Zan Aslett + Dalin Spencer
Physics 513R
Linear Algebra Group

1. $-u'' + Vu' = f(x)$ $V = \gamma$ $u(0) = u(1) = 0$

$f(x) = 1$

$$\Downarrow$$

$\phi(x)$ is smooth

$\phi(0) = \phi(1) = 0$

$$-\int_0^1 \phi u'' dx + \int_0^1 \phi \gamma u' dx = \int_0^1 \phi dx$$

$$\int_0^1 \phi' u' dx + \gamma \int_0^1 \phi u' dx = \int_0^1 \phi dx$$

$$A_1(u,\phi) = \int_0^1 \phi' u' dx \quad A_2(u,\phi) = \int_0^1 \gamma \phi u' dx \quad F(\phi) = \int_0^1 \phi dx$$

$$\boxed{A_1(u,\phi) + A_2(u,\phi) = F(\phi) \Rightarrow A(u,\phi) = F(\phi)}$$

b) let $u(x) = \sum_{i=1}^{m} u_i \phi_i(x)$

$$A\left(\sum_{i=1}^{m} u_i \phi_i(x), \phi_j\right) = F(\phi_j)$$

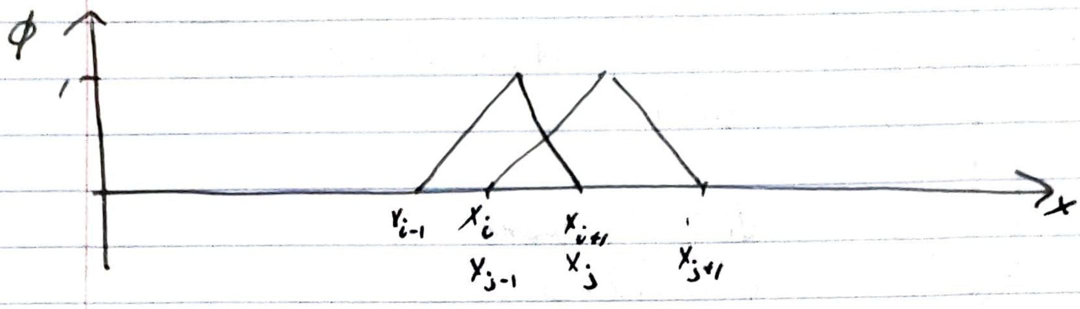
$$\sum_{i=1}^{m} u_i A(\phi_i, \phi_j) = F(\phi_j)$$

let $\hat{A}_{ji} = A(\phi_i, \phi_j)$

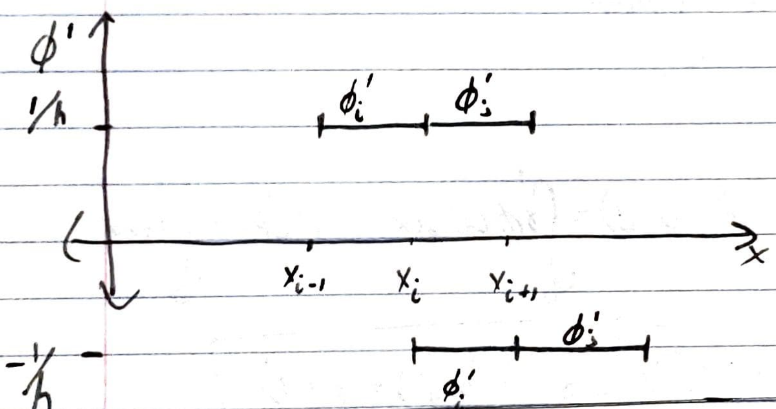
$\hat{x}_i = u_i$

$\hat{b}_j = F(\phi_j)$

$$\boxed{\hat{A}\hat{x} = \hat{b}}$$



$$A_1(\phi_i, \phi_j) = \int_0^1 \phi_i' \cdot \phi_j' \, dx$$



$$\int_0^1 \phi_i' \cdot \phi_j' \, dx = \frac{-1}{h} \quad \text{for } j = i+1$$

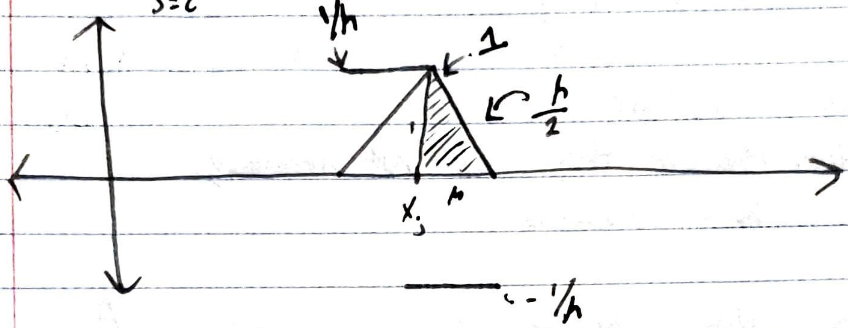
$$\int_0^1 \phi_i' \cdot \phi_j' \, dx = \frac{-1}{h} \quad \text{for } j = i-1$$

