Math/CS 466/666: Programming Project 1

1a. Write a program to find the Cholesky factorization $A = U^T U$ of a symmetric positive definite 2×2 matrix A. Note that if your programming language of choice has a built-in Cholesky factorization subroutine, do not use it for this exercise. Your program should report an error if the input matrix is not symmetric or not positive definite; otherwise it should report the factor U.

My program was

```
using LinearAlgebra

function mycholesky(A)
    n=size(A)[1]
    L=copy(LowerTriangular(A))
    for k=1:n
        for i=1:k-1
            s=0
            for j=1:i-1
                s+=L[i,j]*L[k,j]
            end
            L[k,i]=(L[k,i]-s)/L[i,i]
        end
        v=0
        for j=1:k-1
            v+=L[k,j]^2
        end
        L[k,k]=sqrt(L[k,k]-v)
    end
    return L'
end

function issym(A)
    n=size(A)[1]
    for i=1:n
        for j=i:n
            if A[i,j]!=A[j,i]
                return false
            end
        end
    end
    return true
end

function dowork(A)
    if !issym(A)
```

```julia
        println("The matrix is not symmetric!")
        return
    end
    U=Matrix{Float64}
    try
        U=mycholesky(A)
    catch
        println("The matrix is not positive definite!")
        return
    end
    println("The Cholesky factor:")
    display(U); println("")
end

function main(fn)
    open(fn,"r") do fp
        i=0
        while true
            i+=1
            a=readline(fp)
            if a==""
                break
            end
            d=parse(Int,a)
            A=zeros(d,d)
            for i=1:d
                a=readline(fp)
                r=split(a)
                s=(x->parse(Float64,x)).(r)
                A[i,:]=s
            end
            println("\nMatrix $i:")
            display(A); println("")
            dowork(A)
            a=readline(fp)
        end
    end
end

println("cholesky.jl -- Find the Cholesky factorization\n",
    "Written 2021 by Eric Olson for Math 466/666")
main("prog1b.dat")
```

b. Test your program using the matrices

$$\begin{bmatrix} 1 & 0 \\ 0 & 4 \end{bmatrix}, \quad \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \quad \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}, \quad \begin{bmatrix} 2 & 1 \\ 3 & 4 \end{bmatrix}, \quad \begin{bmatrix} 1 & 2 \\ 2 & 3 \end{bmatrix}, \quad \text{and} \quad \begin{bmatrix} 5 & -1 \\ -1 & 3 \end{bmatrix}.$$

Which ones are symmetric positive definite? For those that are, what is the factor U?

The output was

```
cholesky.jl -- Find the Cholesky factorization
Written 2021 by Eric Olson for Math 466/666

Matrix 1:
2×2 Matrix{Float64}:
 1.0  0.0
 0.0  4.0
The Cholesky factor:
2×2 UpperTriangular{Float64, Adjoint{Float64, Matrix{Float64}}}:
 1.0  0.0
  ⋅   2.0

Matrix 2:
2×2 Matrix{Float64}:
 1.0  0.0
 0.0  0.0
The Cholesky factor:
2×2 UpperTriangular{Float64, Adjoint{Float64, Matrix{Float64}}}:
 1.0  0.0
  ⋅   0.0

Matrix 3:
2×2 Matrix{Float64}:
 2.0  1.0
 1.0  2.0
The Cholesky factor:
2×2 UpperTriangular{Float64, Adjoint{Float64, Matrix{Float64}}}:
 1.41421  0.707107
  ⋅       1.22474

Matrix 4:
2×2 Matrix{Float64}:
 2.0  1.0
 3.0  4.0
The matrix is not symmetric!

Matrix 5:
```

```
2×2 Matrix{Float64}:
 1.0  2.0
 2.0  3.0
The matrix is not positive definite!

Matrix 6:
2×2 Matrix{Float64}:
  5.0  -1.0
 -1.0   3.0
The Cholesky factor:
2×2 UpperTriangular{Float64, Adjoint{Float64, Matrix{Float64}}}:
 2.23607  -0.447214
  ⋅        1.67332
```

c. The Julia programming language has a Cholesky factorization routine included with the linear algebra library. The command

```
julia> using LinearAlgebra
```

loads this library into your workspace. After loading the linear algebra library type

```
julia> A=rand(3,3)
julia> B=A'*A
julia> z=cholesky(B)
julia> propertynames(z)
```

to randomly create a positive definite matrix B, find its Cholesky decomposition and finally display what fields are available for use.

