Class: MATH 466 Numerical Methods
Professor: Eric Olsen
Team 10:

Michael Dorado
Maverick Marsala
Matthew Neill

Date: 10/21/2021
Assignment: Project 1

1a) Cholesky Decomposition Code:

This section of the code uses the library ‘Delimited Files’ to help read in the contents of the file prog1b.txt, which contains a set of square matrices to be factorized. This is done through the function readFileB, which intakes a file pointer and iterates through each matrix, one at a time, until the file is done being read. When one matrix is done being read, it is passed to the function choleskyDecomp, where it is factorized if possible. Once the matrix is factorized (or not depending on its structure), the file pointer will read in the next matrix in the file until the file is completely read.

```
using DelimitedFiles
using LinearAlgebra

#This function returns one matrix from the prog1b.txt file
function readFileB(fp)
    global d
    a = readline(fp)
    if a === ""
        return 99
    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
    a=readline(fp)
    return A
end
```

```
function choleskyDecomp(A)
#k represents rows, i represents columns. j is our iterative variable
#outer row loop
    if A[1,2] == A[2,1] && isposdef(A)
        L=zeros(d,d)
        for k=1:d
            #inner column loop
            for i=1:d
                #if below diag
                if k>i
                    s=0
                    #creation of element L[k,i]
                    for j=1:i-1
                        s+=L[i,j]*L[k,j]
                    end
                    L[k,i]=(A[k,i]-s)/L[i,i]
                #if on diagonal
                elseif k==i
                    s=0
                    #creation of element L[k,k]
                    for j=1:k-1
                        s+=L[k,j]*L[k,j]
                    end
                    L[k,k]=sqrt(A[k,k]-s)
                end
            end
        end
        println('\n')
        print("L factorization of A: ")
        display(L)
    else
        println('\n')
        display("Matrix not Positive Definite!")
    end
end


#This 'open' function resides outside of the while loop to keep continuity of the variable 'fp'. If you were to call 'open' every time the while loop triggered, you would read the first matrix of the file for eternity
open("prog1b.txt","r") do fp
    print("-------------------------------------------------------------------------------------")
    print("\nThis set of answers correspond to part a-b: \nA matrix that is postive definite will be factored into its lower triangular matrix, where a matrix that is not positive definite will display the text 'Matrix not Positive Definite!'. ")

```

```
    #This is the while loop to deal with the prog1b.txt file, it loops until the if statement is reached, which signals the end of the file has been reached
    while true
        #This 'A' variable is a nxn matrix read from the prog1b.dat.txt fil, moving sequentially through the file
        A = readFileB(fp)
        #This line will break out of this while loop once the end of the file is reached
        if A == 99
            break
        end
        #Below this comment and within the while loop is where code from Maverick and Matt should go
        choleskyDecomp(A)
    end
end
```

1b) Results of Cholesky Factorization on the given set of matrices:

```
--------------------------------------------------------------------------------
This set of answers correspond to part a-b:
A matrix that is postive definite will be factored into its lower triangular matrix, where a matrix that is not positive definite will display the text 'Matrix not Positive Definite!'.

L factorization of A: 2×2 Matrix{Float64}:
 1.0  0.0
 0.0  2.0


"Matrix not Positive Definite!"


L factorization of A: 2×2 Matrix{Float64}:
 1.41421   0.0
 0.707107  1.22474


"Matrix not Positive Definite!"


"Matrix not Positive Definite!"


L factorization of A: 2×2 Matrix{Float64}:
  2.23607   0.0
 -0.447214  1.67332
```

<u>Here is a plain text version of the image above, for ease of reading:</u>

L factorization of A: 2×2 Matrix{Float64}:
1.0 0.0
0.0 2.0

"Matrix not Positive Definite!"

L factorization of A: 2×2 Matrix{Float64}:
1.41421   0.0
0.707107 1.22474

"Matrix not Positive Definite!"

"Matrix not Positive Definite!"

