Centro de Investigación en Matemáticas
Maestría en Computo Estadístico

Comparación de método de solución de sistemas lineales

Alumno:

Ing. Zavala Cordero Ricardo Alfredo

Profesor:

Dr. Ángel David Reyes Figueroa

Índice

1. Introducción

La solución de sistemas de ecuaciones es un problema que ha estado presente desde hace mucho tiempo, antes del uso de computadoras para su solución el procedimiento mas sencillo pareciera ser aquel que implique un menor número de operaciones, sin embargo con el uso de computadoras para soluciones cada vez mas grandes y complejos parece no ser esta siempre la mejor alternativa, ya que los distintos programas de solución tiene tiempo y formas distintas de interación con la memoria, lo cual implica no solo tiempo por operaciones si no tambien el tiempo de busqueda de los elementos. Por lo anterior mencionado a continuación se muestra la comparación de los métodos de solución:

- Gauss-Jordan
- LU
- Inversa

Se realizará la implementación de los algoritmos antes mencionados en python, y se compararan los tiempos de realización de cada método, para así decir entre estos cual es el mejor.

2. Marco teórico

2.1. Solución por Gauss-Jordan

Sea el sistema de ecuaciones de forma matricial

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

se utiliza la matriz aumentada

$$\left(\begin{array}{cccc|c} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & b_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} & b_n \end{array}\right)$$

Se divide la primera fila entre la primer componente

$$\left(\begin{array}{cccc|c} 1 & \frac{a_{12}}{a_{11}} & \cdots & \frac{a_{1n}}{a_{11}} & \frac{b_1}{a_{11}} \\ a_{21} & a_{22} & \cdots & a_{2n} & b_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} & b_n \end{array}\right)$$

Se elimina con operaciones entre filas los elementos abajo del elemento 1,1, obteniendo

$$\left(\begin{array}{cccc|c} 1 & \frac{a_{12}}{a_{11}} & \cdots & \frac{a_{1n}}{a_{11}} & \frac{b_1}{a_{11}} \\ 0 & a_{22} - a_{21}\frac{a_{12}}{a_{11}} & \cdots & a_{2n} - a_{21}\frac{a_{1n}}{a_{11}} & b_2 - a_{21}\frac{b_1}{a_{11}} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & a_{n2} - a_{n1}\frac{a_{12}}{a_{11}} & \cdots & a_{nn} - a_{n1}\frac{a_{1n}}{a_{11}} & b_n - a_{n1}\frac{b_1}{a_{11}} \end{array}\right)$$

Repitiendo este procedimiento para los elementos i, i de la matriz con $i \in [1, \ldots, n]$ se obtiene la matriz

$$\left(\begin{array}{cccc|c} 1 & c_{12} & \cdots & c_{1n} & d_1 \\ 0 & 1 & \cdots & c_{2n} & d_2 \\ \vdots & \vdots & \ddots & \vdots & \\ 0 & 0 & \cdots & 1 & d_n \end{array}\right)$$

Realizando el mismo proceso de eliminación pero para los elementos por arriba de la diagonal, se obtiene la solución al sistema, el cual queda expresado de la forma

$$\left(\begin{array}{cccc|c} 1 & 0 & \cdots & 0 & r_1 \\ 0 & 1 & \cdots & 0 & r_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & r_n \end{array}\right)$$

2.2. Solución por factorización LU

Teorema Sea A $\in \mathcal{M}_{n\times n}$ y suponga que A se puede reducir por renglones a una matriz triangular U sin hacer alguna permutación entre sus renglones. Entonces existe una matriz triangular inferior L invertible con unos en la diagonal tal que $A = LU$. Si, ademas, U tiene n pivotes (es decir, A es invertible), entonces esta factorización es única.
Sea el sistema de forma matricial

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

Haciendo la factorización LU, de A el sistema se representa de la forma

$$\begin{pmatrix} 1 & 0 & \cdots & 0 \\ l_{21} & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & \cdots & 1 \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ 0 & u_{22} & \cdots & u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & u_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

Sea $y = ux$ queda el sistema

$$\begin{pmatrix} 1 & 0 & \cdots & 0 \\ l_{21} & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & \cdots & 1 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

