Table of Contents

Dummy

```
%Ignore this. It just makes the published file outputs appear in he
 right spot

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

P = FindP(A);
[L, ~] = lu(P*A);

fprintf("Lower Matrix\n");
disp(L);
```

Part 1: Solve the system using the Doolittle factorization

```
fprintf("Part 1: Solve the system using the Doolittle factorization
\n");

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

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

P = FindP(A);
[L, U] = lu(P*A);

solution = solve(L, U, P, b);

fprintf("Lower Matrix\n");
disp(L);
fprintf("Upper Matrix\n");
disp(U);
fprintf("Solution for x values\n");
disp(solution);
```

```
Part 1: Solve the system using the Doolittle factorization
Lower Matrix
    1.0000         0         0         0
    0.6667    1.0000         0         0
    0.3333         0    1.0000         0
```

```
    0.3333         0         0    1.0000

Upper Matrix
     3    -6     9     3
     0     5    -2    -1
     0     0    -1    -2
     0     0     0    -1

Solution for x values
   -7.2000
    1.4000
    4.6667
   -3.6667
```

Part 2: Solve the system using the Doolittle factorization

```
fprintf("Part 2: Solve the system using the Doolittle factorization \n");

A = [1 1 -1 0; 1 1 4 3; 2 -1 2 4; 2 -1 2 3];
b = [1; 2; 3; 4];

P = FindP(A);
[L, U] = lu(P*A);

solution = solve(L, U, P, b);

fprintf("Lower Matrix\n");
disp(L);
fprintf("Upper Matrix\n");
disp(U);
fprintf("Solution for x values\n");
disp(solution);
```

```
Part 2: Solve the system using the Doolittle factorization
Lower Matrix
     1     0     0     0
     2     1     0     0
     1     0     1     0
     2     1     0     1

Upper Matrix
     1     1    -1     0
     0    -3     4     4
     0     0     5     3
     0     0     0    -1

Solution for x values
    2.4000
   -0.6000
```

```
   0.8000
  -1.0000
```

Part 3: Find the steady state heat distribution

```
fprintf("Part 3: Find the steady state heat distribution\n");

%Part A
fprintf("Part A\n");

%Code to Generate Matrix

n = 4;

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

for i = 1:n-2
    A(i*(n-1), i*(n-1)+1) = 0;
    A(i*(n-1)+1, i*(n-1)) = 0;
end

b = zeros((n-1)*(n-1), 1);

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]';

%Main Code
P = eye(size(A,1));
[L, U] = lu(A);

w = solve(L, U, P, b);

fprintf("w vector is\n");
disp(w);

%Part B
fprintf("Part B\n");

W = zeros(n+1, n+1);
W(2:end-1, 2:end-1) = reshape(w, n-1, n-1)';
W(1,:) = 100/n*[0:n];
W(:,end) = 100/n*[n:-1:0];

x = 0.5/n*[0:n];
y = 0.5/n*[n:-1:0];

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

figure;
surf(xx,yy,W);

%Part C
```

