Práctica Dirigida 4

Análisis y Modelamiento Numérico I

Autor:

▪ Chowdhury Gomez, Junal Johir	20200092K

13 de noviembre de 2024

Ejercicio 8

Encuentra una descomposición en valores singulares (SVD) para las siguientes matrices:

$$A = \begin{pmatrix} 1 & -1 \\ 0 & 1 \\ 1 & 0 \end{pmatrix}$$

$$B = \begin{pmatrix} 1 & 1 & 1 \\ -1 & 0 & -2 \\ 1 & 2 & 0 \end{pmatrix}$$

$$C = \begin{pmatrix} 3 & 2 & 2 \\ 2 & 3 & -2 \end{pmatrix}$$

Código en Python

```python
import numpy as np
def hallarSVD(A):
  U, S, Vt = np.linalg.svd(A)
  print("Matriz U (vectores singulares izquierdos):")
  print(U)
  print("\nValores singulares:")
  print(S)
  print("\nMatriz V^T (transpuesta de los vectores singulares derechos):")
  print(Vt)
def menu():
  print("\na)")
  matrizA = np.array([[1, -1], [0, 1],[1, 0]])
  hallarSVD(matrizA)
  print("\nb)")
  matrizB = np.array([[1, 1, 1], [-1, 0, -2],[1, 2, 0]])
  hallarSVD(matrizB)
  print("\nc)")
  matrizC = np.array([[3, 2, 2], [2, 3, -2]])
  hallarSVD(matrizC)
menu()
```

Output del código

```
a)
Matriz U (vectores singulares izquierdos):
[[-8.16496581e-01  1.85577521e-16 -5.77350269e-01]
 [ 4.08248290e-01 -7.07106781e-01 -5.77350269e-01]
 [-4.08248290e-01 -7.07106781e-01  5.77350269e-01]]

Valores singulares:
[1.73205081 1.        ]

Matriz V^T (transpuesta de los vectores singulares derechos):
[[-0.70710678  0.70710678]
 [-0.70710678 -0.70710678]]

b)
Matriz U (vectores singulares izquierdos):
[[-5.77350269e-01  2.02242588e-16 -8.16496581e-01]
 [ 5.77350269e-01 -7.07106781e-01 -4.08248290e-01]
 [-5.77350269e-01 -7.07106781e-01  4.08248290e-01]]

Valores singulares:
[3. 2. 0.]

Matriz V^T (transpuesta de los vectores singulares derechos):
```

```
[[-5.77350269e-01 -5.77350269e-01 -5.77350269e-01]
 [ 1.33226763e-16 -7.07106781e-01  7.07106781e-01]
 [-8.16496581e-01  4.08248290e-01  4.08248290e-01]]

c)
Matriz U (vectores singulares izquierdos):
[[-0.70710678 -0.70710678]
 [-0.70710678  0.70710678]]

Valores singulares:
[5. 3.]

Matriz V^T (transpuesta de los vectores singulares derechos):
[[-7.07106781e-01 -7.07106781e-01 -6.47932334e-17]
 [-2.35702260e-01  2.35702260e-01 -9.42809042e-01]
 [-6.66666667e-01  6.66666667e-01  3.33333333e-01]]
```

Ejercicio 11

Considere el siguiente sistema de ecuaciones lineales:

$$\begin{pmatrix} 10 & -3 & 6 \\ 1 & -8 & -2 \\ -2 & 4 & 9 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 25 \\ -9 \\ -50 \end{pmatrix}$$

¿Cuál de los métodos iterativos (Richardson, Gauss-Jacobi, Gauss–Seidel) converge?

Código en Python

```python
import numpy as np
A = np.array([[10, -3, 6],
              [1, -8, -2],
              [-2, 4, 9]])
b = np.array([25, -9, -50])
x = np.zeros_like(b)
# numero de iteraciones
n_iter = 10
# Metodo de Gauss-Jacobi
print("Metodo de Gauss-Jacobi:")
for _ in range(n_iter):
    x_new = np.zeros_like(x)
    for i in range(A.shape[0]):
        s = sum(A[i, j] * x[j] for j in range(A.shape[1]) if i != j)
        x_new[i] = (b[i] - s) / A[i, i]
    x = x_new
    print(f"Iteracion {_+1}: {x}")

# Metodo de Gauss-Seidel
print("\nMetodo de Gauss-Seidel:")
for _ in range(n_iter):
    x_new = np.copy(x)
    for i in range(A.shape[0]):
        s1 = sum(A[i, j] * x_new[j] for j in range(i))
        s2 = sum(A[i, j] * x[j] for j in range(i + 1, A.shape[1]))
        x_new[i] = (b[i] - s1 - s2) / A[i, i]
    x = x_new
    print(f"Iteracion {_+1}: {x}")
```

