Universidad Central de Venezuela

Facultad de Ingeniería

Escuela de Ingeniería Eléctrica

Laboratorio N°1

Periodo 2023-3

06/12/2023
CI 29571461 Ricardo Santana

Introducción

En el informe que se presentará, se llevará a cabo un estudio y análisis de los sistemas lineales y su resolución. Se definirán y caracterizarán las ecuaciones que describen los sistemas lineales utilizando matrices; de estas últimas se establecerán varias propiedades útiles para resolver las respectivas ecuaciones.

Para resolver los sistemas de ecuaciones lineales, se requieren métodos numéricos que utilizan cálculos matriciales para simplificar el sistema a través de la computadora. Estos métodos pueden ser directos o iterativos, dependiendo de la forma en que se adapten mejor a la situación o problema que se desea resolver.

Marco Teórico

A continuación se describe y verifica que la terna $(\mathbb{R}^{n\times n}, +, .)$, la cual tiene una estructura de anillo no conmutativo, con unidad y divisores de cero.

Sean las matrices A^{3x3} y B^{3x3} definidas como:

$$A = (6 \quad 6 \quad 4, \ 0 \quad 1 \quad 2, \ 7 \quad 5 \quad 8, \ ) B = (5 \quad 8 \quad 2, \ 3 \quad 4 \quad 0, \ 9 \quad 8 \quad 0, \ )$$

Los productos posibles entre ambas matrices son:

$$AB = (84 \quad 104 \quad 12, \ 21 \quad 20 \quad 0, \ 122 \quad 140 \quad 14, \ ) , , BA = (44 \quad 48 \quad 52, \ 18 \quad 22 \quad 20, \ 54 \quad 62 \quad 52, \ )$$

Por lo tanto queda demostrado que $AB \neq BA$

Sin embargo, para el caso particular en que $B = I(matrizidentidad)$:

$$I = (1 \quad 0 \quad 0, \ 0 \quad 1 \quad 0, \ 0 \quad 0 \quad 1, \ )$$

se tiene que:

$$AI = IA = A = (6 \quad 6 \quad 4, \ 0 \quad 1 \quad 2, \ 7 \quad 5 \quad 8, \ )$$

El teorema que se presenta a continuación proporciona las propiedades básicas sobre la suma de matrices y la multiplicación por escalares [1]

Teorema 2.1.1

Sean A, B y C tres matrices de $m \times n$ y sean a y b dos escalares. Entonces:

1. $A + 0 = A$
2. $0A = 0$
3. $A + B = B + A$ (ley conmutativa para la suma de matrices)
4. $(A + B) + C = A + (B + C)$ (ley asociativa para la suma de matrices)
5. $a(A + B) = aA + aB$ (ley distributiva para la multiplicación por un escalar)
6. $IA = A$
7. $(a + b)A = aA + bA$

Sistema de ecuaciones lineales

Según el libro de texto [2] "un sistema de m ecuaciones lineales con n incognitas $x_1, x_2, \ldots, x_n$ al que podemos llamar simplemente sistema lineal, es un conjunto de m ecuaciones lineales, cada una con n incógnitas. un sistema lineal puede denotarse sin problema mediante"

$$\begin{array}{ccccccc} a_{11}x_1 & + & a_{12}x_2 & + & \cdots & a_{1n}x_n & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & + & \cdots & a_{2n}x_n & = & b_2 \\ \vdots & & \vdots & & & \vdots & & \vdots \\ a_{m1}x_1 & + & a_{m2}x_2 & + & \cdots & a_{mn}x_n & = & b_m \end{array}$$

Cualquier vector de números reales $[x_1, x_2, \ldots, x_n]^T$ que satisface el sistema se denomina como solución

Al usar la notación matricial, el sistema se puede escribir como la matriz aumentada:

$$\left(\begin{array}{cccc|c} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & b_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} & b_m \end{array}\right)$$

Definición 1.2.1 [1]

Sistemas inconsistentes y consistentes:

Se dice que un sistema de ecuaciones lineales es inconsistente si no tiene solución. Se dice que un sistema que tiene al menos una solución es consistente.

En el presente informe diremos que un sistema es incompatible cuando es incosistente y un sistema es compatible cuando es consistente

Consideramos métodos directos e iterativos para resolver un sistema de n ecuaciones lineales en n variables. De manera que se pueda llegar a una solución compatible

Las técnicas directas son métodos que proporcionan teóricamente la solución exacta del sistema en un número finito de pasos.

