U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:35.6
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
70546 ALDREY LARISSA GASTAO
NOGUEIRA 19HEQ111 - 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):

$$A = \begin{pmatrix} 115 & 27 & 33 & 30 & 24 \\ 19 & 74 & 34 & 14 & 3 \\ 8 & 11 & 35 & 10 & 4 \\ 26 & 25 & 6 & 76 & 18 \\ 17 & 20 & 9 & 1 & 50 \end{pmatrix}$$

e

$$B = (1699, 1217, 703, 1537, 879)$$

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

111-70546 - 30/09

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:35.6
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
70553 CAROLINE VEIGA DE MORAES
19HEQ111 - 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):

$$A = \begin{pmatrix} 49 & 13 & 16 & 9 & 5 \\ 23 & 82 & 28 & 1 & 29 \\ 26 & 14 & 80 & 33 & 4 \\ 27 & 8 & 31 & 83 & 15 \\ 35 & 32 & 6 & 21 & 96 \end{pmatrix}$$

e

$$B = (987, 912, 1305, 1453, 1236)$$

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

111-70553 - 30/09

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:35.6
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
70560 DANIEL FERNANDO ALMEIDA
KARGER 19HEQ111 - 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):

$$A = \begin{pmatrix} 80 & 14 & 27 & 5 & 31 \\ 33 & 82 & 12 & 23 & 9 \\ 26 & 25 & 61 & 6 & 3 \\ 20 & 10 & 13 & 63 & 18 \\ 17 & 24 & 29 & 7 & 82 \end{pmatrix}$$

e

$$B = (1791, 2030, 1275, 1422, 1611)$$

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

111-70560 - 30/09

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:35.6
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
71237 DAVI ALEFE COSTA CARNEIRO
19HEQ111 - 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):

$$A = \begin{pmatrix} 81 & 18 & 33 & 16 & 11 \\ 8 & 74 & 1 & 35 & 29 \\ 34 & 25 & 88 & 7 & 21 \\ 3 & 20 & 6 & 35 & 4 \\ 13 & 27 & 19 & 28 & 90 \end{pmatrix}$$

e

$$B = (1437, 1644, 1823, 759, 1364)$$

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

111-71237 - 30/09

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:35.6
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
70577 DAVI RAMOS JORGE 19HEQ111 - 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):

$$A = \begin{pmatrix} 122 & 25 & 28 & 35 & 32 \\ 4 & 93 & 30 & 33 & 24 \\ 21 & 29 & 109 & 26 & 27 \\ 20 & 10 & 23 & 73 & 18 \\ 22 & 1 & 11 & 9 & 49 \end{pmatrix}$$

e

$$B = (3179, 1909, 1672, 1627, 1190)$$

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

111-70577 - 30/09

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:35.6
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
70584 FILIPE LAZZARI PACHECO 19HEQ111 - 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

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

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

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

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

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

Veja por exemplo, se

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

e

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

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

$$x = Bx + g$$

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

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

Agora o processo iterativo pode ser definido como

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

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

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

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

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

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

Critérios de convergência

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

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

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

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

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

Definindo uma matriz como estritamente diagonalmente dominante se

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

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

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

Exemplo

Seja resolver o sistema abaixo, usando o método de Jacobi

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

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

Convergência ?

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

Solução

Dividindo cada equação pelo elemento da diagonal principal, fica

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

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

Iterações

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

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

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

Solução Python

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

☞ Para você fazer

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

$$A = \begin{pmatrix} 100 & 27 & 29 & 31 & 11 \\ 10 & 86 & 33 & 26 & 12 \\ 6 & 22 & 70 & 17 & 24 \\ 18 & 2 & 20 & 45 & 4 \\ 32 & 25 & 1 & 13 & 73 \end{pmatrix}$$

e

$$B = (2227, 1495, 1884, 1142, 1647)$$

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

111-70584 - 30/09

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:35.6
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
70591 FLAVIA ALVES PRATES 19HEQ111 - 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

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

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):

$$A = \begin{pmatrix} 66 & 4 & 20 & 23 & 16 \\ 2 & 46 & 27 & 13 & 1 \\ 10 & 8 & 74 & 30 & 22 \\ 11 & 18 & 14 & 52 & 7 \\ 24 & 15 & 9 & 17 & 67 \end{pmatrix}$$

e

$$B = (1522, 1109, 1632, 908, 1852)$$

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

111-70591 - 30/09

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:35.6
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
70603 FLAVIO PERELLES FILHO 19HEQ111 - 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