Output del código

```
Metodo de Gauss-Jacobi:
Iteracion 1: [ 2  1 -5]
Iteracion 2: [ 5  2 -5]
Iteracion 3: [ 6  3 -5]
Iteracion 4: [ 6  3 -5]
Iteracion 5: [ 6  3 -5]
Iteracion 6: [ 6  3 -5]
Iteracion 7: [ 6  3 -5]
Iteracion 8: [ 6  3 -5]
Iteracion 9: [ 6  3 -5]
Iteracion 10: [ 6  3 -5]

Metodo de Gauss-Seidel:
Iteracion 1: [ 6  3 -5]
Iteracion 2: [ 6  3 -5]
Iteracion 3: [ 6  3 -5]
Iteracion 4: [ 6  3 -5]
Iteracion 5: [ 6  3 -5]
Iteracion 6: [ 6  3 -5]
Iteracion 7: [ 6  3 -5]
Iteracion 8: [ 6  3 -5]
Iteracion 9: [ 6  3 -5]
Iteracion 10: [ 6  3 -5]
```

Ejercicio 12

Consideramos el siguiente sistema de ecuaciones:

$$\begin{cases} x + ay = a \\ ax + y + bz = b \\ by + z = c \end{cases}$$

a) Solución Única

Representamos el sistema en forma matricial $A\mathbf{x} = \mathbf{b}$, donde:

$$A = \begin{bmatrix} 1 & a & 0 \\ a & 1 & b \\ 0 & b & 1 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

El sistema tiene una solución única si y solo si la matriz A es no singular, es decir, si el determinante de A no es cero. Calculamos el determinante de A:

$$\det(A) = 1 \cdot (1 \cdot 1 - b \cdot b) - a \cdot (a \cdot 1 - b \cdot 0) + 0 \cdot (a \cdot b - 1 \cdot 0) = 1 - b^2 - a^2$$

Para que el sistema tenga una solución única, necesitamos que:

$$1 - a^2 - b^2 \neq 0$$

b) Convergencia del Método de Gauss-Jacobi

El método de Gauss-Jacobi converge si la matriz A es estrictamente diagonal dominante. Para que A sea estrictamente diagonal dominante, cada elemento diagonal debe ser mayor en magnitud que la suma de las magnitudes de los otros elementos no diagonales de su fila correspondiente. Por tanto, se debe cumplir:

1. $|1| > |a| + |0|$ implica $1 > |a|$ 2. $|1| > |a| + |b|$ implica $1 > |a| + |b|$ 3. $|1| > |b| + |0|$ implica $1 > |b|$

Por lo tanto, para asegurar la convergencia del método de Gauss-Jacobi:

$$|a| < 1, \quad |b| < 1, \quad \text{y} \quad |a| + |b| < 1$$

c) Convergencia del Método de Gauss-Seidel

Para el método de Gauss-Seidel, también se favorece la dominancia diagonal estricta, aunque el método es más tolerante debido a la actualización inmediata de las variables durante las iteraciones. Sin embargo, se recomienda adherirse a la misma condición de dominancia diagonal estricta para asegurar la convergencia. Por lo tanto, las condiciones son las mismas que para el método de Gauss-Jacobi:

$$|a| < 1, \quad |b| < 1, \quad \text{y} \quad |a| + |b| < 1$$

Estas condiciones deben ayudar a garantizar una convergencia estable y efectiva en ambos métodos iterativos.

Ejercicio 32

Resuelva el siguiente sistema de ecuaciones lineales usando la aceleración de Chebyshev del método de Jacobi:

$$\begin{pmatrix} 4 & -1 & -1 & 0 \\ -1 & 4 & 0 & -1 \\ -1 & 0 & 4 & -1 \\ 0 & -1 & -1 & 4 \end{pmatrix}$$

Código en Python

```python
import numpy as np
A = np.array([[4, -1, -1, 0],
              [-1, 4, 0, -1],
              [-1, 0, 4, -1],
              [0, -1, -1, 4]], dtype=np.double)
b = np.zeros(4)
def chebyshev_acceleration(A, b, n_iterations, l_min, l_max):
    n = len(b)
    x = np.zeros_like(b, dtype=np.double)
    x_old = np.copy(x)
    D_inv = np.diag(1 / np.diag(A))  # Inversa de la matriz diagonal D de A

    d = (l_max + l_min) / 2
    c = (l_max - l_min) / 2

    # Inicializar tau y la iteracion inicial
    tau = 1 / d
    rho_prev = 1
    for k in range(1, n_iterations + 1):
        rho = 1 / (2 * d - rho_prev)
        x[:] = x_old + rho * (x - x_old)
        x_old[:] = x
        for i in range(n):
            s1 = np.dot(A[i, :i], x[:i])
            s2 = np.dot(A[i, i+1:], x[i+1:])
            x[i] = x_old[i] + tau * (b[i] - s1 - s2) * D_inv[i, i]

        rho_prev = rho
        print(f"Iteracion {k}: {x}")
    return x
# Estimar valores propios minimo y maximo
l_max = 6
# Asumiendo un calculo previo o conocimiento del problema
l_min = 2
# Asumiendo un calculo previo o conocimiento del problema
# Llamada al metodo de Jacobi con aceleracion de Chebyshev
chebyshev_acceleration(A, b, 10, l_min, l_max)
```

Output del código

```
Iteración 1: [0. 0. 0. 0.]
Iteración 2: [0. 0. 0. 0.]
Iteración 3: [0. 0. 0. 0.]
Iteración 4: [0. 0. 0. 0.]
Iteración 5: [0. 0. 0. 0.]
Iteración 6: [0. 0. 0. 0.]
Iteración 7: [0. 0. 0. 0.]
Iteración 8: [0. 0. 0. 0.]
Iteración 9: [0. 0. 0. 0.]
Iteración 10: [0. 0. 0. 0.]
```

