29 & 9 & 15 \\ 8 & 72 & 1 & 32 & 30 \\ 11 & 21 & 76 & 22 & 17 \\ 34 & 12 & 5 & 65 & 13 \\ 2 & 16 & 28 & 19 & 69 \end{pmatrix}$$

e

$$B = (1128, 822, 1050, 879, 767)$$

As respostas encontradas, com $\epsilon < 0.0001$ e sugestão inicial igual à coluna B são:

x_1	x_2	x_3	x_4	x_5

111-70391 - 01/10

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:51.2
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
70403 LUCAS FAGANELLO SILVA 19HEL111 - 26 apos 01/10, 50% _____ / _____ / _____

Sistemas Lineares - Jacobi

Este método é iterativo, ou seja, não exato, já que parte-se de um valor atribuído inicial e mediante buscas locais sucessivas, o mesmo converge para a resposta correta. A diferença entre o resultado sugerido e aquele real, é denominada erro, e pode ser tornada tão pequena quanto se queira. O método entretanto tem um senão, que é um critério de convergência bastante rígido. Se não obedecido o algoritmo não converge para a resposta correta, e portanto o método é inútil.

Considere $A.x = b$ um sistema de quações lineares de ordem n e cujo $det(A) \neq 0$. A maneira extensa de escrever tal sistema é

$$\begin{array}{ccccccc} a_{11}x_1 & + & a_{12}x_2 & +...+ & a_{1n}x_n & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & +...+ & a_{2n}x_n & = & b_2 \\ ... & & & & & & \\ a_{n1}x_1 & + & a_{n2}x_2 & +...+ & a_{nn}x_n & = & b_n \end{array}$$

A matriz A dos coeficientes, pode ser decomposta em $A = L + D + R$, onde

L é a matriz inferior de A (o resto é zero)

D é a matriz contendo apenas a diagonal principal de A, o resto é zero.

R é a matriz superior de A (o resto é zero)

Veja por exemplo, se

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \therefore L = \begin{pmatrix} 0 & 0 & 0 \\ 4 & 0 & 0 \\ 7 & 8 & 0 \end{pmatrix},$$

e

$$D = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 9 \end{pmatrix}, R = \begin{pmatrix} 0 & 2 & 3 \\ 0 & 0 & 6 \\ 0 & 0 & 0 \end{pmatrix}$$

Supondo $det(D) \neq 0$, pode-se escrever: $(L + D + R)x = b$ e daqui $Dx = -(L + R)x + b$. e $x = -D^{-1}(L + R)x + D^{-1}b$. Chamando $B = -D^{-1}(L + R)$ e $g = D^{-1}.b$ obtem-se a equação geral deste método que é

$$x = Bx + g$$

. Note que é uma equação recursiva (ou iterativa) já que o x aparece nos 2 lados da mesma. A idéia aqui é atribuir um valor a x e com ele calcular um novo x, que é jogado no lugar do x velho e gera um novo x e assim sucessivamente até haver a convergência esperada (de antemão). Como supusemos $det(D) \neq 0$ pode-se dividir cada equação do sistema pelo coeficiente da diagonal principal, resultando

$$A^* = L^* + I + R^*$$

Agora o processo iterativo pode ser definido como

$$x^{k+1} = -(L^* + R^*)x^k + b^*$$

onde os elementos de L^*, R^* e b^* são:

$l_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i > j$ e zero senão

$d_{ij} = a_{ij}$ se $i = j$ e zero senão

$r_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i < j$ e zero senão

$b_ij^* = \frac{b_i}{a_{ii}}$

Critérios de convergência

Depois de dividida a matriz pelo elemento da diagonal principal, deve-se ter o **critério das linhas** que diz

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}^*| < 1$$

OU então o **critério das colunas** que diz

$$max_{1 \leq j \leq n} \sum_{i=1, i \neq j}^{n} |a_{ij}^*| < 1$$

Note que qualquer um dos dois critérios sendo satisfeitos, isso garante que o sistema tem convergência.

Definindo uma matriz como estritamente diagonalmente dominante se

$$\sum_{j=1, j \neq i}^{n} |a_{ij}| < |a_{ii}|$$

e se A é estritamente diagonalmente dominante, então A satisfaz o critério das linhas, e o sistema converge. Então, um bom critério de convergência pode ser

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}|* < 1$$

Exemplo

Seja resolver o sistema abaixo, usando o método de Jacobi

$$\begin{array}{ccccccc} 10x & + & 2y & + & z & = & 7 \\ x & + & 5y & + & z & = & -8 \\ 2x & + & 3y & + & 10z & = & 6 \end{array}$$

com $x^0 = (0.7, -1.6\ e\ 0.6)^t$ e $\epsilon < 0.11$.

Convergência ?

$|a_{12}| + |a_{13}| < |a_{11}|$? ou $|2| + |1| < |10|$? R: sim
$|a_{21}| + |a_{23}| < |a_{22}|$? ou $|1| + |1| < |5|$? R: sim
$|a_{31}| + |a_{32}| < |a_{33}|$? ou $|2| + |3| < |10|$? R: sim
logo, o sistema convergirá!

Solução

Dividindo cada equação pelo elemento da diagonal principal, fica

$$\begin{array}{ccccccc} x & + & 0.2y & + & 0.1z & = & 0.7 \\ 0.2x & + & y & + & 0.2z & = & -1.6 \\ 0.2x & + & 0.3y & + & z & = & 0.6 \end{array}$$

Apenas para ilustrar o critério das linhas para testar convergência (desnecessário no caso, já que já vimos que o sistema converge, mas em tese...)
$|a_{12}^*| + |a_{13}^*| = |0.2| + |0.1| = 0.3$ e
$|a_{21}^*| + |a_{23}^*| = |0.2| + |0.2| = 0.4$ e
$|a_{31}^*| + |a_{32}^*| = |0.2| + |0.3| = 0.5$ e
$\Rightarrow\ max_{1 \leq i \leq n}(0.3, 0.4\ 0.5) = 0.5 < 1$

Iterações

$x^{k+1} = -0.2y^k - 0.1z^k + 0.7$
$y^{k+1} = -0.2x^k - 0.2z^k - 1.6$
$z^{k+1} = -0.2x^k - 0.3y^k + 0.6$
Começando com $(0.7,\ -1.6,\ 0.6)$ obtemos uma primeira aproximação que é $(0.96,\ -1.86,\ 0.94)$ e jogando estes novos valores na fórmula iterativa, obtem-se:

$x \rightarrow 0.7,\ 0.96,\ 0.978,\ 0.9994,\ 0.9979$
$y \rightarrow -1.6,\ -1.86,\ -1.98,\ -1.9888,\ -1.9996$
$z \rightarrow 0.6,\ 0.94,\ 0.966,\ 0.9984,\ 0.9968$

O critério de parada pode ser
$max(x^{k+1} - x^k) < \epsilon$. No exemplo acima $0.0108 < 0.011$. Ou seja, neste caso, a solução procurada é $x = 0.9979$, $y = -1.9996$ e $z = 0.9978$. Salta aos olhos neste caso que se a tolerância for um pouco menor, a resposta exata será encontrada que é $(1, -2, 1)$. Basta lançar este resultado no sistema original que a exatidão será confirmada.

Solução Python

```
#jacobi
def jacobi(a,b):
 import numpy
 n=len(a) #numero de linhas de a
          # len(a[0])=colunas de a
 xvelho=numpy.zeros((n),float)
 erro=numpy.zeros((n),float)
 xnovo=numpy.zeros((n),float)
 f=numpy.zeros((n,len(a[0])),float)
 d=numpy.zeros((len(a[0])),float)
 tol=0.1
 for i in range(0,n):
  for j in range(0,n):
   if i==j:
    f[i,j]=0
   else:
    f[i,j]=-a[i,j]/a[i,i]
 for i in range(0,n):
  d[i]=b[i]/a[i,i]
 erromax=tol+1
 while erromax>tol:
  for i in range(0,n):
   xnovo[i]=0
   for k in range(0,n):
    xnovo[i]=xnovo[i]+f[i,k]*xvelho[k]
  for i in range(0,n):
   xnovo[i]=xnovo[i]+d[i]
  for i in range(0,n):
   erro[i]=abs(xnovo[i]-xvelho[i])
  erromax=erro[0]
  for i in range(1,n):
   if erro[i]>erromax:
    erromax=erro[i]
  for i in range(0,n):
   xvelho[i]=xnovo[i]
 return(xnovo)
import numpy
# para usar faça:
# a=numpy.array([[10,7,3],[1,5,4],[3,5,9]],float)
# b=numpy.array([22,33,44],float)
# print(jacobi(a,b))
```

☞ Para você fazer

Resolva usando este método e anexe a listagem do programa (ou copie no verso desta folha):

$$A = \begin{pmatrix} 59 & 23 & 12 & 2 & 21 \\ 9 & 71 & 1 & 25 & 33 \\ 19 & 3 & 54 & 8 & 16 \\ 13 & 24 & 5 & 61 & 17 \\ 20 & 10 & 4 & 6 & 44 \end{pmatrix}$$

e

$$B = (938, 1389, 1060, 1464, 574)$$

As respostas encontradas, com $\epsilon < 0.0001$ e sugestão inicial igual à coluna B são:

x_1	x_2	x_3	x_4	x_5

111-70403 - 01/10

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:51.2
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
70410 LUIZ AUGUSTO SHOITI TSURUKAWA
19HEL111 - 27 apos 01/10, 50% _____ / / _____

Sistemas Lineares - Jacobi

Este método é iterativo, ou seja, não exato, já que parte-se de um valor atribuído inicial e mediante buscas locais sucessivas, o mesmo converge para a resposta correta. A diferença entre o resultado sugerido e aquele real, é denominada erro, e pode ser tornada tão pequena quanto se queira. O método entretanto tem um senão, que é um critério de convergência bastante rígido. Se não obedecido o algoritmo não converge para a resposta correta, e portanto o método é inútil.

Considere $A.x = b$ um sistema de quações lineares de ordem n e cujo $det(A) \neq 0$. A maneira extensa de escrever tal sistema é

$$\begin{array}{ccccccc} a_{11}x_1 & + & a_{12}x_2 & +...+ & a_{1n}x_n & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & +...+ & a_{2n}x_n & = & b_2 \\ ... & & & & & & \\ a_{n1}x_1 & + & a_{n2}x_2 & +...+ & a_{nn}x_n & = & b_n \end{array}$$

A matriz A dos coeficientes, pode ser decomposta em $A = L + D + R$, onde

L é a matriz inferior de A (o resto é zero)

D é a matriz contendo apenas a diagonal principal de A, o resto é zero.

R é a matriz superior de A (o resto é zero)

Veja por exemplo, se

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \therefore L = \begin{pmatrix} 0 & 0 & 0 \\ 4 & 0 & 0 \\ 7 & 8 & 0 \end{pmatrix},$$

e

$$D = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 9 \end{pmatrix}, R = \begin{pmatrix} 0 & 2 & 3 \\ 0 & 0 & 6 \\ 0 & 0 & 0 \end{pmatrix}$$

Supondo $det(D) \neq 0$, pode-se escrever: $(L + D + R)x = b$ e daqui $Dx = -(L + R)x + b$. e $x = -D^{-1}(L + R)x + D^{-1}b$. Chamando $B = -D^{-1}(L + R)$ e $g = D^{-1}.b$ obtem-se a equação geral deste método que é

$$x = Bx + g$$

. Note que é uma equação recursiva (ou iterativa) já que o x aparece nos 2 lados da mesma. A idéia aqui é atribuir um valor a x e com ele calcular um novo x, que é jogado no lugar do x velho e gera um novo x e assim sucessivamente até haver a convergência esperada (de antemão). Como supusemos $det(D) \neq 0$ pode-se dividir cada equação do sistema pelo coeficiente da diagonal principal, resultando

$$A^* = L^* + I + R^*$$

Agora o processo iterativo pode ser definido como

$$x^{k+1} = -(L^* + R^*)x^k + b^*$$

onde os elementos de L^*, R^* e b^* são:

$l_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i > j$ e zero senão

$d_{ij} = a_{ij}$ se $i = j$ e zero senão

$r_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i < j$ e zero senão

$b_ij^* = \frac{b_i}{a_{ii}}$

Critérios de convergência

Depois de dividida a matriz pelo elemento da diagonal principal, deve-se ter o **critério das linhas** que diz

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}^*| < 1$$

OU então o **critério das colunas** que diz

$$max_{1 \leq j \leq n} \sum_{i=1, i \neq j}^{n} |a_{ij}^*| < 1$$

Note que qualquer um dos dois critérios sendo satisfeitos, isso garante que o sistema tem convergência.

Definindo uma matriz como estritamente diagonalmente dominante se

$$\sum_{j=1, j \neq i}^{n} |a_{ij}| < |a_{ii}|$$

e se A é estritamente diagonalmente dominante, então A satisfaz o critério das linhas, e o sistema converge. Então, um bom critério de convergência pode ser

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}^|* < 1$$

Exemplo

Seja resolver o sistema abaixo, usando o método de Jacobi

$$\begin{array}{ccccccc} 10x & + & 2y & + & z & = & 7 \\ x & + & 5y & + & z & = & -8 \\ 2x & + & 3y & + & 10z & = & 6 \end{array}$$

com $x^0 = (0.7, -1.6\ e\ 0.6)^t$ e $\epsilon < 0.11$.

Convergência ?

$|a_{12}| + |a_{13}| < |a_{11}|$? ou $|2| + |1| < |10|$? R: sim
$|a_{21}| + |a_{23}| < |a_{22}|$? ou $|1| + |1| < |5|$? R: sim
$|a_{31}| + |a_{32}| < |a_{33}|$? ou $|2| + |3| < |10|$? R: sim
logo, o sistema convergirá!

Solução

Dividindo cada equação pelo elemento da diagonal principal, fica

$$\begin{array}{ccccccc} x & + & 0.2y & + & 0.1z & = & 0.7 \\ 0.2x & + & y & + & 0.2z & = & -1.6 \\ 0.2x & + & 0.3y & + & z & = & 0.6 \end{array}$$

Apenas para ilustrar o critério das linhas para testar convergência (desnecessário no caso, já que já vimos que o sistema converge, mas em tese...)
$|a_{12}^*| + |a_{13}^*| = |0.2| + |0.1| = 0.3$ e
$|a_{21}^*| + |a_{23}^*| = |0.2| + |0.2| = 0.4$ e
$|a_{31}^*| + |a_{32}^*| = |0.2| + |0.3| = 0.5$ e
$\Rightarrow\ max_{1 \leq i \leq n}(0.3, 0.4\ 0.5) = 0.5 < 1$

Iterações

$x^{k+1} = -0.2y^k - 0.1z^k + 0.7$
$y^{k+1} = -0.2x^k - 0.2z^k - 1.6$
$z^{k+1} = -0.2x^k - 0.3y^k + 0.6$
Começando com $(0.7,\ -1.6,\ 0.6)$ obtemos uma primeira aproximação que é $(0.96,\ -1.86,\ 0.94)$ e jogando estes novos valores na fórmula iterativa, obtem-se:

$x \rightarrow 0.7,\ 0.96,\ 0.978,\ 0.9994,\ 0.9979$
$y \rightarrow -1.6,\ -1.86,\ -1.98,\ -1.9888,\ -1.9996$
$z \rightarrow 0.6,\ 0.94,\ 0.966,\ 0.9984,\ 0.9968$

O critério de parada pode ser
$max(x^{k+1} - x^k) < \epsilon$. No exemplo acima $0.0108 < 0.011$. Ou seja, neste caso, a solução procurada é $x = 0.9979$, $y = -1.9996$ e $z = 0.9978$. Salta aos olhos neste caso que se a tolerância for um pouco menor, a resposta exata será encontrada que é $(1, -2, 1)$. Basta lançar este resultado no sistema original que a exatidão será confirmada.

Solução Python

```
#jacobi
def jacobi(a,b):
 import numpy
 n=len(a) #numero de linhas de a
          # len(a[0])=colunas de a
 xvelho=numpy.zeros((n),float)
 erro=numpy.zeros((n),float)
 xnovo=numpy.zeros((n),float)
 f=numpy.zeros((n,len(a[0])),float)
 d=numpy.zeros((len(a[0])),float)
 tol=0.1
 for i in range(0,n):
  for j in range(0,n):
   if i==j:
    f[i,j]=0
   else:
    f[i,j]=-a[i,j]/a[i,i]
 for i in range(0,n):
  d[i]=b[i]/a[i,i]
 erromax=tol+1
 while erromax>tol:
  for i in range(0,n):
   xnovo[i]=0
   for k in range(0,n):
    xnovo[i]=xnovo[i]+f[i,k]*xvelho[k]
  for i in range(0,n):
   xnovo[i]=xnovo[i]+d[i]
  for i in range(0,n):
   erro[i]=abs(xnovo[i]-xvelho[i])
  erromax=erro[0]
  for i in range(1,n):
   if erro[i]>erromax:
    erromax=erro[i]
  for i in range(0,n):
   xvelho[i]=xnovo[i]
 return(xnovo)
import numpy
# para usar faça:
# a=numpy.array([[10,7,3],[1,5,4],[3,5,9]],float)
# b=numpy.array([22,33,44],float)
# print(jacobi(a,b))
```

☞ Para você fazer

Resolva usando este método e anexe a listagem do programa (ou copie no verso desta folha):

$$A = \begin{pmatrix} 82 & 3 & 30 & 28 & 20 \\ 23 & 53 & 2 & 4 & 18 \\ 6 & 16 & 55 & 11 & 19 \\ 32 & 17 & 1 & 89 & 34 \\ 14 & 24 & 35 & 22 & 96 \end{pmatrix}$$

e

$$B = (2251, 936, 1283, 2122, 2393)$$

As respostas encontradas, com $\epsilon < 0.0001$ e sugestão inicial igual à coluna B são:

x_1	x_2	x_3	x_4	x_5

111-70410 - 01/10

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:51.2
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
70427 MARCOS A. DA COSTA 19HEL111 - 28 apos 01/10, 50% _____ / _____ / _____

Sistemas Lineares - Jacobi

Este método é iterativo, ou seja, não exato, já que parte-se de um valor atribuído inicial e mediante buscas locais sucessivas, o mesmo converge para a resposta correta. A diferença entre o resultado sugerido e aquele real, é denominada erro, e pode ser tornada tão pequena quanto se queira. O método entretanto tem um senão, que é um critério de convergência bastante rígido. Se não obedecido o algoritmo não converge para a resposta correta, e portanto o método é inútil.