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

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):

$$A = \begin{pmatrix} 58 & 12 & 9 & 19 & 15 \\ 16 & 62 & 14 & 23 & 8 \\ 26 & 21 & 85 & 1 & 31 \\ 30 & 24 & 20 & 104 & 27 \\ 32 & 18 & 25 & 3 & 85 \end{pmatrix}$$

e

$$B = (829, 1572, 2117, 2070, 1519)$$

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

111-70603 - 30/09

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:35.6
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
70610 GABRIEL LISBOA KOSINSKI 19HEQ111 - 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

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

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):

$$A = \begin{pmatrix} 74 & 1 & 8 & 30 & 34 \\ 11 & 63 & 10 & 32 & 9 \\ 6 & 14 & 65 & 29 & 13 \\ 23 & 28 & 19 & 106 & 35 \\ 17 & 27 & 16 & 24 & 89 \end{pmatrix}$$

e

$$B = (1808, 1428, 918, 2010, 2022)$$

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

111-70610 - 30/09

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:35.6
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
70627 GUILHERMINA FERNANDA C. MORAES 19HEQ111 - 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

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

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):

$$A = \begin{pmatrix} 91 & 32 & 16 & 15 & 27 \\ 11 & 64 & 25 & 7 & 19 \\ 29 & 5 & 74 & 8 & 31 \\ 34 & 18 & 1 & 79 & 24 \\ 26 & 17 & 9 & 3 & 58 \end{pmatrix}$$

e

$$B = (1343, 1167, 1927, 1643, 1332)$$

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

111-70627 - 30/09

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:35.6
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
70634 ISABELLA DO ROSARIO 19HEQ111 - 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

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

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

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

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

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

Veja por exemplo, se

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

e

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

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

$$x = Bx + g$$

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

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

Agora o processo iterativo pode ser definido como

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

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

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

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

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

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

Critérios de convergência

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

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

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

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

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

Definindo uma matriz como estritamente diagonalmente dominante se

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

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

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

Exemplo

Seja resolver o sistema abaixo, usando o método de Jacobi

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

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

Convergência ?

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

Solução

Dividindo cada equação pelo elemento da diagonal principal, fica

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

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

Iterações

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

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

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

Solução Python

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

☞ Para você fazer

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

$$A = \begin{pmatrix} 101 & 19 & 32 & 33 & 16 \\ 18 & 68 & 11 & 2 & 31 \\ 26 & 14 & 76 & 13 & 22 \\ 7 & 24 & 10 & 48 & 6 \\ 9 & 29 & 1 & 3 & 48 \end{pmatrix}$$

e

$$B = (1506, 597, 807, 548, 442)$$

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

111-70634 - 30/09

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:35.6
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
70641 JOAO CARLOS BORSATO 19HEQ111 - 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):

$$A = \begin{pmatrix} 44 & 23 & 10 & 3 & 5 \\ 18 & 100 & 16 & 33 & 32 \\ 15 & 6 & 46 & 20 & 2 \\ 13 & 8 & 17 & 78 & 35 \\ 27 & 4 & 26 & 7 & 71 \end{pmatrix}$$

e

$$B = (1034, 2663, 953, 1656, 1116)$$

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

111-70641 - 30/09

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:35.6
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
70658 JOSE PAULO DE OLIVEIRA NETO
19HEQ111 - 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):

$$A = \begin{pmatrix} 66 & 20 & 25 & 16 & 2 \\ 15 & 84 & 28 & 11 & 23 \\ 7 & 35 & 105 & 26 & 30 \\ 18 & 3 & 1 & 28 & 5 \\ 29 & 6 & 27 & 4 & 68 \end{pmatrix}$$

e

$$B = (1259, 2135, 2633, 535, 908)$$

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

111-70658 - 30/09

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:35.6
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
70665 JULIANA HIROMI SUMI 19HEQ111 - 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):

$$A = \begin{pmatrix} 71 & 20 & 11 & 1 & 34 \\ 7 & 81 & 27 & 30 & 15 \\ 29 & 5 & 78 & 31 & 9 \\ 35 & 33 & 19 & 103 & 13 \\ 17 & 32 & 18 & 12 & 85 \end{pmatrix}$$

e

$$B = (1920, 1716, 1173, 1959, 2187)$$

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

111-70665 - 30/09

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:35.6
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
70672 LEONARDO CHAVES 19HEQ111 - 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):

$$A = \begin{pmatrix} 91 & 20 & 23 & 17 & 30 \\ 31 & 99 & 29 & 27 & 3 \\ 19 & 33 & 88 & 4 & 25 \\ 26 & 32 & 10 & 86 & 16 \\ 21 & 13 & 15 & 18 & 68 \end{pmatrix}$$

e

$$B = (2184, 2393, 1573, 1996, 1856)$$

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

111-70672 - 30/09

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:35.6
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
70689 LEONARDO MARQUES CAMILO
19HEQ111 - 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

