U Positivo - UTFPR - PUCPr – 23/03/2019 - 18:16:58.4
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
68993 ANDRE HEINECK HAHN 19FCN110 - 1
apos 18/04, 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):

$$A = \begin{pmatrix} 112 & 34 & 27 & 26 & 19 \\ 24 & 87 & 33 & 18 & 10 \\ 11 & 29 & 63 & 6 & 16 \\ 15 & 31 & 5 & 82 & 28 \\ 3 & 7 & 9 & 25 & 45 \end{pmatrix}$$

e

$$B = (2522, 2008, 1585, 2061, 852)$$

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

| $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ |
|---|---|---|---|---|
| | | | | |

110-68993 - 18/04

U Positivo - UTFPR - PUCPr – 23/03/2019 - 18:16:58.4
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
69789 BRUNO V GUIMARARES 19FCN110 - 2
apos 18/04, 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):

$$A = \begin{pmatrix} 75 & 28 & 21 & 5 & 20 \\ 23 & 89 & 14 & 17 & 33 \\ 6 & 12 & 30 & 7 & 1 \\ 27 & 15 & 9 & 56 & 3 \\ 19 & 13 & 10 & 25 & 71 \end{pmatrix}$$

e

$$B = (1692, 1956, 711, 1204, 1868)$$

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

| $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ |
|---|---|---|---|---|
| | | | | |

110-69789 - 18/04

U Positivo - UTFPR - PUCPr – 23/03/2019 - 18:16:58.4
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
69008 EDUARDO SCARANTE DEMCZUK
19FCN110 - 3 apos 18/04, 50% _____ / _____ /

## Sistemas Lineares - Jacobi

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

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

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

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

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

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

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

Veja por exemplo, se

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

e

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

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

$$x = Bx + g$$

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

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

Agora o processo iterativo pode ser definido como

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

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

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

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

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

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

## Critérios de convergência

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

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

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

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

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

Definindo uma matriz como estritamente diagonalmente dominante se

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

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

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

## Exemplo

Seja resolver o sistema abaixo, usando o método de Jacobi

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

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

## Convergência ?

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

## Solução

Dividindo cada equação pelo elemento da diagonal principal, fica

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

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

## Iterações

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

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

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

## Solução Python

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

## ☞ Para você fazer

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

$$A = \begin{pmatrix} 67 & 26 & 12 & 3 & 18 \\ 34 & 80 & 19 & 21 & 5 \\ 33 & 7 & 74 & 14 & 16 \\ 23 & 29 & 2 & 86 & 31 \\ 22 & 15 & 6 & 4 & 49 \end{pmatrix}$$

e

$$B = (1443, 1493, 1991, 1865, 923)$$

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

| $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ |
|---|---|---|---|---|
| | | | | |

110-69008 - 18/04

U Positivo - UTFPR - PUCPr – 23/03/2019 - 18:16:58.4
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
69015 FELIPE DE ROSSI REZENDE 19FCN110 - 4 apos 18/04, 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):

$$A = \begin{pmatrix} 66 & 27 & 8 & 12 & 15 \\ 16 & 99 & 33 & 22 & 26 \\ 35 & 5 & 94 & 31 & 18 \\ 29 & 2 & 34 & 92 & 25 \\ 7 & 4 & 1 & 9 & 25 \end{pmatrix}$$

e

$$B = (692, 1561, 817, 880, 289)$$

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

| $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ |
|---|---|---|---|---|
| | | | | |

110-69015 - 18/04

U Positivo - UTFPR - PUCPr – 23/03/2019 - 18:16:58.4
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
69022 GIOVANA STOPANOVSKI BECKER
19FCN110 - 5 apos 18/04, 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):

$$A = \begin{pmatrix} 71 & 14 & 35 & 5 & 10 \\ 24 & 65 & 29 & 2 & 9 \\ 26 & 17 & 65 & 8 & 12 \\ 19 & 30 & 15 & 84 & 18 \\ 16 & 21 & 7 & 1 & 48 \end{pmatrix}$$

e

$$B = (941, 1321, 1122, 1639, 938)$$

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

| $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ |
|---|---|---|---|---|
| | | | | |

110-69022 - 18/04

U Positivo - UTFPR - PUCPr – 23/03/2019 - 18:16:58.4
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
69208 IAN DO AMARAL PIMENTA 19FCN110 - 6 apos 18/04, 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):

$$A = \begin{pmatrix} 83 & 13 & 32 & 9 & 27 \\ 12 & 72 & 16 & 18 & 24 \\ 35 & 21 & 99 & 25 & 15 \\ 20 & 22 & 17 & 66 & 6 \\ 30 & 23 & 10 & 2 & 70 \end{pmatrix}$$

e

$$B = (1789, 1914, 1903, 1828, 1836)$$

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

| $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ |
|---|---|---|---|---|
| | | | | |