Considere $A.x = b$ um sistema de quações lineares de ordem n e cujo $det(A) \neq 0$. A maneira extensa de escrever tal sistema é

$$\begin{array}{ccccccc} a_{11}x_1 & + & a_{12}x_2 & +...+ & a_{1n}x_n & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & +...+ & a_{2n}x_n & = & b_2 \\ ... & & & & & & \\ a_{n1}x_1 & + & a_{n2}x_2 & +...+ & a_{nn}x_n & = & b_n \end{array}$$

A matriz A dos coeficientes, pode ser decomposta em $A = L + D + R$, onde

L é a matriz inferior de A (o resto é zero)

D é a matriz contendo apenas a diagonal principal de A, o resto é zero.

R é a matriz superior de A (o resto é zero)

Veja por exemplo, se

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \therefore L = \begin{pmatrix} 0 & 0 & 0 \\ 4 & 0 & 0 \\ 7 & 8 & 0 \end{pmatrix},$$

e

$$D = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 9 \end{pmatrix}, R = \begin{pmatrix} 0 & 2 & 3 \\ 0 & 0 & 6 \\ 0 & 0 & 0 \end{pmatrix}$$

Supondo $det(D) \neq 0$, pode-se escrever: $(L + D + R)x = b$ e daqui $Dx = -(L + R)x + b$. e $x = -D^{-1}(L + R)x + D^{-1}b$. Chamando $B = -D^{-1}(L + R)$ e $g = D^{-1}.b$ obtem-se a equação geral deste método que é

$$x = Bx + g$$

. Note que é uma equação recursiva (ou iterativa) já que o x aparece nos 2 lados da mesma. A idéia aqui é atribuir um valor a x e com ele calcular um novo x, que é jogado no lugar do x velho e gera um novo x e assim sucessivamente até haver a convergência esperada (de antemão). Como supusemos $det(D) \neq 0$ pode-se dividir cada equação do sistema pelo coeficiente da diagonal principal, resultando

$$A^* = L^* + I + R^*$$

Agora o processo iterativo pode ser definido como

$$x^{k+1} = -(L^* + R^*)x^k + b^*$$

onde os elementos de L^*, R^* e b^* são:

$l_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i > j$ e zero senão

$d_{ij} = a_{ij}$ se $i = j$ e zero senão

$r_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i < j$ e zero senão

$b_ij^* = \frac{b_i}{a_{ii}}$

Critérios de convergência

Depois de dividida a matriz pelo elemento da diagonal principal, deve-se ter o **critério das linhas** que diz

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}^*| < 1$$

OU então o **critério das colunas** que diz

$$max_{1 \leq j \leq n} \sum_{i=1, i \neq j}^{n} |a_{ij}^*| < 1$$

Note que qualquer um dos dois critérios sendo satisfeitos, isso garante que o sistema tem convergência.

Definindo uma matriz como estritamente diagonalmente dominante se

$$\sum_{j=1, j \neq i}^{n} |a_{ij}| < |a_{ii}|$$

e se A é estritamente diagonalmente dominante, então A satisfaz o critério das linhas, e o sistema converge. Então, um bom critério de convergência pode ser

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}|* < 1$$

Exemplo

Seja resolver o sistema abaixo, usando o método de Jacobi

$$\begin{array}{ccccccc} 10x & + & 2y & + & z & = & 7 \\ x & + & 5y & + & z & = & -8 \\ 2x & + & 3y & + & 10z & = & 6 \end{array}$$

com $x^0 = (0.7, -1.6\ e\ 0.6)^t$ e $\epsilon < 0.11$.

Convergência ?

$|a_{12}| + |a_{13}| < |a_{11}|$? ou $|2| + |1| < |10|$? R: sim
$|a_{21}| + |a_{23}| < |a_{22}|$? ou $|1| + |1| < |5|$? R: sim
$|a_{31}| + |a_{32}| < |a_{33}|$? ou $|2| + |3| < |10|$? R: sim
logo, o sistema convergirá!

Solução

Dividindo cada equação pelo elemento da diagonal principal, fica

$$\begin{array}{ccccccc} x & + & 0.2y & + & 0.1z & = & 0.7 \\ 0.2x & + & y & + & 0.2z & = & -1.6 \\ 0.2x & + & 0.3y & + & z & = & 0.6 \end{array}$$

Apenas para ilustrar o critério das linhas para testar convergência (desnecessário no caso, já que já vimos que o sistema converge, mas em tese...)
$|a_{12}^*| + |a_{13}^*| = |0.2| + |0.1| = 0.3$ e
$|a_{21}^*| + |a_{23}^*| = |0.2| + |0.2| = 0.4$ e
$|a_{31}^*| + |a_{32}^*| = |0.2| + |0.3| = 0.5$ e
$\Rightarrow\ max_{1 \leq i \leq n}(0.3, 0.4\ 0.5) = 0.5 < 1$

Iterações

$x^{k+1} = -0.2y^k - 0.1z^k + 0.7$
$y^{k+1} = -0.2x^k - 0.2z^k - 1.6$
$z^{k+1} = -0.2x^k - 0.3y^k + 0.6$
Começando com $(0.7,\ -1.6,\ 0.6)$ obtemos uma primeira aproximação que é $(0.96,\ -1.86,\ 0.94)$ e jogando estes novos valores na fórmula iterativa, obtem-se:

$x \rightarrow 0.7,\ 0.96,\ 0.978,\ 0.9994,\ 0.9979$
$y \rightarrow -1.6,\ -1.86,\ -1.98,\ -1.9888,\ -1.9996$
$z \rightarrow 0.6,\ 0.94,\ 0.966,\ 0.9984,\ 0.9968$

O critério de parada pode ser
$max(x^{k+1} - x^k) < \epsilon$. No exemplo acima $0.0108 < 0.011$. Ou seja, neste caso, a solução procurada é $x = 0.9979$, $y = -1.9996$ e $z = 0.9978$. Salta aos olhos neste caso que se a tolerância for um pouco menor, a resposta exata será encontrada que é $(1, -2, 1)$. Basta lançar este resultado no sistema original que a exatidão será confirmada.

Solução Python

```
#jacobi
def jacobi(a,b):
 import numpy
 n=len(a) #numero de linhas de a
          # len(a[0])=colunas de a
 xvelho=numpy.zeros((n),float)
 erro=numpy.zeros((n),float)
 xnovo=numpy.zeros((n),float)
 f=numpy.zeros((n,len(a[0])),float)
 d=numpy.zeros((len(a[0])),float)
 tol=0.1
 for i in range(0,n):
  for j in range(0,n):
   if i==j:
    f[i,j]=0
   else:
    f[i,j]=-a[i,j]/a[i,i]
 for i in range(0,n):
  d[i]=b[i]/a[i,i]
 erromax=tol+1
 while erromax>tol:
  for i in range(0,n):
   xnovo[i]=0
   for k in range(0,n):
    xnovo[i]=xnovo[i]+f[i,k]*xvelho[k]
  for i in range(0,n):
   xnovo[i]=xnovo[i]+d[i]
  for i in range(0,n):
   erro[i]=abs(xnovo[i]-xvelho[i])
  erromax=erro[0]
  for i in range(1,n):
   if erro[i]>erromax:
    erromax=erro[i]
  for i in range(0,n):
   xvelho[i]=xnovo[i]
 return(xnovo)
import numpy
# para usar faça:
# a=numpy.array([[10,7,3],[1,5,4],[3,5,9]],float)
# b=numpy.array([22,33,44],float)
# print(jacobi(a,b))
```

☞ Para você fazer

Resolva usando este método e anexe a listagem do programa (ou copie no verso desta folha):

$$A = \begin{pmatrix} 101 & 34 & 31 & 8 & 25 \\ 22 & 98 & 32 & 18 & 24 \\ 2 & 27 & 66 & 26 & 10 \\ 33 & 6 & 16 & 68 & 12 \\ 9 & 15 & 29 & 5 & 63 \end{pmatrix}$$

e

$$B = (1557, 1344, 1270, 1620, 943)$$

As respostas encontradas, com $\epsilon < 0.0001$ e sugestão inicial igual à coluna B são:

x_1	x_2	x_3	x_4	x_5

111-70427 - 01/10

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:51.2
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
70434 MARIANA YAEDA ARTUSO 19HEL111 - 29 apos 01/10, 50% _____ / _____ / _____

Sistemas Lineares - Jacobi

Este método é iterativo, ou seja, não exato, já que parte-se de um valor atribuído inicial e mediante buscas locais sucessivas, o mesmo converge para a resposta correta. A diferença entre o resultado sugerido e aquele real, é denominada erro, e pode ser tornada tão pequena quanto se queira. O método entretanto tem um senão, que é um critério de convergência bastante rígido. Se não obedecido o algoritmo não converge para a resposta correta, e portanto o método é inútil.

Considere $A.x = b$ um sistema de quações lineares de ordem n e cujo $det(A) \neq 0$. A maneira extensa de escrever tal sistema é

$$\begin{array}{ccccccccc} a_{11}x_1 & + & a_{12}x_2 & +...+ & a_{1n}x_n & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & +...+ & a_{2n}x_n & = & b_2 \\ ... & & & & & & \\ a_{n1}x_1 & + & a_{n2}x_2 & +...+ & a_{nn}x_n & = & b_n \end{array}$$

A matriz A dos coeficientes, pode ser decomposta em $A = L + D + R$, onde

L é a matriz inferior de A (o resto é zero)

D é a matriz contendo apenas a diagonal principal de A, o resto é zero.

R é a matriz superior de A (o resto é zero)

Veja por exemplo, se

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \therefore L = \begin{pmatrix} 0 & 0 & 0 \\ 4 & 0 & 0 \\ 7 & 8 & 0 \end{pmatrix},$$

e

$$D = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 9 \end{pmatrix}, R = \begin{pmatrix} 0 & 2 & 3 \\ 0 & 0 & 6 \\ 0 & 0 & 0 \end{pmatrix}$$

Supondo $det(D) \neq 0$, pode-se escrever: $(L + D + R)x = b$ e daqui $Dx = -(L + R)x + b$. e $x = -D^{-1}(L + R)x + D^{-1}b$. Chamando $B = -D^{-1}(L + R)$ e $g = D^{-1}.b$ obtem-se a equação geral deste método que é

$$x = Bx + g$$

. Note que é uma equação recursiva (ou iterativa) já que o x aparece nos 2 lados da mesma. A idéia aqui é atribuir um valor a x e com ele calcular um novo x, que é jogado no lugar do x velho e gera um novo x e assim sucessivamente até haver a convergência esperada (de antemão). Como supusemos $det(D) \neq 0$ pode-se dividir cada equação do sistema pelo coeficiente da diagonal principal, resultando

$$A^* = L^* + I + R^*$$

Agora o processo iterativo pode ser definido como

$$x^{k+1} = -(L^* + R^*)x^k + b^*$$

onde os elementos de L^*, R^* e b^* são:

$l_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i > j$ e zero senão

$d_{ij} = a_{ij}$ se $i = j$ e zero senão

$r_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i < j$ e zero senão

$b_ij^* = \frac{b_i}{a_{ii}}$

Critérios de convergência

Depois de dividida a matriz pelo elemento da diagonal principal, deve-se ter o **critério das linhas** que diz

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}^*| < 1$$

OU então o **critério das colunas** que diz

$$max_{1 \leq j \leq n} \sum_{i=1, i \neq j}^{n} |a_{ij}^*| < 1$$

Note que qualquer um dos dois critérios sendo satisfeitos, isso garante que o sistema tem convergência.

Definindo uma matriz como estritamente diagonalmente dominante se

$$\sum_{j=1, j \neq i}^{n} |a_{ij}| < |a_{ii}|$$

e se A é estritamente diagonalmente dominante, então A satisfaz o critério das linhas, e o sistema converge. Então, um bom critério de convergência pode ser

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}|* < 1$$

Exemplo

Seja resolver o sistema abaixo, usando o método de Jacobi

$$\begin{array}{ccccccc} 10x & + & 2y & + & z & = & 7 \\ x & + & 5y & + & z & = & -8 \\ 2x & + & 3y & + & 10z & = & 6 \end{array}$$

com $x^0 = (0.7, -1.6\ e\ 0.6)^t$ e $\epsilon < 0.11$.

Convergência ?

$|a_{12}| + |a_{13}| < |a_{11}|$? ou $|2| + |1| < |10|$? R: sim
$|a_{21}| + |a_{23}| < |a_{22}|$? ou $|1| + |1| < |5|$? R: sim
$|a_{31}| + |a_{32}| < |a_{33}|$? ou $|2| + |3| < |10|$? R: sim
logo, o sistema convergirá!

Solução

Dividindo cada equação pelo elemento da diagonal principal, fica

$$\begin{array}{ccccccc} x & + & 0.2y & + & 0.1z & = & 0.7 \\ 0.2x & + & y & + & 0.2z & = & -1.6 \\ 0.2x & + & 0.3y & + & z & = & 0.6 \end{array}$$

Apenas para ilustrar o critério das linhas para testar convergência (desnecessário no caso, já que já vimos que o sistema converge, mas em tese...)
$|a_{12}^*| + |a_{13}^*| = |0.2| + |0.1| = 0.3$ e
$|a_{21}^*| + |a_{23}^*| = |0.2| + |0.2| = 0.4$ e
$|a_{31}^*| + |a_{32}^*| = |0.2| + |0.3| = 0.5$ e
$\Rightarrow\ max_{1 \leq i \leq n}(0.3, 0.4\ 0.5) = 0.5 < 1$

Iterações

$x^{k+1} = -0.2y^k - 0.1z^k + 0.7$
$y^{k+1} = -0.2x^k - 0.2z^k - 1.6$
$z^{k+1} = -0.2x^k - 0.3y^k + 0.6$
Começando com $(0.7,\ -1.6,\ 0.6)$ obtemos uma primeira aproximação que é $(0.96,\ -1.86,\ 0.94)$ e jogando estes novos valores na fórmula iterativa, obtem-se:

$x \rightarrow 0.7,\ 0.96,\ 0.978,\ 0.9994,\ 0.9979$
$y \rightarrow -1.6,\ -1.86,\ -1.98,\ -1.9888,\ -1.9996$
$z \rightarrow 0.6,\ 0.94,\ 0.966,\ 0.9984,\ 0.9968$

O critério de parada pode ser
$max(x^{k+1} - x^k) < \epsilon$. No exemplo acima $0.0108 < 0.011$. Ou seja, neste caso, a solução procurada é $x = 0.9979$, $y = -1.9996$ e $z = 0.9978$. Salta aos olhos neste caso que se a tolerância for um pouco menor, a resposta exata será encontrada que é $(1, -2, 1)$. Basta lançar este resultado no sistema original que a exatidão será confirmada.

Solução Python

```
#jacobi
def jacobi(a,b):
 import numpy
 n=len(a) #numero de linhas de a
          # len(a[0])=colunas de a
 xvelho=numpy.zeros((n),float)
 erro=numpy.zeros((n),float)
 xnovo=numpy.zeros((n),float)
 f=numpy.zeros((n,len(a[0])),float)
 d=numpy.zeros((len(a[0])),float)
 tol=0.1
 for i in range(0,n):
  for j in range(0,n):
   if i==j:
    f[i,j]=0
   else:
    f[i,j]=-a[i,j]/a[i,i]
 for i in range(0,n):
  d[i]=b[i]/a[i,i]
 erromax=tol+1
 while erromax>tol:
  for i in range(0,n):
   xnovo[i]=0
   for k in range(0,n):
    xnovo[i]=xnovo[i]+f[i,k]*xvelho[k]
  for i in range(0,n):
   xnovo[i]=xnovo[i]+d[i]
  for i in range(0,n):
   erro[i]=abs(xnovo[i]-xvelho[i])
  erromax=erro[0]
  for i in range(1,n):
   if erro[i]>erromax:
    erromax=erro[i]
  for i in range(0,n):
   xvelho[i]=xnovo[i]
 return(xnovo)
import numpy
# para usar faça:
# a=numpy.array([[10,7,3],[1,5,4],[3,5,9]],float)
# b=numpy.array([22,33,44],float)
# print(jacobi(a,b))
```

☞ Para você fazer

Resolva usando este método e anexe a listagem do programa (ou copie no verso desta folha):

$$A = \begin{pmatrix} 101 & 10 & 20 & 35 & 32 \\ 23 & 107 & 19 & 29 & 33 \\ 34 & 9 & 73 & 1 & 26 \\ 8 & 25 & 7 & 70 & 27 \\ 30 & 31 & 21 & 3 & 86 \end{pmatrix}$$

e

$$B = (2700, 2667, 1332, 1633, 2438)$$

As respostas encontradas, com $\epsilon < 0.0001$ e sugestão inicial igual à coluna B são:

x_1	x_2	x_3	x_4	x_5

111-70434 - 01/10

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:51.2
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
71220 MATHEUS CARDOSO CASTILHO DIAS
19HEL111 - 30 apos 01/10, 50% _____ / / _____

Sistemas Lineares - Jacobi

Este método é iterativo, ou seja, não exato, já que parte-se de um valor atribuído inicial e mediante buscas locais sucessivas, o mesmo converge para a resposta correta. A diferença entre o resultado sugerido e aquele real, é denominada erro, e pode ser tornada tão pequena quanto se queira. O método entretanto tem um senão, que é um critério de convergência bastante rígido. Se não obedecido o algoritmo não converge para a resposta correta, e portanto o método é inútil.

Considere $A.x = b$ um sistema de quações lineares de ordem n e cujo $det(A) \neq 0$. A maneira extensa de escrever tal sistema é

$$\begin{array}{ccccccc} a_{11}x_1 & + & a_{12}x_2 & +...+ & a_{1n}x_n & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & +...+ & a_{2n}x_n & = & b_2 \\ ... & & & & & & \\ a_{n1}x_1 & + & a_{n2}x_2 & +...+ & a_{nn}x_n & = & b_n \end{array}$$

A matriz A dos coeficientes, pode ser decomposta em $A = L + D + R$, onde

L é a matriz inferior de A (o resto é zero)

D é a matriz contendo apenas a diagonal principal de A, o resto é zero.

