Universidad Nacional de Ingeniería

Facultad de Ciencias

Escuela Profecional de Ciencias de la Computación

PC2 Informe - Algoritmo LU

Pool Lennin Salvatierra Arevalo [20190282G]

Manuel Ademar Fatama Ruiz [20192148F]

Janice Katherine Escobedo Vasquez [20190468C]

PROGRAMACIÓN PARALELA | CC332

I. Factorización LU Secuencial

Algoritmo de Doolitle

La descomposición LU consiste en encontrar dos matrices L y U construidas de tal manera que

$$A = L \cdot U$$

En el Algoritmo de Doolitle la matriz U es una matriz triangular superior y L es una matriz triangular inferior con diagonal de unos; es decir, las matrices deben tener la forma:

$$L = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ L_{21} & 1 & 0 & \cdots & 0 \\ L_{31} & L_{32} & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ L_{n1} & L_{n2} & L_{n3} & \cdots & 1 \end{bmatrix}, \quad U = \begin{bmatrix} U_{11} & U_{12} & U_{13} & \cdots & U_{1n} \\ 0 & U_{22} & U_{23} & \cdots & U_{2n} \\ 0 & 0 & U_{33} & \cdots & U_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & U_{nn} \end{bmatrix}$$

Así el algoritmo en Julia es como sigue

```
using LinearAlgebra
function LU(A::Matrix{Float64})
    m,n = size(A)
    L = Matrix{Float64}(I, n, n)
    U = copy(A)
    for j = 1:n-1
        for i = j+1:n
            L[i,j] = U[i,j]/U[j,j]
            U[i,j:end] -= L[i,j]*U[j,j:end]
        end
    end
    return L,U
end
```

Prueba:

Probando con una matriz aleatoria $A2$ de orden 1000.

```
A2 = rand(1000,1000)
@time LU(A2)
```

```
  10.421589 seconds (2.00 M allocations: 10.235 GiB, 48.32% gc time)
([1.0 0.0 … 0.0 0.0; 3.4695111639892264 1.0 … 0.0 0.0; … ; 0.18229920593560983 -0.2950560135095423 … 1.0 0.0; 0.20766807579705074 -0.26164627281456243 …
34.77376270028721 1.0], [0.2525691801863449 0.8316343724844454 … 0.49702451956839233 0.005429370030421432; 0.0 -2.2718244651871276 … -0.763091183089621
0.18716081664072315; … ; 0.0 0.0 … 0.19840073310738005 -1.7042053769843837; 0.0 0.0 … 0.0 53.62413348935452])
```

Figura 1: Tiempo de ejecución de Factorización LU serial

II. Factorización LU Paralelo

Comenzamos instalando un kernel con 4 nucleos para poder trabajar de manera paralela en Julia.

Figura 2: Creando Kernel con 4 núcleos

Ya instalado el kernel se tienen 4 hilos disponibles en el entorno.

```
using LinearAlgebra
using Base.Threads

function parallel_LU(A::Matrix{Float64})
    m,n = size(A)
    L = Matrix{Float64}(I, n, n)
    U = copy(A)
    num_threads = Threads.nthreads() # Obtenemos el numero de hilos disponibles en el proceso
    threads = []
    for j = 1:n-1
        for i = j+1:n
            if length(threads) < num_threads
                # Creamos una tarea en el hilo actual, en este caso  la tarea
                # es hacer: L[i,j] = U[i,j]/U[j,j] y U[i,j:end] -= L[i,j]*U[j,j:end]
                push!(threads, Threads.@spawn begin
                    L[i,j] = U[i,j]/U[j,j]
                    U[i,j:end] -= L[i,j]*U[j,j:end]
                end)
            else # Cuando ya tengamos en ejecucion los 4 hilos
                wait(threads[1]) # Esperamos a que la tarea 1 iniciada acabe
                # La tarea 1 acabo y asi lo podemos eliminar
                popfirst!(threads) # Elimina el primer elemento de la lista
                                   # threads; es decir, la primera tarea
            end
        end
    end
    for thread in threads # Recorre las tareas, una en cada hilo
        # Esperamos a que acaben de ejecutarse todos los hilos para finalizar la ejecucion
        wait(thread)
    end
    return L,U
end
```

Prueba:

Probando con la misma matriz aleatoria $A2$ de orden 1000.

