

Pràctica 1 de matemàtiques

A l'arxiu Practica-1.pdf s'explica en què consisteix l'exercici i quins són els criteris d'avaluació.

Integrants del grup: Isaac Digón Donaire

El directori d'entrega consisteix en una carpeta on hi consten dos fitxers de text per a introduir i llegir matrius, dos pdf per a explicar l'exercici (mentonat prèviament) i per a mostrar exemples d'ús del codi, i finalment dos arxius de codi en llenguatge julia (.jl), un proporcionat pel professor per a generar matrius (generator.jl) i un altre anomenat p1.jl que és l'exercici en si, programat i comentat seguint una organització caòtica escollida per mi.

A continuació els exemples resolts per l'algoritme (amb matrius de 2,3 i 4 de dimensió):

```
Generation-----
Your random 2 x 2 matrix is
2x2 Matrix{Int32}:
 6  1
 1  0
Your random vector is
[9, -2]
Your amplified matrix is
2x3 Matrix{Int32}:
 6  1  9
 1  0 -2
Calibration-----
2x3 Matrix{Int32}:
 6  1  9
 1  0 -2
Triangulation-----
Fila2 = 6 * Fila2 - 1 * Fila1
2x3 Matrix{Int32}:
 6  1  9
 0 -1 -21
end
end
end
show(stdout, "text/plain", M)
println("\n\nBacktracking-----")
amp=M[i,n[i]]
```

```

Your triangulated matrix is
2x3 Matrix{Int32}:
 6  1  9
 0 -1 -21
Calibration-----
2x3 Matrix{Int32}:
 6  1  9
 0 -1 -21
Backtracking-----
amp=M[i,n[i]]
result[2] = -21 / -1 = 21.0
result[1] = (9 - (1 * 21.0) - (6 * 0.0)) / 6 = -2.0
result [-2.0, 21.0]
x=A\b = [-2.0, 21.0] (using julia)
Process returned 0 (0x0)   execution time : 2.222 s
Presione una tecla para continuar . . . result[i] = (, amp[i])
```

```

Generation-----
1: random()
// generate()
Your random 3 x 3 matrix is
3x3 Matrix{Int32}:
0 -4 1
1 -3 -4
0 1 0

// generate()
// generate()
Generated a random non-singular matrix
such that the inverse is formed by
//
Your random vector is
[-9, 6, 9]

// generate()
Your amplified matrix is
3x4 Matrix{Int32}:
0 -4 1 -9
1 -3 -4 6
0 1 0 9

println("\nGeneration-----")
using LinearAlgebra
using Random

// generate()
// generate()
Generated a random non-singular matrix
and vectors entre 10 x 10, 10 x 10, 10 x 10
//
39 n = 3 # set the dimension of the matrix
A = rand(-10:10, (n,n))
A = UnitUpperTriangular(A)

Triangulation-----
Fila2 = 1 * Fila2 - 0 * Fila1 A[shuffle(1:end), :]
A = A[Fila2:end, Fila2:end]

```

```

Fila2 = 1 * Fila2 - 0 * Fila1
3x4 Matrix{Int32}:
1 -3 -4 6
0 -4 1 -9
0 1 0 9

// generate()
Fila3 = 1 * Fila3 - 0 * Fila1
3x4 Matrix{Int32}:
1 -3 -4 6
0 -4 1 -9
0 1 0 9

// generate()
Generated a random non-singular matrix
such that the inverse is formed by
//
Fila3 = -4 * Fila3 - 1 * Fila2 println("\nGeneration-----")
3x4 Matrix{Int32}:
1 -3 -4 6
0 -4 1 -9
0 0 -1 -27

using LinearAlgebra
using Random

// generate()
// generate()
Generated a random non-singular matrix
and vectors entre 10 x 10, 10 x 10, 10 x 10
//
Your triangulated matrix is
3x4 Matrix{Int32}:
1 -3 -4 6
0 -4 1 -9
0 0 -1 -27

39 n = 3 # set the dimension of the matrix
A = rand(-10:10, (n,n))
A = UnitUpperTriangular(A)

Calibration-----