R é a matriz superior de A (o resto é zero)

Veja por exemplo, se

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \therefore L = \begin{pmatrix} 0 & 0 & 0 \\ 4 & 0 & 0 \\ 7 & 8 & 0 \end{pmatrix},$$

e

$$D = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 9 \end{pmatrix}, R = \begin{pmatrix} 0 & 2 & 3 \\ 0 & 0 & 6 \\ 0 & 0 & 0 \end{pmatrix}$$

Supondo $det(D) \neq 0$, pode-se escrever: $(L + D + R)x = b$ e daqui $Dx = -(L + R)x + b$. e $x = -D^{-1}(L + R)x + D^{-1}b$. Chamando $B = -D^{-1}(L + R)$ e $g = D^{-1}.b$ obtem-se a equação geral deste método que é

$$x = Bx + g$$

. Note que é uma equação recursiva (ou iterativa) já que o x aparece nos 2 lados da mesma. A idéia aqui é atribuir um valor a x e com ele calcular um novo x, que é jogado no lugar do x velho e gera um novo x e assim sucessivamente até haver a convergência esperada (de antemão). Como supusemos $det(D) \neq 0$ pode-se dividir cada equação do sistema pelo coeficiente da diagonal principal, resultando

$$A^* = L^* + I + R^*$$

Agora o processo iterativo pode ser definido como

$$x^{k+1} = -(L^* + R^*)x^k + b^*$$

onde os elementos de L^*, R^* e b^* são:

$l_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i > j$ e zero senão

$d_{ij} = a_{ij}$ se $i = j$ e zero senão

$r_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i < j$ e zero senão

$b_ij^* = \frac{b_i}{a_{ii}}$

Critérios de convergência

Depois de dividida a matriz pelo elemento da diagonal principal, deve-se ter o **critério das linhas** que diz

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}^*| < 1$$

OU então o **critério das colunas** que diz

$$max_{1 \leq j \leq n} \sum_{i=1, i \neq j}^{n} |a_{ij}^*| < 1$$

Note que qualquer um dos dois critérios sendo satisfeitos, isso garante que o sistema tem convergência.

Definindo uma matriz como estritamente diagonalmente dominante se

$$\sum_{j=1, j \neq i}^{n} |a_{ij}| < |a_{ii}|$$

e se A é estritamente diagonalmente dominante, então A satisfaz o critério das linhas, e o sistema converge. Então, um bom critério de convergência pode ser

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}^|* < 1$$

Exemplo

Seja resolver o sistema abaixo, usando o método de Jacobi

$$\begin{array}{ccccccc} 10x & + & 2y & + & z & = & 7 \\ x & + & 5y & + & z & = & -8 \\ 2x & + & 3y & + & 10z & = & 6 \end{array}$$

com $x^0 = (0.7, -1.6\ e\ 0.6)^t$ e $\epsilon < 0.11$.

Convergência ?

$|a_{12}| + |a_{13}| < |a_{11}|$? ou $|2| + |1| < |10|$? R: sim
$|a_{21}| + |a_{23}| < |a_{22}|$? ou $|1| + |1| < |5|$? R: sim
$|a_{31}| + |a_{32}| < |a_{33}|$? ou $|2| + |3| < |10|$? R: sim
logo, o sistema convergirá!

Solução

Dividindo cada equação pelo elemento da diagonal principal, fica

$$\begin{array}{ccccccc} x & + & 0.2y & + & 0.1z & = & 0.7 \\ 0.2x & + & y & + & 0.2z & = & -1.6 \\ 0.2x & + & 0.3y & + & z & = & 0.6 \end{array}$$

Apenas para ilustrar o critério das linhas para testar convergência (desnecessário no caso, já que já vimos que o sistema converge, mas em tese...)
$|a_{12}^*| + |a_{13}^*| = |0.2| + |0.1| = 0.3$ e
$|a_{21}^*| + |a_{23}^*| = |0.2| + |0.2| = 0.4$ e
$|a_{31}^*| + |a_{32}^*| = |0.2| + |0.3| = 0.5$ e
$\Rightarrow\ max_{1 \leq i \leq n}(0.3, 0.4\ 0.5) = 0.5 < 1$

Iterações

$x^{k+1} = -0.2y^k - 0.1z^k + 0.7$
$y^{k+1} = -0.2x^k - 0.2z^k - 1.6$
$z^{k+1} = -0.2x^k - 0.3y^k + 0.6$
Começando com $(0.7,\ -1.6,\ 0.6)$ obtemos uma primeira aproximação que é $(0.96,\ -1.86,\ 0.94)$ e jogando estes novos valores na fórmula iterativa, obtem-se:

$x \rightarrow 0.7,\ 0.96,\ 0.978,\ 0.9994,\ 0.9979$
$y \rightarrow -1.6,\ -1.86,\ -1.98,\ -1.9888,\ -1.9996$
$z \rightarrow 0.6,\ 0.94,\ 0.966,\ 0.9984,\ 0.9968$

O critério de parada pode ser
$max(x^{k+1} - x^k) < \epsilon$. No exemplo acima $0.0108 < 0.011$. Ou seja, neste caso, a solução procurada é $x = 0.9979$, $y = -1.9996$ e $z = 0.9978$. Salta aos olhos neste caso que se a tolerância for um pouco menor, a resposta exata será encontrada que é $(1, -2, 1)$. Basta lançar este resultado no sistema original que a exatidão será confirmada.

Solução Python

```
#jacobi
def jacobi(a,b):
 import numpy
 n=len(a) #numero de linhas de a
          # len(a[0])=colunas de a
 xvelho=numpy.zeros((n),float)
 erro=numpy.zeros((n),float)
 xnovo=numpy.zeros((n),float)
 f=numpy.zeros((n,len(a[0])),float)
 d=numpy.zeros((len(a[0])),float)
 tol=0.1
 for i in range(0,n):
  for j in range(0,n):
   if i==j:
    f[i,j]=0
   else:
    f[i,j]=-a[i,j]/a[i,i]
 for i in range(0,n):
  d[i]=b[i]/a[i,i]
 erromax=tol+1
 while erromax>tol:
  for i in range(0,n):
   xnovo[i]=0
   for k in range(0,n):
    xnovo[i]=xnovo[i]+f[i,k]*xvelho[k]
  for i in range(0,n):
   xnovo[i]=xnovo[i]+d[i]
  for i in range(0,n):
   erro[i]=abs(xnovo[i]-xvelho[i])
  erromax=erro[0]
  for i in range(1,n):
   if erro[i]>erromax:
    erromax=erro[i]
  for i in range(0,n):
   xvelho[i]=xnovo[i]
 return(xnovo)
import numpy
# para usar faça:
# a=numpy.array([[10,7,3],[1,5,4],[3,5,9]],float)
# b=numpy.array([22,33,44],float)
# print(jacobi(a,b))
```

☞ Para você fazer

Resolva usando este método e anexe a listagem do programa (ou copie no verso desta folha):

$$A = \begin{pmatrix} 105 & 32 & 20 & 18 & 33 \\ 4 & 81 & 35 & 3 & 34 \\ 25 & 12 & 49 & 6 & 5 \\ 13 & 16 & 10 & 71 & 30 \\ 2 & 15 & 1 & 14 & 37 \end{pmatrix}$$

e

$$B = (2035, 1686, 694, 1133, 540)$$

As respostas encontradas, com $\epsilon < 0.0001$ e sugestão inicial igual à coluna B são:

x_1	x_2	x_3	x_4	x_5

111-71220 - 01/10

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:51.2
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
70441 MATHEUS DA SILVA QUADROS
19HEL111 - 31 apos 01/10, 50% _____ / / _____

Sistemas Lineares - Jacobi

Este método é iterativo, ou seja, não exato, já que parte-se de um valor atribuído inicial e mediante buscas locais sucessivas, o mesmo converge para a resposta correta. A diferença entre o resultado sugerido e aquele real, é denominada erro, e pode ser tornada tão pequena quanto se queira. O método entretanto tem um senão, que é um critério de convergência bastante rígido. Se não obedecido o algoritmo não converge para a resposta correta, e portanto o método é inútil.

Considere $A.x = b$ um sistema de quações lineares de ordem n e cujo $det(A) \neq 0$. A maneira extensa de escrever tal sistema é

$$\begin{array}{lcllcl} a_{11}x_1 & + & a_{12}x_2 & +...+ & a_{1n}x_n & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & +...+ & a_{2n}x_n & = & b_2 \\ ... & & & & & & \\ a_{n1}x_1 & + & a_{n2}x_2 & +...+ & a_{nn}x_n & = & b_n \end{array}$$

A matriz A dos coeficientes, pode ser decomposta em $A = L + D + R$, onde

L é a matriz inferior de A (o resto é zero)

D é a matriz contendo apenas a diagonal principal de A, o resto é zero.

R é a matriz superior de A (o resto é zero)

Veja por exemplo, se

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \therefore L = \begin{pmatrix} 0 & 0 & 0 \\ 4 & 0 & 0 \\ 7 & 8 & 0 \end{pmatrix},$$

e

$$D = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 9 \end{pmatrix}, R = \begin{pmatrix} 0 & 2 & 3 \\ 0 & 0 & 6 \\ 0 & 0 & 0 \end{pmatrix}$$

Supondo $det(D) \neq 0$, pode-se escrever: $(L + D + R)x = b$ e daqui $Dx = -(L + R)x + b$. e $x = -D^{-1}(L + R)x + D^{-1}b$. Chamando $B = -D^{-1}(L + R)$ e $g = D^{-1}.b$ obtem-se a equação geral deste método que é

$$x = Bx + g$$

. Note que é uma equação recursiva (ou iterativa) já que o x aparece nos 2 lados da mesma. A idéia aqui é atribuir um valor a x e com ele calcular um novo x, que é jogado no lugar do x velho e gera um novo x e assim sucessivamente até haver a convergência esperada (de antemão). Como supusemos $det(D) \neq 0$ pode-se dividir cada equação do sistema pelo coeficiente da diagonal principal, resultando

$$A^* = L^* + I + R^*$$

Agora o processo iterativo pode ser definido como

$$x^{k+1} = -(L^* + R^*)x^k + b^*$$

onde os elementos de L^*, R^* e b^* são:

$l_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i > j$ e zero senão

$d_{ij} = a_{ij}$ se $i = j$ e zero senão

$r_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i < j$ e zero senão

$b_ij^* = \frac{b_i}{a_{ii}}$

Critérios de convergência

Depois de dividida a matriz pelo elemento da diagonal principal, deve-se ter o **critério das linhas** que diz

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}^*| < 1$$

OU então o **critério das colunas** que diz

$$max_{1 \leq j \leq n} \sum_{i=1, i \neq j}^{n} |a_{ij}^*| < 1$$

Note que qualquer um dos dois critérios sendo satisfeitos, isso garante que o sistema tem convergência.

Definindo uma matriz como estritamente diagonalmente dominante se

$$\sum_{j=1, j \neq i}^{n} |a_{ij}| < |a_{ii}|$$

e se A é estritamente diagonalmente dominante, então A satisfaz o critério das linhas, e o sistema converge. Então, um bom critério de convergência pode ser

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}^|* < 1$$

Exemplo

Seja resolver o sistema abaixo, usando o método de Jacobi

$$\begin{array}{rcrcrcr} 10x & + & 2y & + & z & = & 7 \\ x & + & 5y & + & z & = & -8 \\ 2x & + & 3y & + & 10z & = & 6 \end{array}$$

com $x^0 = (0.7, -1.6\ e\ 0.6)^t$ e $\epsilon < 0.11$.

Convergência ?

$|a_{12}| + |a_{13}| < |a_{11}|$? ou $|2| + |1| < |10|$? R: sim
$|a_{21}| + |a_{23}| < |a_{22}|$? ou $|1| + |1| < |5|$? R: sim
$|a_{31}| + |a_{32}| < |a_{33}|$? ou $|2| + |3| < |10|$? R: sim
logo, o sistema convergirá!

Solução

Dividindo cada equação pelo elemento da diagonal principal, fica

$$\begin{array}{rcrcrcr} x & + & 0.2y & + & 0.1z & = & 0.7 \\ 0.2x & + & y & + & 0.2z & = & -1.6 \\ 0.2x & + & 0.3y & + & z & = & 0.6 \end{array}$$

Apenas para ilustrar o critério das linhas para testar convergência (desnecessário no caso, já que já vimos que o sistema converge, mas em tese...)
$|a_{12}^*| + |a_{13}^*| = |0.2| + |0.1| = 0.3$ e
$|a_{21}^*| + |a_{23}^*| = |0.2| + |0.2| = 0.4$ e
$|a_{31}^*| + |a_{32}^*| = |0.2| + |0.3| = 0.5$ e
$\Rightarrow\ max_{1 \leq i \leq n}(0.3, 0.4\ 0.5) = 0.5 < 1$

Iterações

$x^{k+1} = -0.2y^k - 0.1z^k + 0.7$
$y^{k+1} = -0.2x^k - 0.2z^k - 1.6$
$z^{k+1} = -0.2x^k - 0.3y^k + 0.6$
Começando com $(0.7,\ -1.6,\ 0.6)$ obtemos uma primeira aproximação que é $(0.96,\ -1.86,\ 0.94)$ e jogando estes novos valores na fórmula iterativa, obtem-se:

$x \rightarrow 0.7,\ 0.96,\ 0.978,\ 0.9994,\ 0.9979$
$y \rightarrow -1.6,\ -1.86,\ -1.98,\ -1.9888,\ -1.9996$
$z \rightarrow 0.6,\ 0.94,\ 0.966,\ 0.9984,\ 0.9968$

O critério de parada pode ser
$max(x^{k+1} - x^k) < \epsilon$. No exemplo acima $0.0108 < 0.011$. Ou seja, neste caso, a solução procurada é $x = 0.9979$, $y = -1.9996$ e $z = 0.9978$. Salta aos olhos neste caso que se a tolerância for um pouco menor, a resposta exata será encontrada que é $(1, -2, 1)$. Basta lançar este resultado no sistema original que a exatidão será confirmada.

Solução Python

```
#jacobi
def jacobi(a,b):
 import numpy
 n=len(a) #numero de linhas de a
          # len(a[0])=colunas de a
 xvelho=numpy.zeros((n),float)
 erro=numpy.zeros((n),float)
 xnovo=numpy.zeros((n),float)
 f=numpy.zeros((n,len(a[0])),float)
 d=numpy.zeros((len(a[0])),float)
 tol=0.1
 for i in range(0,n):
  for j in range(0,n):
   if i==j:
    f[i,j]=0
   else:
    f[i,j]=-a[i,j]/a[i,i]
 for i in range(0,n):
  d[i]=b[i]/a[i,i]
 erromax=tol+1
 while erromax>tol:
  for i in range(0,n):
   xnovo[i]=0
   for k in range(0,n):
    xnovo[i]=xnovo[i]+f[i,k]*xvelho[k]
  for i in range(0,n):
   xnovo[i]=xnovo[i]+d[i]
  for i in range(0,n):
   erro[i]=abs(xnovo[i]-xvelho[i])
  erromax=erro[0]
  for i in range(1,n):
   if erro[i]>erromax:
    erromax=erro[i]
  for i in range(0,n):
   xvelho[i]=xnovo[i]
 return(xnovo)
import numpy
# para usar faça:
# a=numpy.array([[10,7,3],[1,5,4],[3,5,9]],float)
# b=numpy.array([22,33,44],float)
# print(jacobi(a,b))
```

☞ Para você fazer

Resolva usando este método e anexe a listagem do programa (ou copie no verso desta folha):

$$A = \begin{pmatrix} 94 & 11 & 28 & 34 & 17 \\ 18 & 84 & 19 & 29 & 13 \\ 2 & 27 & 85 & 26 & 23 \\ 1 & 33 & 7 & 54 & 9 \\ 10 & 22 & 25 & 16 & 81 \end{pmatrix}$$

e

$$B = (1356, 1987, 1397, 1375, 2030)$$

As respostas encontradas, com $\epsilon < 0.0001$ e sugestão inicial igual à coluna B são:

x_1	x_2	x_3	x_4	x_5

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:51.2
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
70458 MICCAELA BOGASZ SANTIAGO
19HEL111 - 32 apos 01/10, 50% _____ / / _____

Sistemas Lineares - Jacobi

Este método é iterativo, ou seja, não exato, já que parte-se de um valor atribuído inicial e mediante buscas locais sucessivas, o mesmo converge para a resposta correta. A diferença entre o resultado sugerido e aquele real, é denominada erro, e pode ser tornada tão pequena quanto se queira. O método entretanto tem um senão, que é um critério de convergência bastante rígido. Se não obedecido o algoritmo não converge para a resposta correta, e portanto o método é inútil.

Considere $A.x = b$ um sistema de quações lineares de ordem n e cujo $det(A) \neq 0$. A maneira extensa de escrever tal sistema é

$$\begin{array}{ccccccc} a_{11}x_1 & + & a_{12}x_2 & +...+ & a_{1n}x_n & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & +...+ & a_{2n}x_n & = & b_2 \\ ... & & & & & & \\ a_{n1}x_1 & + & a_{n2}x_2 & +...+ & a_{nn}x_n & = & b_n \end{array}$$

A matriz A dos coeficientes, pode ser decomposta em $A = L + D + R$, onde

L é a matriz inferior de A (o resto é zero)

D é a matriz contendo apenas a diagonal principal de A, o resto é zero.

R é a matriz superior de A (o resto é zero)

Veja por exemplo, se

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \therefore L = \begin{pmatrix} 0 & 0 & 0 \\ 4 & 0 & 0 \\ 7 & 8 & 0 \end{pmatrix},$$

e

$$D = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 9 \end{pmatrix}, R = \begin{pmatrix} 0 & 2 & 3 \\ 0 & 0 & 6 \\ 0 & 0 & 0 \end{pmatrix}$$

Supondo $det(D) \neq 0$, pode-se escrever: $(L + D + R)x = b$ e daqui $Dx = -(L + R)x + b$. e $x = -D^{-1}(L + R)x + D^{-1}b$. Chamando $B = -D^{-1}(L + R)$ e $g = D^{-1}.b$ obtem-se a equação geral deste método que é

$$x = Bx + g$$

. Note que é uma equação recursiva (ou iterativa) já que o x aparece nos 2 lados da mesma. A idéia aqui é atribuir um valor a x e com ele calcular um novo x, que é jogado no lugar do x velho e gera um novo x e assim sucessivamente até haver a convergência esperada (de antemão). Como supusemos $det(D) \neq 0$ pode-se dividir cada equação do sistema pelo coeficiente da diagonal principal, resultando

$$A^* = L^* + I + R^*$$

Agora o processo iterativo pode ser definido como

$$x^{k+1} = -(L^* + R^*)x^k + b^*$$

onde os elementos de L^*, R^* e b^* são:

$l_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i > j$ e zero senão

$d_{ij} = a_{ij}$ se $i = j$ e zero senão

$r_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i < j$ e zero senão

$b_ij^* = \frac{b_i}{a_{ii}}$

Critérios de convergência

Depois de dividida a matriz pelo elemento da diagonal principal, deve-se ter o **critério das linhas** que diz

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}^*| < 1$$

OU então o **critério das colunas** que diz

$$max_{1 \leq j \leq n} \sum_{i=1, i \neq j}^{n} |a_{ij}^*| < 1$$

Note que qualquer um dos dois critérios sendo satisfeitos, isso garante que o sistema tem convergência.