L factorization of A: 2×2 Matrix{Float64}:
2.23607   0.0
-0.447214 1.67332

1c) This section simply asked us to input the given commands and read their output as shown below:

```
#Part C code
print("This set of answers correspond to part c-d: \nThis section creates a random 3x3 matrix, makes it positive definite by multiplying it by its transpose, creates a cholesky factorization and seperates it into its different fields. Finally it displays the error of the algo by zubtracting the factorization by the original matrix.\n\n")
A=rand(3,3)
B=A'*A
z=cholesky(B)
print("This is the cholesky factorization of the original matrix\n")
display(z)
```

```
--------------------------------------------------------------------------------
This set of answers correspond to part c-d:
This section creates a random 3x3 matrix, makes it positive definite by multiplying it by its transpose, creates a cholesky factorization and seperates it into its different fields. Finally it displays the error of the algo by zubtracting the factorization by the original matrix.

This is the cholesky factorization of the original matrix
Cholesky{Float64, Matrix{Float64}}
U factor:
3×3 UpperTriangular{Float64, Matrix{Float64}}:
 1.44029  1.06265    0.79836
  ⋅       0.266399  -0.411528
  ⋅        ⋅         0.62571

These are the fields that exist within the cholesky factorization variable
(:U, :L, :UL)
```

Here is a plain text version of the image above, for ease of reading:

This is the cholesky factorization of the original matrix
Cholesky{Float64, Matrix{Float64}}
U factor:
3×3 UpperTriangular{Float64, Matrix{Float64}}:
1.44029 1.06265 0.79836
· 0.266399 -0.411528
· · 0.62571

These are the fields that exist within the cholesky factorization variable
(:U, :L, :UL)

1d) This section asked us to compare a decomposed and recomposed matrix to itself in an effort to reveal the types of error that can arise from algorithims solving linear algerbra problems.

```
print("These are the fields that exist within the cholesky factorization variable\n")
fields = propertynames(z)
display(fields)
print("\nThis is the matrix made from the multiplying the factorizations together and subtracting them from the original matrix\n")
display(z.L*z.U-B)
print("\nThe error of this operation is: ")
display(norm(z.L*z.U-B))
print("\nThis is our answer to the question posed at the end of part d:\nWhat line 116 {display(z.Lz.U-B)} computes is the difference between our positive definite matrix B and its  LU which in theory should result in B. In practice we see that there are non-zero elements in the difference which means that LU != B. The norm of this difference is analogous to the “length” or distance (LU-B) is from zero.\n\n")
```

```
This is the matrix made from the multiplying the factorizations together and subtracting them from the original matrix
3×3 Matrix{Float64}:
 0.0  0.0  0.0
 0.0  0.0  0.0
 0.0  0.0  0.0

The error of this operation is: 0.0

This is our answer to the question posed at the end of part d:
What line 116 {display(z.Lz.U-B)} computes is the difference between our positive definite matrix B and its  LU which in theory should result in B. In practice we see that there are non-zero elements in the difference which means that LU != B. The norm of this difference is analogous to the “length” or distance (LU-B) is from zero.
```

Here is a plain text version of the image above, for ease of reading:

This is the matrix made from the multiplying the factorizations together and subtracting them from the original matrix
3×3 Matrix{Float64}:
0.0 0.0 0.0
0.0 0.0 0.0
0.0 0.0 0.0

The error of this operation is: 0.0

This is our answer to the question posed at the end of part d:

What line 116 {display(z.Lz.U-B)} computes is the difference between our positive definite matrix B and its LU which in theory should result in B. In practice we see that there are non-zero elements in the difference which means that LU != B. The norm of this difference is analogous to the "length" or distance (LU-B) is from zero.