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

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

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

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

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

Veja por exemplo, se

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

e

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

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

$$x = Bx + g$$

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

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

Agora o processo iterativo pode ser definido como

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

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

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

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

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

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

Critérios de convergência

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

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

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

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

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

Definindo uma matriz como estritamente diagonalmente dominante se

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

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

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

Exemplo

Seja resolver o sistema abaixo, usando o método de Jacobi

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

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

Convergência ?

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

Solução

Dividindo cada equação pelo elemento da diagonal principal, fica

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

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

Iterações

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

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

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

Solução Python

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

☞ Para você fazer

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

$$A = \begin{pmatrix} 78 & 23 & 20 & 26 & 5 \\ 19 & 67 & 7 & 35 & 1 \\ 24 & 28 & 87 & 25 & 9 \\ 17 & 18 & 29 & 76 & 11 \\ 15 & 13 & 32 & 22 & 88 \end{pmatrix}$$

e

$$B = (2061, 1502, 1838, 1859, 2408)$$

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

111-70689 - 30/09

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:35.6
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
71206 LYANDRA C S LISBOA AMARAL
19HEQ111 - 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

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

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):

$$A = \begin{pmatrix} 88 & 27 & 17 & 9 & 33 \\ 7 & 78 & 23 & 20 & 22 \\ 16 & 12 & 69 & 34 & 5 \\ 2 & 14 & 35 & 63 & 6 \\ 25 & 21 & 15 & 31 & 93 \end{pmatrix}$$

e

$$B = (1827, 1095, 1118, 1148, 1605)$$

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

111-71206 - 30/09

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:35.6
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
71213 MANOELA S VIEIRA 19HEQ111 - 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):

$$A = \begin{pmatrix} 93 & 5 & 33 & 29 & 22 \\ 17 & 82 & 34 & 15 & 10 \\ 14 & 31 & 84 & 3 & 35 \\ 6 & 26 & 20 & 88 & 32 \\ 30 & 4 & 21 & 24 & 84 \end{pmatrix}$$

e

$$B = (2255, 2317, 2447, 1506, 1967)$$

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

111-71213 - 30/09

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:35.6
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
70696 MARCELO EDUARDO MARQUES RIBAS 19HEQ111 - 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):

$$A = \begin{pmatrix} 65 & 19 & 15 & 2 & 26 \\ 1 & 82 & 23 & 22 & 27 \\ 30 & 28 & 93 & 18 & 16 \\ 25 & 29 & 13 & 72 & 4 \\ 7 & 6 & 3 & 34 & 51 \end{pmatrix}$$

e

$$B = (624, 1073, 1743, 1388, 815)$$

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

111-70696 - 30/09

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:35.6
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
70715 MARYNA BORNEMANN DA SILVA
19HEQ111 - 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):

$$A = \begin{pmatrix} 96 & 19 & 34 & 14 & 25 \\ 29 & 72 & 13 & 16 & 11 \\ 18 & 4 & 49 & 5 & 21 \\ 3 & 15 & 27 & 79 & 33 \\ 6 & 1 & 23 & 2 & 41 \end{pmatrix}$$

e

$$B = (1892, 1768, 1239, 1444, 985)$$

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

111-70715 - 30/09

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:35.6
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
70722 MATEUS DE MATOS LEME 19HEQ111 - 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):

$$A = \begin{pmatrix} 48 & 6 & 2 & 16 & 23 \\ 29 & 91 & 3 & 33 & 21 \\ 8 & 1 & 40 & 12 & 11 \\ 35 & 9 & 19 & 81 & 14 \\ 4 & 34 & 30 & 25 & 101 \end{pmatrix}$$

e

$$B = (845, 2290, 622, 1607, 1466)$$

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

111-70722 - 30/09

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:35.6
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
70739 MAYARA MATEUS ROSA 19HEQ111 - 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):