Definindo uma matriz como estritamente diagonalmente dominante se

$$\sum_{j=1, j \neq i}^{n} |a_{ij}| < |a_{ii}|$$

e se A é estritamente diagonalmente dominante, então A satisfaz o critério das linhas, e o sistema converge. Então, um bom critério de convergência pode ser

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}^|* < 1$$

Exemplo

Seja resolver o sistema abaixo, usando o método de Jacobi

$$\begin{array}{ccccccc} 10x & + & 2y & + & z & = & 7 \\ x & + & 5y & + & z & = & -8 \\ 2x & + & 3y & + & 10z & = & 6 \end{array}$$

com $x^0 = (0.7, -1.6\ e\ 0.6)^t$ e $\epsilon < 0.11$.

Convergência ?

$|a_{12}| + |a_{13}| < |a_{11}|$? ou $|2| + |1| < |10|$? R: sim
$|a_{21}| + |a_{23}| < |a_{22}|$? ou $|1| + |1| < |5|$? R: sim
$|a_{31}| + |a_{32}| < |a_{33}|$? ou $|2| + |3| < |10|$? R: sim
logo, o sistema convergirá!

Solução

Dividindo cada equação pelo elemento da diagonal principal, fica

$$\begin{array}{ccccccc} x & + & 0.2y & + & 0.1z & = & 0.7 \\ 0.2x & + & y & + & 0.2z & = & -1.6 \\ 0.2x & + & 0.3y & + & z & = & 0.6 \end{array}$$

Apenas para ilustrar o critério das linhas para testar convergência (desnecessário no caso, já que já vimos que o sistema converge, mas em tese...)
$|a_{12}^*| + |a_{13}^*| = |0.2| + |0.1| = 0.3$ e
$|a_{21}^*| + |a_{23}^*| = |0.2| + |0.2| = 0.4$ e
$|a_{31}^*| + |a_{32}^*| = |0.2| + |0.3| = 0.5$ e
$\Rightarrow\ max_{1 \leq i \leq n}(0.3, 0.4\ 0.5) = 0.5 < 1$

Iterações

$x^{k+1} = -0.2y^k - 0.1z^k + 0.7$
$y^{k+1} = -0.2x^k - 0.2z^k - 1.6$
$z^{k+1} = -0.2x^k - 0.3y^k + 0.6$
Começando com $(0.7,\ -1.6,\ 0.6)$ obtemos uma primeira aproximação que é $(0.96,\ -1.86,\ 0.94)$ e jogando estes novos valores na fórmula iterativa, obtem-se:

$x \rightarrow 0.7,\ 0.96,\ 0.978,\ 0.9994,\ 0.9979$
$y \rightarrow -1.6,\ -1.86,\ -1.98,\ -1.9888,\ -1.9996$
$z \rightarrow 0.6,\ 0.94,\ 0.966,\ 0.9984,\ 0.9968$

O critério de parada pode ser
$max(x^{k+1} - x^k) < \epsilon$. No exemplo acima $0.0108 < 0.011$. Ou seja, neste caso, a solução procurada é $x = 0.9979$, $y = -1.9996$ e $z = 0.9978$. Salta aos olhos neste caso que se a tolerância for um pouco menor, a resposta exata será encontrada que é $(1, -2, 1)$. Basta lançar este resultado no sistema original que a exatidão será confirmada.

Solução Python

```
#jacobi
def jacobi(a,b):
 import numpy
 n=len(a) #numero de linhas de a
          # len(a[0])=colunas de a
 xvelho=numpy.zeros((n),float)
 erro=numpy.zeros((n),float)
 xnovo=numpy.zeros((n),float)
 f=numpy.zeros((n,len(a[0])),float)
 d=numpy.zeros((len(a[0])),float)
 tol=0.1
 for i in range(0,n):
  for j in range(0,n):
   if i==j:
    f[i,j]=0
   else:
    f[i,j]=-a[i,j]/a[i,i]
 for i in range(0,n):
  d[i]=b[i]/a[i,i]
 erromax=tol+1
 while erromax>tol:
  for i in range(0,n):
   xnovo[i]=0
   for k in range(0,n):
    xnovo[i]=xnovo[i]+f[i,k]*xvelho[k]
  for i in range(0,n):
   xnovo[i]=xnovo[i]+d[i]
  for i in range(0,n):
   erro[i]=abs(xnovo[i]-xvelho[i])
  erromax=erro[0]
  for i in range(1,n):
   if erro[i]>erromax:
    erromax=erro[i]
  for i in range(0,n):
   xvelho[i]=xnovo[i]
 return(xnovo)
import numpy
# para usar faça:
# a=numpy.array([[10,7,3],[1,5,4],[3,5,9]],float)
# b=numpy.array([22,33,44],float)
# print(jacobi(a,b))
```

☞ Para você fazer

Resolva usando este método e anexe a listagem do programa (ou copie no verso desta folha):

$$A = \begin{pmatrix} 86 & 10 & 20 & 24 & 30 \\ 14 & 83 & 34 & 13 & 21 \\ 1 & 29 & 46 & 11 & 4 \\ 33 & 16 & 18 & 94 & 23 \\ 12 & 26 & 9 & 8 & 59 \end{pmatrix}$$

e

$$B = (1804, 1666, 1128, 2228, 1491)$$

As respostas encontradas, com $\epsilon < 0.0001$ e sugestão inicial igual à coluna B são:

x_1	x_2	x_3	x_4	x_5

111-70458 - 01/10

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:51.2
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
71194 NATALIA GONCALVES P CHOMA
19HEL111 - 33 apos 01/10, 50% _____ / / _____

Sistemas Lineares - Jacobi

Este método é iterativo, ou seja, não exato, já que parte-se de um valor atribuído inicial e mediante buscas locais sucessivas, o mesmo converge para a resposta correta. A diferença entre o resultado sugerido e aquele real, é denominada erro, e pode ser tornada tão pequena quanto se queira. O método entretanto tem um senão, que é um critério de convergência bastante rígido. Se não obedecido o algoritmo não converge para a resposta correta, e portanto o método é inútil.

Considere $A.x = b$ um sistema de quações lineares de ordem n e cujo $det(A) \neq 0$. A maneira extensa de escrever tal sistema é

$$\begin{array}{lllllll} a_{11}x_1 & + & a_{12}x_2 & +...+ & a_{1n}x_n & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & +...+ & a_{2n}x_n & = & b_2 \\ ... & & & & & & \\ a_{n1}x_1 & + & a_{n2}x_2 & +...+ & a_{nn}x_n & = & b_n \end{array}$$

A matriz A dos coeficientes, pode ser decomposta em $A = L + D + R$, onde

L é a matriz inferior de A (o resto é zero)

D é a matriz contendo apenas a diagonal principal de A, o resto é zero.

R é a matriz superior de A (o resto é zero)

Veja por exemplo, se

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \therefore L = \begin{pmatrix} 0 & 0 & 0 \\ 4 & 0 & 0 \\ 7 & 8 & 0 \end{pmatrix},$$

e

$$D = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 9 \end{pmatrix}, R = \begin{pmatrix} 0 & 2 & 3 \\ 0 & 0 & 6 \\ 0 & 0 & 0 \end{pmatrix}$$

Supondo $det(D) \neq 0$, pode-se escrever: $(L + D + R)x = b$ e daqui $Dx = -(L + R)x + b$. e $x = -D^{-1}(L + R)x + D^{-1}b$. Chamando $B = -D^{-1}(L + R)$ e $g = D^{-1}.b$ obtem-se a equação geral deste método que é

$$x = Bx + g$$

. Note que é uma equação recursiva (ou iterativa) já que o x aparece nos 2 lados da mesma. A idéia aqui é atribuir um valor a x e com ele calcular um novo x, que é jogado no lugar do x velho e gera um novo x e assim sucessivamente até haver a convergência esperada (de antemão). Como supusemos $det(D) \neq 0$ pode-se dividir cada equação do sistema pelo coeficiente da diagonal principal, resultando

$$A^* = L^* + I + R^*$$

Agora o processo iterativo pode ser definido como

$$x^{k+1} = -(L^* + R^*)x^k + b^*$$

onde os elementos de L^*, R^* e b^* são:

$l_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i > j$ e zero senão

$d_{ij} = a_{ij}$ se $i = j$ e zero senão

$r_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i < j$ e zero senão

$b_ij^* = \frac{b_i}{a_{ii}}$

Critérios de convergência

Depois de dividida a matriz pelo elemento da diagonal principal, deve-se ter o **critério das linhas** que diz

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}^*| < 1$$

OU então o **critério das colunas** que diz

$$max_{1 \leq j \leq n} \sum_{i=1, i \neq j}^{n} |a_{ij}^*| < 1$$

Note que qualquer um dos dois critérios sendo satisfeitos, isso garante que o sistema tem convergência.

Definindo uma matriz como estritamente diagonalmente dominante se

$$\sum_{j=1, j \neq i}^{n} |a_{ij}| < |a_{ii}|$$

e se A é estritamente diagonalmente dominante, então A satisfaz o critério das linhas, e o sistema converge. Então, um bom critério de convergência pode ser

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}^|* < 1$$

Exemplo

Seja resolver o sistema abaixo, usando o método de Jacobi

$$\begin{array}{lllllll} 10x & + & 2y & + & z & = & 7 \\ x & + & 5y & + & z & = & -8 \\ 2x & + & 3y & + & 10z & = & 6 \end{array}$$

com $x^0 = (0.7, -1.6\ e\ 0.6)^t$ e $\epsilon < 0.11$.

Convergência ?

$|a_{12}| + |a_{13}| < |a_{11}|$? ou $|2| + |1| < |10|$? R: sim
$|a_{21}| + |a_{23}| < |a_{22}|$? ou $|1| + |1| < |5|$? R: sim
$|a_{31}| + |a_{32}| < |a_{33}|$? ou $|2| + |3| < |10|$? R: sim
logo, o sistema convergirá!

Solução

Dividindo cada equação pelo elemento da diagonal principal, fica

$$\begin{array}{lllllll} x & + & 0.2y & + & 0.1z & = & 0.7 \\ 0.2x & + & y & + & 0.2z & = & -1.6 \\ 0.2x & + & 0.3y & + & z & = & 0.6 \end{array}$$

Apenas para ilustrar o critério das linhas para testar convergência (desnecessário no caso, já que já vimos que o sistema converge, mas em tese...)
$|a_{12}^*| + |a_{13}^*| = |0.2| + |0.1| = 0.3$ e
$|a_{21}^*| + |a_{23}^*| = |0.2| + |0.2| = 0.4$ e
$|a_{31}^*| + |a_{32}^*| = |0.2| + |0.3| = 0.5$ e
$\Rightarrow\ max_{1 \leq i \leq n}(0.3, 0.4\ 0.5) = 0.5 < 1$

Iterações

$x^{k+1} = -0.2y^k - 0.1z^k + 0.7$
$y^{k+1} = -0.2x^k - 0.2z^k - 1.6$
$z^{k+1} = -0.2x^k - 0.3y^k + 0.6$
Começando com $(0.7,\ -1.6,\ 0.6)$ obtemos uma primeira aproximação que é $(0.96,\ -1.86,\ 0.94)$ e jogando estes novos valores na fórmula iterativa, obtem-se:

$x \rightarrow 0.7,\ 0.96,\ 0.978,\ 0.9994,\ 0.9979$
$y \rightarrow -1.6,\ -1.86,\ -1.98,\ -1.9888,\ -1.9996$
$z \rightarrow 0.6,\ 0.94,\ 0.966,\ 0.9984,\ 0.9968$

O critério de parada pode ser
$max(x^{k+1} - x^k) < \epsilon$. No exemplo acima $0.0108 < 0.011$. Ou seja, neste caso, a solução procurada é $x = 0.9979$, $y = -1.9996$ e $z = 0.9978$. Salta aos olhos neste caso que se a tolerância for um pouco menor, a resposta exata será encontrada que é $(1, -2, 1)$. Basta lançar este resultado no sistema original que a exatidão será confirmada.

Solução Python

```
#jacobi
def jacobi(a,b):
 import numpy
 n=len(a) #numero de linhas de a
          # len(a[0])=colunas de a
 xvelho=numpy.zeros((n),float)
 erro=numpy.zeros((n),float)
 xnovo=numpy.zeros((n),float)
 f=numpy.zeros((n,len(a[0])),float)
 d=numpy.zeros((len(a[0])),float)
 tol=0.1
 for i in range(0,n):
  for j in range(0,n):
   if i==j:
    f[i,j]=0
   else:
    f[i,j]=-a[i,j]/a[i,i]
 for i in range(0,n):
  d[i]=b[i]/a[i,i]
 erromax=tol+1
 while erromax>tol:
  for i in range(0,n):
   xnovo[i]=0
   for k in range(0,n):
    xnovo[i]=xnovo[i]+f[i,k]*xvelho[k]
  for i in range(0,n):
   xnovo[i]=xnovo[i]+d[i]
  for i in range(0,n):
   erro[i]=abs(xnovo[i]-xvelho[i])
  erromax=erro[0]
  for i in range(1,n):
   if erro[i]>erromax:
    erromax=erro[i]
  for i in range(0,n):
   xvelho[i]=xnovo[i]
 return(xnovo)
import numpy
# para usar faça:
# a=numpy.array([[10,7,3],[1,5,4],[3,5,9]],float)
# b=numpy.array([22,33,44],float)
# print(jacobi(a,b))
```

☞ Para você fazer

Resolva usando este método e anexe a listagem do programa (ou copie no verso desta folha):

$$A = \begin{pmatrix} 62 & 23 & 5 & 13 & 16 \\ 17 & 88 & 11 & 32 & 24 \\ 33 & 27 & 126 & 34 & 29 \\ 30 & 12 & 25 & 77 & 4 \\ 28 & 10 & 31 & 7 & 77 \end{pmatrix}$$

e

$$B = (740, 1328, 2531, 1813, 1115)$$

As respostas encontradas, com $\epsilon < 0.0001$ e sugestão inicial igual à coluna B são:

x_1	x_2	x_3	x_4	x_5

111-71194 - 01/10

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:51.2
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
70465 PEDRO AFONSO CARDOSO SZPAK
19HEL111 - 34 apos 01/10, 50% _____ / / _____

Sistemas Lineares - Jacobi

Este método é iterativo, ou seja, não exato, já que parte-se de um valor atribuído inicial e mediante buscas locais sucessivas, o mesmo converge para a resposta correta. A diferença entre o resultado sugerido e aquele real, é denominada erro, e pode ser tornada tão pequena quanto se queira. O método entretanto tem um senão, que é um critério de convergência bastante rígido. Se não obedecido o algoritmo não converge para a resposta correta, e portanto o método é inútil.

Considere $A.x = b$ um sistema de quações lineares de ordem n e cujo $det(A) \neq 0$. A maneira extensa de escrever tal sistema é

$$\begin{array}{lcllcl} a_{11}x_1 & + & a_{12}x_2 & +...+ & a_{1n}x_n & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & +...+ & a_{2n}x_n & = & b_2 \\ ... & & & & & & \\ a_{n1}x_1 & + & a_{n2}x_2 & +...+ & a_{nn}x_n & = & b_n \end{array}$$

A matriz A dos coeficientes, pode ser decomposta em $A = L + D + R$, onde

L é a matriz inferior de A (o resto é zero)

D é a matriz contendo apenas a diagonal principal de A, o resto é zero.

R é a matriz superior de A (o resto é zero)

Veja por exemplo, se

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \therefore L = \begin{pmatrix} 0 & 0 & 0 \\ 4 & 0 & 0 \\ 7 & 8 & 0 \end{pmatrix},$$

e

$$D = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 9 \end{pmatrix}, R = \begin{pmatrix} 0 & 2 & 3 \\ 0 & 0 & 6 \\ 0 & 0 & 0 \end{pmatrix}$$

Supondo $det(D) \neq 0$, pode-se escrever: $(L + D + R)x = b$ e daqui $Dx = -(L + R)x + b$. e $x = -D^{-1}(L + R)x + D^{-1}b$. Chamando $B = -D^{-1}(L + R)$ e $g = D^{-1}.b$ obtem-se a equação geral deste método que é

$$x = Bx + g$$

. Note que é uma equação recursiva (ou iterativa) já que o x aparece nos 2 lados da mesma. A idéia aqui é atribuir um valor a x e com ele calcular um novo x, que é jogado no lugar do x velho e gera um novo x e assim sucessivamente até haver a convergência esperada (de antemão). Como supusemos $det(D) \neq 0$ pode-se dividir cada equação do sistema pelo coeficiente da diagonal principal, resultando

$$A^* = L^* + I + R^*$$

Agora o processo iterativo pode ser definido como

$$x^{k+1} = -(L^* + R^*)x^k + b^*$$

onde os elementos de L^*, R^* e b^* são:

$l_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i > j$ e zero senão

$d_{ij} = a_{ij}$ se $i = j$ e zero senão

$r_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i < j$ e zero senão

$b_ij^* = \frac{b_i}{a_{ii}}$

Critérios de convergência

Depois de dividida a matriz pelo elemento da diagonal principal, deve-se ter o **critério das linhas** que diz

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}^*| < 1$$

OU então o **critério das colunas** que diz

$$max_{1 \leq j \leq n} \sum_{i=1, i \neq j}^{n} |a_{ij}^*| < 1$$

Note que qualquer um dos dois critérios sendo satisfeitos, isso garante que o sistema tem convergência.

Definindo uma matriz como estritamente diagonalmente dominante se

$$\sum_{j=1, j \neq i}^{n} |a_{ij}| < |a_{ii}|$$

e se A é estritamente diagonalmente dominante, então A satisfaz o critério das linhas, e o sistema converge. Então, um bom critério de convergência pode ser

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}^|* < 1$$

Exemplo

Seja resolver o sistema abaixo, usando o método de Jacobi

$$\begin{array}{rcrcrcr} 10x & + & 2y & + & z & = & 7 \\ x & + & 5y & + & z & = & -8 \\ 2x & + & 3y & + & 10z & = & 6 \end{array}$$

com $x^0 = (0.7, -1.6\ e\ 0.6)^t$ e $\epsilon < 0.11$.

