Questão 6

In [73]:
```
using LinearAlgebra
```

In [68]:
```
A = [1.133 5.281; 24.140 -1.210]
b = [6.414, 22.93]
```

Out[68]:
```
2-element Vector{Float64}:
  6.414
 22.93
```

In [69]:
```
function subs_reg(A, b)
    n = length(b)
    x = Vector{Float64}(undef, n)
    for i = n:-1:1
        ld = b[i]
        for j = i + 1:n
            ld -= round(A[i, j]*x[j], digits = 3)
        end
        x[i] = round(ld / A[i, i], digits = 3)
    end
    x .= round.(x, digits = 3)
    return x
end

function subs_prog(A, b)
    n = length(b)
    x = Vector{Float64}(undef, n)
    x[1] = b[1]/A[1, 1]
    for i = 2:n
        x[i] = (b[i] - dot(A[i, 1:i], x[1:i]))/A[i, i]
    end
    x .= round.(x, digits = 3)
    return x
end
```

Out[69]:
```
subs_prog (generic function with 1 method)
```

In [72]:
```
# Fatoracao LU de uma matriz A sem pivoteamento
function preLU(A)
```

```
    n, _ = size(A)
    L = one(A)
    U = copy(A)
    for i = 1:n - 1
        for j = i + 1:n
            coef = round(U[j, i] / U[i, i], digits = 3)
            L[j, i] = coef
            U[j, i] = 0.0
            U[j, i + 1:end] .-= coef .* U[i, i + 1:end]
        end
    end
    L .= round.(L, digits = 3)
    U .= round.(U, digits = 3)
    return L, U
end

preL, preU = preLU(A)
prey = subs_prog(preL, b)
prex = subs_reg(preU, prey)
println("Solucao         = ", prex)
println("Verificacao, Ax = ", A*prex)
println("lado direito    = ", b)
```

```
Solucao         = [0.48, 4.239]
Verificacao, Ax = [22.929999, 6.45801]
lado direito    = [22.93, 6.414]
```

In [71]:

```
# Fatoracao LU de uma matriz A com pivoteamento parcial
function PLU(A)
    n, _ = size(A)
    P = collect(1:n)
    L = one(A)
    U = copy(A)
    for i = 1:n - 1
        # Busca o maior pivot em valor absoluto
        maxind = argmax(abs.(U[i:end, i])) + i - 1
        # Troca as linhas de lugar e guarda a informação.
        U[i, i:end], U[maxind, i:end] = U[maxind, i:end], U[i, i:end]
        L[i, 1:i-1], L[maxind, 1:i-1] = L[maxind, 1:i-1], L[i, 1:i-1]
        P[i], P[maxind] = P[maxind], P[i]
```

```
        # Continua com a fatoração LU.
        for j = i + 1:n
            coef = round(U[j, i] / U[i, i], digits = 3)
            L[j, i] = coef
            U[j, i] = 0.0
            U[j, i + 1:end] .-= coef .* U[i, i + 1:end]
        end
    end
    P .= round.(P, digits = 3)
    L .= round.(L, digits = 3)
    U .= round.(U, digits = 3)
    return P, L, U
end

P, L, U = PLU(A)
b = b[P]
y = subs_prog(L, b)
x = subs_reg(U, y)
println("Solucao         = ", x)
println("Verificacao, Ax = ", A*x)
println("lado direito    = ", b)
```

```
Solucao         = [1.0, 1.0]
Verificacao, Ax = [6.414, 22.93]
lado direito    = [22.93, 6.414]
```

Como podemos ver, a solução do sistema sem pivoteamento foi mais distante da resposta exata que a solução com pivoteamento. Isso aconteceu, pois provavelmente houve erro de cancelamento no primeiro caso, mas não no segundo.