Ejercicio 33

Dado un sistema lineal de ecuaciones con la matriz de Hilbert modificada definida como $a_{ij} = (1 + i + j)^{-1}$ y el vector $\mathbf{b}$ calculado como $\mathbf{b}_i = \frac{1}{3}\sum_{j=1}^{n} a_{ij}$, implemente y pruebe:

a) El método de Jacobi

b) El método de Gauss-Seidel

Código en Python

```python
import numpy as np
def crear_matriz_hilbert(n):
    H = np.zeros((n, n))
    for i in range(n):
        for j in range(n):
            H[i, j] = 1.0 / (1 + i + j)
    return H
def vector_b(H):
    b = np.sum(H, axis=1) / 3
    return b

n = 5  # Tamanio de la matriz
H = crear_matriz_hilbert(n)
b = vector_b(H)
def jacobi(A, b, n_iterations=25):
    n = len(b)
    x = np.zeros_like(b, dtype=np.double)
    for it in range(n_iterations):
        x_new = np.zeros_like(x, dtype=np.double)
        for i in range(n):
            s1 = np.dot(A[i, :i], x[:i])
            s2 = np.dot(A[i, i+1:], x[i+1:])
            x_new[i] = (b[i] - s1 - s2) / A[i, i]
        x = x_new
        print(f"Jacobi Iter {it+1}: {x}")
    return x
def gauss_seidel(A, b, n_iterations=25):
    n = len(b)
    x = np.zeros_like(b, dtype=np.double)
    for it in range(n_iterations):
        x_new = np.copy(x)
        for i in range(n):
            s1 = np.dot(A[i, :i], x_new[:i])
            s2 = np.dot(A[i, i+1:], x_new[i+1:])
            x_new[i] = (b[i] - s1 - s2) / A[i, i]
        x = x_new
        print(f"Gauss-Seidel Iter {it+1}: {x}")
    return x
print("Metodo de jacobi:")
x_jacobi = jacobi(H, b)
print("\nMetodo Gauss-Seidel:")
x_gauss_seidel = gauss_seidel(H, b)
```