Convergência ?

$|a_{12}| + |a_{13}| < |a_{11}|$? ou $|2| + |1| < |10|$? R: sim
$|a_{21}| + |a_{23}| < |a_{22}|$? ou $|1| + |1| < |5|$? R: sim
$|a_{31}| + |a_{32}| < |a_{33}|$? ou $|2| + |3| < |10|$? R: sim
logo, o sistema convergirá!

Solução

Dividindo cada equação pelo elemento da diagonal principal, fica

$$\begin{array}{rcrcrcr} x & + & 0.2y & + & 0.1z & = & 0.7 \\ 0.2x & + & y & + & 0.2z & = & -1.6 \\ 0.2x & + & 0.3y & + & z & = & 0.6 \end{array}$$

Apenas para ilustrar o critério das linhas para testar convergência (desnecessário no caso, já que já vimos que o sistema converge, mas em tese...)
$|a_{12}^*| + |a_{13}^*| = |0.2| + |0.1| = 0.3$ e
$|a_{21}^*| + |a_{23}^*| = |0.2| + |0.2| = 0.4$ e
$|a_{31}^*| + |a_{32}^*| = |0.2| + |0.3| = 0.5$ e
$\Rightarrow\ max_{1 \leq i \leq n}(0.3, 0.4\ 0.5) = 0.5 < 1$

Iterações

$x^{k+1} = -0.2y^k - 0.1z^k + 0.7$
$y^{k+1} = -0.2x^k - 0.2z^k - 1.6$
$z^{k+1} = -0.2x^k - 0.3y^k + 0.6$
Começando com $(0.7,\ -1.6,\ 0.6)$ obtemos uma primeira aproximação que é $(0.96,\ -1.86,\ 0.94)$ e jogando estes novos valores na fórmula iterativa, obtem-se:

$x \rightarrow 0.7,\ 0.96,\ 0.978,\ 0.9994,\ 0.9979$
$y \rightarrow -1.6,\ -1.86,\ -1.98,\ -1.9888,\ -1.9996$
$z \rightarrow 0.6,\ 0.94,\ 0.966,\ 0.9984,\ 0.9968$

O critério de parada pode ser
$max(x^{k+1} - x^k) < \epsilon$. No exemplo acima $0.0108 < 0.011$. Ou seja, neste caso, a solução procurada é $x = 0.9979$, $y = -1.9996$ e $z = 0.9978$. Salta aos olhos neste caso que se a tolerância for um pouco menor, a resposta exata será encontrada que é $(1, -2, 1)$. Basta lançar este resultado no sistema original que a exatidão será confirmada.

Solução Python

```
#jacobi
def jacobi(a,b):
 import numpy
 n=len(a) #numero de linhas de a
          # len(a[0])=colunas de a
 xvelho=numpy.zeros((n),float)
 erro=numpy.zeros((n),float)
 xnovo=numpy.zeros((n),float)
 f=numpy.zeros((n,len(a[0])),float)
 d=numpy.zeros((len(a[0])),float)
 tol=0.1
 for i in range(0,n):
  for j in range(0,n):
   if i==j:
    f[i,j]=0
   else:
    f[i,j]=-a[i,j]/a[i,i]
 for i in range(0,n):
  d[i]=b[i]/a[i,i]
 erromax=tol+1
 while erromax>tol:
  for i in range(0,n):
   xnovo[i]=0
   for k in range(0,n):
    xnovo[i]=xnovo[i]+f[i,k]*xvelho[k]
  for i in range(0,n):
   xnovo[i]=xnovo[i]+d[i]
  for i in range(0,n):
   erro[i]=abs(xnovo[i]-xvelho[i])
  erromax=erro[0]
  for i in range(1,n):
   if erro[i]>erromax:
    erromax=erro[i]
  for i in range(0,n):
   xvelho[i]=xnovo[i]
 return(xnovo)
import numpy
# para usar faça:
# a=numpy.array([[10,7,3],[1,5,4],[3,5,9]],float)
# b=numpy.array([22,33,44],float)
# print(jacobi(a,b))
```

☞ Para você fazer

Resolva usando este método e anexe a listagem do programa (ou copie no verso desta folha):

$$A = \begin{pmatrix} 96 & 15 & 17 & 22 & 35 \\ 8 & 94 & 28 & 32 & 24 \\ 26 & 13 & 59 & 7 & 11 \\ 2 & 10 & 9 & 57 & 31 \\ 5 & 29 & 27 & 1 & 63 \end{pmatrix}$$

e

$$B = (2543, 2666, 1300, 1713, 1741)$$

As respostas encontradas, com $\epsilon < 0.0001$ e sugestão inicial igual à coluna B são:

x_1	x_2	x_3	x_4	x_5

111-70465 - 01/10

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:51.2
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
71163 PEDRO HENRIQUE L ARAUJO
19HEL111 - 35 apos 01/10, 50% _____ / / _____

Sistemas Lineares - Jacobi

Este método é iterativo, ou seja, não exato, já que parte-se de um valor atribuído inicial e mediante buscas locais sucessivas, o mesmo converge para a resposta correta. A diferença entre o resultado sugerido e aquele real, é denominada erro, e pode ser tornada tão pequena quanto se queira. O método entretanto tem um senão, que é um critério de convergência bastante rígido. Se não obedecido o algoritmo não converge para a resposta correta, e portanto o método é inútil.

Considere $A.x = b$ um sistema de quações lineares de ordem n e cujo $det(A) \neq 0$. A maneira extensa de escrever tal sistema é

$$\begin{array}{lllllll} a_{11}x_1 & + & a_{12}x_2 & +...+ & a_{1n}x_n & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & +...+ & a_{2n}x_n & = & b_2 \\ ... & & & & & & \\ a_{n1}x_1 & + & a_{n2}x_2 & +...+ & a_{nn}x_n & = & b_n \end{array}$$

A matriz A dos coeficientes, pode ser decomposta em $A = L + D + R$, onde

L é a matriz inferior de A (o resto é zero)

D é a matriz contendo apenas a diagonal principal de A, o resto é zero.

R é a matriz superior de A (o resto é zero)

Veja por exemplo, se

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \therefore L = \begin{pmatrix} 0 & 0 & 0 \\ 4 & 0 & 0 \\ 7 & 8 & 0 \end{pmatrix},$$

e

$$D = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 9 \end{pmatrix}, R = \begin{pmatrix} 0 & 2 & 3 \\ 0 & 0 & 6 \\ 0 & 0 & 0 \end{pmatrix}$$

Supondo $det(D) \neq 0$, pode-se escrever: $(L + D + R)x = b$ e daqui $Dx = -(L + R)x + b$. e $x = -D^{-1}(L + R)x + D^{-1}b$. Chamando $B = -D^{-1}(L + R)$ e $g = D^{-1}.b$ obtem-se a equação geral deste método que é

$$x = Bx + g$$

. Note que é uma equação recursiva (ou iterativa) já que o x aparece nos 2 lados da mesma. A idéia aqui é atribuir um valor a x e com ele calcular um novo x, que é jogado no lugar do x velho e gera um novo x e assim sucessivamente até haver a convergência esperada (de antemão). Como supusemos $det(D) \neq 0$ pode-se dividir cada equação do sistema pelo coeficiente da diagonal principal, resultando

$$A^* = L^* + I + R^*$$

Agora o processo iterativo pode ser definido como

$$x^{k+1} = -(L^* + R^*)x^k + b^*$$

onde os elementos de L^*, R^* e b^* são:

$l_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i > j$ e zero senão

$d_{ij} = a_{ij}$ se $i = j$ e zero senão

$r_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i < j$ e zero senão

$b_ij^* = \frac{b_i}{a_{ii}}$

Critérios de convergência

Depois de dividida a matriz pelo elemento da diagonal principal, deve-se ter o **critério das linhas** que diz

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}^*| < 1$$

OU então o **critério das colunas** que diz

$$max_{1 \leq j \leq n} \sum_{i=1, i \neq j}^{n} |a_{ij}^*| < 1$$

Note que qualquer um dos dois critérios sendo satisfeitos, isso garante que o sistema tem convergência.

Definindo uma matriz como estritamente diagonalmente dominante se

$$\sum_{j=1, j \neq i}^{n} |a_{ij}| < |a_{ii}|$$

e se A é estritamente diagonalmente dominante, então A satisfaz o critério das linhas, e o sistema converge. Então, um bom critério de convergência pode ser

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}^|* < 1$$

Exemplo

Seja resolver o sistema abaixo, usando o método de Jacobi

$$\begin{array}{lllllll} 10x & + & 2y & + & z & = & 7 \\ x & + & 5y & + & z & = & -8 \\ 2x & + & 3y & + & 10z & = & 6 \end{array}$$

com $x^0 = (0.7, -1.6\ e\ 0.6)^t$ e $\epsilon < 0.11$.

Convergência ?

$|a_{12}| + |a_{13}| < |a_{11}|$? ou $|2| + |1| < |10|$? R: sim
$|a_{21}| + |a_{23}| < |a_{22}|$? ou $|1| + |1| < |5|$? R: sim
$|a_{31}| + |a_{32}| < |a_{33}|$? ou $|2| + |3| < |10|$? R: sim
logo, o sistema convergirá!

Solução

Dividindo cada equação pelo elemento da diagonal principal, fica

$$\begin{array}{lllllll} x & + & 0.2y & + & 0.1z & = & 0.7 \\ 0.2x & + & y & + & 0.2z & = & -1.6 \\ 0.2x & + & 0.3y & + & z & = & 0.6 \end{array}$$

Apenas para ilustrar o critério das linhas para testar convergência (desnecessário no caso, já que já vimos que o sistema converge, mas em tese...)
$|a_{12}^*| + |a_{13}^*| = |0.2| + |0.1| = 0.3$ e
$|a_{21}^*| + |a_{23}^*| = |0.2| + |0.2| = 0.4$ e
$|a_{31}^*| + |a_{32}^*| = |0.2| + |0.3| = 0.5$ e
$\Rightarrow\ max_{1 \leq i \leq n}(0.3, 0.4\ 0.5) = 0.5 < 1$

Iterações

$x^{k+1} = -0.2y^k - 0.1z^k + 0.7$
$y^{k+1} = -0.2x^k - 0.2z^k - 1.6$
$z^{k+1} = -0.2x^k - 0.3y^k + 0.6$
Começando com $(0.7,\ -1.6,\ 0.6)$ obtemos uma primeira aproximação que é $(0.96,\ -1.86,\ 0.94)$ e jogando estes novos valores na fórmula iterativa, obtem-se:

$x \rightarrow 0.7,\ 0.96,\ 0.978,\ 0.9994,\ 0.9979$
$y \rightarrow -1.6,\ -1.86,\ -1.98,\ -1.9888,\ -1.9996$
$z \rightarrow 0.6,\ 0.94,\ 0.966,\ 0.9984,\ 0.9968$

O critério de parada pode ser
$max(x^{k+1} - x^k) < \epsilon$. No exemplo acima $0.0108 < 0.011$. Ou seja, neste caso, a solução procurada é $x = 0.9979$, $y = -1.9996$ e $z = 0.9978$. Salta aos olhos neste caso que se a tolerância for um pouco menor, a resposta exata será encontrada que é $(1, -2, 1)$. Basta lançar este resultado no sistema original que a exatidão será confirmada.

Solução Python

```
#jacobi
def jacobi(a,b):
 import numpy
 n=len(a) #numero de linhas de a
          # len(a[0])=colunas de a
 xvelho=numpy.zeros((n),float)
 erro=numpy.zeros((n),float)
 xnovo=numpy.zeros((n),float)
 f=numpy.zeros((n,len(a[0])),float)
 d=numpy.zeros((len(a[0])),float)
 tol=0.1
 for i in range(0,n):
  for j in range(0,n):
   if i==j:
    f[i,j]=0
   else:
    f[i,j]=-a[i,j]/a[i,i]
 for i in range(0,n):
  d[i]=b[i]/a[i,i]
 erromax=tol+1
 while erromax>tol:
  for i in range(0,n):
   xnovo[i]=0
   for k in range(0,n):
    xnovo[i]=xnovo[i]+f[i,k]*xvelho[k]
  for i in range(0,n):
   xnovo[i]=xnovo[i]+d[i]
  for i in range(0,n):
   erro[i]=abs(xnovo[i]-xvelho[i])
  erromax=erro[0]
  for i in range(1,n):
   if erro[i]>erromax:
    erromax=erro[i]
  for i in range(0,n):
   xvelho[i]=xnovo[i]
 return(xnovo)
import numpy
# para usar faça:
# a=numpy.array([[10,7,3],[1,5,4],[3,5,9]],float)
# b=numpy.array([22,33,44],float)
# print(jacobi(a,b))
```

☞ Para você fazer

Resolva usando este método e anexe a listagem do programa (ou copie no verso desta folha):

$$A = \begin{pmatrix} 121 & 28 & 27 & 32 & 30 \\ 17 & 70 & 4 & 23 & 24 \\ 20 & 25 & 68 & 15 & 7 \\ 5 & 35 & 34 & 89 & 8 \\ 29 & 10 & 3 & 16 & 64 \end{pmatrix}$$

e

$$B = (2364, 891, 1424, 1888, 932)$$

As respostas encontradas, com $\epsilon < 0.0001$ e sugestão inicial igual à coluna B são:

x_1	x_2	x_3	x_4	x_5

111-71163 - 01/10

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:51.2
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
70472 RODOLPHO DE SOUZA DAVID
19HEL111 - 36 apos 01/10, 50% _____ / / _____

Sistemas Lineares - Jacobi

Este método é iterativo, ou seja, não exato, já que parte-se de um valor atribuído inicial e mediante buscas locais sucessivas, o mesmo converge para a resposta correta. A diferença entre o resultado sugerido e aquele real, é denominada erro, e pode ser tornada tão pequena quanto se queira. O método entretanto tem um senão, que é um critério de convergência bastante rígido. Se não obedecido o algoritmo não converge para a resposta correta, e portanto o método é inútil.

Considere $A.x = b$ um sistema de quações lineares de ordem n e cujo $det(A) \neq 0$. A maneira extensa de escrever tal sistema é

$$\begin{array}{ccccccc} a_{11}x_1 & + & a_{12}x_2 & +...+ & a_{1n}x_n & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & +...+ & a_{2n}x_n & = & b_2 \\ ... & & & & & & \\ a_{n1}x_1 & + & a_{n2}x_2 & +...+ & a_{nn}x_n & = & b_n \end{array}$$

A matriz A dos coeficientes, pode ser decomposta em $A = L + D + R$, onde

L é a matriz inferior de A (o resto é zero)

D é a matriz contendo apenas a diagonal principal de A, o resto é zero.

R é a matriz superior de A (o resto é zero)

Veja por exemplo, se

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \therefore L = \begin{pmatrix} 0 & 0 & 0 \\ 4 & 0 & 0 \\ 7 & 8 & 0 \end{pmatrix},$$

e

$$D = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 9 \end{pmatrix}, R = \begin{pmatrix} 0 & 2 & 3 \\ 0 & 0 & 6 \\ 0 & 0 & 0 \end{pmatrix}$$

Supondo $det(D) \neq 0$, pode-se escrever: $(L + D + R)x = b$ e daqui $Dx = -(L + R)x + b$. e $x = -D^{-1}(L + R)x + D^{-1}b$. Chamando $B = -D^{-1}(L + R)$ e $g = D^{-1}.b$ obtem-se a equação geral deste método que é

$$x = Bx + g$$

. Note que é uma equação recursiva (ou iterativa) já que o x aparece nos 2 lados da mesma. A idéia aqui é atribuir um valor a x e com ele calcular um novo x, que é jogado no lugar do x velho e gera um novo x e assim sucessivamente até haver a convergência esperada (de antemão). Como supusemos $det(D) \neq 0$ pode-se dividir cada equação do sistema pelo coeficiente da diagonal principal, resultando

$$A^* = L^* + I + R^*$$

Agora o processo iterativo pode ser definido como

$$x^{k+1} = -(L^* + R^*)x^k + b^*$$

onde os elementos de L^*, R^* e b^* são:

$l_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i > j$ e zero senão

$d_{ij} = a_{ij}$ se $i = j$ e zero senão

$r_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i < j$ e zero senão

$b_ij^* = \frac{b_i}{a_{ii}}$

Critérios de convergência

Depois de dividida a matriz pelo elemento da diagonal principal, deve-se ter o **critério das linhas** que diz

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}^*| < 1$$

OU então o **critério das colunas** que diz

$$max_{1 \leq j \leq n} \sum_{i=1, i \neq j}^{n} |a_{ij}^*| < 1$$

Note que qualquer um dos dois critérios sendo satisfeitos, isso garante que o sistema tem convergência.

Definindo uma matriz como estritamente diagonalmente dominante se

$$\sum_{j=1, j \neq i}^{n} |a_{ij}| < |a_{ii}|$$

e se A é estritamente diagonalmente dominante, então A satisfaz o critério das linhas, e o sistema converge. Então, um bom critério de convergência pode ser

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}^|* < 1$$

Exemplo

Seja resolver o sistema abaixo, usando o método de Jacobi

$$\begin{array}{ccccccc} 10x & + & 2y & + & z & = & 7 \\ x & + & 5y & + & z & = & -8 \\ 2x & + & 3y & + & 10z & = & 6 \end{array}$$

com $x^0 = (0.7, -1.6\ e\ 0.6)^t$ e $\epsilon < 0.11$.

Convergência ?

$|a_{12}| + |a_{13}| < |a_{11}|$? ou $|2| + |1| < |10|$? R: sim
$|a_{21}| + |a_{23}| < |a_{22}|$? ou $|1| + |1| < |5|$? R: sim
$|a_{31}| + |a_{32}| < |a_{33}|$? ou $|2| + |3| < |10|$? R: sim
logo, o sistema convergirá!

