# Table of Contents



# Problem 1

Consider an nxn matrix A where n is a random integer between 4 and 6 (inclusive) and entries of A are random numbers between 0 and 1 (exclusive). Determine if A is nonsingular; if it is, and the inverse of A (by either directly solving a system or by extracting entries from an augmented matrix in reduced row echelon form). If it is singular, print a message stating this.

```
clear
clc

n = 3 + randi(3);

A = rand(n)

if det(A) == 0
    fprintf('The matrix A is singular\n');
else
    AM = [A eye(n)];
    RRAM = rref(AM);
    invA = RRAM(:,n+1:end)
end
```

```
A =

    0.3063    0.6443    0.9390    0.2077    0.1948
    0.5085    0.3786    0.8759    0.3012    0.2259
    0.5108    0.8116    0.5502    0.4709    0.1707
    0.8176    0.5328    0.6225    0.2305    0.2277
    0.7948    0.3507    0.5870    0.8443    0.4357


invA =

   -2.7623    2.1694    0.6990    1.2835   -0.8346
    2.0489   -3.1304    0.5345    0.7343    0.1142
   -1.0711    3.4430    0.3566   -1.1318   -0.8549
   -3.4804    4.1083    2.7559   -2.7732   -0.2052
   11.5776  -14.0379   -7.5264    3.9664    5.2752
```

# Problem 2

Find the Doolittle factorization for A (or PA if necessary) and use it to solve the system Ax = b where

```
clear
clc

A = [ 3, -6, 9,  3;
      2,  1, 4,  1;
      1, -2, 2, -1;
      1, -2, 3, 0];

b = [ 1;
      2;
      3;
      4];

x = DoolitteFactorization(A , b)
```

# Probelm 3

Repeat Problem 2 for the following cofficient matrix:

```
clear
clc
A = [ 1,  1, -1, 0;
      1,  1,  4, 3;
      2, -1,  2, 4;
      2, -1,  2. 3;];

b = [ 1;
      2;
      3;
      4];

x = DoolitteFactorization(A , b)
```

# Problem 4

Determine the steady-state heat distribution in a thin square metal plate with dimensions 0.5 m by 0.5 m using n = 4 subintervals. Two adjacent boundaries are held at 0C, and the heat on the other boundaries increases linearly from 0C at one corner to 100C where the sides meet. To find a numerical solution for this problem, solve the linear system given in the lab instructions for temperatures wi at interior grid points

```
clear
clc

% specifies the number of sub intervals in either direction
n = 4;

[A, b] = CreateCoefficientMatrix(n);

% (a) What do you notice about the coefficient matrix?
```

```
%       The coefficient matrix is a banded matrix

% Use the LU  decomposition of the coeffcient matrix to solve the
 system
% and save the temperatures in a vector w.

[L, U, P] = lu(A); % note: no permutations needed to for LU decomp

w = DoolitteFactorization(A, b);

% (b) Produce the plot for the numerical solution of the temperature
% distribution with n = 4 intervals.

PlotNumericSol(n, w)


% (c) Redo parts (a) and (b) for n = 32 intervals.

n = 32;

[A, b] = CreateCoefficientMatrix(n);

[L, U, P] = lu(A); % note: no permutations needed to for LU decomp

w = DoolitteFactorization(A, b);

PlotNumericSol(n, w)

% (d) Update the code to use the temperature distributions f(x) =
 1600x^4 and
% f(y) = 1600y^4 at the non-zero boundaries and use n = 32 intervals.

% I do not know how to do that.
```

# Solve system using Doolittle

```
function x = DoolitteFactorization(A, b)
[L, U, P] = lu(A);

if ~isequal(P,eye(size(A)))
    %    PLUx = b
    % =>  LUx = (P^T)b
    b = (P.')*b
end

AM = [L, b];
y = ForwardSub(AM);
AM = [U, y];
x = BackSub(AM);
end


b =
```

```
3
2
1
4
```

# Backsubstitution Function

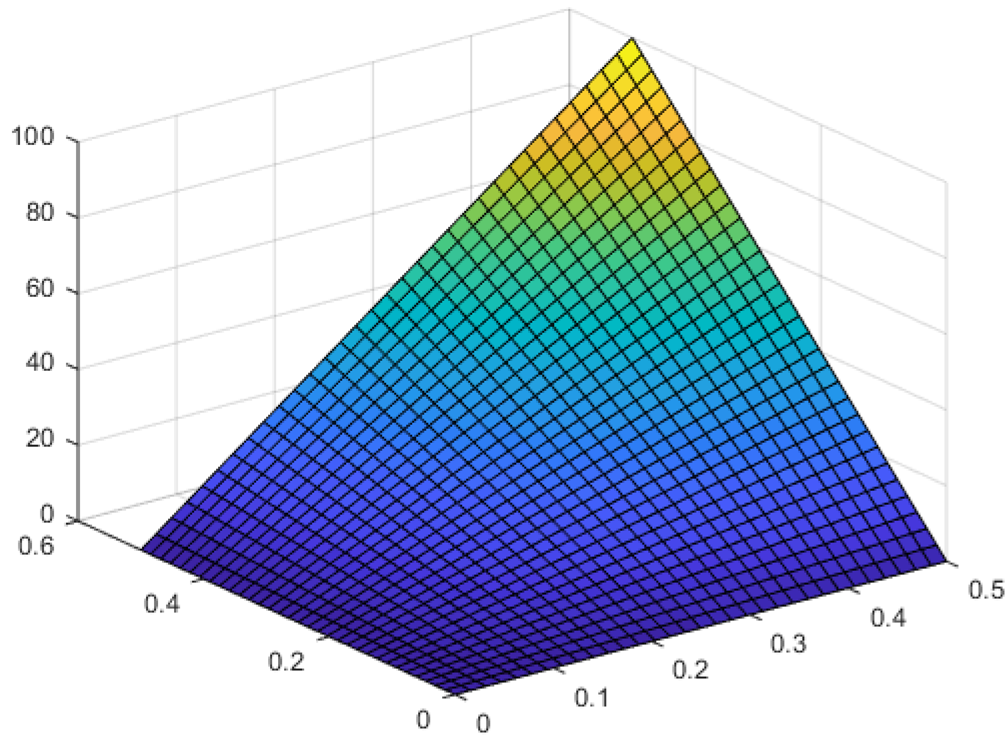
```
function x = BackSub(AM)
    [numRows, numCols] = size(AM);
    U = AM(:,1:end-1);
    b = AM(:,end);
    x = zeros(numRows,1);
    x(numRows) = b(numRows)/U(numRows,numRows);
    for i = 1: numRows-1
        AM;
        j= numRows - i;
        UsubT = U(j,j:end)';
        xsub = x(j+1:end,:);
        bsub = b(j);
        x(numRows-i) = (bsub-dot(UsubT(2:end,:),xsub))./UsubT(1);
    end
    x;
end
```

```
x =

   -7.2000
    1.4000
    4.6667
   -3.6667
```