Output del código

```
Metodo de jacobi:
Jacobi Iter 1: [0.76111111 1.45       1.82142857 2.06388889 2.23690476]
Jacobi Iter 2: [-1.53438492 -3.41452381 -4.57728647 -5.38034722 -5.97180697]
Jacobi Iter 3: [ 6.53358337 13.39865405 17.39609063 20.09489448 22.05855656]
Jacobi Iter 4: [-21.17234773 -44.483658   -58.31805185 -67.72467375 -74.59468492]
Jacobi Iter 5: [ 74.29239615 154.87920718 202.4323932  234.70069507 258.24322801]
Jacobi Iter 6: [-254.47977625 -531.75492018 -695.6416018  -806.91264434 -888.11672087]
Jacobi Iter 7: [ 877.87061039 1833.10681277 2397.44432917 2780.5443851  3060.10595234]
Jacobi Iter 8: [ -3022.09769174  -6311.8187697   -8255.59005825  -9575.1703095
 -10538.13126956]
Jacobi Iter 9: [10409.95267999 21740.45690177 28434.96958928 32979.66772249
```

```
 36296.13757751]
Jacobi Iter 10: [ -35851.93498233  -74875.5756342   -97932.53982274 -113585.2445314
 -125007.73607448]
Jacobi Iter 11: [123480.58721687 257883.77209663 337295.12643791 391205.15485422
 430545.74890847]
Jacobi Iter 12: [ -425283.27224509  -888186.7430205  -1161690.84589061 -1347364.75581505
 -1482859.44000457]
Jacobi Iter 13: [1464737.49153958 3059043.06627691 4001029.9814142  4640517.22742288
 5107180.10791368]
Jacobi Iter 14: [ -5044762.7609372  -10535777.66378059 -13780114.64486951
 -15982599.74516742 -17589853.13280798]
Jacobi Iter 15: [17374881.70414469 36286717.98856238 47460675.76617105 55046347.53504322
 60581956.67326532]
Jacobi Iter 16: [-5.98415617e+07 -1.24976615e+08 -1.63461314e+08 -1.89587447e+08
 -2.08652872e+08]
Jacobi Iter 17: [2.06102849e+08 4.30437234e+08 5.62984011e+08 6.52966125e+08
 7.18630160e+08]
Jacobi Iter 18: [-7.09847516e+08 -1.48248703e+09 -1.93899698e+09 -2.24890818e+09
 -2.47506445e+09]
Jacobi Iter 19: [2.44481578e+09 5.10589615e+09 6.67818132e+09 7.74555955e+09
 8.52447391e+09]
Jacobi Iter 20: [-8.42029318e+09 -1.75854323e+10 -2.30006060e+10 -2.66768084e+10
 -2.93595002e+10]
Jacobi Iter 21: [2.90006870e+10 6.05667295e+10 7.92173577e+10 9.18787214e+10
 1.01118293e+11]
Jacobi Iter 22: [-9.98824897e+10 -2.08600428e+11 -2.72835844e+11 -3.16443381e+11
 -3.48265781e+11]
Jacobi Iter 23: [3.44009497e+11 7.18449536e+11 9.39685441e+11 1.08987600e+12
 1.19947687e+12]
Jacobi Iter 24: [-1.18481762e+12 -2.47444236e+12 -3.23641027e+12 -3.75368791e+12
 -4.13116889e+12]
Jacobi Iter 25: [4.08068036e+12 8.52233133e+12 1.11466572e+13 1.29282349e+13
 1.42283331e+13]

Metodo Gauss-Seidel:
Gauss-Seidel Iter 1: [0.76111111 0.30833333 0.16749339 0.10486883 0.07107869]
Gauss-Seidel Iter 2: [0.51068037 0.45989876 0.25725998 0.16400947 0.11255846]
Gauss-Seidel Iter 3: [0.38189434 0.52952859 0.30595381 0.198799   0.138184  ]
Gauss-Seidel Iter 4: [0.31702566 0.55662474 0.33290284 0.22052169 0.15521662]
Gauss-Seidel Iter 5: [0.28565738 0.56191548 0.3483015  0.23514051 0.16749897]
Gauss-Seidel Iter 6: [0.27176795 0.55628816 0.35752925 0.2458125  0.17707069]
Gauss-Seidel Iter 7: [0.26692335 0.54544519 0.36342701 0.25421472 0.18502008]
Gauss-Seidel Iter 8: [0.26668848 0.53235814 0.36749729 0.26124324 0.19193312]
Gauss-Seidel Iter 9: [0.26873551 0.51856127 0.37053669 0.26738169 0.19813023]
Gauss-Seidel Iter 10: [0.27184678 0.50483318 0.37296946 0.27289558 0.20379109]
Gauss-Seidel Iter 11: [0.27538925 0.49155613 0.37502326 0.27793476 0.20902053]
Gauss-Seidel Iter 12: [0.27903749 0.47890519 0.37682192 0.28258748 0.21388326]
Gauss-Seidel Iter 13: [0.28262769 0.46694791 0.37843421 0.28690893 0.21842224]
Gauss-Seidel Iter 14: [0.28608074 0.45569676 0.37989973 0.29093632 0.22266842]
Gauss-Seidel Iter 15: [0.28936172 0.44513661 0.38124246 0.29469687 0.22664588]
Gauss-Seidel Iter 16: [0.29245859 0.43523921 0.38247794 0.29821206 0.23037457]
Gauss-Seidel Iter 17: [0.29537093 0.42597062 0.38361709 0.30149986 0.23387183]
Gauss-Seidel Iter 18: [0.2981041  0.41729519 0.38466821 0.304576   0.23715316]
Gauss-Seidel Iter 19: [0.30066614 0.40917744 0.38563808 0.30745461 0.24023272]
Gauss-Seidel Iter 20: [0.30306617 0.40158306 0.38653249 0.31014861 0.24312354]
Gauss-Seidel Iter 21: [0.30531356 0.39447937 0.38735659 0.31266993 0.24583773]
Gauss-Seidel Iter 22: [0.30741753 0.38783543 0.38811507 0.31502967 0.24838659]
Gauss-Seidel Iter 23: [0.30938697 0.38162214 0.38881222 0.31723817 0.25078063]
```

```
Gauss-Seidel Iter 24: [0.3112303  0.37581217 0.38945203 0.31930507 0.25302972]
Gauss-Seidel Iter 25: [0.31295547 0.37037987 0.39003821 0.32123941 0.25514307]
```

Ejercicio 34

Considere el siguiente sistema de ecuaciones lineales:

$$\begin{pmatrix} 10 & 1 & 2 & 3 & 4 \\ 1 & 9 & -1 & 2 & -3 \\ 2 & -1 & 7 & 3 & -5 \\ 3 & 2 & 3 & 12 & -1 \\ 4 & 3 & -5 & -1 & 15 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} = \begin{pmatrix} 12 \\ -27 \\ 14 \\ -17 \\ 12 \end{pmatrix}$$

Código en Python

```python
import numpy as np
def jacobi(A, b, x=None, max_iterations=100, tolerance=1e-10):
    n = len(b)
    if x is None:
        x = np.zeros_like(b, dtype=np.double)
    x_new = np.zeros_like(x, dtype=np.double)

    for it in range(max_iterations):
        for i in range(n):
            s = np.dot(A[i, :], x) - A[i, i] * x[i]
            x_new[i] = (b[i] - s) / A[i, i]

        # Convergencia
        if np.linalg.norm(x_new - x, ord=np.inf) < tolerance:
            print(f"Jacobi converge  en la iteracion {it+1}")
            return x_new
        x[:] = x_new
        print(f"Iteracion Jacobi  {it+1}: {x}")
    return x

def gauss_seidel(A, b, x=None, max_iterations=100, tolerance=1e-10):
    n = len(b)
    if x is None:
        x = np.zeros_like(b, dtype=np.double)

    for it in range(max_iterations):
        x_old = x.copy()
        for i in range(n):
            s1 = np.dot(A[i, :i], x[:i])
            s2 = np.dot(A[i, i+1:], x[i+1:])
            x[i] = (b[i] - s1 - s2) / A[i, i]

        # Convergencia
        if np.linalg.norm(x - x_old, ord=np.inf) < tolerance:
            print(f"Gauss-Seidel converge en laiteracion {it+1}")
            return x
        print(f"Iteracion Gauss-Seidel {it+1}: {x}")
    return x

A = np.array([
    [10, 1, 2, 3, 4],
    [1, 9, -1, 2, -3],
    [2, -1, 7, 3, -5],
    [3, 2, 3, 12, -1],
    [4, -3, -5, -1, 15]
    ], dtype=np.double)
b = np.array([12, -27, 14, -17, 12], dtype=np.double)
print("Testing Jacobi Method:")
x_jacobi = jacobi(A, b)
print("\nTesting Gauss-Seidel Method:")
x_gauss_seidel = gauss_seidel(A, b)
```