1e) This section asked us to use the internal Julia 'cholesky' function to solve for larger and larger matrices located in the pro1c.txt file. Note: we tested this file with our own version of the Cholesky Factorization and received the same results.

```
#This is part of the file reading for part e of the project
open("prog1c.txt", "r") do fp
    #This is the while loop to deal with the prog1c.txt file, it loops until the if statement is reached, which signals the end of the file has been reached
    print("--------------------------------------------------------------------------------\n")
    print("This set of answers correspond to part e: \nThis section reads in a set of matrices and uses the julia function 'cholesky' to factor them out as well as determine if they are positive definite matrices. If they are not positive definite matrices, it will display 'Matrix not Positive Definite!'. Otherwise it will display the L factorization. \n\n")
    while true
        #This 'A' variable is a nxn matrix read from the prog1b.dat.txt fil, moving sequentially through the file
        A = readFileC(fp)
        #This line will break out of this while loop once the end of the file is reached
        if A == 99
            break
        end
        #Below this comment and within the while loop is where code from Maverick and Matt should go
        if isposdef(A)
            print("\nL factorization of matrix A:\n")
            display(cholesky(A).L)
        else
            print('\n')
            display("Matrix not Positive Definite!")
        end
    end
end
```

```
--------------------------------------------------------------------------------
This set of answers correspond to part e:
This section reads in a set of matrices and uses the julia function 'cholesky' to factor them out as well as determine if they are positive definite matrices. If they are not positive definite matrices, it will display 'Matrix not Positive Definite!'. Otherwise it will display the L factorization.


L factorization of matrix A:
3×3 LowerTriangular{Float64, Matrix{Float64}}:
  3.74166   ⋅         ⋅
  8.55236  1.96396    ⋅
 14.1648   3.49149  0.408248

"Matrix not Positive Definite!"

L factorization of matrix A:
4×4 LowerTriangular{Float64, Matrix{Float64}}:
 1.0   ⋅    ⋅     ⋅
 2.0  5.0   ⋅     ⋅
 3.0  6.0  8.0    ⋅
 4.0  7.0  9.0  10.0

L factorization of matrix A:
5×5 LowerTriangular{Float64, Matrix{Float64}}:
 1.22491    ⋅          ⋅           ⋅          ⋅
 0.86684   0.840492    ⋅           ⋅          ⋅
 1.12125   0.522972   0.794623     ⋅          ⋅
 0.616258  0.410345  -0.137927   0.407503     ⋅
 0.643779  0.774078   0.0562672  0.082128  0.293004

"Matrix not Positive Definite!"

"Matrix not Positive Definite!"

"Matrix not Positive Definite!"

L factorization of matrix A:
6×6 LowerTriangular{Float64, Matrix{Float64}}:
  0.562231    ⋅           ⋅          ⋅          ⋅          ⋅
  0.573649   0.500819     ⋅          ⋅          ⋅          ⋅
  0.0814434 -0.358015    0.787863    ⋅          ⋅          ⋅
  0.226344  -0.0600356  -0.115487   0.67844     ⋅          ⋅
 -0.223845   0.253891   -0.294512  -0.417802   0.616474    ⋅
 -0.0343774  0.491001   -0.384284   0.42769   -0.174615  0.237077
```

<u>Here is a plain text version of the image above, for ease of reading:</u>

This set of answers correspond to part e:

This section reads in a set of matrices and uses the julia function 'cholesky' to factor them out as well as determine if they are positive definite matrices. If they are not positive definite matrices, it will display 'Matrix not Positive Definite!'. Otherwise it will display the L factorization.

L factorization of matrix A:

3×3 LowerTriangular{Float64, Matrix{Float64}}:

3.74166 · ·

```
  8.55236  1.96396   ⋅
 14.1648   3.49149  0.408248

"Matrix not Positive Definite!"

L factorization of matrix A:
4×4 LowerTriangular{Float64, Matrix{Float64}}:
 1.0   ⋅    ⋅     ⋅
 2.0  5.0   ⋅     ⋅
 3.0  6.0  8.0    ⋅
 4.0  7.0  9.0  10.0

L factorization of matrix A:
5×5 LowerTriangular{Float64, Matrix{Float64}}:
 1.22491    ⋅          ⋅          ⋅         ⋅
 0.86684   0.840492    ⋅          ⋅         ⋅
 1.12125   0.522972   0.794623    ⋅         ⋅
 0.616258  0.410345  -0.137927   0.407503   ⋅
 0.643779  0.774078   0.0562672  0.082128  0.293004

"Matrix not Positive Definite!"

"Matrix not Positive Definite!"

"Matrix not Positive Definite!"

L factorization of matrix A:
6×6 LowerTriangular{Float64, Matrix{Float64}}:
  0.562231    ⋅          ⋅          ⋅          ⋅         ⋅
  0.573649   0.500819    ⋅          ⋅          ⋅         ⋅
  0.0814434 -0.358015   0.787863    ⋅          ⋅         ⋅
  0.226344  -0.0600356 -0.115487   0.67844     ⋅         ⋅
 -0.223845   0.253891  -0.294512  -0.417802   0.616474   ⋅
 -0.0343774  0.491001  -0.384284   0.42769   -0.174615  0.237077
```