Una técnica iterativa para resolver el sistema lineal $n \times n$, $Ax = b$ inicia con una aproximación $x(0)$ para la solución x y genera una sucesión de vectores $\{x(k)\}_{k=0}^{\infty}$ que convergen a x. [3]

Las diferencias, semejanzas y condiciones requeridas para resolver sistemas de ecuaciones lineales, se analizaran con detalle al finalizar la practica.

Práctica

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
#importando documentos y librerias necesarias para realizar la práctica

import numpy as np
import math as mt
import pandas as pd
import sys
sys.path.append('/content/drive/MyDrive/Colab-Notebooks/29571461-CN/29571461-Proy1/29571461-Lab1')
from resolver_SEL import SEL


ruta_carpeta = '/content/drive/MyDrive/Colab-Notebooks/29571461-CN/29571461-Proy1/29571461-Lab1'
matriz_A = np.loadtxt(f'{ruta_carpeta}/A20.txt',skiprows = 0, delimiter=';')
matriz_b = np.loadtxt(f'{ruta_carpeta}/b20.txt',skiprows = 0, delimiter=',')
#asignando instancia de la clase
Ax_b = SEL(matriz_A, matriz_b)
```

Metodos Directos

1. Eliminación Gaussiana

Construyendo sistema de ecuciones lineales con $Ax = b$, dados:

Matriz A

```
pd.DataFrame(Ax_b.A)
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	3.0	9.0	8.0	7.0	3.0	1.0	5.0	8.0	3.0	4.0	5.0	4.0	4.0	0.0	1.0	5.0	6.0	0.0	0.0	6.0
1	1.0	3.0	2.0	6.0	8.0	2.0	3.0	9.0	0.0	8.0	2.0	3.0	1.0	1.0	8.0	4.0	6.0	6.0	2.0	0.0
2	3.0	8.0	6.0	0.0	0.0	4.0	2.0	6.0	7.0	4.0	5.0	2.0	0.0	4.0	7.0	1.0	5.0	9.0	3.0	8.0
3	2.0	8.0	0.0	6.0	6.0	4.0	4.0	2.0	2.0	2.0	6.0	7.0	3.0	7.0	4.0	9.0	8.0	3.0	7.0	2.0
4	3.0	9.0	6.0	9.0	2.0	8.0	3.0	3.0	9.0	4.0	0.0	9.0	4.0	8.0	5.0	3.0	6.0	9.0	7.0	2.0
5	4.0	1.0	1.0	8.0	6.0	3.0	1.0	2.0	3.0	7.0	8.0	0.0	7.0	2.0	7.0	3.0	7.0	9.0	2.0	0.0
6	2.0	5.0	8.0	3.0	7.0	0.0	5.0	5.0	3.0	5.0	9.0	1.0	9.0	5.0	7.0	1.0	0.0	2.0	4.0	1.0
7	5.0	6.0	3.0	9.0	9.0	0.0	3.0	9.0	6.0	2.0	5.0	8.0	4.0	2.0	0.0	1.0	2.0	1.0	9.0	4.0
8	0.0	1.0	0.0	5.0	9.0	7.0	3.0	1.0	4.0	2.0	3.0	2.0	6.0	2.0	4.0	0.0	5.0	8.0	4.0	8.0
9	3.0	9.0	3.0	6.0	6.0	0.0	0.0	7.0	3.0	2.0	6.0	3.0	4.0	5.0	6.0	3.0	9.0	7.0	5.0	7.0
10	9.0	4.0	9.0	6.0	2.0	4.0	1.0	8.0	2.0	3.0	3.0	2.0	8.0	3.0	1.0	4.0	4.0	9.0	1.0	3.0
11	9.0	2.0	8.0	7.0	3.0	2.0	9.0	0.0	8.0	1.0	8.0	5.0	3.0	3.0	8.0	2.0	1.0	3.0	9.0	8.0
12	0.0	9.0	4.0	9.0	8.0	4.0	5.0	1.0	6.0	5.0	0.0	4.0	2.0	1.0	8.0	7.0	1.0	9.0	7.0	5.0
13	9.0	3.0	7.0	4.0	3.0	9.0	0.0	6.0	6.0	2.0	4.0	1.0	1.0	9.0	6.0	5.0	2.0	3.0	6.0	6.0
14	2.0	1.0	5.0	1.0	5.0	9.0	6.0	3.0	1.0	1.0	7.0	6.0	1.0	5.0	7.0	0.0	1.0	6.0	0.0	3.0
15	9.0	1.0	2.0	2.0	1.0	9.0	3.0	7.0	1.0	4.0	2.0	5.0	4.0	3.0	3.0	3.0	0.0	2.0	0.0	5.0
16	1.0	2.0	2.0	7.0	7.0	8.0	6.0	7.0	0.0	2.0	7.0	4.0	0.0	0.0	8.0	5.0	1.0	1.0	3.0	1.0
17	9.0	8.0	3.0	1.0	6.0	5.0	9.0	3.0	1.0	5.0	2.0	2.0	4.0	2.0	1.0	9.0	9.0	4.0	8.0	0.0
18	3.0	8.0	1.0	8.0	5.0	2.0	0.0	3.0	3.0	6.0	6.0	0.0	2.0	5.0	7.0	0.0	5.0	2.0	9.0	1.0
19	4.0	0.0	2.0	6.0	0.0	0.0	9.0	1.0	0.0	0.0	5.0	7.0	5.0	4.0	0.0	4.0	1.0	6.0	3.0	4.0