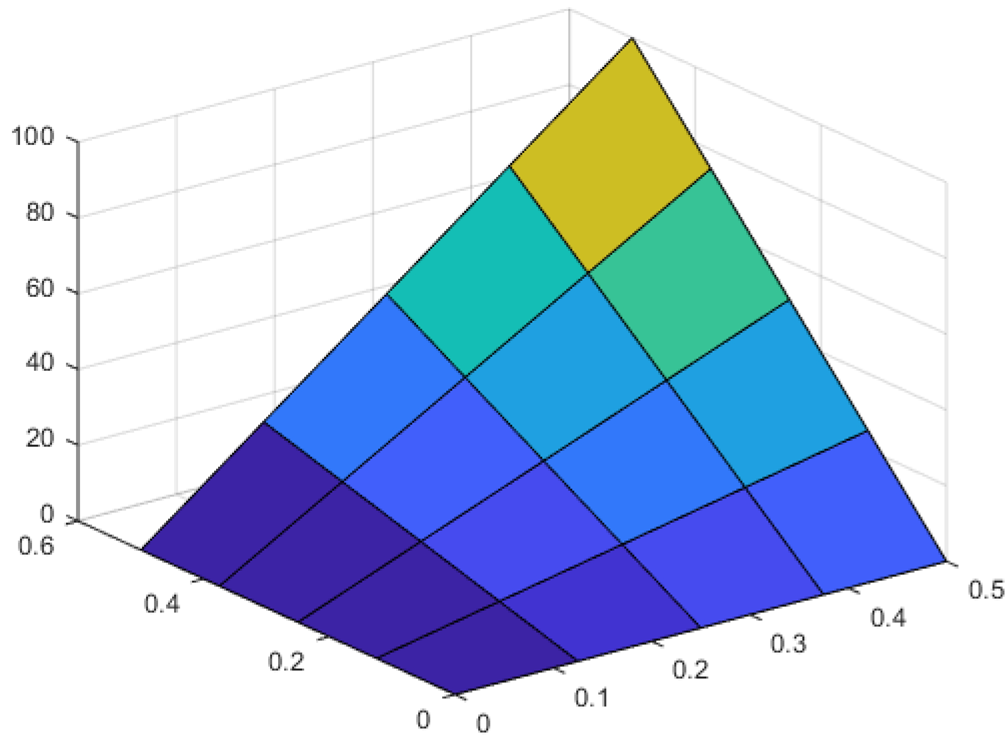
```
x =

    2.4000
   -0.6000
    0.8000
   -1.0000
```




# Forward-substitution Function

```
function y = ForwardSub(AM)
[numRows, numCols] = size(AM);
L = AM(:,1:end-1);
b = AM(:,end);
y = zeros(numRows,1);
y(1) = b(1)/L(1,1);
for i = 2: numRows
    i;
    AM;
    LsubT = L(i,1:i)';
    ysub = y(1:i-1,:);
    bsub = b(i);
    LsubT(1:i-1,:);
    y(i) = (bsub-dot(LsubT(1:i-1,:),ysub))./LsubT(i);
end
y;
end
```




# Helper functions for question 4. Pulled from lab.

```
function [A, b] = CreateCoefficientMatrix(n)

% creates coeffient matrix
A = toeplitz([4 -1 zeros(1,n-3) -1 zeros(1, (n-2)*(n-1)-1)]);

% updates specific coeffient matrix entries in the identy blocks
for i = 1:n-2
    A(i*(n-1), i*(n-1)+1) = 0;
    A(i*(n-1)+1, i*(n-1)) = 0;
end

% creates the right-hand side vector
b = zeros((n-1)*(n-1), 1);

% updates entries based on the boundry conditions
b(1:n-1) = b(1:n-1) + 100/n*[1:n-1]';
b(n-1:n-1:end) = b(n-1:n-1:end) + 100/n*[n-1:-1:1]';

end

function [] = PlotNumericSol(n, w)
```

```
% creates a matrix for temperatures at all grid points
W = zeros(n+1, n+1);

% inserts temperatures at interior grid points
W(2:end-1, 2:end-1) = reshape(w, n-1, n-1)';

% inserts temperatures on the boundaries
W(1, :) = 100/n*[0:n];
W(:, end) = 100/n*[n:-1:0];

% creates vectors for plotting
x = 0.5/n*[0:n];
y = 0.5/n*[n:-1:0];

[xx, yy] = meshgrid(x,y);

surf(xx, yy, W)
end
```

*Published with MATLAB® R2018b*