Plantación de Patetse

Antón Riveiro Alonso, Laura Silva Loureiro, Tom Willemse Ríos, Isaac Noya Vázquez

Cálculo y Análisis Numérico

Grupo Miércoles

{anton.riveiro, laura.silva.loureiro, tom.willemse, isaac.noya}@rai.usc.es

ÍNDICE

Presentación del problema

El objetivo es encontrar el punto óptimo para construir una plantación de patatas. Este debe cumplir ciertos requisitos de viento, humedad y exposición al sol. Estas características están parametrizadas por las ecuaciones que forman nuestra matriz, siendo x, y, z las coordenadas en el espacio.

- **Matriz y sistema de ecuaciones:**

El viento, la humedad y la exposición al sol están parametrizadas bajo estas ecuaciones, siendo (x, y, z) las coordenadas en el espacio.

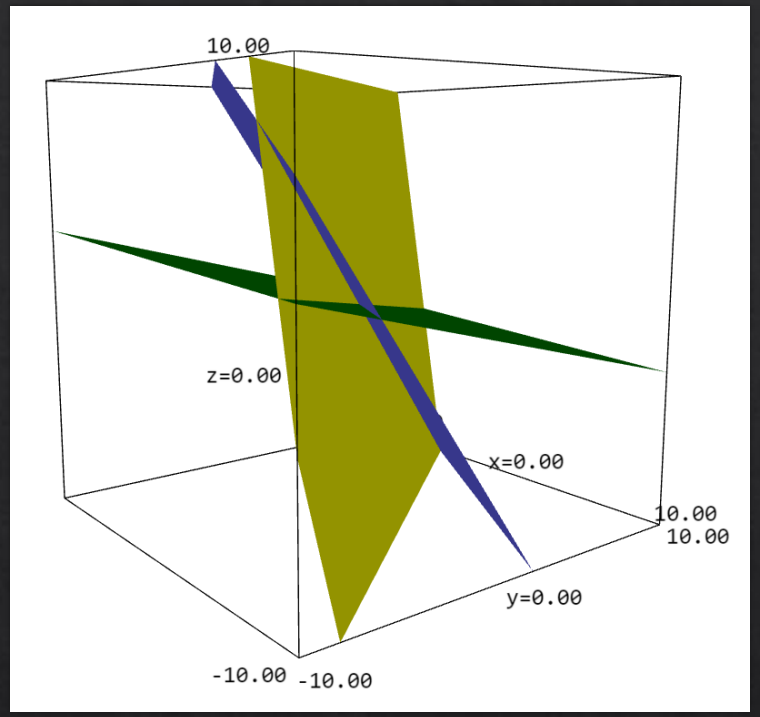
$$5x + 4y + 1z = 7$$
$$4x + 7y + 5z = 1$$
$$x + 5y + 20z = 2$$

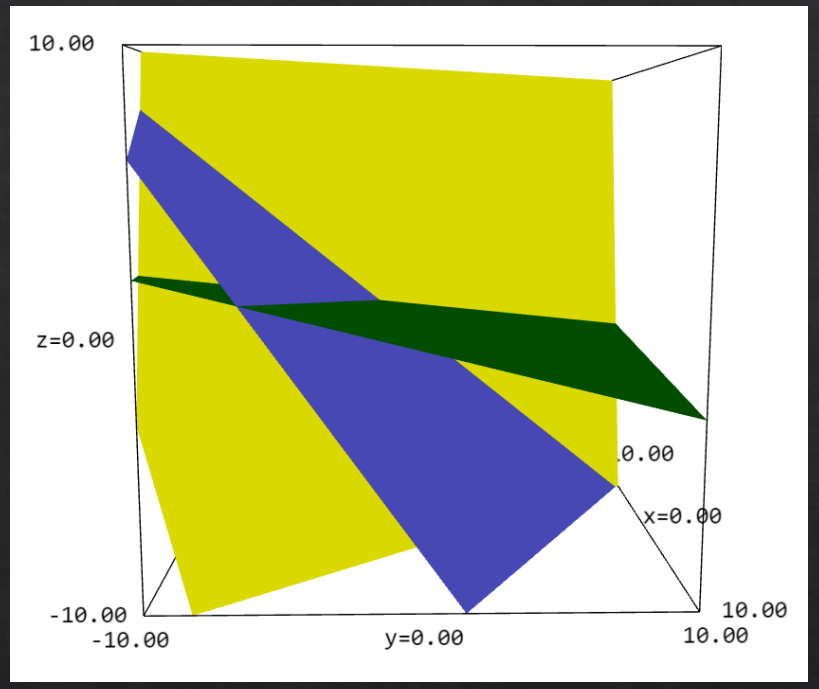
Matriz y sistema de ecuaciones

El viento, la humedad y la exposición al sol están parametrizadas bajo estas ecuaciones, siendo (x, y, z) las coordenadas en el espacio.