```

```

Calibration-----
3x4 Matrix{Int32}:
1 -3 -4 6
0 -4 1 -9
0 0 -1 -27

// generate()
// generate()
Generated a random non-singular matrix
and vectors entre 10 x 10, 10 x 10, 10 x 10
//

Backtracking-----
// generate()
// generate()
result[3] = -27 / -1 = 27.0
result[2] = (-9 - (1 * 27.0) - (-4 * 0.0)) / -4 = 9.0
result[1] = (6 - (-4 * 27.0) - (-3 * 9.0) - (1 * 0.0)) / 1 = 141.0

result [141.0, 9.0, 27.0]
using LinearAlgebra

x=A\b = [141.0, 9.0, 27.0] (using julia)

Process returned 0 (0x0) execution time : 2.250 s
Presione una tecla para continuar . . .

```

```

Generation-----
Your random 4 x 4 matrix is
4x4 Matrix{Int32}:
 1  0  0  0
-10 1  7  9
-5  0 10  1
 5  0  1  0

Your random vector is
[4, -6, 7, 10]

Your amplified matrix is
4x5 Matrix{Int32}:
 1  0  0  0  4
-10 1  7  9 -6
-5  0 10  1  7
 5  0  1  0 10

Calibration-----
4x5 Matrix{Int32}:
 1  0  0  0  4
-10 1  7  9 -6
-5  0 10  1  7
 5  0  1  0 10

```

```

Triangulation-----
Fila2 = 1 * Fila2 - -10 * Fila1
4x5 Matrix{Int32}:
 1  0  0  0  4
 0  1  7  9 34
-5  0 10  1  7
 5  0  1  0 10

Fila3 = 1 * Fila3 - -5 * Fila1
4x5 Matrix{Int32}:
 1  0  0  0  4
 0  1  7  9 34
 0  0 10  1 27
 5  0  1  0 10

Fila4 = 1 * Fila4 - 5 * Fila1
4x5 Matrix{Int32}:
 1  0  0  0  4
 0  1  7  9 34
 0  0 10  1 27
 0  0  1  0 -10

Fila3 = 1 * Fila3 - 0 * Fila2

```

```

Fila3 = 1 * Fila3 - 0 * Fila2
4x5 Matrix{Int32}:
 1  0  0  0  4
 0  1  7  9 34
 0  0 10  1 27
 0  0  1  0 -10

Fila4 = 1 * Fila4 - 0 * Fila2
4x5 Matrix{Int32}:
 1  0  0  0  4
 0  1  7  9 34
 0  0 10  1 27
 0  0  1  0 -10

Fila4 = 10 * Fila4 - 1 * Fila3
4x5 Matrix{Int32}:
 1  0  0  0  4
 0  1  7  9 34
 0  0 10  1 27
 0  0  0 -1 -127

Your triangulated matrix is
4x5 Matrix{Int32}:
 1  0  0  0  4
 0  1  7  9 34
 0  0 10  1 27
 0  0  0 -1 -127

```

```

Your triangulated matrix is
4x5 Matrix{Int32}:
 1  0  0  0  4
 0  1  7  9 34
 0  0 10  1 27
 0  0  0 -1 -127

Calibration----- (again) -----
4x5 Matrix{Int32}:
 1  0  0  0  4
 0  1  7  9 34
 0  0 10  1 27
 0  0  0 -1 -127

Backtracking-----
result[4] = -127 / -1 = 127.0
result[3] = (27 - (1 * 127.0) - (10 * 0.0)) / 10 = -10.0
result[2] = (34 - (9 * 127.0) - (7 * -10.0) - (1 * 0.0)) / 1 = -1039.0
result[1] = (4 - (0 * 127.0) - (0 * -10.0) - (0 * -1039.0) - (1 * 0.0)) / 1 = 4.0

result [4.0, -1039.0, -10.0, 127.0]
x=A\b = [4.000000000000023, -1039.000000000003, -10.00000000000034, 127.0000000000037] (using julia)

Process returned 0 (0x0)   execution time : 2.276 s
Presione una tecla para continuar . . .

```