Solução

Dividindo cada equação pelo elemento da diagonal principal, fica

$$\begin{array}{ccccccc} x & + & 0.2y & + & 0.1z & = & 0.7 \\ 0.2x & + & y & + & 0.2z & = & -1.6 \\ 0.2x & + & 0.3y & + & z & = & 0.6 \end{array}$$

Apenas para ilustrar o critério das linhas para testar convergência (desnecessário no caso, já que já vimos que o sistema converge, mas em tese...)
$|a_{12}^*| + |a_{13}^*| = |0.2| + |0.1| = 0.3$ e
$|a_{21}^*| + |a_{23}^*| = |0.2| + |0.2| = 0.4$ e
$|a_{31}^*| + |a_{32}^*| = |0.2| + |0.3| = 0.5$ e
$\Rightarrow\ max_{1 \leq i \leq n}(0.3, 0.4\ 0.5) = 0.5 < 1$

Iterações

$x^{k+1} = -0.2y^k - 0.1z^k + 0.7$
$y^{k+1} = -0.2x^k - 0.2z^k - 1.6$
$z^{k+1} = -0.2x^k - 0.3y^k + 0.6$
Começando com $(0.7,\ -1.6,\ 0.6)$ obtemos uma primeira aproximação que é $(0.96,\ -1.86,\ 0.94)$ e jogando estes novos valores na fórmula iterativa, obtem-se:

$x \rightarrow 0.7,\ 0.96,\ 0.978,\ 0.9994,\ 0.9979$
$y \rightarrow -1.6,\ -1.86,\ -1.98,\ -1.9888,\ -1.9996$
$z \rightarrow 0.6,\ 0.94,\ 0.966,\ 0.9984,\ 0.9968$

O critério de parada pode ser
$max(x^{k+1} - x^k) < \epsilon$. No exemplo acima $0.0108 < 0.011$. Ou seja, neste caso, a solução procurada é $x = 0.9979$, $y = -1.9996$ e $z = 0.9978$. Salta aos olhos neste caso que se a tolerância for um pouco menor, a resposta exata será encontrada que é $(1, -2, 1)$. Basta lançar este resultado no sistema original que a exatidão será confirmada.

Solução Python

```
#jacobi
def jacobi(a,b):
 import numpy
 n=len(a) #numero de linhas de a
          # len(a[0])=colunas de a
 xvelho=numpy.zeros((n),float)
 erro=numpy.zeros((n),float)
 xnovo=numpy.zeros((n),float)
 f=numpy.zeros((n,len(a[0])),float)
 d=numpy.zeros((len(a[0])),float)
 tol=0.1
 for i in range(0,n):
  for j in range(0,n):
   if i==j:
    f[i,j]=0
   else:
    f[i,j]=-a[i,j]/a[i,i]
 for i in range(0,n):
  d[i]=b[i]/a[i,i]
 erromax=tol+1
 while erromax>tol:
  for i in range(0,n):
   xnovo[i]=0
   for k in range(0,n):
    xnovo[i]=xnovo[i]+f[i,k]*xvelho[k]
  for i in range(0,n):
   xnovo[i]=xnovo[i]+d[i]
  for i in range(0,n):
   erro[i]=abs(xnovo[i]-xvelho[i])
  erromax=erro[0]
  for i in range(1,n):
   if erro[i]>erromax:
    erromax=erro[i]
  for i in range(0,n):
   xvelho[i]=xnovo[i]
 return(xnovo)
import numpy
# para usar faça:
# a=numpy.array([[10,7,3],[1,5,4],[3,5,9]],float)
# b=numpy.array([22,33,44],float)
# print(jacobi(a,b))
```

☞ Para você fazer

Resolva usando este método e anexe a listagem do programa (ou copie no verso desta folha):

$$A = \begin{pmatrix} 78 & 29 & 16 & 27 & 2 \\ 22 & 89 & 25 & 32 & 7 \\ 5 & 8 & 39 & 17 & 6 \\ 14 & 23 & 18 & 87 & 28 \\ 10 & 13 & 20 & 3 & 47 \end{pmatrix}$$

e

$$B = (1459, 1185, 653, 1697, 539)$$

As respostas encontradas, com $\epsilon < 0.0001$ e sugestão inicial igual à coluna B são:

x_1	x_2	x_3	x_4	x_5

111-70472 - 01/10

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:51.2
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
70489 TAYANE CORDEIRO FARIA 19HEL111 - 37 apos 01/10, 50% _____ / _____ / _____

Sistemas Lineares - Jacobi

Este método é iterativo, ou seja, não exato, já que parte-se de um valor atribuído inicial e mediante buscas locais sucessivas, o mesmo converge para a resposta correta. A diferença entre o resultado sugerido e aquele real, é denominada erro, e pode ser tornada tão pequena quanto se queira. O método entretanto tem um senão, que é um critério de convergência bastante rígido. Se não obedecido o algoritmo não converge para a resposta correta, e portanto o método é inútil.

Considere $A.x = b$ um sistema de quações lineares de ordem n e cujo $det(A) \neq 0$. A maneira extensa de escrever tal sistema é

$$\begin{array}{ccccccccc} a_{11}x_1 & + & a_{12}x_2 & +...+ & a_{1n}x_n & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & +...+ & a_{2n}x_n & = & b_2 \\ ... & & & & & & \\ a_{n1}x_1 & + & a_{n2}x_2 & +...+ & a_{nn}x_n & = & b_n \end{array}$$

A matriz A dos coeficientes, pode ser decomposta em $A = L + D + R$, onde

L é a matriz inferior de A (o resto é zero)

D é a matriz contendo apenas a diagonal principal de A, o resto é zero.

R é a matriz superior de A (o resto é zero)

Veja por exemplo, se

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \therefore L = \begin{pmatrix} 0 & 0 & 0 \\ 4 & 0 & 0 \\ 7 & 8 & 0 \end{pmatrix},$$

e

$$D = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 9 \end{pmatrix}, R = \begin{pmatrix} 0 & 2 & 3 \\ 0 & 0 & 6 \\ 0 & 0 & 0 \end{pmatrix}$$

Supondo $det(D) \neq 0$, pode-se escrever: $(L + D + R)x = b$ e daqui $Dx = -(L + R)x + b$. e $x = -D^{-1}(L + R)x + D^{-1}b$. Chamando $B = -D^{-1}(L + R)$ e $g = D^{-1}.b$ obtem-se a equação geral deste método que é

$$x = Bx + g$$

. Note que é uma equação recursiva (ou iterativa) já que o x aparece nos 2 lados da mesma. A idéia aqui é atribuir um valor a x e com ele calcular um novo x, que é jogado no lugar do x velho e gera um novo x e assim sucessivamente até haver a convergência esperada (de antemão). Como supusemos $det(D) \neq 0$ pode-se dividir cada equação do sistema pelo coeficiente da diagonal principal, resultando

$$A^* = L^* + I + R^*$$

Agora o processo iterativo pode ser definido como

$$x^{k+1} = -(L^* + R^*)x^k + b^*$$

onde os elementos de L^*, R^* e b^* são:

$l_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i > j$ e zero senão

$d_{ij} = a_{ij}$ se $i = j$ e zero senão

$r_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i < j$ e zero senão

$b_ij^* = \frac{b_i}{a_{ii}}$

Critérios de convergência

Depois de dividida a matriz pelo elemento da diagonal principal, deve-se ter o **critério das linhas** que diz

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}^*| < 1$$

OU então o **critério das colunas** que diz

$$max_{1 \leq j \leq n} \sum_{i=1, i \neq j}^{n} |a_{ij}^*| < 1$$

Note que qualquer um dos dois critérios sendo satisfeitos, isso garante que o sistema tem convergência.

Definindo uma matriz como estritamente diagonalmente dominante se

$$\sum_{j=1, j \neq i}^{n} |a_{ij}| < |a_{ii}|$$

e se A é estritamente diagonalmente dominante, então A satisfaz o critério das linhas, e o sistema converge. Então, um bom critério de convergência pode ser

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}|* < 1$$

Exemplo

Seja resolver o sistema abaixo, usando o método de Jacobi

$$\begin{array}{ccccccc} 10x & + & 2y & + & z & = & 7 \\ x & + & 5y & + & z & = & -8 \\ 2x & + & 3y & + & 10z & = & 6 \end{array}$$

com $x^0 = (0.7, -1.6\ e\ 0.6)^t$ e $\epsilon < 0.11$.

Convergência ?

$|a_{12}| + |a_{13}| < |a_{11}|$? ou $|2| + |1| < |10|$? R: sim
$|a_{21}| + |a_{23}| < |a_{22}|$? ou $|1| + |1| < |5|$? R: sim
$|a_{31}| + |a_{32}| < |a_{33}|$? ou $|2| + |3| < |10|$? R: sim
logo, o sistema convergirá!

Solução

Dividindo cada equação pelo elemento da diagonal principal, fica

$$\begin{array}{ccccccc} x & + & 0.2y & + & 0.1z & = & 0.7 \\ 0.2x & + & y & + & 0.2z & = & -1.6 \\ 0.2x & + & 0.3y & + & z & = & 0.6 \end{array}$$

Apenas para ilustrar o critério das linhas para testar convergência (desnecessário no caso, já que já vimos que o sistema converge, mas em tese...)
$|a_{12}^*| + |a_{13}^*| = |0.2| + |0.1| = 0.3$ e
$|a_{21}^*| + |a_{23}^*| = |0.2| + |0.2| = 0.4$ e
$|a_{31}^*| + |a_{32}^*| = |0.2| + |0.3| = 0.5$ e
$\Rightarrow\ max_{1 \leq i \leq n}(0.3, 0.4\ 0.5) = 0.5 < 1$

Iterações

$x^{k+1} = -0.2y^k - 0.1z^k + 0.7$
$y^{k+1} = -0.2x^k - 0.2z^k - 1.6$
$z^{k+1} = -0.2x^k - 0.3y^k + 0.6$
Começando com $(0.7,\ -1.6,\ 0.6)$ obtemos uma primeira aproximação que é $(0.96,\ -1.86,\ 0.94)$ e jogando estes novos valores na fórmula iterativa, obtem-se:

$x \rightarrow 0.7,\ 0.96,\ 0.978,\ 0.9994,\ 0.9979$
$y \rightarrow -1.6,\ -1.86,\ -1.98,\ -1.9888,\ -1.9996$
$z \rightarrow 0.6,\ 0.94,\ 0.966,\ 0.9984,\ 0.9968$

O critério de parada pode ser
$max(x^{k+1} - x^k) < \epsilon$. No exemplo acima $0.0108 < 0.011$. Ou seja, neste caso, a solução procurada é $x = 0.9979$, $y = -1.9996$ e $z = 0.9978$. Salta aos olhos neste caso que se a tolerância for um pouco menor, a resposta exata será encontrada que é $(1, -2, 1)$. Basta lançar este resultado no sistema original que a exatidão será confirmada.

Solução Python

```
#jacobi
def jacobi(a,b):
 import numpy
 n=len(a) #numero de linhas de a
          # len(a[0])=colunas de a
 xvelho=numpy.zeros((n),float)
 erro=numpy.zeros((n),float)
 xnovo=numpy.zeros((n),float)
 f=numpy.zeros((n,len(a[0])),float)
 d=numpy.zeros((len(a[0])),float)
 tol=0.1
 for i in range(0,n):
  for j in range(0,n):
   if i==j:
    f[i,j]=0
   else:
    f[i,j]=-a[i,j]/a[i,i]
 for i in range(0,n):
  d[i]=b[i]/a[i,i]
 erromax=tol+1
 while erromax>tol:
  for i in range(0,n):
   xnovo[i]=0
   for k in range(0,n):
    xnovo[i]=xnovo[i]+f[i,k]*xvelho[k]
  for i in range(0,n):
   xnovo[i]=xnovo[i]+d[i]
  for i in range(0,n):
   erro[i]=abs(xnovo[i]-xvelho[i])
  erromax=erro[0]
  for i in range(1,n):
   if erro[i]>erromax:
    erromax=erro[i]
  for i in range(0,n):
   xvelho[i]=xnovo[i]
 return(xnovo)
import numpy
# para usar faça:
# a=numpy.array([[10,7,3],[1,5,4],[3,5,9]],float)
# b=numpy.array([22,33,44],float)
# print(jacobi(a,b))
```

☞ Para você fazer

Resolva usando este método e anexe a listagem do programa (ou copie no verso desta folha):

$$A = \begin{pmatrix} 48 & 30 & 4 & 2 & 5 \\ 25 & 65 & 10 & 15 & 13 \\ 22 & 24 & 75 & 3 & 23 \\ 29 & 31 & 6 & 90 & 18 \\ 19 & 27 & 21 & 8 & 78 \end{pmatrix}$$

e

$$B = (836, 1243, 1661, 1066, 1262)$$

As respostas encontradas, com $\epsilon < 0.0001$ e sugestão inicial igual à coluna B são:

x_1	x_2	x_3	x_4	x_5

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:51.2
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
70496 THIAGO ENRIQUE B. FIORESE
19HEL111 - 38 apos 01/10, 50% _____ / / _____

Sistemas Lineares - Jacobi

Este método é iterativo, ou seja, não exato, já que parte-se de um valor atribuído inicial e mediante buscas locais sucessivas, o mesmo converge para a resposta correta. A diferença entre o resultado sugerido e aquele real, é denominada erro, e pode ser tornada tão pequena quanto se queira. O método entretanto tem um senão, que é um critério de convergência bastante rígido. Se não obedecido o algoritmo não converge para a resposta correta, e portanto o método é inútil.

Considere $A.x = b$ um sistema de quações lineares de ordem n e cujo $det(A) \neq 0$. A maneira extensa de escrever tal sistema é

$$\begin{array}{lllllll} a_{11}x_1 & + & a_{12}x_2 & +...+ & a_{1n}x_n & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & +...+ & a_{2n}x_n & = & b_2 \\ ... & & & & & & \\ a_{n1}x_1 & + & a_{n2}x_2 & +...+ & a_{nn}x_n & = & b_n \end{array}$$

A matriz A dos coeficientes, pode ser decomposta em $A = L + D + R$, onde

L é a matriz inferior de A (o resto é zero)

D é a matriz contendo apenas a diagonal principal de A, o resto é zero.

R é a matriz superior de A (o resto é zero)

Veja por exemplo, se

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \therefore L = \begin{pmatrix} 0 & 0 & 0 \\ 4 & 0 & 0 \\ 7 & 8 & 0 \end{pmatrix},$$

e

$$D = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 9 \end{pmatrix}, R = \begin{pmatrix} 0 & 2 & 3 \\ 0 & 0 & 6 \\ 0 & 0 & 0 \end{pmatrix}$$

Supondo $det(D) \neq 0$, pode-se escrever: $(L + D + R)x = b$ e daqui $Dx = -(L + R)x + b$. e $x = -D^{-1}(L + R)x + D^{-1}b$. Chamando $B = -D^{-1}(L + R)$ e $g = D^{-1}.b$ obtem-se a equação geral deste método que é

$$x = Bx + g$$

. Note que é uma equação recursiva (ou iterativa) já que o x aparece nos 2 lados da mesma. A idéia aqui é atribuir um valor a x e com ele calcular um novo x, que é jogado no lugar do x velho e gera um novo x e assim sucessivamente até haver a convergência esperada (de antemão). Como supusemos $det(D) \neq 0$ pode-se dividir cada equação do sistema pelo coeficiente da diagonal principal, resultando

$$A^* = L^* + I + R^*$$

Agora o processo iterativo pode ser definido como

$$x^{k+1} = -(L^* + R^*)x^k + b^*$$

onde os elementos de L^*, R^* e b^* são:

$l_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i > j$ e zero senão

$d_{ij} = a_{ij}$ se $i = j$ e zero senão

$r_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i < j$ e zero senão

$b_ij^* = \frac{b_i}{a_{ii}}$

Critérios de convergência

Depois de dividida a matriz pelo elemento da diagonal principal, deve-se ter o **critério das linhas** que diz

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}^*| < 1$$

OU então o **critério das colunas** que diz

$$max_{1 \leq j \leq n} \sum_{i=1, i \neq j}^{n} |a_{ij}^*| < 1$$

Note que qualquer um dos dois critérios sendo satisfeitos, isso garante que o sistema tem convergência.

Definindo uma matriz como estritamente diagonalmente dominante se

$$\sum_{j=1, j \neq i}^{n} |a_{ij}| < |a_{ii}|$$

e se A é estritamente diagonalmente dominante, então A satisfaz o critério das linhas, e o sistema converge. Então, um bom critério de convergência pode ser

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}^|* < 1$$

Exemplo

Seja resolver o sistema abaixo, usando o método de Jacobi

$$\begin{array}{lllllll} 10x & + & 2y & + & z & = & 7 \\ x & + & 5y & + & z & = & -8 \\ 2x & + & 3y & + & 10z & = & 6 \end{array}$$

com $x^0 = (0.7, -1.6\ e\ 0.6)^t$ e $\epsilon < 0.11$.

Convergência ?

$|a_{12}| + |a_{13}| < |a_{11}|$? ou $|2| + |1| < |10|$? R: sim
$|a_{21}| + |a_{23}| < |a_{22}|$? ou $|1| + |1| < |5|$? R: sim
$|a_{31}| + |a_{32}| < |a_{33}|$? ou $|2| + |3| < |10|$? R: sim
logo, o sistema convergirá!

Solução

Dividindo cada equação pelo elemento da diagonal principal, fica

$$\begin{array}{lllllll} x & + & 0.2y & + & 0.1z & = & 0.7 \\ 0.2x & + & y & + & 0.2z & = & -1.6 \\ 0.2x & + & 0.3y & + & z & = & 0.6 \end{array}$$

Apenas para ilustrar o critério das linhas para testar convergência (desnecessário no caso, já que já vimos que o sistema converge, mas em tese...)
$|a_{12}^*| + |a_{13}^*| = |0.2| + |0.1| = 0.3$ e
$|a_{21}^*| + |a_{23}^*| = |0.2| + |0.2| = 0.4$ e
$|a_{31}^*| + |a_{32}^*| = |0.2| + |0.3| = 0.5$ e
$\Rightarrow\ max_{1 \leq i \leq n}(0.3, 0.4\ 0.5) = 0.5 < 1$

Iterações

$x^{k+1} = -0.2y^k - 0.1z^k + 0.7$
$y^{k+1} = -0.2x^k - 0.2z^k - 1.6$
$z^{k+1} = -0.2x^k - 0.3y^k + 0.6$
Começando com $(0.7,\ -1.6,\ 0.6)$ obtemos uma primeira aproximação que é $(0.96,\ -1.86,\ 0.94)$ e jogando estes novos valores na fórmula iterativa, obtem-se:

$x \rightarrow 0.7,\ 0.96,\ 0.978,\ 0.9994,\ 0.9979$
$y \rightarrow -1.6,\ -1.86,\ -1.98,\ -1.9888,\ -1.9996$
$z \rightarrow 0.6,\ 0.94,\ 0.966,\ 0.9984,\ 0.9968$

O critério de parada pode ser
$max(x^{k+1} - x^k) < \epsilon$. No exemplo acima $0.0108 < 0.011$. Ou seja, neste caso, a solução procurada é $x = 0.9979$, $y = -1.9996$ e $z = 0.9978$. Salta aos olhos neste caso que se a tolerância for um pouco menor, a resposta exata será encontrada que é $(1, -2, 1)$. Basta lançar este resultado no sistema original que a exatidão será confirmada.

