U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:26.3
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
70865 ALEXANDRE S A DELDUQUE
MACEDO 19HEA111 - 1 apos 30/09, 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):

$$A = \begin{pmatrix} 69 & 6 & 20 & 30 & 11 \\ 9 & 90 & 10 & 34 & 35 \\ 23 & 33 & 120 & 27 & 31 \\ 14 & 16 & 19 & 60 & 8 \\ 2 & 21 & 17 & 22 & 66 \end{pmatrix}$$

e

$$B = (1665, 2217, 2498, 1614, 1798)$$

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

111-70865 - 30/09

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:26.3
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
70872 ARIANE APARECIDA CRUZ SILVA
19HEA111 - 2 apos 30/09, 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):

$$A = \begin{pmatrix} 109 & 34 & 30 & 27 & 15 \\ 29 & 76 & 23 & 19 & 3 \\ 25 & 11 & 67 & 28 & 2 \\ 22 & 32 & 35 & 104 & 12 \\ 5 & 10 & 14 & 16 & 46 \end{pmatrix}$$

e

$$B = (2742, 2093, 1707, 2017, 720)$$

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

111-70872 - 30/09

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:26.3
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
71170 ERICK BUENO C DOS SANTOS
19HEA111 - 3 apos 30/09, 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):

$$A = \begin{pmatrix} 59 & 18 & 12 & 14 & 13 \\ 3 & 22 & 6 & 7 & 5 \\ 27 & 9 & 81 & 28 & 15 \\ 19 & 25 & 20 & 91 & 26 \\ 24 & 23 & 8 & 33 & 94 \end{pmatrix}$$

e

$$B = (1498, 605, 2124, 2422, 1757)$$

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

111-71170 - 30/09

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:26.3
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
70889 FABIANA ANDREA SILVA MORALES
19HEA111 - 4 apos 30/09, 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):

$$A = \begin{pmatrix} 88 & 31 & 29 & 19 & 7 \\ 20 & 84 & 28 & 5 & 27 \\ 26 & 25 & 89 & 24 & 13 \\ 3 & 22 & 10 & 50 & 11 \\ 1 & 6 & 15 & 2 & 25 \end{pmatrix}$$

e

$$B = (2125, 1634, 1419, 1205, 314)$$

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

111-70889 - 30/09

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:26.3
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
70896 FELIPE BORTOLLETTO CIVITATE
19HEA111 - 5 apos 30/09, 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):

$$A = \begin{pmatrix} 100 & 25 & 18 & 28 & 27 \\ 5 & 55 & 26 & 17 & 6 \\ 8 & 10 & 86 & 33 & 34 \\ 16 & 20 & 31 & 78 & 3 \\ 29 & 23 & 32 & 30 & 122 \end{pmatrix}$$

e

$$B = (1989, 1151, 1200, 1779, 1897)$$

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

111-70896 - 30/09

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:26.3
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
71187 GABRIEL CHIMALESKI COSTA
19HEA111 - 6 apos 30/09, 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):

$$A = \begin{pmatrix} 103 & 22 & 28 & 21 & 27 \\ 30 & 94 & 23 & 18 & 20 \\ 24 & 14 & 61 & 3 & 13 \\ 35 & 29 & 9 & 83 & 6 \\ 4 & 12 & 19 & 16 & 56 \end{pmatrix}$$

e

$$B = (1593, 2207, 1045, 1187, 803)$$

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

111-71187 - 30/09

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:26.3
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
70908 GABRIELA MONTEIRO XAVIER
19HEA111 - 7 apos 30/09, 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

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

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):

$$A = \begin{pmatrix} 98 & 35 & 24 & 27 & 9 \\ 12 & 48 & 19 & 5 & 11 \\ 23 & 6 & 76 & 10 & 32 \\ 33 & 7 & 29 & 75 & 1 \\ 20 & 17 & 28 & 16 & 86 \end{pmatrix}$$

e

$$B = (2228, 692, 1246, 1981, 1349)$$

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

111-70908 - 30/09

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:26.3
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
70915 GRENDA IZABELI MENEZES SILVA
19HEA111 - 8 apos 30/09, 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):

$$A = \begin{pmatrix} 61 & 19 & 13 & 8 & 16 \\ 6 & 87 & 15 & 26 & 32 \\ 29 & 30 & 111 & 35 & 10 \\ 5 & 7 & 11 & 31 & 2 \\ 22 & 27 & 3 & 9 & 65 \end{pmatrix}$$

e

$$B = (1266, 1189, 1882, 331, 1388)$$

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

111-70915 - 30/09

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:26.3
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
70922 HALLYPHER GUERREIRO ZOLLNER
19HEA111 - 9 apos 30/09, 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):

$$A = \begin{pmatrix} 118 & 29 & 34 & 22 & 26 \\ 10 & 111 & 35 & 31 & 33 \\ 30 & 6 & 85 & 27 & 21 \\ 7 & 28 & 24 & 83 & 19 \\ 5 & 9 & 15 & 13 & 45 \end{pmatrix}$$