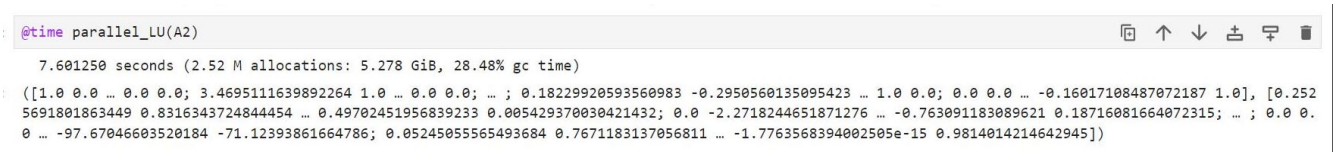

Figura 3: Tiempo de ejecución de Factorización LU paralelo

III. Factorización LU por bloques

Busca descomponer a la matriz A de orden n, con n (par), en una matriz 2×2 por bloques y así la descomposición obtenida se da de la forma

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} = \begin{bmatrix} L_{11} & 0_{n/2} \\ L_{21} & L_{22} \end{bmatrix} \begin{bmatrix} U_{11} & U_{12} \\ 0_{n/2} & U_{22} \end{bmatrix}$$

Donde

$$\begin{aligned} A_{11} &= L_{11}U_{11} \text{ (aplicamos el algoritmo de Doolitle)} \\ A_{21} &= L_{21}U_{11} \text{ , entonces podemos conocer } L_{21} \\ A_{12} &= L_{11}U_{12} \text{ , entonces podemos conocer } U_{12} \\ A_{22} &= L_{21}U_{12} + L_{22}U_{22} \text{ , entonces podemos conocer } L_{22} \text{ y } U_{22} \text{ aplicando el algoritmo} \end{aligned}$$

y así es como hallamos las matrices de descomposición por bloques.

```
using LinearAlgebra

function is_singular(A)
    return det(A) == 0
end

function block_LU(A)
    n = size(A, 1) # Numero de filas, debe ser par
    if n == 1
        return A, A, A
    end
    # Dividimos a la matriz en bloques de (n/2)*(n/2)
    #     [ A_11 | A_12 ]
    # A = | ------------ |
    #     [ A_21 | A_22 ]

    m = n / 2
    A11 = A[1:m, 1:m]
    A12 = A[1:m, m+1:end]
    A21 = A[m+1:end, 1:m]
    A22 = A[m+1:end, m+1:end]

    if is_singular(A11) || is_singular(A22)
        error("La matriz es singular")
    end

    # Efectuar el algoritmo Doolitle como si tuvieramos una matriz 2x2
    L11, U11 = impLU(A11)
    U12 = L11 \ A12
    L21 = A21 / U11
    L22, U22 = impLU(A22 - L21*U12)

    # Formacion de matrices L, U por bloques tambien
    L = [L11 zeros(m, n-m);
         L21 L22]
    U = [U11 U12;
         zeros(n-m, m) U22]

    return L, U
```

```
end

# Algoritmo Doolitle
function impLU(A::Matrix{Float64})
    m,n = size(A)
    L = Matrix{Float64}(I, n, n)
    U = copy(A)
    for j = 1:n-1
        for i = j+1:n
            L[i,j] = U[i,j]/U[j,j]
            U[i,j:end] -= L[i,j]*U[j,j:end]
        end
    end
    return L,U
end
```

Prueba:

Probando con la misma matriz aleatoria $A2$ de orden 1000.

```
@time block_LU(A2)
```

```
  2.380511 seconds (998.04 k allocations: 2.668 GiB, 12.49% gc time)
([1.0 0.0 … 0.0 0.0; 3.4695111639892264 1.0 … 0.0 0.0; … ; 0.18229920592933546 -0.2950560135122865 … 1.0 0.0; 0.20766807578972465 -0.26164627281750924 …
34.773762731020696 1.0], [0.2525691801863449 0.8316343724844454 … 0.49702451956839233 0.0054293700304214 32; 0.0 -2.2718244651871276 … -0.763091183089621
0.18716081664072312; … ; 0.0 0.0 … 0.19840073292648652 -1.7042053766435008; 0.0 0.0 … 0.0 53.62413353024458])
```