Solução Python

```
#jacobi
def jacobi(a,b):
 import numpy
 n=len(a) #numero de linhas de a
          # len(a[0])=colunas de a
 xvelho=numpy.zeros((n),float)
 erro=numpy.zeros((n),float)
 xnovo=numpy.zeros((n),float)
 f=numpy.zeros((n,len(a[0])),float)
 d=numpy.zeros((len(a[0])),float)
 tol=0.1
 for i in range(0,n):
  for j in range(0,n):
   if i==j:
    f[i,j]=0
   else:
    f[i,j]=-a[i,j]/a[i,i]
 for i in range(0,n):
  d[i]=b[i]/a[i,i]
 erromax=tol+1
 while erromax>tol:
  for i in range(0,n):
   xnovo[i]=0
   for k in range(0,n):
    xnovo[i]=xnovo[i]+f[i,k]*xvelho[k]
  for i in range(0,n):
   xnovo[i]=xnovo[i]+d[i]
  for i in range(0,n):
   erro[i]=abs(xnovo[i]-xvelho[i])
  erromax=erro[0]
  for i in range(1,n):
   if erro[i]>erromax:
    erromax=erro[i]
  for i in range(0,n):
   xvelho[i]=xnovo[i]
 return(xnovo)
import numpy
# para usar faça:
# a=numpy.array([[10,7,3],[1,5,4],[3,5,9]],float)
# b=numpy.array([22,33,44],float)
# print(jacobi(a,b))
```

☞ Para você fazer

Resolva usando este método e anexe a listagem do programa (ou copie no verso desta folha):

$$A = \begin{pmatrix} 68 & 27 & 6 & 7 & 21 \\ 33 & 89 & 2 & 16 & 32 \\ 11 & 17 & 72 & 13 & 29 \\ 19 & 9 & 4 & 58 & 23 \\ 1 & 24 & 26 & 35 & 90 \end{pmatrix}$$

e

$$B = (1623, 1712, 1132, 1185, 1250)$$

As respostas encontradas, com $\epsilon < 0.0001$ e sugestão inicial igual à coluna B são:

x_1	x_2	x_3	x_4	x_5

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:51.2
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
70515 VALTER WERGENSKI 19HEL111 - 39
apos 01/10, 50% _____ / _____ / _____

Sistemas Lineares - Jacobi

Este método é iterativo, ou seja, não exato, já que parte-se de um valor atribuído inicial e mediante buscas locais sucessivas, o mesmo converge para a resposta correta. A diferença entre o resultado sugerido e aquele real, é denominada erro, e pode ser tornada tão pequena quanto se queira. O método entretanto tem um senão, que é um critério de convergência bastante rígido. Se não obedecido o algoritmo não converge para a resposta correta, e portanto o método é inútil.

Considere $A.x = b$ um sistema de quações lineares de ordem n e cujo $det(A) \neq 0$. A maneira extensa de escrever tal sistema é

$$\begin{array}{lllllll} a_{11}x_1 & + & a_{12}x_2 & +...+ & a_{1n}x_n & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & +...+ & a_{2n}x_n & = & b_2 \\ ... & & & & & & \\ a_{n1}x_1 & + & a_{n2}x_2 & +...+ & a_{nn}x_n & = & b_n \end{array}$$

A matriz A dos coeficientes, pode ser decomposta em $A = L + D + R$, onde

L é a matriz inferior de A (o resto é zero)

D é a matriz contendo apenas a diagonal principal de A, o resto é zero.

R é a matriz superior de A (o resto é zero)

Veja por exemplo, se

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \therefore L = \begin{pmatrix} 0 & 0 & 0 \\ 4 & 0 & 0 \\ 7 & 8 & 0 \end{pmatrix},$$

e

$$D = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 9 \end{pmatrix}, R = \begin{pmatrix} 0 & 2 & 3 \\ 0 & 0 & 6 \\ 0 & 0 & 0 \end{pmatrix}$$

Supondo $det(D) \neq 0$, pode-se escrever: $(L + D + R)x = b$ e daqui $Dx = -(L + R)x + b$. e $x = -D^{-1}(L + R)x + D^{-1}b$. Chamando $B = -D^{-1}(L + R)$ e $g = D^{-1}.b$ obtem-se a equação geral deste método que é

$$x = Bx + g$$

. Note que é uma equação recursiva (ou iterativa) já que o x aparece nos 2 lados da mesma. A idéia aqui é atribuir um valor a x e com ele calcular um novo x, que é jogado no lugar do x velho e gera um novo x e assim sucessivamente até haver a convergência esperada (de antemão). Como supusemos $det(D) \neq 0$ pode-se dividir cada equação do sistema pelo coeficiente da diagonal principal, resultando

$$A^* = L^* + I + R^*$$

Agora o processo iterativo pode ser definido como

$$x^{k+1} = -(L^* + R^*)x^k + b^*$$

onde os elementos de L^*, R^* e b^* são:

$l_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i > j$ e zero senão

$d_{ij} = a_{ij}$ se $i = j$ e zero senão

$r_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i < j$ e zero senão

$b_ij^* = \frac{b_i}{a_{ii}}$

Critérios de convergência

Depois de dividida a matriz pelo elemento da diagonal principal, deve-se ter o **critério das linhas** que diz

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}^*| < 1$$

OU então o **critério das colunas** que diz

$$max_{1 \leq j \leq n} \sum_{i=1, i \neq j}^{n} |a_{ij}^*| < 1$$

Note que qualquer um dos dois critérios sendo satisfeitos, isso garante que o sistema tem convergência.

Definindo uma matriz como estritamente diagonalmente dominante se

$$\sum_{j=1, j \neq i}^{n} |a_{ij}| < |a_{ii}|$$

e se A é estritamente diagonalmente dominante, então A satisfaz o critério das linhas, e o sistema converge. Então, um bom critério de convergência pode ser

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}|* < 1$$

Exemplo

Seja resolver o sistema abaixo, usando o método de Jacobi

$$\begin{array}{lllllll} 10x & + & 2y & + & z & = & 7 \\ x & + & 5y & + & z & = & -8 \\ 2x & + & 3y & + & 10z & = & 6 \end{array}$$

com $x^0 = (0.7, -1.6\ e\ 0.6)^t$ e $\epsilon < 0.11$.

Convergência ?

$|a_{12}| + |a_{13}| < |a_{11}|$? ou $|2| + |1| < |10|$? R: sim
$|a_{21}| + |a_{23}| < |a_{22}|$? ou $|1| + |1| < |5|$? R: sim
$|a_{31}| + |a_{32}| < |a_{33}|$? ou $|2| + |3| < |10|$? R: sim
logo, o sistema convergirá!

Solução

Dividindo cada equação pelo elemento da diagonal principal, fica

$$\begin{array}{lllllll} x & + & 0.2y & + & 0.1z & = & 0.7 \\ 0.2x & + & y & + & 0.2z & = & -1.6 \\ 0.2x & + & 0.3y & + & z & = & 0.6 \end{array}$$

Apenas para ilustrar o critério das linhas para testar convergência (desnecessário no caso, já que já vimos que o sistema converge, mas em tese...)
$|a_{12}^*| + |a_{13}^*| = |0.2| + |0.1| = 0.3$ e
$|a_{21}^*| + |a_{23}^*| = |0.2| + |0.2| = 0.4$ e
$|a_{31}^*| + |a_{32}^*| = |0.2| + |0.3| = 0.5$ e
$\Rightarrow\ max_{1 \leq i \leq n}(0.3, 0.4\ 0.5) = 0.5 < 1$

Iterações

$x^{k+1} = -0.2y^k - 0.1z^k + 0.7$
$y^{k+1} = -0.2x^k - 0.2z^k - 1.6$
$z^{k+1} = -0.2x^k - 0.3y^k + 0.6$
Começando com $(0.7,\ -1.6,\ 0.6)$ obtemos uma primeira aproximação que é $(0.96,\ -1.86,\ 0.94)$ e jogando estes novos valores na fórmula iterativa, obtem-se:

$x \rightarrow 0.7,\ 0.96,\ 0.978,\ 0.9994,\ 0.9979$
$y \rightarrow -1.6,\ -1.86,\ -1.98,\ -1.9888,\ -1.9996$
$z \rightarrow 0.6,\ 0.94,\ 0.966,\ 0.9984,\ 0.9968$

O critério de parada pode ser
$max(x^{k+1} - x^k) < \epsilon$. No exemplo acima $0.0108 < 0.011$. Ou seja, neste caso, a solução procurada é $x = 0.9979$, $y = -1.9996$ e $z = 0.9978$. Salta aos olhos neste caso que se a tolerância for um pouco menor, a resposta exata será encontrada que é $(1, -2, 1)$. Basta lançar este resultado no sistema original que a exatidão será confirmada.

Solução Python

```
#jacobi
def jacobi(a,b):
 import numpy
 n=len(a) #numero de linhas de a
          # len(a[0])=colunas de a
 xvelho=numpy.zeros((n),float)
 erro=numpy.zeros((n),float)
 xnovo=numpy.zeros((n),float)
 f=numpy.zeros((n,len(a[0])),float)
 d=numpy.zeros((len(a[0])),float)
 tol=0.1
 for i in range(0,n):
  for j in range(0,n):
   if i==j:
    f[i,j]=0
   else:
    f[i,j]=-a[i,j]/a[i,i]
 for i in range(0,n):
  d[i]=b[i]/a[i,i]
 erromax=tol+1
 while erromax>tol:
  for i in range(0,n):
   xnovo[i]=0
   for k in range(0,n):
    xnovo[i]=xnovo[i]+f[i,k]*xvelho[k]
  for i in range(0,n):
   xnovo[i]=xnovo[i]+d[i]
  for i in range(0,n):
   erro[i]=abs(xnovo[i]-xvelho[i])
  erromax=erro[0]
  for i in range(1,n):
   if erro[i]>erromax:
    erromax=erro[i]
  for i in range(0,n):
   xvelho[i]=xnovo[i]
 return(xnovo)
import numpy
# para usar faça:
# a=numpy.array([[10,7,3],[1,5,4],[3,5,9]],float)
# b=numpy.array([22,33,44],float)
# print(jacobi(a,b))
```

☞ Para você fazer

Resolva usando este método e anexe a listagem do programa (ou copie no verso desta folha):

$$A = \begin{pmatrix} 98 & 28 & 24 & 10 & 34 \\ 1 & 78 & 14 & 30 & 32 \\ 18 & 26 & 60 & 6 & 9 \\ 22 & 7 & 8 & 46 & 5 \\ 35 & 12 & 29 & 13 & 94 \end{pmatrix}$$

e

$$B = (2024, 1941, 1493, 718, 2483)$$

As respostas encontradas, com $\epsilon < 0.0001$ e sugestão inicial igual à coluna B são:

x_1	x_2	x_3	x_4	x_5

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:51.2
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
70522 VICTOR EMANUEL VENTURI
19HEL111 - 40 apos 01/10, 50% _____ / / _____

Sistemas Lineares - Jacobi

Este método é iterativo, ou seja, não exato, já que parte-se de um valor atribuído inicial e mediante buscas locais sucessivas, o mesmo converge para a resposta correta. A diferença entre o resultado sugerido e aquele real, é denominada erro, e pode ser tornada tão pequena quanto se queira. O método entretanto tem um senão, que é um critério de convergência bastante rígido. Se não obedecido o algoritmo não converge para a resposta correta, e portanto o método é inútil.

Considere $A.x = b$ um sistema de quações lineares de ordem n e cujo $det(A) \neq 0$. A maneira extensa de escrever tal sistema é

$$\begin{array}{ccccccc} a_{11}x_1 & + & a_{12}x_2 & +...+ & a_{1n}x_n & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & +...+ & a_{2n}x_n & = & b_2 \\ ... & & & & & & \\ a_{n1}x_1 & + & a_{n2}x_2 & +...+ & a_{nn}x_n & = & b_n \end{array}$$

A matriz A dos coeficientes, pode ser decomposta em $A = L + D + R$, onde

L é a matriz inferior de A (o resto é zero)

D é a matriz contendo apenas a diagonal principal de A, o resto é zero.

R é a matriz superior de A (o resto é zero)

Veja por exemplo, se

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \therefore L = \begin{pmatrix} 0 & 0 & 0 \\ 4 & 0 & 0 \\ 7 & 8 & 0 \end{pmatrix},$$

e

$$D = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 9 \end{pmatrix}, R = \begin{pmatrix} 0 & 2 & 3 \\ 0 & 0 & 6 \\ 0 & 0 & 0 \end{pmatrix}$$

Supondo $det(D) \neq 0$, pode-se escrever: $(L + D + R)x = b$ e daqui $Dx = -(L + R)x + b$. e $x = -D^{-1}(L + R)x + D^{-1}b$. Chamando $B = -D^{-1}(L + R)$ e $g = D^{-1}.b$ obtem-se a equação geral deste método que é

$$x = Bx + g$$

. Note que é uma equação recursiva (ou iterativa) já que o x aparece nos 2 lados da mesma. A idéia aqui é atribuir um valor a x e com ele calcular um novo x, que é jogado no lugar do x velho e gera um novo x e assim sucessivamente até haver a convergência esperada (de antemão). Como supusemos $det(D) \neq 0$ pode-se dividir cada equação do sistema pelo coeficiente da diagonal principal, resultando

$$A^* = L^* + I + R^*$$

Agora o processo iterativo pode ser definido como

$$x^{k+1} = -(L^* + R^*)x^k + b^*$$

onde os elementos de L^*, R^* e b^* são:

$l_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i > j$ e zero senão

$d_{ij} = a_{ij}$ se $i = j$ e zero senão

$r_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i < j$ e zero senão

$b_ij^* = \frac{b_i}{a_{ii}}$

Critérios de convergência

Depois de dividida a matriz pelo elemento da diagonal principal, deve-se ter o **critério das linhas** que diz

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}^*| < 1$$

OU então o **critério das colunas** que diz

$$max_{1 \leq j \leq n} \sum_{i=1, i \neq j}^{n} |a_{ij}^*| < 1$$

Note que qualquer um dos dois critérios sendo satisfeitos, isso garante que o sistema tem convergência.

Definindo uma matriz como estritamente diagonalmente dominante se

$$\sum_{j=1, j \neq i}^{n} |a_{ij}| < |a_{ii}|$$

e se A é estritamente diagonalmente dominante, então A satisfaz o critério das linhas, e o sistema converge. Então, um bom critério de convergência pode ser

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}^|* < 1$$

Exemplo

Seja resolver o sistema abaixo, usando o método de Jacobi

$$\begin{array}{ccccccc} 10x & + & 2y & + & z & = & 7 \\ x & + & 5y & + & z & = & -8 \\ 2x & + & 3y & + & 10z & = & 6 \end{array}$$

com $x^0 = (0.7, -1.6\ e\ 0.6)^t$ e $\epsilon < 0.11$.

Convergência ?

$|a_{12}| + |a_{13}| < |a_{11}|$? ou $|2| + |1| < |10|$? R: sim
$|a_{21}| + |a_{23}| < |a_{22}|$? ou $|1| + |1| < |5|$? R: sim
$|a_{31}| + |a_{32}| < |a_{33}|$? ou $|2| + |3| < |10|$? R: sim
logo, o sistema convergirá!

Solução

Dividindo cada equação pelo elemento da diagonal principal, fica

$$\begin{array}{ccccccc} x & + & 0.2y & + & 0.1z & = & 0.7 \\ 0.2x & + & y & + & 0.2z & = & -1.6 \\ 0.2x & + & 0.3y & + & z & = & 0.6 \end{array}$$

Apenas para ilustrar o critério das linhas para testar convergência (desnecessário no caso, já que já vimos que o sistema converge, mas em tese...)
$|a_{12}^*| + |a_{13}^*| = |0.2| + |0.1| = 0.3$ e
$|a_{21}^*| + |a_{23}^*| = |0.2| + |0.2| = 0.4$ e
$|a_{31}^*| + |a_{32}^*| = |0.2| + |0.3| = 0.5$ e
$\Rightarrow\ max_{1 \leq i \leq n}(0.3, 0.4\ 0.5) = 0.5 < 1$

Iterações

$x^{k+1} = -0.2y^k - 0.1z^k + 0.7$
$y^{k+1} = -0.2x^k - 0.2z^k - 1.6$
$z^{k+1} = -0.2x^k - 0.3y^k + 0.6$
Começando com $(0.7,\ -1.6,\ 0.6)$ obtemos uma primeira aproximação que é $(0.96,\ -1.86,\ 0.94)$ e jogando estes novos valores na fórmula iterativa, obtem-se:

$x \rightarrow 0.7,\ 0.96,\ 0.978,\ 0.9994,\ 0.9979$
$y \rightarrow -1.6,\ -1.86,\ -1.98,\ -1.9888,\ -1.9996$
$z \rightarrow 0.6,\ 0.94,\ 0.966,\ 0.9984,\ 0.9968$

O critério de parada pode ser
$max(x^{k+1} - x^k) < \epsilon$. No exemplo acima $0.0108 < 0.011$. Ou seja, neste caso, a solução procurada é $x = 0.9979$, $y = -1.9996$ e $z = 0.9978$. Salta aos olhos neste caso que se a tolerância for um pouco menor, a resposta exata será encontrada que é $(1, -2, 1)$. Basta lançar este resultado no sistema original que a exatidão será confirmada.