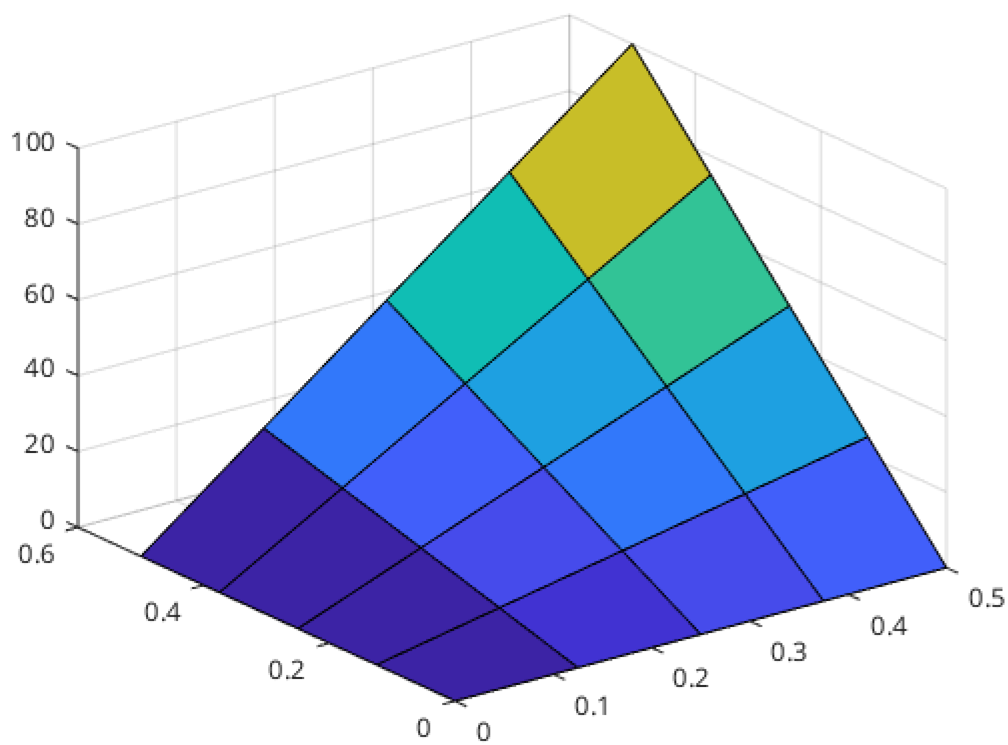
```
fprintf("Part C\n");

%Code to Generate Matrix

n = 32;

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

for i = 1:n-2
    A(i*(n-1), i*(n-1)+1) = 0;
    A(i*(n-1)+1, i*(n-1)) = 0;
end

b = zeros((n-1)*(n-1), 1);

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]';

%Main Code

P = eye(size(A,1));
[L, U] = lu(P*A);

w = solve(L, U, P, b);

W = zeros(n+1, n+1);
W(2:end-1, 2:end-1) = reshape(w, n-1, n-1)';
W(1,:) = 100/n*[0:n];
W(:,end) = 100/n*[n:-1:0];

x = 0.5/n*[0:n];
y = 0.5/n*[n:-1:0];

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

figure;
surf(xx,yy,W);

%Part D
fprintf("Part D\n");

b = zeros((n-1)*(n-1), 1);

b(1:n-1) = b(1:n-1) + 100*([1:n-1]'/n).^4;
b(n-1:n-1:end) = b(n-1:n-1:end) + 100*([n-1:-1:1]'/n).^4;

w = solve(L, U, P, b);

W = zeros(n+1, n+1);
W(2:end-1, 2:end-1) = reshape(w, n-1, n-1)';
W(1,:) = 100*([0:n]/n).^4;
W(:,end) = 100*([n:-1:0]'/n).^4;

x = 0.5/n*[0:n];
```

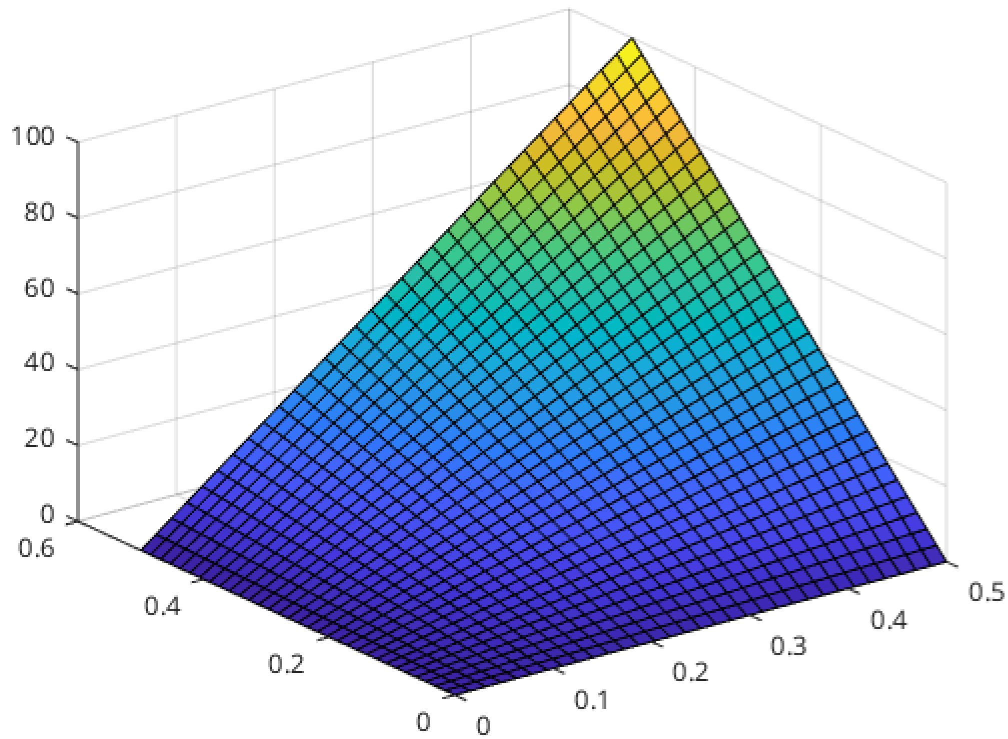
```
y = 0.5/n*[n:-1:0];

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

figure;
surf(xx,yy,W);
```