Figura 4: Tiempo de ejecución de Factorización LU por bloques serial

IV. Factorización LU por bloques con LU paralelo

Por último, paralelizando la Factorización por bloques usando 4 hilos tenemos el siguiente algoritmo.

```
function is_singular(A)
    return det(A) == 0
end

function pseudo_parallel_LU_block_LU(A)
    n = size(A, 1)
    if n == 1
        return A, A, A
    end

    m = n / 2
    A11 = A[1:m, 1:m]
    A12 = A[1:m, m+1:end]
    A21 = A[m+1:end, 1:m]
    A22 = A[m+1:end, m+1:end]

    if is_singular(A11) || is_singular(A22)
        error("La matriz es singular")
    end

    L11, U11 = parallel_LU(A11)
    U12 = L11 \ A12
    L21 = A21 / U11
    L22, U22 = parallel_LU(A22 - L21*U12)

    L = [L11 zeros(m, n-m);
         L21 L22]
    U = [U11 U12;
         zeros(n-m, m) U22]

    return L, U
end
```

```
function parallel_LU(A::Matrix{Float64})
    m,n = size(A)
    L = Matrix{Float64}(I, n, n)
    U = copy(A)
    num_threads = Threads.nthreads()
    threads = []
    for j = 1:n-1
        for i = j+1:n
            if length(threads) < num_threads
                push!(threads, Threads.@spawn begin
                    L[i,j] = U[i,j]/U[j,j]
                    U[i,j:end] -= L[i,j]*U[j,j:end]
                end)
            else
                wait(threads[1])
                popfirst!(threads)
            end
        end
    end
    for thread in threads
        wait(thread)
    end
    return L,U
end
```

Prueba:

Probando con la misma matriz aleatoria $A2$ de orden 1000.

```
@time pseudo_parallel_LU_block_LU(A2)
```

```
  1.558353 seconds (1.26 M allocations: 1.435 GiB)
([1.0 0.0 … 0.0 0.0; 3.4695111639892264 1.0 … 0.0 0.0; … ; 2.291869903351849 1.7297873138518978 … 1.0 0.0; 0.29581462481853893 0.37243890197836665 … 0.05035868127299998 1.0], [0.2525691801863449 0.8316343724844454 … 0.49702451956839233 0.0054293700304214 32; 0.0 -2.2718244651871276 … -0.763091183089621 0.18716081664072312; … ; 0.0 0.0 … -34.663708055927806 -25.346913532126663; 0.0 0.0 … 2.220446049250313e-16 -1.07726921088336])
```

Figura 5: Tiempo de ejecución de Factorización LU por bloques en paralelo

V. Comparando Tiempos de ejecución para matrices de orden mayor

```
A3 = rand(2000,2000)
```

```
2000×2000 Matrix{Float64}:
 0.0549109  0.0202591  0.51384    …  0.818147   0.683864   0.809701
 0.170037   0.913431   0.684139      0.862794   0.983926   0.55613
 0.704906   0.655008   0.56884       0.776266   0.254161   0.298475
 0.0554005  0.315595   0.71282       0.0627926  0.163593   0.809113
 0.704343   0.076775   0.776898      0.20593    0.517136   0.336386
 0.298656   0.450285   0.392252   …  0.384504   0.202468   0.889039
 0.876766   0.102646   0.661861      0.327693   0.331203   0.364921
 0.942986   0.304658   0.91167       0.847318   0.904285   0.432371
 0.227969   0.259491   0.0670966     0.86364    0.284437   0.199149
 0.825595   0.687794   0.54205       0.249374   0.830727   0.969441
 0.772652   0.450444   0.94256    …  0.407625   0.727248   0.183077
 0.755856   0.31815    0.211984      0.468109   0.845889   0.0877412
 0.0261066  0.405011   0.83543       0.997032   0.0891597  0.618466
 ⋮                                ⋱
 0.571677   0.28162    0.871071      0.567263   0.639375   0.0515111
 0.733301   0.282459   0.242371      0.082602   0.250032   0.0997331
 0.222525   0.563998   0.256499   …  0.532905   0.178466   0.880472
 0.994238   0.608261   0.491329      0.341637   0.771734   0.929753
 0.160713   0.482084   0.366703      0.926191   0.841292   0.205428
 0.37494    0.804401   0.257672      0.601722   0.405466   0.964106
 0.426578   0.853388   0.243276      0.357391   0.577533   0.483382
 0.673519   0.281379   0.596539   …  0.0941084  0.195892   0.378382
 0.948491   0.380228   0.444858      0.457169   0.0595779  0.86419
 0.991201   0.753323   0.385257      0.721011   0.560735   0.648206
 0.835164   0.644813   0.300657      0.909372   0.143351   0.480445
 0.391773   0.146328   0.765649      0.216253   0.347588   0.487241
```

```
@time LU(A3)
```

```
 55.049873 seconds (8.00 M allocations: 80.690 GiB, 9.71% gc time)

([1.0 0.0 … 0.0 0.0; 3.0965922842084077 1.0 … 0.0 0.0; … ; 15.209444148889675 0.39577317022402086 … 1.0 0.0; 7.13470924973447 0.00209883896903328 … -5.2
69290414079614 1.0], [0.054910881633198905 0.02025913121896805 … 0.6838641259466856 0.809700851562404; 0.0 0.8506970735473804 … -1.1337227773796799 -1.9
511837272862618; … ; 0.0 0.0 … 3.9673600769847734 5.929271422913388; 0.0 2.168404344971009e-19 … 0.0 24.413557992267275])
```

```
@time parallel_LU(A3)
```

```
 26.141261 seconds (10.10 M allocations: 40.988 GiB, 11.66% gc time)

([1.0 0.0 … 0.0 0.0; 3.0965922842084077 1.0 … 0.0 0.0; … ; 15.209444148889675 0.39577317022402086 … 1.0 0.0; 0.0 0.0 … -3.0889000021489204 1.0], [0.0549
10881633198905 0.02025913121896805 … 0.6838641259466856 0.809700851562404; 0.0 0.8506970735473804 … -1.1337227773796799 -1.9511837272862618; … ; 0.0 0.0
… -79.32413552050515 -50.16481433996946; 0.39177317509945886 0.14632848706835955 … 0.0 -47.68112934439564])
```