Solução Python

```
#jacobi
def jacobi(a,b):
 import numpy
 n=len(a) #numero de linhas de a
          # len(a[0])=colunas de a
 xvelho=numpy.zeros((n),float)
 erro=numpy.zeros((n),float)
 xnovo=numpy.zeros((n),float)
 f=numpy.zeros((n,len(a[0])),float)
 d=numpy.zeros((len(a[0])),float)
 tol=0.1
 for i in range(0,n):
  for j in range(0,n):
   if i==j:
    f[i,j]=0
   else:
    f[i,j]=-a[i,j]/a[i,i]
 for i in range(0,n):
  d[i]=b[i]/a[i,i]
 erromax=tol+1
 while erromax>tol:
  for i in range(0,n):
   xnovo[i]=0
   for k in range(0,n):
    xnovo[i]=xnovo[i]+f[i,k]*xvelho[k]
  for i in range(0,n):
   xnovo[i]=xnovo[i]+d[i]
  for i in range(0,n):
   erro[i]=abs(xnovo[i]-xvelho[i])
  erromax=erro[0]
  for i in range(1,n):
   if erro[i]>erromax:
    erromax=erro[i]
  for i in range(0,n):
   xvelho[i]=xnovo[i]
 return(xnovo)
import numpy
# para usar faça:
# a=numpy.array([[10,7,3],[1,5,4],[3,5,9]],float)
# b=numpy.array([22,33,44],float)
# print(jacobi(a,b))
```

☞ Para você fazer

Resolva usando este método e anexe a listagem do programa (ou copie no verso desta folha):

$$A = \begin{pmatrix} 89 & 29 & 14 & 15 & 30 \\ 21 & 113 & 24 & 33 & 27 \\ 17 & 35 & 82 & 13 & 11 \\ 1 & 7 & 20 & 39 & 4 \\ 12 & 3 & 25 & 10 & 54 \end{pmatrix}$$

e

$$B = (1197, 1225, 935, 319, 819)$$

As respostas encontradas, com $\epsilon < 0.0001$ e sugestão inicial igual à coluna B são:

x_1	x_2	x_3	x_4	x_5

111-70522 - 01/10

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:51.2
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
70539 VINICIUS GUILHERME F. SILVA
19HEL111 - 41 apos 01/10, 50% _____ / / _____

Sistemas Lineares - Jacobi

Este método é iterativo, ou seja, não exato, já que parte-se de um valor atribuído inicial e mediante buscas locais sucessivas, o mesmo converge para a resposta correta. A diferença entre o resultado sugerido e aquele real, é denominada erro, e pode ser tornada tão pequena quanto se queira. O método entretanto tem um senão, que é um critério de convergência bastante rígido. Se não obedecido o algoritmo não converge para a resposta correta, e portanto o método é inútil.

Considere $A.x = b$ um sistema de quações lineares de ordem n e cujo $det(A) \neq 0$. A maneira extensa de escrever tal sistema é

$$\begin{array}{lcllcl} a_{11}x_1 & + & a_{12}x_2 & +...+ & a_{1n}x_n & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & +...+ & a_{2n}x_n & = & b_2 \\ ... & & & & & & \\ a_{n1}x_1 & + & a_{n2}x_2 & +...+ & a_{nn}x_n & = & b_n \end{array}$$

A matriz A dos coeficientes, pode ser decomposta em $A = L + D + R$, onde

L é a matriz inferior de A (o resto é zero)

D é a matriz contendo apenas a diagonal principal de A, o resto é zero.

R é a matriz superior de A (o resto é zero)

Veja por exemplo, se

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \therefore L = \begin{pmatrix} 0 & 0 & 0 \\ 4 & 0 & 0 \\ 7 & 8 & 0 \end{pmatrix},$$

e

$$D = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 9 \end{pmatrix}, R = \begin{pmatrix} 0 & 2 & 3 \\ 0 & 0 & 6 \\ 0 & 0 & 0 \end{pmatrix}$$

Supondo $det(D) \neq 0$, pode-se escrever: $(L + D + R)x = b$ e daqui $Dx = -(L + R)x + b$. e $x = -D^{-1}(L + R)x + D^{-1}b$. Chamando $B = -D^{-1}(L + R)$ e $g = D^{-1}.b$ obtem-se a equação geral deste método que é

$$x = Bx + g$$

. Note que é uma equação recursiva (ou iterativa) já que o x aparece nos 2 lados da mesma. A idéia aqui é atribuir um valor a x e com ele calcular um novo x, que é jogado no lugar do x velho e gera um novo x e assim sucessivamente até haver a convergência esperada (de antemão). Como supusemos $det(D) \neq 0$ pode-se dividir cada equação do sistema pelo coeficiente da diagonal principal, resultando

$$A^* = L^* + I + R^*$$

Agora o processo iterativo pode ser definido como

$$x^{k+1} = -(L^* + R^*)x^k + b^*$$

onde os elementos de L^*, R^* e b^* são:

$l_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i > j$ e zero senão

$d_{ij} = a_{ij}$ se $i = j$ e zero senão

$r_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i < j$ e zero senão

$b_ij^* = \frac{b_i}{a_{ii}}$

Critérios de convergência

Depois de dividida a matriz pelo elemento da diagonal principal, deve-se ter o **critério das linhas** que diz

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}^*| < 1$$

OU então o **critério das colunas** que diz

$$max_{1 \leq j \leq n} \sum_{i=1, i \neq j}^{n} |a_{ij}^*| < 1$$

Note que qualquer um dos dois critérios sendo satisfeitos, isso garante que o sistema tem convergência.

Definindo uma matriz como estritamente diagonalmente dominante se

$$\sum_{j=1, j \neq i}^{n} |a_{ij}| < |a_{ii}|$$

e se A é estritamente diagonalmente dominante, então A satisfaz o critério das linhas, e o sistema converge. Então, um bom critério de convergência pode ser

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}^|* < 1$$

Exemplo

Seja resolver o sistema abaixo, usando o método de Jacobi

$$\begin{array}{rcrcrcr} 10x & + & 2y & + & z & = & 7 \\ x & + & 5y & + & z & = & -8 \\ 2x & + & 3y & + & 10z & = & 6 \end{array}$$

com $x^0 = (0.7, -1.6\ e\ 0.6)^t$ e $\epsilon < 0.11$.

Convergência ?

$|a_{12}| + |a_{13}| < |a_{11}|$? ou $|2| + |1| < |10|$? R: sim
$|a_{21}| + |a_{23}| < |a_{22}|$? ou $|1| + |1| < |5|$? R: sim
$|a_{31}| + |a_{32}| < |a_{33}|$? ou $|2| + |3| < |10|$? R: sim
logo, o sistema convergirá!

Solução

Dividindo cada equação pelo elemento da diagonal principal, fica

$$\begin{array}{rcrcrcr} x & + & 0.2y & + & 0.1z & = & 0.7 \\ 0.2x & + & y & + & 0.2z & = & -1.6 \\ 0.2x & + & 0.3y & + & z & = & 0.6 \end{array}$$

Apenas para ilustrar o critério das linhas para testar convergência (desnecessário no caso, já que já vimos que o sistema converge, mas em tese...)
$|a_{12}^*| + |a_{13}^*| = |0.2| + |0.1| = 0.3$ e
$|a_{21}^*| + |a_{23}^*| = |0.2| + |0.2| = 0.4$ e
$|a_{31}^*| + |a_{32}^*| = |0.2| + |0.3| = 0.5$ e
$\Rightarrow\ max_{1 \leq i \leq n}(0.3, 0.4\ 0.5) = 0.5 < 1$

Iterações

$x^{k+1} = -0.2y^k - 0.1z^k + 0.7$
$y^{k+1} = -0.2x^k - 0.2z^k - 1.6$
$z^{k+1} = -0.2x^k - 0.3y^k + 0.6$
Começando com $(0.7,\ -1.6,\ 0.6)$ obtemos uma primeira aproximação que é $(0.96,\ -1.86,\ 0.94)$ e jogando estes novos valores na fórmula iterativa, obtem-se:

$x \rightarrow 0.7,\ 0.96,\ 0.978,\ 0.9994,\ 0.9979$
$y \rightarrow -1.6,\ -1.86,\ -1.98,\ -1.9888,\ -1.9996$
$z \rightarrow 0.6,\ 0.94,\ 0.966,\ 0.9984,\ 0.9968$

O critério de parada pode ser
$max(x^{k+1} - x^k) < \epsilon$. No exemplo acima $0.0108 < 0.011$. Ou seja, neste caso, a solução procurada é $x = 0.9979$, $y = -1.9996$ e $z = 0.9978$. Salta aos olhos neste caso que se a tolerância for um pouco menor, a resposta exata será encontrada que é $(1, -2, 1)$. Basta lançar este resultado no sistema original que a exatidão será confirmada.

Solução Python

```
#jacobi
def jacobi(a,b):
 import numpy
 n=len(a) #numero de linhas de a
          # len(a[0])=colunas de a
 xvelho=numpy.zeros((n),float)
 erro=numpy.zeros((n),float)
 xnovo=numpy.zeros((n),float)
 f=numpy.zeros((n,len(a[0])),float)
 d=numpy.zeros((len(a[0])),float)
 tol=0.1
 for i in range(0,n):
  for j in range(0,n):
   if i==j:
    f[i,j]=0
   else:
    f[i,j]=-a[i,j]/a[i,i]
 for i in range(0,n):
  d[i]=b[i]/a[i,i]
 erromax=tol+1
 while erromax>tol:
  for i in range(0,n):
   xnovo[i]=0
   for k in range(0,n):
    xnovo[i]=xnovo[i]+f[i,k]*xvelho[k]
  for i in range(0,n):
   xnovo[i]=xnovo[i]+d[i]
  for i in range(0,n):
   erro[i]=abs(xnovo[i]-xvelho[i])
  erromax=erro[0]
  for i in range(1,n):
   if erro[i]>erromax:
    erromax=erro[i]
  for i in range(0,n):
   xvelho[i]=xnovo[i]
 return(xnovo)
import numpy
# para usar faça:
# a=numpy.array([[10,7,3],[1,5,4],[3,5,9]],float)
# b=numpy.array([22,33,44],float)
# print(jacobi(a,b))
```

☞ Para você fazer

Resolva usando este método e anexe a listagem do programa (ou copie no verso desta folha):

$$A = \begin{pmatrix} 101 & 33 & 14 & 23 & 27 \\ 24 & 84 & 20 & 5 & 34 \\ 2 & 25 & 81 & 35 & 15 \\ 17 & 12 & 21 & 77 & 26 \\ 29 & 3 & 18 & 4 & 58 \end{pmatrix}$$

e

$$B = (2010, 1966, 1359, 1897, 1033)$$

As respostas encontradas, com $\epsilon < 0.0001$ e sugestão inicial igual à coluna B são:

x_1	x_2	x_3	x_4	x_5

111-70539 - 01/10

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:51.2
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
71244 VINICIUS LUIS ALFIERI 19HEL111 - 42
apos 01/10, 50% _____ / _____ / _____

Sistemas Lineares - Jacobi

Este método é iterativo, ou seja, não exato, já que parte-se de um valor atribuído inicial e mediante buscas locais sucessivas, o mesmo converge para a resposta correta. A diferença entre o resultado sugerido e aquele real, é denominada erro, e pode ser tornada tão pequena quanto se queira. O método entretanto tem um senão, que é um critério de convergência bastante rígido. Se não obedecido o algoritmo não converge para a resposta correta, e portanto o método é inútil.

Considere $A.x = b$ um sistema de quações lineares de ordem n e cujo $det(A) \neq 0$. A maneira extensa de escrever tal sistema é

$$\begin{array}{ccccccccc} a_{11}x_1 & + & a_{12}x_2 & +...+ & a_{1n}x_n & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & +...+ & a_{2n}x_n & = & b_2 \\ ... & & & & & & \\ a_{n1}x_1 & + & a_{n2}x_2 & +...+ & a_{nn}x_n & = & b_n \end{array}$$

A matriz A dos coeficientes, pode ser decomposta em $A = L + D + R$, onde

L é a matriz inferior de A (o resto é zero)

D é a matriz contendo apenas a diagonal principal de A, o resto é zero.

R é a matriz superior de A (o resto é zero)

Veja por exemplo, se

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \therefore L = \begin{pmatrix} 0 & 0 & 0 \\ 4 & 0 & 0 \\ 7 & 8 & 0 \end{pmatrix},$$

e

$$D = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 9 \end{pmatrix}, R = \begin{pmatrix} 0 & 2 & 3 \\ 0 & 0 & 6 \\ 0 & 0 & 0 \end{pmatrix}$$

Supondo $det(D) \neq 0$, pode-se escrever: $(L + D + R)x = b$ e daqui $Dx = -(L + R)x + b.$ e $x = -D^{-1}(L + R)x + D^{-1}b$. Chamando $B = -D^{-1}(L + R)$ e $g = D^{-1}.b$ obtem-se a equação geral deste método que é

$$x = Bx + g$$

. Note que é uma equação recursiva (ou iterativa) já que o x aparece nos 2 lados da mesma. A idéia aqui é atribuir um valor a x e com ele calcular um novo x, que é jogado no lugar do x velho e gera um novo x e assim sucessivamente até haver a convergência esperada (de antemão). Como supusemos $det(D) \neq 0$ pode-se dividir cada equação do sistema pelo coeficiente da diagonal principal, resultando

$$A^* = L^* + I + R^*$$

Agora o processo iterativo pode ser definido como

$$x^{k+1} = -(L^* + R^*)x^k + b^*$$

onde os elementos de L^*, R^* e b^* são:

$l_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i > j$ e zero senão

$d_{ij} = a_{ij}$ se $i = j$ e zero senão

$r_ij^* = \frac{a_{ij}}{a_{ii}}$ se $i < j$ e zero senão

$b_ij^* = \frac{b_i}{a_{ii}}$

Critérios de convergência

Depois de dividida a matriz pelo elemento da diagonal principal, deve-se ter o **critério das linhas** que diz

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}^*| < 1$$

OU então o **critério das colunas** que diz

$$max_{1 \leq j \leq n} \sum_{i=1, i \neq j}^{n} |a_{ij}^*| < 1$$

Note que qualquer um dos dois critérios sendo satisfeitos, isso garante que o sistema tem convergência.

Definindo uma matriz como estritamente diagonalmente dominante se

$$\sum_{j=1, j \neq i}^{n} |a_{ij}| < |a_{ii}|$$

e se A é estritamente diagonalmente dominante, então A satisfaz o critério das linhas, e o sistema converge. Então, um bom critério de convergência pode ser

$$max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} |a_{ij}|* < 1$$

Exemplo

Seja resolver o sistema abaixo, usando o método de Jacobi

$$\begin{array}{ccccccc} 10x & + & 2y & + & z & = & 7 \\ x & + & 5y & + & z & = & -8 \\ 2x & + & 3y & + & 10z & = & 6 \end{array}$$

com $x^0 = (0.7, -1.6\ e\ 0.6)^t$ e $\epsilon < 0.11$.

Convergência ?

$|a_{12}| + |a_{13}| < |a_{11}|$? ou $|2| + |1| < |10|$? R: sim
$|a_{21}| + |a_{23}| < |a_{22}|$? ou $|1| + |1| < |5|$? R: sim
$|a_{31}| + |a_{32}| < |a_{33}|$? ou $|2| + |3| < |10|$? R: sim
logo, o sistema convergirá!

Solução

Dividindo cada equação pelo elemento da diagonal principal, fica

$$\begin{array}{ccccccc} x & + & 0.2y & + & 0.1z & = & 0.7 \\ 0.2x & + & y & + & 0.2z & = & -1.6 \\ 0.2x & + & 0.3y & + & z & = & 0.6 \end{array}$$

Apenas para ilustrar o critério das linhas para testar convergência (desnecessário no caso, já que já vimos que o sistema converge, mas em tese...)
$|a_{12}^*| + |a_{13}^*| = |0.2| + |0.1| = 0.3$ e
$|a_{21}^*| + |a_{23}^*| = |0.2| + |0.2| = 0.4$ e
$|a_{31}^*| + |a_{32}^*| = |0.2| + |0.3| = 0.5$ e
$\Rightarrow\ max_{1 \leq i \leq n}(0.3, 0.4\ 0.5) = 0.5 < 1$

Iterações

$x^{k+1} = -0.2y^k - 0.1z^k + 0.7$
$y^{k+1} = -0.2x^k - 0.2z^k - 1.6$
$z^{k+1} = -0.2x^k - 0.3y^k + 0.6$
Começando com $(0.7,\ -1.6,\ 0.6)$ obtemos uma primeira aproximação que é $(0.96,\ -1.86,\ 0.94)$ e jogando estes novos valores na fórmula iterativa, obtem-se:

$x \rightarrow 0.7,\ 0.96,\ 0.978,\ 0.9994,\ 0.9979$
$y \rightarrow -1.6,\ -1.86,\ -1.98,\ -1.9888,\ -1.9996$
$z \rightarrow 0.6,\ 0.94,\ 0.966,\ 0.9984,\ 0.9968$

O critério de parada pode ser
$max(x^{k+1} - x^k) < \epsilon$. No exemplo acima $0.0108 < 0.011$. Ou seja, neste caso, a solução procurada é $x = 0.9979$, $y = -1.9996$ e $z = 0.9978$. Salta aos olhos neste caso que se a tolerância for um pouco menor, a resposta exata será encontrada que é $(1, -2, 1)$. Basta lançar este resultado no sistema original que a exatidão será confirmada.

Solução Python

```
#jacobi
def jacobi(a,b):
 import numpy
 n=len(a) #numero de linhas de a
          # len(a[0])=colunas de a
 xvelho=numpy.zeros((n),float)
 erro=numpy.zeros((n),float)
 xnovo=numpy.zeros((n),float)
 f=numpy.zeros((n,len(a[0])),float)
 d=numpy.zeros((len(a[0])),float)
 tol=0.1
 for i in range(0,n):
  for j in range(0,n):
   if i==j:
    f[i,j]=0
   else:
    f[i,j]=-a[i,j]/a[i,i]
 for i in range(0,n):
  d[i]=b[i]/a[i,i]
 erromax=tol+1
 while erromax>tol:
  for i in range(0,n):
   xnovo[i]=0
   for k in range(0,n):
    xnovo[i]=xnovo[i]+f[i,k]*xvelho[k]
  for i in range(0,n):
   xnovo[i]=xnovo[i]+d[i]
  for i in range(0,n):
   erro[i]=abs(xnovo[i]-xvelho[i])
  erromax=erro[0]
  for i in range(1,n):
   if erro[i]>erromax:
    erromax=erro[i]
  for i in range(0,n):
   xvelho[i]=xnovo[i]
 return(xnovo)
import numpy
# para usar faça:
# a=numpy.array([[10,7,3],[1,5,4],[3,5,9]],float)
# b=numpy.array([22,33,44],float)
# print(jacobi(a,b))
```

☞ Para você fazer

Resolva usando este método e anexe a listagem do programa (ou copie no verso desta folha):

$$A = \begin{pmatrix} 74 & 9 & 31 & 25 & 8 \\ 33 & 114 & 24 & 22 & 30 \\ 15 & 23 & 59 & 18 & 1 \\ 21 & 4 & 32 & 65 & 6 \\ 13 & 12 & 34 & 29 & 91 \end{pmatrix}$$

e

$$B = (981, 2394, 872, 693, 2015)$$

As respostas encontradas, com $\epsilon < 0.0001$ e sugestão inicial igual à coluna B são:

x_1	x_2	x_3	x_4	x_5

111-71244 - 01/10