Universidad Autónoma de Querétaro

Facultad de Ingeniería

Descomposición LU

Análisis numérico

Autor:
David Gómez Torres

8 Septiembre del 2021

ÍNDICE GENERAL

CAPÍTULO 1

INTRODUCCIÓN

1.1. Descomposición LU

Los métodos de descomposición LU separan el tiempo usado en las eliminaciones para la matriz [A] de las manipulaciones en el lado derecho B. Una vez que [A] se ha "descompuesto", los múltiples vectores del lado derecho B se pueden evaluar de manera eficiente.

1.1.1. Revisión

Supongamos un sistema de la forma

$$[A]X{-}B = 0 \tag{1.1}$$

que se pueda expresar como una matriz triangular superior:

$$[A] = \left[\begin{array}{ccc|c} u_{11} & u_{12} & u_{13} & x_1 \\ 0 & u_{22} & u_{23} & x_2 \\ 0 & 0 & u_{33} & x_3 \end{array}\right] \tag{1.2}$$

La ecuación (1.2) también se expresa en notación matricial y se reordena como

$$[U]X{-}D = 0 \tag{1.3}$$

Ahora, suponga que existe una matriz diagonal inferior con números 1 en la diagonal,

$$[L] = \begin{bmatrix} 1 & 0 & 0 \\ l_1 & 1 & 0 \\ l_2 & l_3 & 1 \end{bmatrix} \tag{1.4}$$

que tiene la propiedad de que cuando se premultiplica por la ecuación (1.3), el resultado es la ecuación (1.1).

$$[L][U]X{-}D = [A]X{-}B \tag{1.5}$$

Si esta ecuación se satisface, según las reglas de multiplicación entre matrices, se obtendrá

$$[L][U] = [A]$$

y

$$[L]D = B$$

1.1.2. Versión de la eliminación de Gauss

El primer paso en la eliminación de Gauss consiste en multiplicar el renglón 1 por el factor

$$f_{21} = \frac{a_{21}}{a_{11}}$$

y restar el resultado al segundo renglón para eliminar a 2l . De forma similar, el renglón 1 se multiplica por

$$f_{31} = \frac{a_{31}}{a_{11}}$$

y el resultado se resta al tercer renglón para eliminar a_{31} . El paso final es multiplicar el segundo renglón modificado por

$$f_{32} = \frac{a'_{32}}{a'_{22}}$$

y restar el resultado al tercer renglón para eliminar a_{32}.
Después de la eliminación la matriz [A], por lo tanto, se describe como

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ f_{21} & a'_{22} & a'_{23} \\ f_{31} & f_{32} & a''_{33} \end{bmatrix}$$

De hecho, esta matriz representa un almacenamiento eficiente de la descomposición LU de [A],

$$[A] \to [L][U] \tag{1.6}$$

donde

$$[U] = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ 0 & a'_{22} & a'_{23} \\ 0 & 0 & a''_{33} \end{bmatrix} \tag{1.7}$$

y

$$[L] = \begin{bmatrix} 1 & 0 & 0 \\ f_{21} & 1 & 0 \\ f_{31} & f_{32} & 1 \end{bmatrix} \tag{1.8}$$

Después de descomponer la matriz, se puede generar una solución para un vector particular B. Esto se lleva a cabo en dos pasos. Primero, se realiza un paso de sustitución hacia adelante al resolver la ecuación para D. Es importante notar que esto sólo se refiere a la realización de las operaciones de la eliminación en B. De esta forma, al final del procedimiento, el lado derecho estará en el mismo estado que si se hubiesen realizado las operaciones hacia adelante sobre [A] y B en forma simultánea.
El paso de la sustitución hacia adelante se representa en forma concisa como