e

$$B = (1928, 1330, 1196, 1175, 760)$$

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

111-70922 - 30/09

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:26.3
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
70939 HIAN DA SILVA PINTO 19HEA111 - 10
apos 30/09, 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):

$$A = \begin{pmatrix} 59 & 21 & 8 & 1 & 27 \\ 22 & 54 & 6 & 11 & 13 \\ 26 & 3 & 68 & 15 & 18 \\ 29 & 12 & 30 & 104 & 28 \\ 19 & 25 & 24 & 16 & 89 \end{pmatrix}$$

e

$$B = (1182, 1192, 1049, 2298, 2115)$$

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

111-70939 - 30/09

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:26.3
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
70946 IGOR HERNAN ANDRADE SANTOS
19HEA111 - 11 apos 30/09, 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):

$$A = \begin{pmatrix} 75 & 15 & 18 & 11 & 27 \\ 14 & 42 & 7 & 6 & 10 \\ 29 & 32 & 98 & 26 & 5 \\ 13 & 9 & 33 & 78 & 16 \\ 24 & 1 & 31 & 12 & 71 \end{pmatrix}$$

e

$$B = (1323, 1028, 1938, 1218, 1348)$$

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

111-70946 - 30/09

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:26.3
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
70953 ISADORA FONSECA CARRARI
19HEA111 - 12 apos 30/09, 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):

$$A = \begin{pmatrix} 80 & 20 & 23 & 27 & 8 \\ 19 & 98 & 21 & 30 & 22 \\ 10 & 33 & 84 & 13 & 26 \\ 11 & 34 & 25 & 105 & 31 \\ 14 & 16 & 2 & 4 & 39 \end{pmatrix}$$

e

$$B = (1614, 2187, 2378, 2297, 943)$$

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

111-70953 - 30/09

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:26.3
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
70960 JESSICA FURTUOSO HARMATIUK
19HEA111 - 13 apos 30/09, 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):

$$A = \begin{pmatrix} 51 & 16 & 15 & 5 & 12 \\ 25 & 92 & 33 & 21 & 4 \\ 30 & 27 & 80 & 7 & 13 \\ 34 & 28 & 18 & 104 & 17 \\ 26 & 6 & 1 & 9 & 45 \end{pmatrix}$$

e

$$B = (1048, 1328, 1983, 1800, 933)$$

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

111-70960 - 30/09

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:26.3
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
70977 JOAO GUILHERME LIMA MARIANO
19HEA111 - 14 apos 30/09, 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):

$$A = \begin{pmatrix} 66 & 13 & 17 & 26 & 8 \\ 15 & 62 & 23 & 20 & 1 \\ 28 & 10 & 76 & 11 & 22 \\ 18 & 32 & 31 & 87 & 2 \\ 24 & 33 & 14 & 29 & 101 \end{pmatrix}$$

e

$$B = (1252, 1574, 1163, 1888, 1598)$$

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

111-70977 - 30/09

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:26.3
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
70984 KAREN LETICIA SARTORI 19HEA111 - 15 apos 30/09, 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):

$$A = \begin{pmatrix} 99 & 35 & 26 & 28 & 5 \\ 18 & 63 & 16 & 12 & 15 \\ 4 & 30 & 63 & 25 & 1 \\ 22 & 3 & 13 & 51 & 10 \\ 20 & 8 & 11 & 31 & 74 \end{pmatrix}$$

e

$$B = (2447, 1129, 1212, 1097, 1057)$$

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

111-70984 - 30/09

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:26.3
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
70991 KARLA KOSOSKI BARTH MACEDO
19HEA111 - 16 apos 30/09, 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

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

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

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

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):

$$A = \begin{pmatrix} 87 & 33 & 2 & 20 & 24 \\ 5 & 49 & 22 & 7 & 14 \\ 34 & 30 & 113 & 25 & 19 \\ 23 & 32 & 13 & 103 & 29 \\ 16 & 31 & 6 & 3 & 59 \end{pmatrix}$$

e

$$B = (1246, 979, 2741, 2234, 1140)$$

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

111-70991 - 30/09

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:26.3
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
71006 LETICIA ERTHAL CORDEIRO
19HEA111 - 17 apos 30/09, 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):

$$A = \begin{pmatrix} 82 & 14 & 16 & 17 & 34 \\ 19 & 75 & 10 & 35 & 9 \\ 2 & 28 & 76 & 25 & 18 \\ 12 & 5 & 23 & 56 & 11 \\ 29 & 8 & 27 & 32 & 98 \end{pmatrix}$$

e

$$B = (1055, 1433, 1607, 673, 1459)$$

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

111-71006 - 30/09

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:26.3
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
71013 LIAH CHIURATTO BUBA 19HEA111 - 18 apos 30/09, 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):

$$A = \begin{pmatrix} 102 & 27 & 29 & 33 & 7 \\ 15 & 80 & 30 & 8 & 22 \\ 5 & 21 & 68 & 17 & 23 \\ 32 & 35 & 6 & 106 & 26 \\ 20 & 19 & 13 & 12 & 68 \end{pmatrix}$$

