Taller análisis Numérico

Jenifer Medina Yepez

Cristian Camilo Contreras Borja

Kevin Andrés Garzón Ospina

Oscar Andrés Pacheco Turizo

Julian Carrillo

22/03/2021

Punto 4: Dado el sistema

$$
\begin{aligned}
2x - z &= 1.5 \\
\alpha x + 2y - z &= 2 \\
-x + y + \delta z &= 1.6666...
\end{aligned}
$$

a) Determine los valores de α, δ, para que asegura la convergencia por el metodo de Jacobi y para Gauss-Seidel. Sugerencia: utilice el teorema convergencia.

Tras hallar la determinante de la matriz se encuentran los valores siguientes:

2 -1 0 = 1.5

alfa(5) 2 -1 = 2

-1 1 delta(-1) = 1.6666

=(4*delta-0-1)-(0-2-alfadelta)

d=-1

a=5

```
            break

"""-------------------------------------------------------------
Pruebas
-------------------------------------------------------------"""

#matrices
A = array([[2,-1,0],[5, 2, .1],[-1,1,-1]]) # Matriz
B = array([1.5, 2.1 ,1.666]) #
X = array([1, 2, 3]) #

Jacobi(A, B, 20, X)

a = array([[2,-1,0],[5, 2, .1],[-1,1,-1]])
b = array([1.5, 2.1 ,1.666])
x = array([1, 2, 3])          #Iteration initial value

Gauss_seidel(A, B, X, 20)
```

b) Genere una tabla que tenga 20 iteraciones, del metodo de Jacobi con vector inicial $x_0 = [1, 2, 3]$

Se utilizo Python para realizar la codificación y las respectivas pruebas:

```python
from numpy import array, diag, diagflat, dot

def Jacobi(A, b, N, x):

    # (1) Create a vector using the diagonal elemts of A
    D = diag(A)
    # (2) Subtract D vector from A into the new vector R
    R = A - diagflat(D)

    # iteraciones
    for i in range(N):
        x = (b - dot(R,x)) / D
        print("iteracion: ",i+1)
        print("solucion aproximada: ",x)
```

```
iteracion:  1
solucion aproximada:  [ 1.75  -1.6   -0.666]
iteracion:  2
solucion aproximada:  [-0.05   -3.2917 -5.016 ]
iteracion:  3
solucion aproximada:  [-0.89585  1.4258  -4.9077 ]
iteracion:  4
solucion aproximada:  [1.4629  3.53501 0.65565]
iteracion:  5
solucion aproximada:  [ 2.517505  -2.6400325  0.40611  ]
iteracion:  6
solucion aproximada:  [-0.57001625 -5.264068   -6.8235375 ]
iteracion:  7
solucion aproximada:  [-1.882034    2.8162175  -6.36005175]
iteracion:  8
solucion aproximada:  [2.15810875 6.07308759 3.0322515 ]
iteracion:  9
solucion aproximada:  [ 3.78654379 -4.49688445  2.24897884]
iteracion:  10
solucion aproximada:  [-1.49844222 -8.52880843 -9.94942824]
iteracion:  11
solucion aproximada:  [-3.51440421  5.29357697 -8.6963662 ]
iteracion:  12
solucion aproximada:  [ 3.39678849 10.27082884  7.14198119]
iteracion:  13
solucion aproximada:  [ 5.88541442 -7.79907028  5.20804036]
iteracion:  14
solucion aproximada:  [ -3.14953514 -13.92393807 -15.3504847 ]
iteracion:  15
```

```
iteracion:  15
solucion aproximada:  [ -6.21196904   9.69136208 -12.44040293]
iteracion:  16
solucion aproximada:  [ 5.59568104 17.20194274 14.23733112]
iteracion:  17
solucion aproximada:  [  9.35097137 -13.65106916   9.94026169]
iteracion:  18
solucion aproximada:  [ -6.07553458 -22.8244415  -24.66804053]
iteracion:  19
solucion aproximada:  [-10.66222075  17.47223847 -18.41490693]
iteracion:  20
solucion aproximada:  [ 9.48611924 28.62629723 26.46845923]
```

$c)$ Genere una tabla que tenga 20 iteraciones, del metodo Gauss-Seidel con vector inicial $x_0 = [1, 2, 3]$

Se utilizo Python para realizar la codificación y las respectivas pruebas:

```python
def Gauss_seidel(a, b, x, g):
    # Set the precision of x
    x = x.astype(float)
    m, n = a.shape
    times = 0
    while True:
        for i in range(n):
            s1 = 0
            for j in range(n):
                if i != j:
                    s1 += x[j] * a[i][j]
            x[i] = (b[i] - s1) / a[i][i]
            times += 1
            print("iteracion: ",times)
            print("solucion aproximada: ",x)
            if times >= g:
                break
        if times >= g:
            break
```

```
iteracion:  1
solucion aproximada:  [1.75 2.   3.  ]
iteracion:  2
solucion aproximada:  [ 1.75  -3.475  3.   ]
iteracion:  3
solucion aproximada:  [ 1.75  -3.475 -6.891]
iteracion:  4
solucion aproximada:  [-0.9875 -3.475  -6.891 ]
iteracion:  5
solucion aproximada:  [-0.9875  3.8633 -6.891 ]
iteracion:  6
solucion aproximada:  [-0.9875  3.8633  3.1848]
iteracion:  7
solucion aproximada:  [2.68165 3.8633  3.1848 ]
iteracion:  8
solucion aproximada:  [ 2.68165  -5.813365  3.1848  ]
iteracion:  9
solucion aproximada:  [  2.68165   -5.813365 -10.161015]
iteracion:  10
solucion aproximada:  [ -2.1566825  -5.813365  -10.161015 ]
iteracion:  11
solucion aproximada:  [ -2.1566825   6.949757  -10.161015 ]
iteracion:  12
solucion aproximada:  [-2.1566825  6.949757   7.4404395]
iteracion:  13
solucion aproximada:  [4.2248785 6.949757  7.4404395]
```

```
iteracion:  14
solucion aproximada:  [ 4.2248785  -9.88421823  7.4404395 ]
iteracion:  15
solucion aproximada:  [  4.2248785   -9.88421823 -15.77509673]
iteracion:  16
solucion aproximada:  [ -4.19210911  -9.88421823 -15.77509673]
iteracion:  17
solucion aproximada:  [ -4.19210911  12.31902762 -15.77509673]
iteracion:  18
solucion aproximada:  [-4.19210911 12.31902762 14.84513673]
iteracion:  19
solucion aproximada:  [ 6.90951381 12.31902762 14.84513673]
iteracion:  20
solucion aproximada:  [  6.90951381 -16.96604136  14.84513673]
```