$$d_i = b_i - \sum_{j=1}^{i-1} a_{ij} d_j \tag{1.9}$$

En el segundo paso, entonces, tan sólo se realiza la sustitución hacia atrás

$$x_i = \frac{d_i - \sum_{j=1+1}^{n} a_{ij} x_j}{a_{ii}} \tag{1.10}$$

1.1.3. Descomposición Crout

Un método alternativo a la ecuación (1.4), usa una matriz [U] con números 1 sobre la diagonal. Esto se conoce como **descomposición Crout**. Aunque hay algunas diferencias entre estos métodos, su funcionamiento es comparable.
La descomposición de Crout se puede implementar mediante la siguiente serie concisa de fórmulas:

$$l_{i,1} = a_{i,1} \tag{1.11}$$

$$u_{ij} = \frac{a_{1j}}{l_{11}} \tag{1.12}$$

$$l_{ij} = a_{ij} - \sum_{k=1}^{j-1} l_{ik} u_{kj} \tag{1.13}$$

$$u_{jk} = \frac{a_{jk} - \sum_{i=1}^{j-1} l_{ji} u_{ik}}{l_{jj}} \tag{1.14}$$

$$l_{nn} = a_{nn} - \sum_{i=1}^{n-1} l_{nk} u_{kn} \tag{1.15}$$

No hay necesidad de guardar los números 1 que están en la diagonal de [U] o los números cero de [L] o [U], ya que se dan en el método.
Por lo tanto, conforme se va calculando cada elemento de [L] y [U], se puede sustituir por el elemento correspondiente de [A].

1.2. Matriz inversa

En el estudio de las operaciones con matrices, vimos que si una matriz [A] es cuadrada, existe otra matriz $[A]^{-1}$, conocida como la inversa de [A]

$$[A][A]^{-1} = [A]^{-1}[A] = [I]$$

1.2.1. Cálculo de la inversa

La inversa se puede calcular en forma de columna por columna, generando soluciones con vectores unitarios como las constantes del lado derecho.

$$\{b\} = \begin{Bmatrix} 1 \\ 0 \\ 0 \end{Bmatrix}$$

La solución resultante será la primera columna de la matriz inversa.

La mejor forma de realizar un cálculo como éste es con el algoritmo de descomposición LU.Recuerde que una de las ventajas más importantes de la descomposición LU es que proporciona un medio eficiente para evaluar diversos vectores del lado derecho.

CAPÍTULO 2

METODOLOGÍA

2.1. Multiplicación de matrices

Problema 10.2

a) Use la eliminación simple de Gauss para descomponer el siguiente sistema

$$
\begin{aligned}
10x_1 + 2x_2 - x_3 &= 27 \\
-3x_1 + -6x_2 + 2x_3 &= -61.5 \\
x_1 + x_2 + 5x_3 &= -21.5
\end{aligned}
$$

Después, multiplique las matrices [L] y [U] resultantes para demostrar que se genera [A].
b) Emplee la descomposición LU para resolver el sistema. Realice todos los pasos del cálculo.
c) También resuelva el sistema para un vector alternativo del lado derecho:
B^T = [12 18 -6]

1. Solución por método de Gauss

```
import numpy

m=int(input("Numero de renglones:"))
n=int(input("Numero de columnas:"))

matrix = numpy.zeros((m,n))
vector= numpy.zeros((n))
x=numpy.zeros((m))

print("Ingresa la matriz y el vector solucion")

for r in range(0,m):
    for c in range(0,n):

        matrix[(r),(c)]=(input("ElementoM["+str(r+1)+","+
        str(c+1)+"]:"))

    vector[(r)]=(input("b["+str(r+1)+"]: "))

print(matrix)

for k in range (0,m):
    for r in range(k+1,m):

        factor=(matrix[r,k]/matrix[k,k])

        vector[r]=vector[r]-(factor*vector[k])

        for c in range(0,n):

            matrix[r,c]=matrix[r,c]-(factor*matrix[k,c])

#sustitucion hacia atras
x[m-1]=vector[m-1]/matrix[m-1,m-1]

print(x[m-1])

for r in range(m-2,-1,-1):
    suma=0
    for c in range(0,n):
        suma=suma+matrix[r,c]*x[c]

    x[r]=(vector[r]-suma)/matrix[r,r]

print("Matriz resultante\n", matrix, "\n")
print("Vector resultante\n", vector, "\n")
print("Soluciones al sistema: \n", x, "\n")
```