Matriz b

```
Ax_b.b
```

```
array([[2.],
       [1.],
       [9.],
       [1.],
       [5.],
       [1.],
       [7.],
       [1.],
       [1.],
       [1.],
       [4.],
       [1.],
       [6.],
       [1.],
       [1.],
       [1.],
       [2.],
       [1.],
       [9.],
       [1.]], dtype=float32)
```

Metodos Directos

1. Eliminación Gaussiana

```
xg = Ax_b.metodo_directo('gauss')
xg
```

```
matrix([[-1.11964968],
        [ 0.04417053],
        [ 0.90653156],
        [-0.4281711 ],
        [-1.11672739],
        [ 0.61707654],
        [-0.8546822 ],
        [-0.03388646],
        [-1.15426103],
        [ 1.35999402],
        [ 0.7177617 ],
        [ 0.3899889 ],
        [ 0.32225134],
        [-0.77879468],
        [-0.37055536],
        [ 0.02795713],
        [-0.46055414],
        [ 0.35415948],
        [ 1.886938  ],
        [ 0.42718354]])
```

2. PALU

```
xp = Ax_b.metodo_directo('palu')
xp
```

```
matrix([[-1.11964986],
        [ 0.04417033],
        [ 0.90653174],
        [-0.42817125],
        [-1.11672746],
        [ 0.61707666],
        [-0.85468243],
        [-0.03388655],
        [-1.15426121],
        [ 1.35999442],
        [ 0.71776185],
        [ 0.38998904],
        [ 0.32225131],
        [-0.77879477],
        [-0.3705556 ],
        [ 0.02795726],
        [-0.46055424],
        [ 0.35415955],
        [ 1.88693833],
        [ 0.42718363]])
```

Cabe señalar que por ambos métodos dió aproximadamente igual en relación a 5 cifras decimales.

3. Cholesky

```
xch = Ax_b.metodo_directo('cholesky')
xch
```

```
'La matriz A no es simétrica'
```

```
simetrica = Ax_b.D - Ax_b.Au - Ax_b.Au.T
#definiendo nuevo sistema con matriz simetrica basandose en la descomposición PALU
Asx_b = SEL(simetrica, matriz_b)

xch = Asx_b.metodo_directo('cholesky')
xch
```

```
'La matriz A no es definida positiva'
```

```
#importando matriz que cumpla las condiciones de cholesky
matriz_A_ch = np.loadtxt(f'{ruta_carpeta}/A_ch.txt',skiprows = 0, delimiter=';')
matriz_b_ch = np.loadtxt(f'{ruta_carpeta}/b_ch.txt',skiprows = 0, delimiter=',')
Achx_b = SEL(matriz_A_ch,matriz_b_ch)
xch = Achx_b.metodo_directo('cholesky')
```

```
#mostrando sistema agregado
'matriz A', Achx_b.A, 'matriz_b', Achx_b.b,'solucion', xch
```

```
('matriz A',
 array([[ 4.  , -1.  ,  1.  ],
        [-1.  ,  4.25,  2.75],
        [ 1.  ,  2.75,  3.5 ]], dtype=float32),
 'matriz_b',
 array([[  4.],
        [  2.],
        [-12.]], dtype=float32),
 'solucion',
 matrix([[  7.859375],
         [ 12.1875  ],
         [-15.25    ]]))
```

```
#comparando con gauss (ya demostrada su efectividad)
xg = Achx_b.metodo_directo('gauss')
xg
```

```
matrix([[  7.859375],
        [ 12.1875  ],
        [-15.25    ]])
```