$$
\begin{aligned}
5x + 4y + 1z &= 7 \\
4x + 7y + 5z &= 1 \\
x + 5y + 20z &= 2
\end{aligned}
$$

Matriz y sistema de ecuaciones






```
x, y, z = var('x y z')
h1 = lambda x,y,z: 5*x + 4*y + z - 7
h2 = lambda x,y,z: 4*x + 7*y + 5*z - 1
h3 = lambda x,y,z: x + 5*y + 20*z - 2
args = [(x, -10, 10), (y, -10, 10), (z, -10, 10)]
```

```
g1 = implicit_plot3d(h1, *args, color="yellow")
g2 = implicit_plot3d(h2, *args)
g3 = implicit_plot3d(h3, *args, color="green")
g1 + g3 + g2
```

Comandos en sage creación de la matriz

```
Sage:
A = matrix(RDF,[[5, 4, 1], [4, 7, 5], [1, 5, 20]])
b = matrix([[7], [1], [2]])
Ab = matrix([[5, -4, 20, 7], [4, 7, 5, 1], [0, 5, -1, 2]])
```

$$\begin{pmatrix} 5 & -4 & 20 & 7 \\ 4 & 7 & 5 & 1 \\ 0 & 5 & -1 & 2 \end{pmatrix}$$

Rango y comprobación de que es simétrica

```
>>rank(A)
3
>>rank(Ab)
3
>>transpose(A) == A
True
```

Comprobación de que es definida positiva

```
>>A.eigenvalues()
[1.550043437771786?, 8.442742734473602?, 22.007213827754627?]
>>s1 = matrix([[A[0, 0], A[0, 1]], [A[1, 0], A[1, 1]]])
>>s2 = matrix([[A[1, 0], A[1, 1]], [A[2, 0], A[2, 1]]])
>>s3 = matrix([[A[0, 1], A[0, 2]], [A[1, 1], A[1, 2]]])
>>s4 = matrix([[A[1, 1], A[1, 2]], [A[2, 1], A[2, 2]]])
>>det(s1)>0 and det(s2)>0 and det(s3)>0 and det(s4)>0
True
```

INVERSA

Podemos resolver el sistema haciendo:

$$A \cdot x = b$$

$$x = A^{-1} \cdot b$$

Siempre que A sea invertible, es decir:

$$\exists A^{-1} \in \mathbb{R}^{n \times n} / A^{-1} \cdot A = I$$

$$|A| \neq 0$$

$$rg(A) = n$$

$$0 \notin \sigma(A)$$

COMPROBACIONES

```
A = matrix([[5, 4, 1], [4, 7, 5], [1, 5, 20]])
A.det()
OUT: 288
[i.n(digits=4) for i in A.eigenvalues()]
OUT:  [1.550, 8.443, 22.01]
rank(A)
OUT: 3
A*A^-1
OUT:[1 0 0]
    [0 1 0]
    [0 0 1]
```

Matriz Inversa

La matriz inversa A^-1:

$$\begin{bmatrix} 115/288 & -25/96 & 13/288 \\ -25/96 & 11/32 & -7/96 \\ 13/288 & -7/96 & 19/288 \end{bmatrix}$$