2. Descomposición LU

```python
import numpy as np

m = int(input("Dimension de la matriz: "))

#crear matrices con ceros
a = np.zeros([m,m])
l = np.zeros([m,m])
u = np.zeros([m,m])
b = np.zeros([m])
x = np.zeros([m]) #soluciones

#pedir datos de la matriz
for i in range(0,m):
    for j in range(0,m):
        a[i,j] = (input("Elemento a[" + str(i+1)
        + "," + str(j+1)+"]: "))
        a[i,j] = float(a[i,j])
        u[i,j] = a[i,j] #definimos matriz u
    b[i] = (input("Elemento b[" + str(i+1)+"]: "))

#fase de descomposicion
for k in range(0, m):
    for i in range(0, m):
        if (k==i):
            l[k,i]=1
        if(k<i):
            factor = (a[i,k] / a[k,k])
            l[i,k] = factor
            for j in range(0,m):
                a[i,j] = a[i,j] - (factor * a[k,j])
                u[i,j] = a[i,j]
#sustitucion:
   #hacia atras
x[m-1] = b[m-1]/a[m-1,m-1]

for i in range(m-2,-1,-1):
    sum = 0
    for j in range(0,m):
        sum = sum + a[i,j] * x[j]
    x[i] = (b[i] - sum)/a[i,i]

print("Matriz U\n", u)
print("Matriz L\n", l)
print("Soluciones al sistema: ", x)
```

3. El código para este apartado es el mismo que para el número anterior. Ir a la sección de anexo (3.3) y comprobar la diferencia en el resultado al cambiar el vector alternativo B^T.

2.2. Sistema de ecuaciones lineales (1)

Problema 10.3

Resuelva el siguiente sistema de ecuaciones mediante descomposición LU:

$$
\begin{aligned}
8x_1 + 4x_2 - x_3 &= 11 \\
-2x_1 + 5x_2 + x_3 &= 4 \\
2x_1 - x_2 + 6x_3 &= 7
\end{aligned}
$$

Determine la matriz inversa. Compruebe sus resultados por medio de verificar que $[\mathrm{A}][\mathrm{A}]^{-1} = [I]$

1. Descomposición LU con su matriz inversa

```
import numpy as np

m = int(input("Dimension de la matriz: "))
#crear matrices con ceros
a = np.zeros([m,m])
l = np.zeros([m,m])
u = np.zeros([m,m])

b = np.zeros([m])
x = np.zeros([m]) #soluciones
#pedir datos de la matriz
for i in range(0,m):
    for j in range(0,m):
        a[i,j] = (input("Elemento a[" + str(i+1) + "," +
        str(j+1)+"]: "))
        a[i,j] = float(a[i,j])
        u[i,j] = a[i,j] #definimos matriz u
    b[i] = (input("Elemento b[" + str(i+1)+"]: "))

#fase de descomposicion
for k in range(0, m):
    for i in range(0, m):
        if (k==i):
            l[k,i]=1
        if(k<i):
            factor = (a[i,k] / a[k,k])
            l[i,k] = factor

            for j in range(0,m):
                a[i,j] = a[i,j] - (factor * a[k,j])
                u[i,j] = a[i,j]
#sustitucion:
```

```
    #hacia atras
x[m-1] = b[m-1]/a[m-1,m-1]

for i in range(m-2,-1,-1):
    sum = 0
    for j in range(0,m):
        sum = sum + a[i,j] * x[j]
    x[i] = (b[i] - sum)/a[i,i]

#calculo de la inversa de A
u_inv = np.linalg.inv(u)
l_inv = np.linalg.inv(l)
a_inv = u_inv.dot(l_inv)

print("Matriz U\n", u)
print("Matriz L\n", l)
print("Inversa\n", a_inv)
```