e

$$B = (1333, 1711, 1076, 2309, 1301)$$

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

111-71013 - 30/09

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:26.3
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
71020 LISA ISABELLA FISCHER 19HEA111 - 19 apos 30/09, 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):

$$A = \begin{pmatrix} 72 & 19 & 34 & 13 & 4 \\ 1 & 75 & 25 & 23 & 24 \\ 14 & 22 & 104 & 35 & 31 \\ 28 & 10 & 33 & 94 & 20 \\ 9 & 27 & 17 & 7 & 64 \end{pmatrix}$$

e

$$B = (1362, 1737, 2529, 2456, 1481)$$

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

111-71020 - 30/09

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:26.3
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
71037 LUIS GABRIEL COUTINHO ROCHA F
19HEA111 - 20 apos 30/09, 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):

$$A = \begin{pmatrix} 87 & 11 & 34 & 35 & 6 \\ 27 & 76 & 28 & 2 & 13 \\ 15 & 1 & 39 & 3 & 18 \\ 5 & 21 & 25 & 77 & 19 \\ 26 & 32 & 4 & 8 & 72 \end{pmatrix}$$

e

$$B = (2060, 1691, 967, 1249, 1946)$$

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

111-71037 - 30/09

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:26.3
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
71044 MARIA GABRIELA FERNANDES LOPES 19HEA111 - 21 apos 30/09, 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):

$$A = \begin{pmatrix} 56 & 16 & 27 & 1 & 3 \\ 2 & 57 & 4 & 12 & 32 \\ 22 & 18 & 77 & 26 & 9 \\ 20 & 34 & 31 & 102 & 15 \\ 24 & 14 & 19 & 17 & 80 \end{pmatrix}$$

e

$$B = (1429, 1053, 1803, 2731, 1424)$$

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

111-71044 - 30/09

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:26.3
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
71051 MARIA LUIZA MOCELIM DE MATTOS
19HEA111 - 22 apos 30/09, 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):

$$A = \begin{pmatrix} 71 & 12 & 14 & 33 & 8 \\ 13 & 76 & 24 & 11 & 26 \\ 4 & 15 & 85 & 30 & 32 \\ 10 & 28 & 9 & 76 & 23 \\ 20 & 34 & 3 & 18 & 76 \end{pmatrix}$$

e

$$B = (1474, 1252, 1503, 811, 969)$$

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

111-71051 - 30/09

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:26.3
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
71068 MATEUS DOS SANTOS NEVES
19HEA111 - 23 apos 30/09, 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):

$$A = \begin{pmatrix} 36 & 12 & 6 & 14 & 1 \\ 18 & 71 & 7 & 24 & 19 \\ 4 & 33 & 78 & 29 & 11 \\ 35 & 20 & 17 & 83 & 2 \\ 34 & 22 & 9 & 31 & 100 \end{pmatrix}$$

e

$$B = (497, 1147, 1422, 1486, 1877)$$

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

111-71068 - 30/09

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:26.3
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
71075 PAOLA GEBAUER MATEOS 19HEA111 - 24 apos 30/09, 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):

$$A = \begin{pmatrix} 111 & 28 & 32 & 26 & 22 \\ 7 & 59 & 2 & 33 & 14 \\ 25 & 34 & 98 & 27 & 11 \\ 3 & 29 & 24 & 75 & 15 \\ 9 & 6 & 20 & 16 & 53 \end{pmatrix}$$

e

$$B = (1875, 573, 1586, 884, 578)$$

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

111-71075 - 30/09

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:26.3
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
71101 RAUL BECKERT VESSARO 19HEA111 - 25 apos 30/09, 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

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

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

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

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):

$$A = \begin{pmatrix} 58 & 13 & 10 & 15 & 12 \\ 11 & 73 & 16 & 21 & 22 \\ 1 & 34 & 100 & 29 & 31 \\ 30 & 32 & 26 & 100 & 8 \\ 27 & 7 & 19 & 25 & 80 \end{pmatrix}$$

e

$$B = (1403, 1895, 1975, 2872, 1387)$$

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

111-71101 - 30/09

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:26.3
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
71118 VICTORIA FERRO MAESTRELLI
19HEA111 - 26 apos 30/09, 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):

$$A = \begin{pmatrix} 106 & 32 & 24 & 18 & 26 \\ 21 & 76 & 9 & 27 & 14 \\ 4 & 16 & 48 & 19 & 6 \\ 17 & 20 & 1 & 43 & 3 \\ 23 & 12 & 31 & 8 & 78 \end{pmatrix}$$

e

$$B = (1332, 1365, 766, 749, 1145)$$

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

111-71118 - 30/09

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:26.3
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
71125 VITOR KURTEN FEITOSA 19HEA111 - 27 apos 30/09, 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):

$$A = \begin{pmatrix} 68 & 13 & 26 & 19 & 8 \\ 22 & 47 & 9 & 4 & 11 \\ 24 & 34 & 77 & 6 & 10 \\ 15 & 31 & 30 & 98 & 20 \\ 32 & 3 & 28 & 1 & 67 \end{pmatrix}$$

e

$$B = (1708, 1122, 2035, 2525, 1788)$$

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

111-71125 - 30/09