MÉTODO DE GAUSS

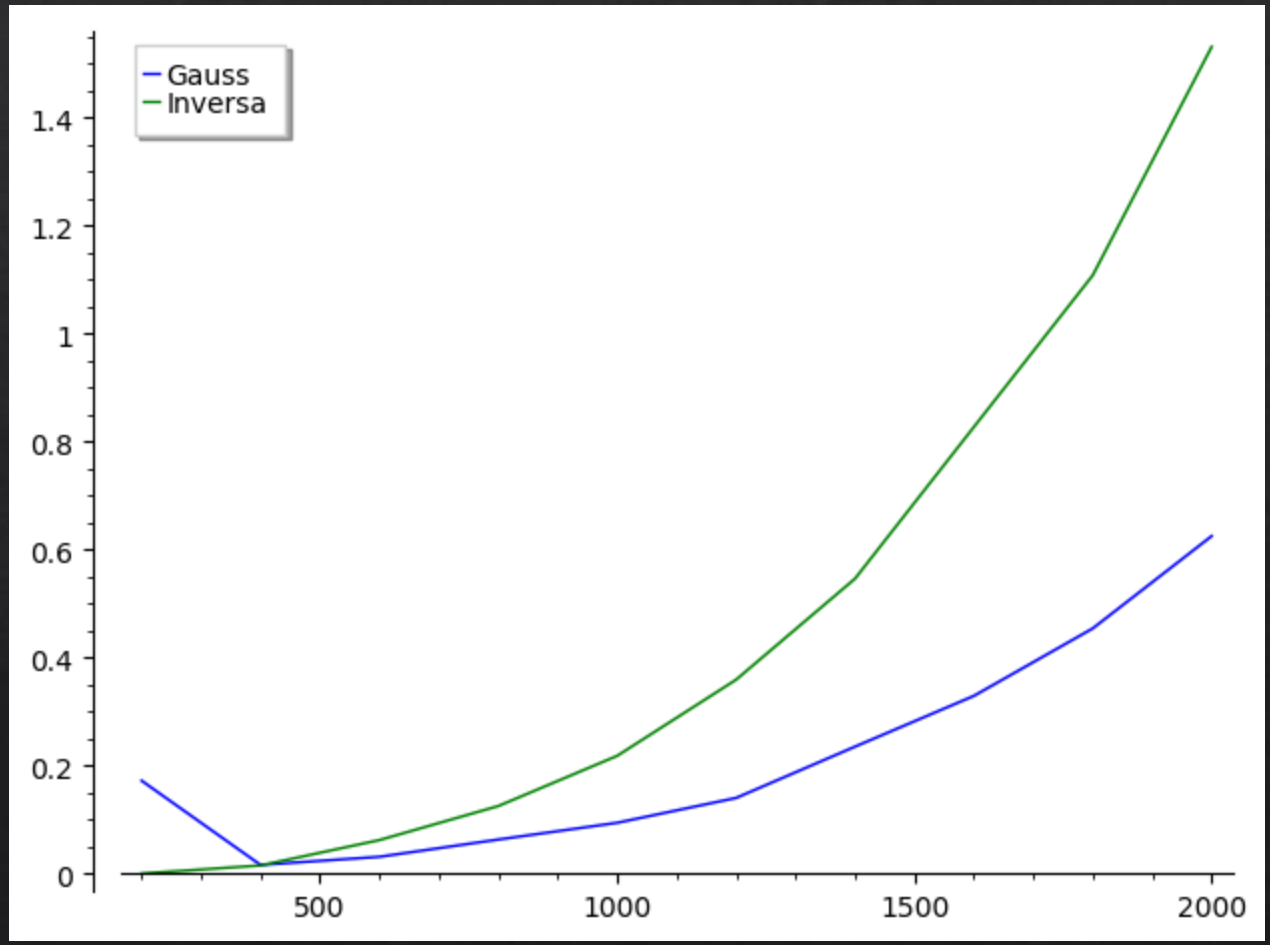
- Aplicamos transformaciones elementales para obtener un sistema equivalente de la forma:

$$U \cdot x = \tilde{b}$$

-U es una matriz triangular superior (tiene 0 debajo de la diagonal superior)

- En la siguiente diapositiva se muestra una comparción del coste operacional de los métodos de la inversa y de Gauss.