2.3. Sistema de ecuaciones lineales (2)

Problema 10.4

Resuelva el siguiente sistema de ecuaciones mediante descomposición LU con pivote parcial:

$$\begin{aligned} -2x_1 - -6x_2 - x_3 &= -9.5 \\ -3x_1 - x_2 + 7x_3 &= -34 \\ -8x_1 + x_2 - 2x_3 &= -20 \end{aligned}$$

1. Descomposición LU con pivote

```
import numpy as np

a = np.array([[-2   ,-6,-1],[3, -1   ,7],[-8,1, -2]])
b = np.array([-9.5,-34,-20])
ab = np.copy(a)
    #pivote
size = np.shape(AB)
n = size[0]
m = size[1]

for i in range(0,n-1,1):
    column = abs(ab[i:,i])
    max = np.argmax(column)

# si el max no esta en la diagonal
    if (max !=0):
        #intercambia filas
        temp = np.copy(ab[i,:])
        ab[i,:] = ab[max+i,:]
        ab[max+i,:] = temp
ab1 = np.copy(ab)
U = np.copy(AB)

#sustitucion
   #hacia atras
x[m-1] = b[m-1]/a[m-1,m-1]
for i in range(m-2,-1,-1):
    sum = 0
    for j in range(0,m):
        sum = sum + a[i,j] * x[j]
    x[i] = (b[i] - sum)/a[i,i]
print("Matriz U", U )
print("Matriz L:", L)
```

2.4. Descomposición de Crout

Problema 10.7

Ejecute la descomposición de Crout sobre el sistema

$$\begin{aligned} 2x_1 - 5x_2 + x_3 &= 12 \\ -x_1 + 3x_2 - x_3 &= -8 \\ 3x_1 - 4x_2 + 2x_3 &= 16 \end{aligned}$$

Después multiplique las matrices [L] y [U] resultantes para determinar que se produce [A]

1. Descomposición Crout

```
import numpy as np

m,n = a.shape
#declarar matriz y vector
a = np.array([[2,-5,1],[-1,3,-1],[3,-4,2]])
b = np.array([12,8,16])

#inicializar l,u y sumas
l = np.zeros((n,n))
u = np.zeros((n,n))
s1, s2 = 0

#descomp. crout
    for i in range(n):
        l[i][0] = a[i][0]
        u[i][i] = 1
    for j in range(1, n):
        u[0][j] = a[0][j] / l[0][0]
    for k in range(1, n):
        for i in range(k, n):
            for r in range(k):
                s1 += l[i][r] * u[r][k]
                l[i][k] = a[i][k] - s1
                s1 = 0
        for j in range(k+1, n):
            for r in range(k):
                s2 += l[k][r] * u[r][j]
                u[k][j] = (a[k][j] - s2) / l[k][k]
                s2 = 0
#matrices de la descomp
    print("Matriz U\n", u)
    print("Matriz L\n", l)
```

```python

    y = np.zeros(n)
    s3 = 0
    y[0] = b[0] / l[0][0]
    for k in range(1, n):
        for r in range(k):
            s3 += l[k][r] * y[r]
        y[k] = (b[k]-s3) / l[k][k]
        s3 = 0

    x = np.zeros(n)
    s4 = 0
    x[n-1] = y[n-1]
    for k in range(n-2, -1, -1):
        for r in range(k+1, n):
            s4 += u[k][r] * x[r]
        x[k] = y[k] - s4
        s4 = 0

    for i in range(n):
        print("x" + str(i + 1) + " = ", x[i])
    print("x" " = ", x)
```

2.5. Interfaz gráfica

Problema 10.19

Realice un programa amigable para el usuario para calcular la descomposición LU, que incluya la capacidad de evaluar la matriz inversa.