quedando demostrado así que el metodo de cholesky es igual de efectivo que el de Gauss, ya que $x_g = x_{ch}$. Sin embargo se tienen que cumplir las condiciones suficientes y necesarias para la aplicación de cholesky

Metodos Iterativos

1. Jacobi

```
max_iter = 100
tol = 0.000001
w = 1.2
xj = Ax_b.metodo_iterativo('jacobi', max_iter, tol, w)
xj
```

```
'El metodo no converge'
```

Se intentará con el sistema $A_{ch}x_{ch} = b_{ch}$

```
xj = Achx_b.metodo_iterativo('jacobi', max_iter, tol, w)
xj[0]
```

```
array([[  7.85937334],
       [ 12.18749729],
       [-15.24999699]])
```

```
#variando parametros
iterp = max_iter
tolp = tol
wp = w
for k in range(3):
    xj = Achx_b.metodo_iterativo('jacobi', iterp, tolp, wp)
    print(xj[0])
    iterp *= 2
    tolp *= 2
    wp += 0.2
```

```
[[  7.85937334]
 [ 12.18749729]
 [-15.24999699]]
[[  7.8593742 ]
 [ 12.1874987 ]
 [-15.24999855]]
[[  7.85937334]
 [ 12.18749729]
 [-15.24999699]]
```

2. Gauss-Seidel

```
xse = Ax_b.metodo_iterativo('seidel', max_iter, tol, w)
xse
```

```
'El metodo no converge'
```

Se intentará con el sistema $A_{ch}x_{ch} = b_{ch}$

```
xj = Achx_b.metodo_iterativo('seidel', max_iter, tol, w)
xj[0]
```

```
array([[  7.85937459],
       [ 12.18749939],
       [-15.2499994 ]])
```

```
#variando parametros
iterp = max_iter
tolp = tol
wp = w
for k in range(3):
    xj = Achx_b.metodo_iterativo('seidel', iterp, tolp, wp)
    print(xj[0])
    iterp *= 2
    tolp *= 2
    wp += 0.2
```

```
 [[  7.85937459]
  [ 12.18749939]
  [-15.2499994 ]]
 [[  7.85937427]
  [ 12.1874989 ]
  [-15.24999893]]
 [[  7.85937369]
  [ 12.18749802]
  [-15.24999807]]
```

3. SOR

```
xso = Ax_b.metodo_iterativo('seidel', max_iter, tol, w)
xso
```

```
'El metodo no converge'
```

Se intentará con el sistema $A_{ch}x_{ch} = b_{ch}$

```
xj = Achx_b.metodo_iterativo('sor', max_iter, tol, w)
xj[0]
```

```
array([[  7.85937511],
       [ 12.1875002 ],
       [-15.25000035]])
```

```
#variando parametros
iterp = max_iter
tolp = tol
wp = w
for k in range(3):
    xj = Achx_b.metodo_iterativo('sor', iterp, tolp, wp)
    print(xj[0])
    iterp *= 2
    tolp *= 2
    wp += 0.2
```

```
 [[  7.85937511]
  [ 12.1875002 ]
  [-15.25000035]]
 [[  7.85937495]
  [ 12.18749951]
  [-15.24999945]]
 [[  7.85937468]
  [ 12.18750026]
  [-15.25000019]]
```

Análisis, comparaciones y semejanzas entre los metodos para resolver Sistemas de ecuaciones lineales

I. Metodos directos:

Semejanzas:

- En todos los casos se tiene que triangularizar la matriz de una u otra forma, en concreto, el metodo cholesky utiliza directamente la triangularizacion del metodo PALU.
- Tienen que ser matrices cuadradas, ya que tiene que haber misma cantidad de incognitas que ecuaciones.
- Determinante tiene que ser diferente de 0. Esto asegura que el sistema tenga una solución compatible.