110-69208 - 18/04

U Positivo - UTFPR - PUCPr – 23/03/2019 - 18:16:58.4
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
69039 KHAREN INGRID VIDAL BELO
19FCN110 - 7 apos 18/04, 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):

$$A = \begin{pmatrix} 106 & 33 & 35 & 16 & 18 \\ 17 & 78 & 30 & 13 & 12 \\ 10 & 34 & 100 & 22 & 25 \\ 1 & 4 & 27 & 55 & 14 \\ 6 & 19 & 31 & 23 & 81 \end{pmatrix}$$

e

$$B = (2354, 1226, 2355, 1381, 1969)$$

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

| $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ |
|---|---|---|---|---|
| | | | | |

U Positivo - UTFPR - PUCPr – 23/03/2019 - 18:16:58.4
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
69046 MARCELL ANDREY MACHADO PINTO
19FCN110 - 8 apos 18/04, 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):

$$A = \begin{pmatrix} 57 & 14 & 4 & 11 & 27 \\ 10 & 65 & 15 & 2 & 31 \\ 7 & 21 & 53 & 9 & 12 \\ 25 & 8 & 28 & 82 & 17 \\ 20 & 3 & 19 & 26 & 75 \end{pmatrix}$$

e

$$B = (843, 1021, 1174, 1106, 1619)$$

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

| $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ |
|---|---|---|---|---|
| | | | | |

110-69046 - 18/04

U Positivo - UTFPR - PUCPr – 23/03/2019 - 18:16:58.4
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
69060 MELISSA SCHELLIN BECKER 19FCN110 - 9 apos 18/04, 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):

$$A = \begin{pmatrix} 53 & 31 & 13 & 2 & 6 \\ 32 & 77 & 18 & 8 & 15 \\ 33 & 28 & 69 & 4 & 3 \\ 7 & 17 & 16 & 69 & 26 \\ 34 & 29 & 1 & 22 & 91 \end{pmatrix}$$

e

$$B = (1234, 1344, 1842, 1154, 2027)$$

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

| $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ |
|---|---|---|---|---|
| | | | | |

110-69060 - 18/04

U Positivo - UTFPR - PUCPr – 23/03/2019 - 18:16:58.4
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
69077 TAYANE CORDEIRO FARIA 19FCN110 - 10 apos 18/04, 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):

$$A = \begin{pmatrix} 73 & 32 & 6 & 10 & 22 \\ 11 & 95 & 23 & 29 & 30 \\ 27 & 9 & 55 & 1 & 15 \\ 17 & 2 & 25 & 67 & 21 \\ 24 & 12 & 16 & 18 & 75 \end{pmatrix}$$

e

$$B = (1392, 1574, 901, 919, 1629)$$

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

| $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ |
|---|---|---|---|---|
| | | | | |

110-69077 - 18/04

U Positivo - UTFPR - PUCPr – 23/03/2019 - 18:16:58.4
Prof Dr P Kantek (pkantek@gmail.com)
Sistemas Lineares - Jacobi VIVO736p, V: 1.05
69796 THONNY G FERRAZ 19FCN110 - 11 apos 18/04, 50% _____ / _____ / _____

## Sistemas Lineares - Jacobi

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

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

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

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

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

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

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

Veja por exemplo, se

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

e

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

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

$$x = Bx + g$$

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

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

Agora o processo iterativo pode ser definido como

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

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

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

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

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

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

## Critérios de convergência

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

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

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

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

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

Definindo uma matriz como estritamente diagonalmente dominante se

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

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

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

## Exemplo

Seja resolver o sistema abaixo, usando o método de Jacobi

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

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

## Convergência ?

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

## Solução

Dividindo cada equação pelo elemento da diagonal principal, fica

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

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

## Iterações

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

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

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

## Solução Python

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

## ☞ Para você fazer

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

$$A = \begin{pmatrix} 105 & 10 & 27 & 33 & 32 \\ 17 & 45 & 1 & 20 & 3 \\ 21 & 19 & 80 & 26 & 13 \\ 7 & 2 & 16 & 37 & 9 \\ 11 & 6 & 35 & 31 & 87 \end{pmatrix}$$

e

$$B = (1637, 925, 1153, 648, 2062)$$

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

| $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ |
|---|---|---|---|---|
| | | | | |

110-69796 - 18/04