1. Interfaz del programa con módulo Tkinter

```
import numpy as np
from tkinter import *

### Funcion

def LU():
    a = np.array( [ [float(a11.get()), float(a12.get()),
    float(a13.get())], [float(a21.get()), float(a22.get()),
    float(a23.get())], [float(a31.get()), float(a32.get()),
    float(a33.get())] ] )

    b = np.array([float(b1.get()), float(b2.get()),
    float(b3.get())])

    #fase de descomposicion
    for k in range(0, m):
        for i in range(0, m):
            if (k==i):
                l[k,i]=1
            if(k<i):
                factor = (a[i,k] / a[k,k])
                l[i,k] = factor

        for j in range(0,m):
            a[i,j] = a[i,j] - (factor * a[k,j])
            u[i,j] = a[i,j]
#sustitucion:
   #hacia atras
    x[m-1] = b[m-1]/a[m-1,m-1]

    for i in range(m-2,-1,-1):
        sum = 0

        for j in range(0,m):
            sum = sum + a[i,j] * x[j]

    x[i] = (b[i] - sum)/a[i,i]

    print("Matriz U\n", u)
```

```
    print("Matriz L\n", l)

    print("Soluciones al sistema: ", x)

def Inversa():
#calculo de la inversa de A
    a = np.array( [ [float(a11.get()), float(a12.get()),
    float(a13.get())], [float(a21.get()), float(a22.get()),
    float(a23.get())], [float(a31.get()), float(a32.get()),
    float(a33.get())] ] )

    b = np.array([float(b1.get()), float(b2.get()),
    float(b3.get())])

    a_inv = np.linalg.inv(a)
    b_inv = np.linalg.inv(b)
    mat_inv = u_inv.dot(l_inv)

    print("Inversa\n", mat_inv)
### Interfaz ####
window = Tk()
window.title('Factorizacion LU')
window.config(padx=70, pady=70)

a_label = Label(text="Matriz A", font=("Inconsolata Bold", 16,
"bold"))
a_label.grid(column=0, row=0, columnspan=3)

b_label = Label(text="Vector b", font=("Inconsolata Bold", 16,
"bold"))
b_label.grid(column=3, row=0)

a11 = Entry(width=5)
a11.grid(column=0, row=1)

a12 = Entry(width=5)
a12.grid(column=1, row=1)

a13 = Entry(width=5)
a13.grid(column=2, row=1)

a21 = Entry(width=5)
a21.grid(column=0, row=2)

a22 = Entry(width=5)
a22.grid(column=1, row=2)

a23 = Entry(width=5)
```

```
a23.grid(column=2, row=2)

a31 = Entry(width=5)
a31.grid(column=0, row=3)

a32 = Entry(width=5)
a32.grid(column=1, row=3)

a33 = Entry(width=5)
a33.grid(column=2, row=3)

b1 = Entry(width=7)
b1.grid(column=3, row=1)

b2 = Entry(width=7)
b2.grid(column=3, row=2)

b3 = Entry(width=7)
b3.grid(column=3, row=3)

boton = Button(text="Calcular", width=20, font=("Incosolata Bold"
,14,"bold"), bg="red", command=LU)
boton.grid(column=0, row=4, columnspan=4)

boton1 = Button(text="Inversa", width=20, font=("Incosolata Bold"
,14,"bold"), bg="blue", command=Inversa)
boton1.grid(column=0, row=5, columnspan=4)

window.mainloop()
```

CAPÍTULO 3

ANEXOS

3.1. Anexo A: Evidencia del funcionamiento de los código reportados

3.1.1. Ejercicio 10.2

```
Matriz resultante
 [[10.          2.          -1.        ]
 [ 0.         -5.4         1.7        ]
 [ 0.          0.          5.35185185]]

Vector resultante
 [ 27.          -53.4         -32.11111111]

Soluciones al sistema:
 [ 0.5  8.  -6. ]
```