$$A = \begin{pmatrix} 74 & 20 & 4 & 25 & 23 \\ 27 & 102 & 1 & 35 & 33 \\ 24 & 21 & 107 & 29 & 30 \\ 28 & 8 & 31 & 75 & 5 \\ 19 & 9 & 32 & 14 & 80 \end{pmatrix}$$

e

$$B = (1417, 2749, 2312, 1369, 1993)$$

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

111-70739 - 30/09

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:35.6
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
70746 PHILLIP YAGYU MORIBAYASHI
19HEQ111 - 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):

$$A = \begin{pmatrix} 33 & 13 & 2 & 4 & 10 \\ 6 & 69 & 32 & 8 & 17 \\ 16 & 25 & 73 & 22 & 5 \\ 31 & 12 & 24 & 102 & 34 \\ 29 & 14 & 23 & 7 & 82 \end{pmatrix}$$

e

$$B = (421, 1432, 1498, 1145, 1148)$$

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

111-70746 - 30/09

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:35.6
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
70753 RAFAELA BORIN OLSEMANN
19HEQ111 - 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):

$$A = \begin{pmatrix} 75 & 6 & 22 & 24 & 17 \\ 34 & 86 & 8 & 10 & 33 \\ 23 & 29 & 96 & 30 & 11 \\ 16 & 15 & 26 & 77 & 18 \\ 31 & 3 & 4 & 35 & 75 \end{pmatrix}$$

e

$$B = (1609, 1518, 1887, 1543, 1302)$$

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

111-70753 - 30/09

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:35.6
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
70760 RODRIGO ENZO V. SCHWITZNER
19HEQ111 - 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

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

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):

$$A = \begin{pmatrix} 50 & 3 & 19 & 9 & 13 \\ 7 & 87 & 27 & 23 & 28 \\ 2 & 34 & 61 & 6 & 15 \\ 25 & 18 & 14 & 84 & 26 \\ 16 & 10 & 11 & 12 & 51 \end{pmatrix}$$

e

$$B = (959, 1315, 1334, 1630, 1066)$$

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

111-70760 - 30/09

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:35.6
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
71149 STHEFANNY GRANZOTI PEREIRA
19HEQ111 - 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):

$$A = \begin{pmatrix} 101 & 31 & 19 & 35 & 8 \\ 23 & 103 & 33 & 18 & 21 \\ 9 & 12 & 64 & 29 & 11 \\ 28 & 14 & 13 & 88 & 24 \\ 27 & 20 & 17 & 2 & 72 \end{pmatrix}$$

e

$$B = (2014, 2201, 1112, 1942, 943)$$

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

111-71149 - 30/09

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:35.6
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
70777 VINICIUS GABRIEL DE AQUINO
19HEQ111 - 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):

$$A = \begin{pmatrix} 43 & 1 & 3 & 30 & 4 \\ 2 & 54 & 11 & 19 & 18 \\ 17 & 22 & 61 & 5 & 15 \\ 28 & 16 & 21 & 103 & 35 \\ 8 & 23 & 33 & 12 & 77 \end{pmatrix}$$

e

$$B = (648, 1351, 1415, 1615, 1806)$$

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

111-70777 - 30/09

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:35.6
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
70784 VINICIUS RICARDO NUNES 19HEQ111 - 28 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):

$$A = \begin{pmatrix} 44 & 20 & 10 & 3 & 9 \\ 16 & 70 & 4 & 35 & 14 \\ 12 & 27 & 83 & 22 & 21 \\ 19 & 5 & 29 & 65 & 7 \\ 8 & 15 & 18 & 30 & 76 \end{pmatrix}$$

e

$$B = (729, 1279, 1180, 1387, 1181)$$

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

111-70784 - 30/09

U Positivo - UTFPR - PUCPr – 09/09/2019 - 09:14:35.6
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
70791 VINICIUS SILVEIRA MALESKI
19HEQ111 - 29 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):

$$A = \begin{pmatrix} 57 & 34 & 5 & 11 & 1 \\ 13 & 54 & 2 & 21 & 15 \\ 33 & 29 & 116 & 22 & 30 \\ 9 & 14 & 6 & 54 & 23 \\ 35 & 16 & 27 & 12 & 95 \end{pmatrix}$$

e

$$B = (1060, 1137, 2177, 1297, 2442)$$

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

111-70791 - 30/09