Output del código

```
Resolviendo por el metodo de  Jacobi:
Iteracion Jacobi  1: [ 1.2        -3.          2.          -1.41666667  0.8        ]
Iteracion Jacobi  2: [ 1.205      -2.32962963  2.40714286 -1.65        0.45222222]
Iteracion Jacobi  3: [ 1.2656455  -2.34902116  2.35306878 -1.89374559  0.70512169]
Iteracion Jacobi  4: [ 1.25036336 -2.22330227  2.61807615 -1.87108157  0.65079685]
Iteracion Jacobi  5: [ 1.19972073 -2.21530372  2.59188571 -1.95899309  0.7698626 ]
Iteracion Jacobi  6: [ 1.18290612 -2.15336234  2.73022101 -1.93119577  0.77037609]
Iteracion Jacobi  7: [ 1.14050033 -2.14212948  2.7323276  -1.97185672  0.83521319]
Iteracion Jacobi  8: [ 1.12521917 -2.10653553  2.80978657 -1.95825097  0.84675944]
Iteracion Jacobi  9: [ 1.09746776 -2.09540582  2.82165375 -1.97876556  0.88467991]
Iteracion Jacobi  10: [ 1.08496754 -2.0738048   2.8670507  -1.97348941  0.89689431]
Iteracion Jacobi  11: [ 1.06725944 -2.06447279  2.88017142 -1.9842959   0.92003197]
Iteracion Jacobi  12: [ 1.05768898 -2.05093337  2.90772227 -1.98277625  0.93027367]
Iteracion Jacobi  13: [ 1.04627229 -2.04373258  2.91905511 -1.98867444  0.94481861]
Iteracion Jacobi  14: [ 1.03923713 -2.03504583  2.93626274 -1.98864154  0.95268762]
Iteracion Jacobi  15: [ 1.03176945 -2.0297365   2.94512037 -1.99197669  0.96203908]
Iteracion Jacobi  16: [ 1.02672695 -2.02406427  2.95612144 -1.99242978  0.96782253]
Iteracion Jacobi  17: [ 1.02178206 -2.02025315  2.96269769 -1.99438284  0.97393845]
Iteracion Jacobi  18: [ 1.01822525 -2.01650037  2.9698605  -1.99491621  0.97808119]
Iteracion Jacobi  19: [ 1.01492032 -2.01380986  2.97460053 -1.99609794  0.98213228]
Iteracion Jacobi  20: [ 1.01243735 -2.01130301  2.97932925 -1.99656755  0.98505292]
Iteracion Jacobi  21: [ 1.01021355 -2.00942381  2.98268422 -1.9973034   0.98776135]
Iteracion Jacobi  22: [ 1.00849202 -2.00773761  2.98583801 -1.99767369  0.98979947]
Iteracion Jacobi  23: [ 1.00698848 -2.00643424  2.98818526 -1.99814295  0.99162236]
Iteracion Jacobi  24: [ 1.00580031 -2.00529447  2.99030421 -1.9984192   0.99303511]
Iteracion Jacobi  25: [ 1.00478032 -2.00439471  2.99193401 -1.99872413  0.99426781]
Iteracion Jacobi  26: [ 1.00396278 -2.00362162  2.99336516 -1.99892381  0.9952427 ]
Iteracion Jacobi  27: [ 1.00326919 -2.00300243  2.99449111 -1.99912482  0.99607906]
Iteracion Jacobi  28: [ 1.00270784 -2.00247681  2.99546128 -1.99926641  0.99674978]
Iteracion Jacobi  29: [ 1.00223544 -2.0020516   2.99623652 -1.99940033  0.99731855]
Iteracion Jacobi  30: [ 1.00185054 -2.00169362  2.99689589 -1.99949951  0.99777905]
Iteracion Jacobi  31: [ 1.00152842 -2.00140205  2.99742844 -1.99958942  0.99816646]
Iteracion Jacobi  32: [ 1.00126476 -2.00115797  2.99787738 -1.99965833  0.9984822 ]
Iteracion Jacobi  33: [ 1.00104494 -2.00095824  2.99824264 -1.99971902  0.99874638]
Iteracion Jacobi  34: [ 1.00086445 -2.00079168  2.99854869 -1.99976666  0.99896265]
Iteracion Jacobi  35: [ 1.00071437 -2.00065494  2.99879895 -1.99980778  0.99914293]
Iteracion Jacobi  36: [ 1.00059087 -2.00054123  2.99900776 -1.99984059  0.99929098]
Iteracion Jacobi  37: [ 1.00048836 -2.00044767  2.9991791  -1.99986854  0.99941407]
Iteracion Jacobi  38: [ 1.00040388 -2.00037     2.99932165 -1.99989108  0.99951537]
Iteracion Jacobi  39: [ 1.00033385 -2.000306    2.9994389  -1.9999101   0.99959944]
Iteracion Jacobi  40: [ 1.00027607 -2.00025293  2.99953626 -1.99992557  0.99966874]