Figura 3.1: Método de Gauss simple

```
Matriz U
 [[10.          2.          -1.        ]
 [ 0.         -5.4         1.7        ]
 [ 0.          0.          5.35185185]]
Matriz L
 [[ 1.          0.          0.        ]
 [-0.3         1.          0.        ]
 [ 0.1        -0.14814815  1.        ]]
Soluciones al sistema:  [ 0.27343329 10.12418301 -4.01730104]
```

Figura 3.2: Descomposición LU

```
Matriz U
 [[10.   2.  -1. ]
 [ 0.   2.6 -1.3]
 [ 0.   0.   5.5]]
Matriz L
 [[ 1.          0.          0.        ]
 [-0.3         1.          0.        ]
 [ 0.1         0.30769231  1.        ]]
Soluciones al sistema:  [-0.18461538  6.37762238 -1.09090909]
```

Figura 3.3: Descomposición LU con vector alternativo

3.1.2. Ejercicio 10.3

```
Matriz U
 [[ 8.    4.    -1.   ]
 [ 0.    6.     0.75]
 [ 0.    0.     6.5 ]]
Matriz L
 [[ 1.          0.          0.        ]
 [-0.25        1.          0.        ]
 [ 0.25       -0.33333333  1.        ]]
Inversa
 [[ 0.09935897 -0.07371795  0.02884615]
 [ 0.04487179  0.16025641 -0.01923077]
 [-0.02564103  0.05128205  0.15384615]]
```

Figura 3.4: Descomposición LU e inversa de la matriz

3.1.3. Ejercicio 10.4

```
Matriz U
 [[ -2.       -6.       -1.    ]
 [  0.        8.        8.5   ]
 [  0.        0.      -24.5625]]
Matriz L
 [[1.    0.    0.   ]
 [1.5   1.    0.   ]
 [4.    3.125 1.   ]]
Soluciones al sistema:  [33.93829517 -5.11513995  0.81424936]
```

Figura 3.5: Descomposición LU con pivote

3.1.4. Ejercicio 10.7

```
Matriz U
 [[ 1.  -2.5  0.5]
 [ 0.   1.  -1. ]
 [ 0.   0.   1. ]]
Matriz L
 [[ 2.   0.   0. ]
 [-1.   0.5  0. ]
 [ 3.   3.5  4. ]]
x1 =  26.0
x2 =  3.0
x3 =  -25.0
x = [ 26.   3. -25.]
```

Figura 3.6: Descomposición de Crout

3.1.5. Ejercicio 10.19

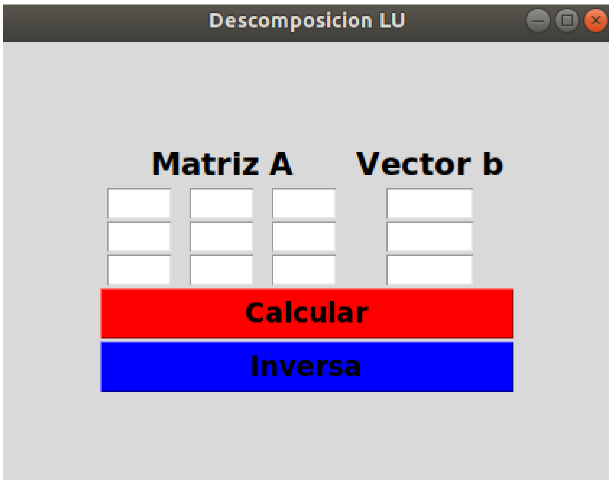

Figura 3.7: Interfaz gráfica para calcular la descomposición LU y la inversa

CAPÍTULO 4

BIBLIOGRAFÍA

1. Chapra, S. C., Canale, R. P. (2011). *Numerical methods for engineers* (Vol. 1221). New York: Mcgraw-hill.