INVERSA VS MÉTODO DE GAUSS



FACTORIZACIÓN LU PARA EL M.E.G

- Si podemos aplicar el M.E.G, entonces:

$$A = LU,\ LU \in \mathbb{M}_{n\times n}$$

1. U matriz triangular superior obtenida después de la eliminación

2. L es la matriz triangular inferior formada por los multiplicadores

$$Ax = b \leftrightarrow L(Ux) = b \leftrightarrow \begin{cases} Ly = b \\ Ux = y \end{cases}$$

Coste resolución: $2n^2$

Coste factorización (hacer 1 y 2): $\frac{2}{3} \cdot n^3$

Costo total: $2n^2 \ll \frac{2}{3} \cdot n^3$

Factorización LU (solución)

```
x = U\(L\(P*b)) #y=L(P*b)
x.n(digits=3)
```

```
[ 2.62]
[-1.62]
[0.375]
```

Factorización LU (residuo)

```
x = U\(L\(P*b)) #y=L(P*b)
#residuo
norm(b-A*x)
```

```
0.0
```

Coste computacional

Facotrización de Cholesky

- Este método solo es aplicable para casos en los que la matriz A del sistema lineal Ax = b que queremos resolver es simétrica y definida positiva.

  - Sabemos que una matriz es simétrica cuando esta coincide con su traspuesta

  - Sabemos que una matriz es definida positiva cuando todos sus menores principales son positivos.

- En Sage existen ya unos comandos para verificar estas propiedades:

  >>A.is_symmetric() and A.is_positive_definite()

  Out: True

- Por lo tanto, podemos realizar el sistema lineal por Cholesky

Sistema lineal por Cholesky

- Siguiendo el método de Cholesky, podemos factorizar nuestra matriz A en: A = LLt
- No se necesita niguna estrategia de pivoteo, por lo que obtenemos el siguiente sistema:

$$Ax = b \iff LL^t x = b \iff \begin{cases} Ly = b \\ \\ L^t x = b \end{cases}$$

- Como solo necesitamos obtener la matriz L a partir de la de A, el costo operacional para obtener la factorización es de: $\frac{1}{3}n^3$. Por otra parte, la resolución del sistema de ecuaciones, como ya hemos visto, conlleva un coste de: $2n^2$
- El coste operacional del método de Cholesky es, por lo tanto, la mitad que en el M.E.G.

Comandos Sage para Cholesky

```
>>L = A.cholesky()

    L.n(digits = 2)
```

$$L = \begin{pmatrix} 2.2 & 0.0 & 0.0 \\ 1.8 & 1.9 & 0.0 \\ 0.45 & 2.2 & 3.9 \end{pmatrix}$$

```
>> x = L.transpose()\(L\b)
    x
        [2.6249999999999996]
  Out:  [-1.6249999999999996]
        [0.37499999999999983]
```

Propagación de errores

```
n = 3
A = matrix([[5, 4, 1], [4, 7, 5], [1, 5, 20]])
dA = matrix(RDF,n,n, [1e-2*(2*random()-1) for j in range((n)^2)])
A1 = A+dA  # matriz con errores
b = vector([7, 1, 2])
db = vector(RDF, [1e-6*(2*random()-1) for j in range(len(b))])
b1 = b + db # vector con errores
#Resolvemos ambos sistemas para obtener sus soluciones:
x = A\b
x1 = A1\b1
```

```
err_b = norm(b1-b)/norm(b)
err_A = norm(A-A1)/norm(A)
err_x = norm(x-x1)/norm(x)
err_A, err_b, err_x
```

```
(0.0006635392767215572,
 (6.31571410773923e-08)*sqrt(6),
 (9.656982121695038e-05)*sqrt(619))
```

Propagación de errores

```
show(A.condition(2)*err_A<1)
show(err_x<=(A.condition(2)/(1-A.condition(2)*err_A))*(err_A+err_b))
```

True

$$0.0018630194709608598 \leq \left(1.0563078188887012 \times 10^{-06}\right) \sqrt{6} + 0.00841115256130845$$

```
err_x <=  A.condition(2)*err_A/(1-A.condition(2)*err_A)
```

```
True
```

Conclusión

Una vez realizados los dos métodos conseguimos los siguientes resultados:

Factorización LU:

- Coste operacional: $\frac{2}{3}n^3$ flops.
- Resultados:

Cholesky:

- Coste operacional: $\frac{1}{3}n^3$ flops
- Resultados:

Podemos observar que ambos métodos llegan al mismo resultado. Sin embargo, el coste operacional en el método de Cholesky es la mitad que si utilizamos la factorización LU, por lo que es más apropiado si queremos obtener soluciones para n muy grande.

Bibliografía

- SageMath documentation. [en línea], [sin fecha]. [Consulta: 30 Abril 2022]. Disponible en: https://doc.sagemath.org/.