Se resuelve el sistema $LY = b$, despejando los valores de Y, de la forma

$$\begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 - l_{21} \\ b_3 - \sum_{i=1}^{2} l_{3i} \\ \vdots \\ b_n - \sum_{i=1}^{n-1} l_{ni} \end{pmatrix}$$

Encontrados valores se da solución al sistema

$$\begin{pmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ 0 & u_{22} & \cdots & u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & u_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$$

Despejando de este sistema, la solución del sistema original es

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} \frac{y_1 - \sum_{i=2}^{n} u_{1i}}{u_{11}} \\ \frac{y_2 - \sum_{i=3}^{n} u_{2i}}{u_{22}} \\ \frac{y_3 - \sum_{i=4}^{n} u_{3i}}{u_{33}} \\ \vdots \\ \frac{y_n}{u_{nn}} \end{pmatrix}$$

2.3. Solución de la matriz inversa

Sea el sistema de forma matricial

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

Si la inversa de la matriz A existe, entonces el sistema

$$Ax = b$$

tiene solución única y es obtenida al multiplicar por la izquierda ambos lados de la ecuación

$$\begin{aligned} A^{-1}Ax &= A^{-1}b \\ Ix &= A^{-1}b \\ x &= A^{-1}b \end{aligned}$$

Las soluciones del sistema estan dadas por $x = A^{-1}b$

3. Desarrollo

El programa fue realizaco en python 3.9 con las librerias

- math: para algunas operaciones aritmeticas
- random: Generación de números aleatorios para las matrices
- time: Utilizada para medir el tiempo de ejecución
- Statistics: Obtener media y desviación estándar de los tiempos
- numpy: Calculo del determinante para conocer si el sistema tiene solución

Para conocer los códigos realizados puede consultar el archivo llamado comparación, en este mismo repositorio o ir al apendice a consultarlo.
Para la realización experimental se considera una muestra de 100 sistemas lineales donde se tiene 100 ecuaciones con 100 incognitas, donde se encontraron los siguientes resultados

Método	Media	Desviación estándar
Gauss-Jordan	2.004	0.014
LU	3.033	0.425
Inversa	5.627	0.209

De los resultados obtenidos se puede ver que el resultado menos conveniente para calcular las soluciones de un sistema de ecuaciones lineales es por el método de la inversa, esto es ya que el método utilizado para calcular la inversa de la matriz es por reducción gaussiana es decir aplicando las mismas operaciones que en el primer método pero a dos matrices, se realizan el doble de operaciones, mas aparte la multiplicación de la inversa por el vector resultado en un número operaciones mucho mayor, por lo tanto un tiempo mucho mayor. Se puede ver que el promedio de las medias para el método LU es es mayor que para el método Gauss-Jordan, lo cual aparenta ser significante ,pero ¿Es este significativo?, por los valores obtenidos, podemos decir que el tiempo promedio en realizar la solución por el método LU es mayor, por medio de una prueba de hipotesis, demostraremos que esto es así

$$\begin{aligned} H_0 &: \bar{\mu}_{GJ} - \bar{\mu}_{LU} \geq 0 \\ H_a &: \bar{\mu}_{GJ} - \bar{\mu}_{LU} < 0 \end{aligned}$$

El estadistico de prueba de hipotesis para una diferencia de medias es
El estadistico es

$$\frac{\bar{t_{GJ}} - \bar{t_{LU}} - (\mu_{GL} - \mu_{LU})}{\sqrt{\frac{s_{GJ}^2}{n_{GJ}} + \frac{s_{LU}^2}{n_{LU}}}} \sim N(0,1)$$

Considerando un nivel de significancia del 95 %, el valor $\alpha = 0{,}05$, asi el punto critico considerando una prueba de cola izquierda es

$$Z_{0,05} = -1{,}64$$

Calculando el estadistico

$$\frac{\bar{t_{GJ}} - \bar{t_{LU}} - (\mu_{GL} - \mu_{LU})}{\sqrt{\frac{s_{GJ}^2}{n_{GJ}} + \frac{s_{LU}^2}{n_{LU}}}} =$$

$$\frac{2{,}0004 - 2{,}033 - 0}{\sqrt{\frac{0{,}014^2}{100} + \frac{0{,}425^2}{100}}} = -24{,}20$$

Por lo tanto rechazamos la hipotesis nula, es decir rechazamos que el tiempo de solución un sistema lineal de 100 ecuaciones con 100 incognitas programado en python, por el método de Gauss-Jorda, sea mayor o igual al tiempo de solución por el método LU, bajo los algoritmos planteados.