Figura 6: Comparando tiempos de ejecución LU con matriz de orden 2000

```
@time block_LU(A3)
```

```
 15.385267 seconds (4.00 M allocations: 20.634 GiB, 38.40% gc time)

([1.0 0.0 … 0.0 0.0; 3.0965922842084077 1.0 … 0.0 0.0; … ; 15.209444148874223 0.39577317022226405 … 1.0 0.0; 7.134709249677594 0.00209883897407292 … -5.2
69290399185727 1.0], [0.054910881633198905 0.02025913121896805 … 0.6838641259466856 0.809700851562404; 0.0 0.8506970735473804 … -1.1337227773796799 -1.9
511837272862615; … ; 0.0 0.0 … 3.967360082211501 5.929271419913377; 0.0 0.0 … 0.0 24.413557853668813])
```

```
@time pseudo_parallel_LU_block_LU(A3)
```

```
  5.500196 seconds (5.04 M allocations: 10.720 GiB, 16.27% gc time)

([1.0 0.0 … 0.0 0.0; 3.0965922842084077 1.0 … 0.0 0.0; … ; -19.62939864104914 -3.7750364653245403 … 1.0 0.0; 31.65464250079439 3.51842662767697 … 0.3693
4231528069605 1.0], [0.054910881633198905 0.02025913121896805 … 0.6838641259466856 0.809700851562404; 0.0 0.8506970735473804 … -1.1337227773796799 -1.95
11837272862615; … ; 0.0 0.0 … -83.46921423455646 61.29024519686027; 0.0 0.0 … 0.0 8.712907272404756])
```