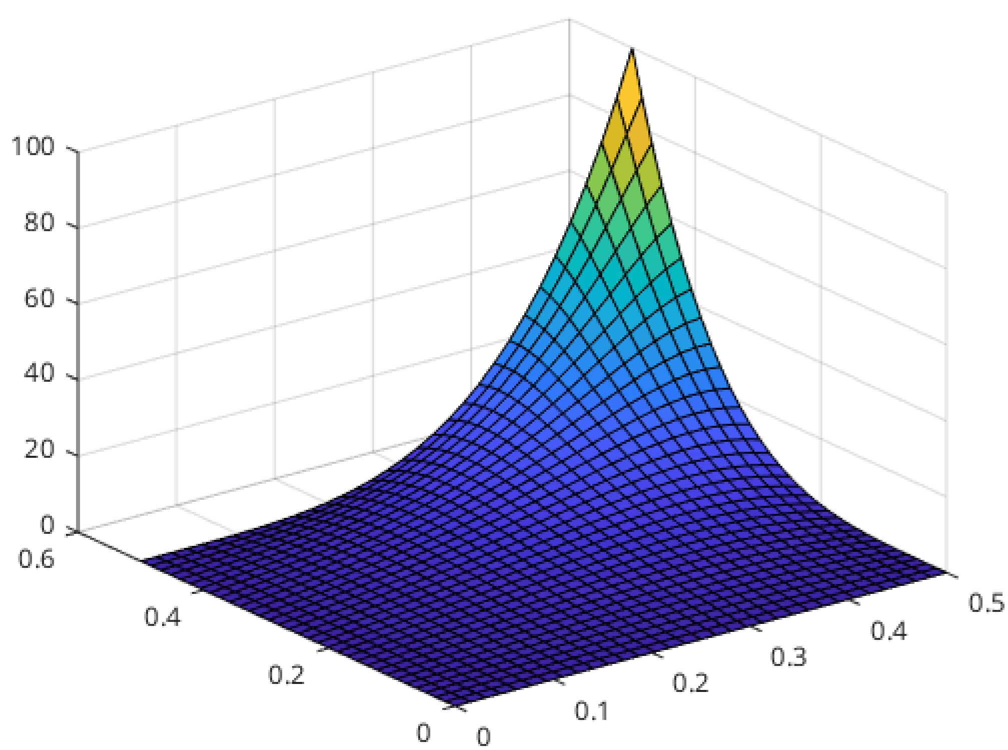
Part 3: Find the steady state heat distribution
Part A
w vector is

```
   18.7500
   37.5000
   56.2500
   12.5000
   25.0000
   37.5000
    6.2500
   12.5000
   18.7500
```

Part B
Part C
Part D




100
80
60
40
20
0
0.6
0.4
0.2
0
0
0.1
0.2
0.3
0.4
0.5


100
80
60
40
20
0
0.6
0.4
0.2
0
0
0.1
0.2
0.3
0.4
0.5

Local Functions

```
function P = FindP(A)
n = size(A, 1);
P = eye(n);

for i=1:n-1
    if (A(i,i) == 0)
       rowToSwapWith = find(A(i+1:end,i),1) + i; % finds first row
 after i with a non-zero value to swap with
       tempRowP = P(rowToSwapWith,:);
       P(rowToSwapWith,:) = P(i,:);
       P(i,:) = tempRowP;
       tempRowA = A(rowToSwapWith,:);
       A(rowToSwapWith,:) = A(i,:);
       A(i,:) = tempRowA;
    end
    for j=i+1:n
        m = A(j,i)/A(i,i);
        A(j,:) = A(j,:) - (m*A(i,:));
    end
end

end

function [L, U] = lu(A)
n = size(A, 1);
L = zeros(n);
U = zeros(n);

for i = 1:n
    L(i,i) = 1;
    for j = i:n
        U(i,j) = A(i,j);
        for m = 1:(i-1)
            U(i,j) = U(i,j) - L(i,m)*U(m,j);
        end
    end
    for k = (i+1):n
        L(k,i) = A(k,i);
        for m = 1:(i-1)
            L(k,i) = L(k,i) - L(k,m)*U(m,i);
        end
        L(k,i) = L(k,i)/U(i,i);
    end
end
end

function x = solve(L, U, P, b)
y = ForwardSubstitution([L P*b]);
x = BackSubstitution([U y]);
end
```

```
function x = BackSubstitution(C)
n = size(C, 1);
solution = zeros(n,1);
solution(end) = C(end,end)/C(end,end-1);
for i=n-1:-1:1
    C(i,end) = C(i,end)-sum(C(i,i+1:end-1).*(solution(i+1:end))');
    solution(i) = (C(i,end))/C(i,i);
end
x = solution;
end

function x = ForwardSubstitution(C)
n = size(C, 1);
solution = zeros(n,1);
solution(1) = C(1,end)/C(1,1);
for i=2:n
    C(i,end) = C(i,end)-sum(C(i,1:i-1).*(solution(1:i-1))');
    solution(i) = (C(i,end))/C(i,i);
end
x = solution;
end
```

```
Lower Matrix
    1.0000         0         0         0
    0.6667    1.0000         0         0
    0.3333         0    1.0000         0
    0.3333         0         0    1.0000
```

Published with MATLAB® R2020a