4. Conclusiones

Los métodos de solución de ecuaciones lineales son distintos algunos necesitan la implementación implicita de otros para dar solución a estos, como es el caso de la inversa, el cual mostro no ser la mejor opción cuando la inversa es calculada por el método Gauss Jordan. El método que obtuvo mejores resultados, fue el método gauss-jordan ya que los demas métodos necesitan de la implementación indirecta de este, es decir se mostro que en python 3.9, bajo los algoritmos descritos en este trabajo, e implemetados de la forma mostrada en el apendice, la mejor opción para la solución de un sistema de ecuaciones lineales es el método Gauss-Jordan ya que implica un tiempo promedio menor considerable a diferencia de métodos.

5. Apendice

5.1. Método Gauss-Jordan

```
def solucion_gj(A,b):
  n=len(A)
  aux=A[0][0]
  A[0][0]=1
  #Division de la primera fila entre el primer elemento
  for i in range(1,n):
      A[0][i]=A[0][i]/aux
  b[0]=b[0]/aux
  #Llevando matriz A a una matriz triangular superior
  for i in range(n-1):
      for k in range(i+1,n):
          aux=A[k][i]
          for j in range(n):
              A[k][j]=A[k][j]-aux*A[i][j]/A[i][i]
          b[k]=b[k]-aux*b[i]/A[i][i]
      aux=A[i+1][i+1]
      for j in range(i+1,n):
          A[i+1][j]=A[i+1][j]/aux
      b[i+1]=b[i+1]/aux
  #Realizando las operaciones para realizar una matriz diagonal a la
  #matriz A resultante del paso anterior, pero con operaciones solo al vector
  #para reducir tiempos
  for i in range(n-1,-1,-1):
      for j in range(i,0,-1):
          aux=A[j-1][i]
          b[j-1]=b[j-1]-aux*b[i]
  return b
```

5.2. Método LU

```
def solucion_LU(A,b):
    n=len(A)
    l=[]
    #Generación de la matriz L
    for i in range(n):
        aux=[]
        for j in range(n):
            aux.append(0)
        l.append(aux)
    #Realizanción de factorización LU
    for i in range(n):
        for j in range(i,n):
            if i==j:
                l[j][i]=1
            else:
                l[j][i]=A[j][i]/A[i][i]
        if i<n-1:
            for k in range(i+1,n):
                aux=A[k][i]
                for j in range(n):
                        A[k][j]=A[k][j]-aux*A[i][j]/A[i][i]
    #Realizando solución del Sistema LY=B
    for i in range(1,n):
        for j in range(0,i):
            b[i]=b[i]-l[i][j]*b[j]
    #Realizando solución del sistema UX=Y
    for i in range(n-1,-1,-1):
        for j in range(n-1,i,-1):
            b[i]=b[i]-A[i][j]*b[j]
        b[i]=b[i]/A[i][i]
    return b
```

5.3. Método Inversa

```
def solucion_inversa(A,b):
    n=len(A)
    inversa=[]
    #Generando la matriz identidad
    for i in range(n):
        aux=[]
        for j in range(n):
            if i==j:
                aux.append(1)
            else:
                aux.append(0)
        inversa.append(aux)
    #Calculando la matriz inversa por el método Gauss-Jordan
    for i in range(n):
        for j in range(i+1,n):
            aux=A[j][i]
            for k in range(n):
                A[j][k]=A[j][k]-aux*A[i][k]/A[i][i]
                inversa[j][k]=inversa[j][k]-aux*inversa[i][k]/A[i][i]
    for i in range(n):
        aux=A[i][i]
        for j in range(n):
            A[i][j]=A[i][j]/aux
            inversa[i][j]=inversa[i][j]/aux
    for i in range(n-1,-1,-1):
        for j in range(i,0,-1):
            aux=A[j-1][i]
            for k in range(n):
                inversa[j-1][k]=inversa[j-1][k]-aux*inversa[i][k]
    sol=[]
    #Multiplicación de la inversa por el vector b
    for i in range(n):
        a=0
        for j in range(n):
            a+=inversa[i][j]*b[j]
        sol.append(a)
    return sol
```