Iteracion Jacobi  41: [ 1.00022822 -2.00020916  2.99961647 -1.99993853  0.99972618]
Iteracion Jacobi  42: [ 1.00018871 -2.0001729   2.99968298 -1.99994913  0.99977357]
Iteracion Jacobi  43: [ 1.00015601 -2.00014297  2.99973784 -1.99995798  0.99981282]
Iteracion Jacobi  44: [ 1.000129   -2.0001182   2.99978329 -1.99996523  0.99984522]
Iteracion Jacobi  45: [ 1.00010664 -2.00009773  2.9998208  -1.99997127  0.99987204]
Iteracion Jacobi  46: [ 1.00008818 -2.0000808   2.99985186 -1.99997624  0.9998942 ]
Iteracion Jacobi  47: [ 1.0000729  -2.00006681  2.99987751 -1.99998036  0.99991253]
Iteracion Jacobi  48: [ 1.00006028 -2.00005523  2.99989873 -1.99998376  0.99992768]
Iteracion Jacobi  49: [ 1.00004983 -2.00004567  2.99991627 -1.99998657  0.99994021]
Iteracion Jacobi  50: [ 1.0000412  -2.00003776  2.99993077 -1.9999889   0.99995056]
Iteracion Jacobi  51: [ 1.00003406 -2.00003122  2.99994276 -1.99999082  0.99995913]
Iteracion Jacobi  52: [ 1.00002816 -2.00002581  2.99995268 -1.99999241  0.99996621]
Iteracion Jacobi  53: [ 1.00002329 -2.00002134  2.99996087 -1.99999373  0.99997206]
Iteracion Jacobi  54: [ 1.00001925 -2.00001764  2.99996765 -1.99999481  0.9999769 ]
Iteracion Jacobi  55: [ 1.00001592 -2.00001459  2.99997325 -1.99999571  0.9999809 ]
Iteracion Jacobi  56: [ 1.00001316 -2.00001206  2.99997789 -1.99999645  0.99998421]
Iteracion Jacobi  57: [ 1.00001088 -2.00000997  2.99998172 -1.99999707  0.99998694]
Iteracion Jacobi  58: [ 1.000009   -2.00000824  2.99998488 -1.99999758  0.99998921]
Iteracion Jacobi  59: [ 1.00000744 -2.00000682  2.9999875  -1.999998    0.99999108]
Iteracion Jacobi  60: [ 1.00000615 -2.00000564  2.99998967 -1.99999834  0.99999262]
Iteracion Jacobi  61: [ 1.00000508 -2.00000466  2.99999146 -1.99999863  0.9999939 ]
Iteracion Jacobi  62: [ 1.0000042  -2.00000385  2.99999294 -1.99999887  0.99999496]
Iteracion Jacobi  63: [ 1.00000348 -2.00000318  2.99999416 -1.99999906  0.99999583]
Iteracion Jacobi  64: [ 1.00000287 -2.00000263  2.99999517 -1.99999923  0.99999655]
Iteracion Jacobi  65: [ 1.00000238 -2.00000218  2.99999601 -1.99999936  0.99999715]
Iteracion Jacobi  66: [ 1.00000196 -2.0000018   2.9999967  -1.99999947  0.99999764]
Iteracion Jacobi  67: [ 1.00000162 -2.00000149  2.99999727 -1.99999956  0.99999805]
Iteracion Jacobi  68: [ 1.00000134 -2.00000123  2.99999774 -1.99999964  0.99999839]
Iteracion Jacobi  69: [ 1.00000111 -2.00000102  2.99999813 -1.9999997   0.99999867]
Iteracion Jacobi  70: [ 1.00000092 -2.00000084  2.99999846 -1.99999975  0.9999989 ]
Iteracion Jacobi  71: [ 1.00000076 -2.0000007   2.99999872 -1.9999998   0.99999909]
Iteracion Jacobi  72: [ 1.00000063 -2.00000057  2.99999895 -1.99999983  0.99999925]
Iteracion Jacobi  73: [ 1.00000052 -2.00000048  2.99999913 -1.99999986  0.99999938]
Iteracion Jacobi  74: [ 1.00000043 -2.00000039  2.99999928 -1.99999988  0.99999949]
Iteracion Jacobi  75: [ 1.00000035 -2.00000032  2.9999994  -1.9999999   0.99999957]
Iteracion Jacobi  76: [ 1.00000029 -2.00000027  2.99999951 -1.99999992  0.99999965]
Iteracion Jacobi  77: [ 1.00000024 -2.00000022  2.99999959 -1.99999993  0.99999971]
Iteracion Jacobi  78: [ 1.0000002  -2.00000018  2.99999966 -1.99999995  0.99999976]
Iteracion Jacobi  79: [ 1.00000017 -2.00000015  2.99999972 -1.99999996  0.9999998 ]
Iteracion Jacobi  80: [ 1.00000014 -2.00000013  2.99999977 -1.99999996  0.99999984]