Diferencias:

- Las diferentes formas en las que se puede factorizar la matriz A, de tal manera que el sistema $Ax = b$ puede sobreescribirse para diversificar la resolucion computacional.
- El método de cholesky está limitado por poseer obligatoriamente una matriz de coefientes que se sea simétrica, lo cual reduce considerablemente el numero de sistemas que se pueden resolver.

Condiciones necesarias:

a. GAUSS:

- Matriz cuadrada (mismo numero de incognitas que de ecuaciones)
- Matriz invertible (sistema incompatible)
- Filas de vector igual al numero de filas y columnas de la matriz de coeficientes

b. PALU:

- Matriz cuadrada
- Matriz invertible
- Filas de vector igual al numero de filas y columnas de la matriz de coeficientes

c. Cholesky:

- Matriz cuadrada
- Matriz definida positiva
- Matriz simétrica

II. Metodos iterativos:

Semejanzas:

- Los métodos iterativos Jacobi, Gauss-Seidel y SOR son utilizados para resolver sistemas de ecuaciones lineales algebraicas. Los tres parten de una solución inicial, la cual se va mejorando hasta alcanzar una solución aproximada. Ésto a traves de una sucesión $x_{k+1} = Mx_k + v$
- todos los metodos son iterativos, es decir, se basan en aproximaciones; esto significa que la solución obtenida no es exacta y su precisión depende del número de iteraciones realizadas.
- El método SOR es una variante del método de Gauss-Seidel, en el que se introduce un factor de sobrerelajación para acelerar la convergencia. En general, el método SOR es más rápido que el método de Gauss-Seidel aunque más complejo de implementar.

Diferencias:

- Una diferencia notable entre los métodos es que en el método de Jacobi, las soluciones de cada variable se actualizan con los valores de la iteración anterior. En cambio, en el método de Gauss-Seidel, las soluciones se van actualizando a medida que se van calculando, lo que permite una convergencia más rápida.
- Las distintas maneras en la que se puede expresar la matriz M y el vector v, basandose en algebra matricial, para que existan ditintas formas de iterar computacionalmente un resultado
- Los tres métodos son iterativos y aproximados, pero difieren en la forma en que actualizan las soluciones. Además, el método SOR es una extensión del método de Gauss-Seidel que introduce un factor de sobrerelajación.

Condiciones necesarias:

- Una condicion suficiente y necesaria es que el radio espectral de la matriz M de la ecuacion $x_{k+1} = Mx_k + v$,sea menor a 1 en cada metodo iterativo.

a. Jacobi:

- Si la matriz coeficientes es estrictamente diagonal dominante, la solución del sistema converge para cualquier iterado inicial.

b. Gauss-Seidel:

- Si la matriz coeficientes es estrictamente diagonal dominante, la solución del sistema converge para cualquier iterado inicial.

c. SOR:

- Si la matriz de coeficientes es simétrica con elementos de la diagonal positivos y definida positiva, se puede asegurar la convergencia de la solución del sistema para cualquier iterado.
- Es necesario elegir un valor adecuado para el parámetro de sobrerelajación (ω) para asegurar la convergencia del método, generalmente suele estar entre 1 y 2. Sin embargo, el valor óptimo de ω depende de la matriz de coeficientes y puede variar en cada iteración.

Coclusión

Se puede llegar a la conclusión de que el método de Gauss es efectivo y corto en términos de código. Sin embargo, los métodos de Cholesky y PALU son un poco más densos y requieren más procesos. El método de Cholesky también depende de ciertos factores, como la simetría y la positividad de la matriz, para poder aplicarse. Por otro lado, los métodos iterativos están limitados por el radio espectral para poder converger. Además, se puede decir que el método SOR llega a la solución más rápidamente, lo que implica menos tiempo de procesamiento.

Es importante tener en cuenta que estas conclusiones se basan en los resultados obtenidos y pueden variar dependiendo del contexto y las condiciones específicas de cada problema.

Referencias

[1] **Algebra Lineal**, *Stanley I. Grossman S., José Job Flores Godoy* 7ma edición

[2] **Algebra Lineal**, *Bernard Kolman, David R. Hill*, 8va edición

[3] **Analisis Numérico**, *Richard L. Burden • Douglas J. Faires • Annette M. Burden*, 10ma edición