Figura 7: Comparando tiempos de ejecución LU por bloques con matriz de orden 2000

```
A2 = rand(4096,4096)
```

```
4096x4096 Matrix{Float64}:
 0.0296867  0.0950193  0.446196   …  0.249777   0.307937    0.747337
 0.114082   0.16329    0.17419       0.440044   0.421668    0.759555
 0.973958   0.197145   0.154852      0.590097   0.00850266  0.883437
 0.389139   0.313659   0.0889511     0.318088   0.634308    0.018493
 0.253464   0.870436   0.874104      0.645054   0.815052    0.735412
 0.737576   0.0393733  0.862329   …  0.922918   0.345616    0.826692
 0.686894   0.0998984  0.149798      0.590153   0.830671    0.261188
 0.672854   0.228385   0.0100861     0.684087   0.123277    0.653609
 0.212338   0.82499    0.131326      0.0707359  0.365844    0.0832599
 0.441277   0.729894   0.948379      0.919134   0.267154    0.545103
 0.223587   0.141336   0.204631   …  0.833694   0.891934    0.0823051
 0.700929   0.314508   0.461097      0.679484   0.841496    0.811495
 0.629377   0.497909   0.752857      0.307292   0.761614    0.340696
 ⋮                                  ⋱                      ⋮
 0.707018   0.876267   0.597344      0.0472267  0.518836    0.99897
 0.180315   0.50231    0.691955   …  0.615231   0.0685899   0.629606
 0.238825   0.450095   0.948723      0.193594   0.0747071   0.845829
 0.932164   0.143257   0.914694      0.525618   0.702401    0.160248
 0.369989   0.0600504  0.375638      0.224979   0.152941    0.558437
 0.631881   0.529587   0.527452      0.042239   0.26876     0.688622
 0.692166   0.157845   0.854132   …  0.406863   0.173126    0.471164
 0.978879   0.820833   0.559851      0.197599   0.944187    0.00822356
 0.775663   0.986533   0.574041      0.0340259  0.478105    0.796138
 0.283312   0.127474   0.55666       0.950472   0.911421    0.823891
 0.685679   0.28717    0.563926      0.31087    0.479407    0.352653
 0.741664   0.798445   0.50776    …  0.91631    0.436561    0.240475
```

```
@time LU(A2)
```

```
1238.513556 seconds (58.71 M allocations: 685.911 GiB, 3.34% gc time)
```

```
([1.0 0.0 … 0.0 0.0; 3.842847284695305 1.0 … 0.0 0.0; … ; 23.097172596194024 9.449922040015881 … 1.0 0.0; 24.9830135166097 7.804756308297792 … -0.202102
43609981285 1.0], [0.029686712457045905 0.09501928952400829 … 0.30793705086947887 0.7473365855671892; 0.0 -0.20185421985590496 … -0.7616874784991267 -2.
112345302372907; … ; 0.0 0.0 … -39.01874088286546 14.333403505755157; 0.0 0.0 … 0.0 20.02316683760623])
```

```
@time parallel_LU(A2)
```

```
636.156260 seconds (54.87 M allocations: 345.653 GiB, 43.13% gc time)
```

```
([1.0 0.0 … 0.0 0.0; 3.842847284695305 1.0 … 0.0 0.0; … ; 23.097172596194024 9.449922040015881 … 1.0 0.0; 0.0 0.0 … 0.9198529122592702 1.0], [0.02968671
2457045905 0.09501928952400829 … 0.30793705086947887 0.7473365855671892; 0.0 -0.20185421985590496 … -0.7616874784991267 -2.112345302372907; … ; 0.0 0.0
… 85.36761665097292 -110.98361947853168; 0.7416635385780833 0.7984451987400459 … 0.0 -17.721876429933218])
```

Figura 8: Comparando tiempos de ejecución LU con matriz de orden 4096

```
@time block_LU(A2)
```

```
166.100383 seconds (16.77 M allocations: 173.906 GiB, 7.36% gc time)
```

```
([1.0 0.0 … 0.0 0.0; 3.842847284695305 1.0 … 0.0 0.0; … ; 23.097172592099238 9.449922038146914 … 1.0 0.0; 24.983013504816952 7.804756302440132 … -0.20210
243017316412 1.0], [0.029686712457045905 0.09501928952400829 … 0.30793705086947887 0.7473365855671892; 0.0 -0.20185421985590496 … -0.7616874784991267 -2.
112345302372907; … ; 0.0 0.0 … -39.018739970251346 14.33340296885359; 0.0 0.0 … -8.881784197001252e-16 20.02316668829928])
```

```
@time pseudo_parallel_LU_block_LU(A2)
```

```
 76.289873 seconds (21.19 M allocations: 88.647 GiB, 21.75% gc time)
```

```
([1.0 0.0 … 0.0 0.0; 3.842847284695305 1.0 … 0.0 0.0; … ; 818.963370573739 37.39543171241954 … 1.0 0.0; -450.28472635076247 -20.875320908926785 … 1.47574
8585787109 1.0], [0.029686712457045905 0.09501928952400829 … 0.30793705086947887 0.7473365855671892; 0.0 -0.20185421985590496 … -0.7616874784991267 -2.11
2345302372907; … ; 0.0 0.0 … -115.84960485625751 196.51790670981362; 0.0 0.0 … 77.51793865194651 -213.5523371883619])
```

Figura 9: Comparando tiempos de ejecución LU por bloques con matriz de orden 4096

pool.salvatierra.a@uni.pe
manuel.fatama.r@uni.pe
janice.escobedo.v@uni.pe

Documento realizado en
LaTeX