Iteracion Jacobi  81: [ 1.00000011 -2.0000001   2.99999981 -1.99999997  0.99999986]
Iteracion Jacobi  82: [ 1.00000009 -2.00000009  2.99999984 -1.99999997  0.99999989]
Iteracion Jacobi  83: [ 1.00000008 -2.00000007  2.99999987 -1.99999998  0.99999991]
Iteracion Jacobi  84: [ 1.00000006 -2.00000006  2.99999989 -1.99999998  0.99999992]
Iteracion Jacobi  85: [ 1.00000005 -2.00000005  2.99999991 -1.99999999  0.99999994]
Iteracion Jacobi  86: [ 1.00000004 -2.00000004  2.99999993 -1.99999999  0.99999995]
Iteracion Jacobi  87: [ 1.00000004 -2.00000003  2.99999994 -1.99999999  0.99999996]
Iteracion Jacobi  88: [ 1.00000003 -2.00000003  2.99999995 -1.99999999  0.99999996]
Iteracion Jacobi  89: [ 1.00000002 -2.00000002  2.99999996 -1.99999999  0.99999997]
Iteracion Jacobi  90: [ 1.00000002 -2.00000002  2.99999997 -1.99999999  0.99999998]
Iteracion Jacobi  91: [ 1.00000002 -2.00000002  2.99999997 -2.          0.99999998]
Iteracion Jacobi  92: [ 1.00000001 -2.00000001  2.99999998 -2.          0.99999998]
Iteracion Jacobi  93: [ 1.00000001 -2.00000001  2.99999998 -2.          0.99999999]
Iteracion Jacobi  94: [ 1.00000001 -2.00000001  2.99999998 -2.          0.99999999]
Iteracion Jacobi  95: [ 1.00000001 -2.00000001  2.99999999 -2.          0.99999999]
Iteracion Jacobi  96: [ 1.00000001 -2.00000001  2.99999999 -2.          0.99999999]
Iteracion Jacobi  97: [ 1.00000001 -2.          2.99999999 -2.          0.99999999]
Iteracion Jacobi  98: [ 1.         -2.          2.99999999 -2.          0.99999999]
Iteracion Jacobi  99: [ 1.         -2.          2.99999999 -2.          1.        ]
Iteracion Jacobi  100: [ 1.         -2.          2.99999999 -2.          1.        ]

Resolviendo por el metodo de Gauss-Seidel:
Iteracion Gauss-Seidel 1: [ 1.2        -3.13333333  1.20952381 -1.4968254   0.15671958]
Iteracion Gauss-Seidel 2: [ 1.65778836 -2.66493945  1.8990797  -1.84866714  0.33471731]
Iteracion Gauss-Seidel 3: [ 1.50739123 -2.4340917   2.25295911 -1.92317919  0.53398509]
Iteracion Gauss-Seidel 4: [ 1.35617707 -2.29498937  2.49030279 -1.95128965  0.67936986]
Iteracion Gauss-Seidel 5: [ 1.24507733 -2.20156507  2.6512855  -1.96721571  0.78028049]
Iteracion Gauss-Seidel 6: [ 1.16795192 -2.13793262  2.76131615 -1.97763821  0.8495558 ]
Iteracion Gauss-Seidel 7: [ 1.11499917 -2.09441547  2.8366114  -1.98470375  0.89700734]
Iteracion Gauss-Seidel 8: [ 1.07872745 -2.06463184  2.8881516  -1.98953051  0.92949481]
Iteracion Gauss-Seidel 9: [ 1.05389409 -2.04424411  2.92343333 -1.99283327  0.95173498]
Iteracion Gauss-Seidel 10: [ 1.03689373 -2.03028766  2.94758565 -1.99509399  0.96695976]
Iteracion Gauss-Seidel 11: [ 1.02525593 -2.02073367  2.96411932 -1.99664155  0.97738202]
Iteracion Gauss-Seidel 12: [ 1.01728916 -2.01419341  2.97543758 -1.99770095  0.98451667]
Iteracion Gauss-Seidel 13: [ 1.01183544 -2.00971622  2.98318559 -1.99842616  0.98940076]
Iteracion Gauss-Seidel 14: [ 1.00810205 -2.00665132  2.98848955 -1.99892262  0.9927442 ]
Iteracion Gauss-Seidel 15: [ 1.00554633 -2.00455322  2.99212042 -1.99926247  0.99503298]
Iteracion Gauss-Seidel 16: [ 1.00379679 -2.00311694  2.99460597 -1.99949512  0.99659978]
Iteracion Gauss-Seidel 17: [ 1.00259912 -2.00213373  2.99630747 -1.99965438  0.99767235]
Iteracion Gauss-Seidel 18: [ 1.00177925 -2.00146066  2.99747225 -1.9997634   0.99840659]
Iteracion Gauss-Seidel 19: [ 1.001218   -2.00099991  2.99826961 -1.99983803  0.99890922]