5.4. Función de pruebas

```
def pruebas(repeticiones,dimensiones):
    tiempo_gj=[]             #Lista para guarda los tiempos de ejecuación de metodo
    aux=0                    #Variable auxiliar en caso de error de computo
    for i in range(repeticiones+aux):
        A=gen_A(dimensiones) #Obteniendo matriz de tamaño dimensiones x dimensiones
        b=gen_b(dimensiones) #Generando vector b de  tamaño dimensiones
        try:
            inicio=time()        #Registrado tiempo donde inicial el método
            c=solucion_gj(A,b)   #Ejecución del método
            final=time()         #Registrando timmpo donde termina el método
            tiempo_gj.append(final-inicio) #Calculando tiempo de duración  y guardando ti
        except:
            aux+=1
    aux=0
    tiempo_lu=[]             #Lista para guarda los tiempos de ejecuación de metodo
    for i in range(repeticiones+aux):
        A=gen_A(dimensiones) #Obteniendo matriz de tamaño dimensiones x dimensiones
        b=gen_b(dimensiones) #Generando vector b de  tamaño dimensiones
        try:
            inicio=time()        #Registrado tiempo donde inicial el método
            c=solucion_LU(A,b)   #Ejecución del método
            final=time()         #Registrando timmpo donde termina el método
            tiempo_lu.append(final-inicio) #Calculando tiempo de duración  y guardando ti
        except:
            aux+=1
    aux=0
    tiempo_inversa=[]             #Lista para guarda los tiempos de ejecuación de metodo
    for i in range(repeticiones+aux):
        A=gen_A(dimensiones) #Obteniendo matriz de tamaño dimensiones x dimensiones
        b=gen_b(dimensiones) #Generando vector b de  tamaño dimensiones
        try:
            inicio=time()        #Registrado tiempo donde inicial el método
            c=solucion_inversa(A,b)   #Ejecución del método
            final=time()         #Registrando timmpo donde termina el método
            tiempo_inversa.append(final-inicio) #Calculando tiempo de duración  y guardar
        except:
            aux+=1
    return [tiempo_gj,tiempo_lu,tiempo_inversa]
```

5.5. Función generadora de matrices

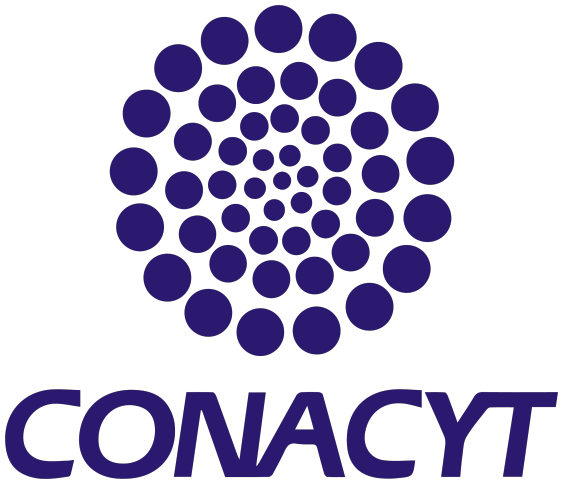
```
def gen_A(n):
    d=0
    while d==0: #Verificación para que el sistema solución
        c=[]
        #Llenando los elementos de la matriz
        for i in range(n):
            aux=[]
            for i in range(n):
                aux.append(randint(0,1000000))
            c.append(aux)
        c= np.array(c)
        #Calculando determinante para conocer si el sistema con esta matriz
        #tiene solución
        d=np.linalg.det(c)
    return c
```

5.6. Función generadora de vectores

```
def gen_b(n):
    d=[]
    #Generación de vector vector b
    for i in range(n):
        d.append(randint(0,1000000))
    return d
```

Referencias

[1] Grossman, Stanley I. Algebra Lineal. México, D.F., Mcgraw-Hill, 2012.


CONACYT