The results were

```
$ julia
               _
   _       _ _(_)_     |  Documentation: https://docs.julialang.org
  (_)     | (_) (_)    |
   _ _   _| |_  __ _   |  Type "?" for help, "]?" for Pkg help.
  | | | | | | |/ _` |  |
  | | |_| | | | (_| |  |  Version 1.6.1 (2021-04-23)
 _/ |\__'_|_|_|\__'_|  |
|__/                   |

julia> using LinearAlgebra

julia> A=rand(3,3)
3×3 Matrix{Float64}:
 0.709776  0.920373  0.290763
 0.511919  0.893039  0.0561106
 0.846738  0.773582  0.906861
```

```
julia> B=A'*A
3×3 Matrix{Float64}:
 1.48281  1.76544  1.00297
 1.76544  2.24303  1.01925
 1.00297  1.01925  0.910089

julia> z=cholesky(B)
Cholesky{Float64, Matrix{Float64}}
U factor:
3×3 UpperTriangular{Float64, Matrix{Float64}}:
 1.21771  1.44981   0.823659
  ⋅       0.37561  -0.465639
  ⋅        ⋅        0.121881

julia> propertynames(z)
(:U, :L, :UL)
```

d. Compute the residual error of the factorization found in the previous problem using

```
julia> z.L*z.U-B
julia> norm(z.L*z.U-B)
```

Comment on the accuracy of the factorization.

The residual error was

```
julia> z.L*z.U-B
3×3 Matrix{Float64}:
  2.22045e-16  -2.22045e-16  0.0
 -2.22045e-16   0.0          0.0
  0.0           0.0          0.0

julia> norm(z.L*z.U-B)
3.8459253727671276e-16
```

Note that an error on the order 10^{-16} is consistent with double precision arithmetic being accurate to about 15 decimal digits.

e. The file `prog1c.dat` contains square matrices of varying sizes separated by blank lines. The description of each matrix consists of an integer specifying the dimension of the matrix followed by the entries of the matrix in row-major order. Use the Julia command `cholesky` to check which of these matrices are positive definite and to find the Cholesky decomposition for each one that is.

I deleted the call `mycholesky(A)` from the program writen for part a and replaced it with `cholesky(A).U` to use the built-in function.