Iteracion Gauss-Seidel 20: [ 1.00083379 -2.0006845   2.99881544 -1.99988913  0.9992533 ]
Iteracion Gauss-Seidel 21: [ 1.00057078 -2.00046858  2.9991891  -1.9999241   0.99948884]
Iteracion Gauss-Seidel 22: [ 1.00039073 -2.00032077  2.99944489 -1.99994804  0.99965008]
Iteracion Gauss-Seidel 23: [ 1.00026748 -2.00021958  2.99962    -1.99996443  0.99976046]
Iteracion Gauss-Seidel 24: [ 1.00018311 -2.00015032  2.99973987 -1.99997565  0.99983602]
Iteracion Gauss-Seidel 25: [ 1.00012535 -2.0001029   2.99982192 -1.99998333  0.99988775]
Iteracion Gauss-Seidel 26: [ 1.00008581 -2.00007044  2.9998781  -1.99998859  0.99992316]
Iteracion Gauss-Seidel 27: [ 1.00005874 -2.00004822  2.99991655 -1.99999219  0.9999474 ]
Iteracion Gauss-Seidel 28: [ 1.00004021 -2.00003301  2.99994287 -1.99999465  0.99996399]
Iteracion Gauss-Seidel 29: [ 1.00002753 -2.0000226   2.99996089 -1.99999634  0.99997535]
Iteracion Gauss-Seidel 30: [ 1.00001884 -2.00001547  2.99997323 -1.99999749  0.99998312]
Iteracion Gauss-Seidel 31: [ 1.0000129  -2.00001059  2.99998167 -1.99999828  0.99998845]
Iteracion Gauss-Seidel 32: [ 1.00000883 -2.00000725  2.99998745 -1.99999883  0.99999209]
Iteracion Gauss-Seidel 33: [ 1.00000604 -2.00000496  2.99999141 -1.9999992   0.99999459]
Iteracion Gauss-Seidel 34: [ 1.00000414 -2.0000034   2.99999412 -1.99999945  0.99999629]
Iteracion Gauss-Seidel 35: [ 1.00000283 -2.00000233  2.99999598 -1.99999962  0.99999746]
Iteracion Gauss-Seidel 36: [ 1.00000194 -2.00000159  2.99999724 -1.99999974  0.99999826]
Iteracion Gauss-Seidel 37: [ 1.00000133 -2.00000109  2.99999811 -1.99999982  0.99999881]
Iteracion Gauss-Seidel 38: [ 1.00000091 -2.00000075  2.99999871 -1.99999988  0.99999919]
Iteracion Gauss-Seidel 39: [ 1.00000062 -2.00000051  2.99999912 -1.99999992  0.99999944]
Iteracion Gauss-Seidel 40: [ 1.00000043 -2.00000035  2.99999939 -1.99999994  0.99999962]
Iteracion Gauss-Seidel 41: [ 1.00000029 -2.00000024  2.99999959 -1.99999996  0.99999974]
Iteracion Gauss-Seidel 42: [ 1.0000002  -2.00000016  2.99999972 -1.99999997  0.99999982]
Iteracion Gauss-Seidel 43: [ 1.00000014 -2.00000011  2.99999981 -1.99999998  0.99999988]
Iteracion Gauss-Seidel 44: [ 1.00000009 -2.00000008  2.99999987 -1.99999999  0.99999992]
Iteracion Gauss-Seidel 45: [ 1.00000006 -2.00000005  2.99999991 -1.99999999  0.99999994]
Iteracion Gauss-Seidel 46: [ 1.00000004 -2.00000004  2.99999994 -1.99999999  0.99999996]
Iteracion Gauss-Seidel 47: [ 1.00000003 -2.00000002  2.99999996 -2.          0.99999997]
Iteracion Gauss-Seidel 48: [ 1.00000002 -2.00000002  2.99999997 -2.          0.99999998]
Iteracion Gauss-Seidel 49: [ 1.00000001 -2.00000001  2.99999998 -2.          0.99999999]
Iteracion Gauss-Seidel 50: [ 1.00000001 -2.00000001  2.99999999 -2.          0.99999999]
Iteracion Gauss-Seidel 51: [ 1.00000001 -2.00000001  2.99999999 -2.          0.99999999]
Iteracion Gauss-Seidel 52: [ 1.         -2.          2.99999999 -2.          1.        ]
Iteracion Gauss-Seidel 53: [ 1. -2.  3. -2.  1.]
Iteracion Gauss-Seidel 54: [ 1. -2.  3. -2.  1.]
Iteracion Gauss-Seidel 55: [ 1. -2.  3. -2.  1.]
Iteracion Gauss-Seidel 56: [ 1. -2.  3. -2.  1.]
Iteracion Gauss-Seidel 57: [ 1. -2.  3. -2.  1.]
Iteracion Gauss-Seidel 58: [ 1. -2.  3. -2.  1.]
Iteracion Gauss-Seidel 59: [ 1. -2.  3. -2.  1.]
Iteracion Gauss-Seidel 60: [ 1. -2.  3. -2.  1.]
Gauss-Seidel converge en la iteracion 61
```