The output was

```
cholesky.jl -- Find the Cholesky factorization
```

```
Written 2021 by Eric Olson for Math 466/666

Matrix 1:
3×3 Matrix{Float64}:
 14.0   32.0   53.0
 32.0   77.0  128.0
 53.0  128.0  213.0
The Cholesky factor:
3×3 UpperTriangular{Float64, Matrix{Float64}}:
 3.74166  8.55236  14.1648
  ⋅       1.96396   3.49149
  ⋅        ⋅        0.408248

Matrix 2:
3×3 Matrix{Float64}:
 14.0   32.0   50.0
 32.0   77.0  122.0
 50.0  122.0  194.0
The matrix is not positive definite!

Matrix 3:
4×4 Matrix{Float64}:
 1.0   2.0    3.0    4.0
 2.0  29.0   36.0   43.0
 3.0  36.0  109.0  126.0
 4.0  43.0  126.0  246.0
The Cholesky factor:
4×4 UpperTriangular{Float64, Matrix{Float64}}:
 1.0  2.0  3.0   4.0
  ⋅   5.0  6.0   7.0
  ⋅    ⋅   8.0   9.0
  ⋅    ⋅    ⋅   10.0

Matrix 4:
5×5 Matrix{Float64}:
 1.5004   1.0618   1.37343  0.75486  0.78857
 1.0618   1.45784  1.4115   0.87909  1.20866
 1.37343  1.4115   2.16213  0.79598  1.17137
 0.75486  0.87909  0.79598  0.73324  0.74008
 0.78857  1.20866  1.17137  0.74008  1.10941
The Cholesky factor:
5×5 UpperTriangular{Float64, Matrix{Float64}}:
 1.22491  0.86684   1.12125    0.616258  0.643779
  ⋅       0.840492  0.522972   0.410345  0.774078
  ⋅        ⋅        0.794623  -0.137927  0.0562672
```

```
  ⋅        ⋅         ⋅         0.407503  0.082128
  ⋅        ⋅         ⋅          ⋅        0.293004

Matrix 5:
5×5 Matrix{Float64}:
 1.63947  1.10207  0.60042  0.87735  0.24427
 1.10207  0.9435   1.36237  1.00498  0.47145
 0.60042  1.36237  1.42103  0.66141  1.34568
 0.87735  1.00498  0.66141  1.1112   0.9995
 0.24427  0.47145  1.34568  0.9995   0.32154
The matrix is not positive definite!

Matrix 6:
5×5 Matrix{Float64}:
  0.0       0.50002  -0.02633  -0.23598  -0.16588
 -0.50002   0.0      -0.5074    0.89786   0.00493
  0.02633   0.5074    0.0       0.1858    0.0765
  0.23598  -0.89786  -0.1858    0.0      -0.5294
  0.16588  -0.00493  -0.0765    0.5294    0.0
The matrix is not symmetric!

Matrix 7:
5×5 Matrix{Float64}:
  1.5004    1.0618    1.37343   0.75486  0.78857
 -1.0618    1.45784   1.4115    0.87909  1.20866
 -1.37343  -1.4115    2.16213   0.79598  1.17137
 -0.75486  -0.87909  -0.79598   0.73324  0.74008
 -0.78857  -1.20866  -1.17137  -0.74008  1.10941
The matrix is not symmetric!

Matrix 8:
6×6 Matrix{Float64}:
  0.316103   0.322523    0.04579     0.127258   -0.125852   -0.019328
  0.322523   0.579893   -0.132581    0.099775   -0.0012549   0.226182
  0.04579   -0.132581    0.755535   -0.0510603  -0.341162   -0.481348
  0.127258   0.099775   -0.0510603   0.528454   -0.31535     0.297283
 -0.125852  -0.0012549  -0.341162   -0.31535     0.755903   -0.0408033
 -0.019328   0.226182   -0.481348    0.297283   -0.0408033   0.659553
The Cholesky factor:
6×6 UpperTriangular{Float64, Matrix{Float64}}:
 0.562231  0.573649   0.0814434   0.226344   -0.223845  -0.0343774
  ⋅        0.500819  -0.358015   -0.0600356   0.253891   0.491001
  ⋅         ⋅         0.787863   -0.115487   -0.294512  -0.384284
  ⋅         ⋅          ⋅          0.67844    -0.417802   0.42769
  ⋅         ⋅          ⋅           ⋅          0.616474  -0.174615
```

```
  .         .          .           .           .         0.237077
```

For reference the above output was produced by the code

```
using LinearAlgebra

function issym(A)
    n=size(A)[1]
    for i=1:n
        for j=i:n
            if A[i,j]!=A[j,i]
                return false
            end
        end
    end
    return true
end

function dowork(A)
    if !issym(A)
        println("The matrix is not symmetric!")
        return
    end
    U=Matrix{Float64}
    try
        U=cholesky(A).U
    catch
        println("The matrix is not positive definite!")
        return
    end
    println("The Cholesky factor:")
    display(U); println("")
end

function main(fn)
    open(fn,"r") do fp
        i=0
        while true
            i+=1
            a=readline(fp)
            if a==""
                break
            end
            d=parse(Int,a)
            A=zeros(d,d)
            for i=1:d
```

```
                a=readline(fp)
                r=split(a)
                s=(x->parse(Float64,x)).(r)
                A[i,:]=s
            end
            println("\nMatrix $i:")
            display(A); println("")
            dowork(A)
            a=readline(fp)
        end
    end
end

println("cholesky.jl -- Find the Cholesky factorization\n",
    "Written 2021 by Eric Olson for Math 466/666")
main("prog1c.dat")
```