$$\int_0^1 \phi_i' \cdot \phi_j' \, dx = \frac{2}{h} \quad \text{for } j = i$$

0 otherwise

$$A_2(\phi_i, \phi_j) = \gamma \int \phi_j \cdot \phi_i' \, dx$$

$j=i$



$$\boxed{\gamma \int_0^1 \phi_j \cdot \phi_i' \, dx = \begin{cases} 0, & j=i \\ \gamma/2, & j=i-1 \\ -\gamma/2, & j=i+1 \\ 0, & \text{elsewhere} \end{cases}}$$

$$A = \begin{bmatrix} 2/h & -1/h - \gamma/2 & 0 & 0 \\ -1/h - \gamma/2 & 2/h & & 0 \\ 0 & -1/h - \gamma/2 & \ddots & \\ 0 & 0 & & 2/h \end{bmatrix}$$

$$F(\phi_j) = \int_0^1 \phi_j \, dx = \begin{cases} h/2 & j = 0, n-1 \\ h & \text{elsewhere} \end{cases}$$

4. $Ax = b$ let $M = A$,

$M^{-1}Ax = M^{-1}b$

$\tilde{A}x = \tilde{b}$ let $M^{-1}A = \tilde{A}$

$M^{-1}b = \tilde{b}$

thus these two equations are equivalent. However they will not necessarily produce the same residuals.

b) even though they require a matrix inverse, A_1 can be chosen such that it's inverse is relatively simple to compute

```
function [A,b] = popMatrices(n, gamma)
    h = 1/n;
    A = diag((2/h)*ones(1,n)) + diag((-1/h-gamma/2)*ones(1,n-1),-1) +
 diag((-1/h+gamma/2)*ones(1,n-1),1);
    b = zeros(n,1) + h;
    b(1) = h/2;
    b(end) = h/2;
```

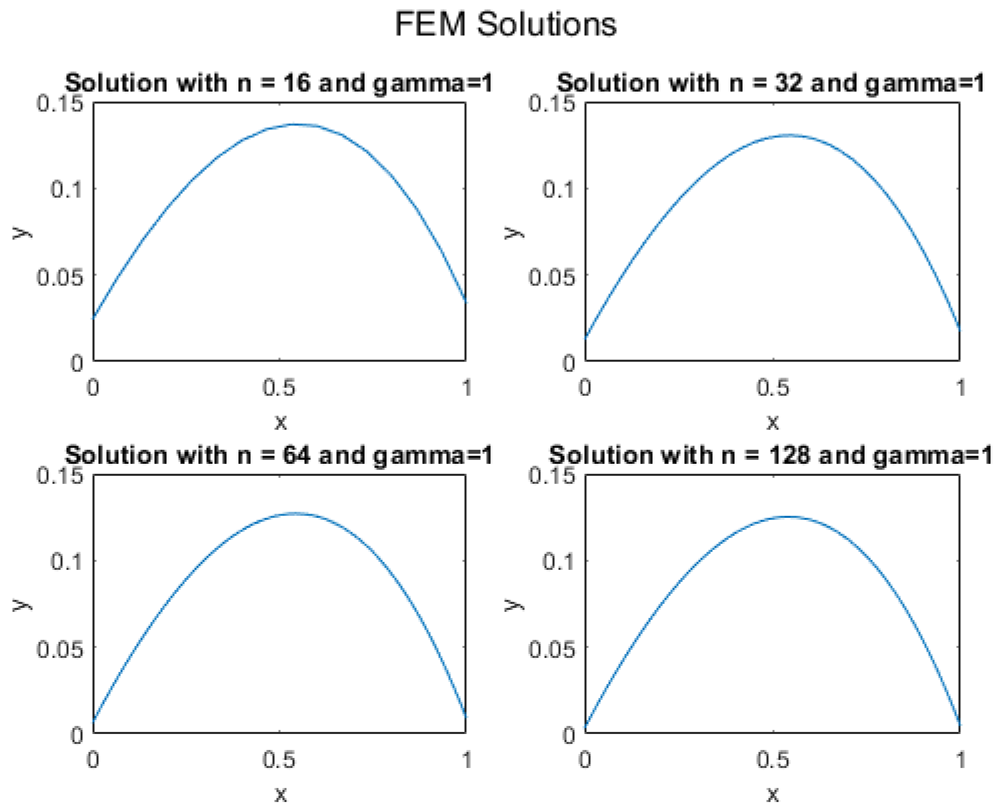
```
Not enough input arguments.

Error in popMatrices (line 2)
    h = 1/n;
```

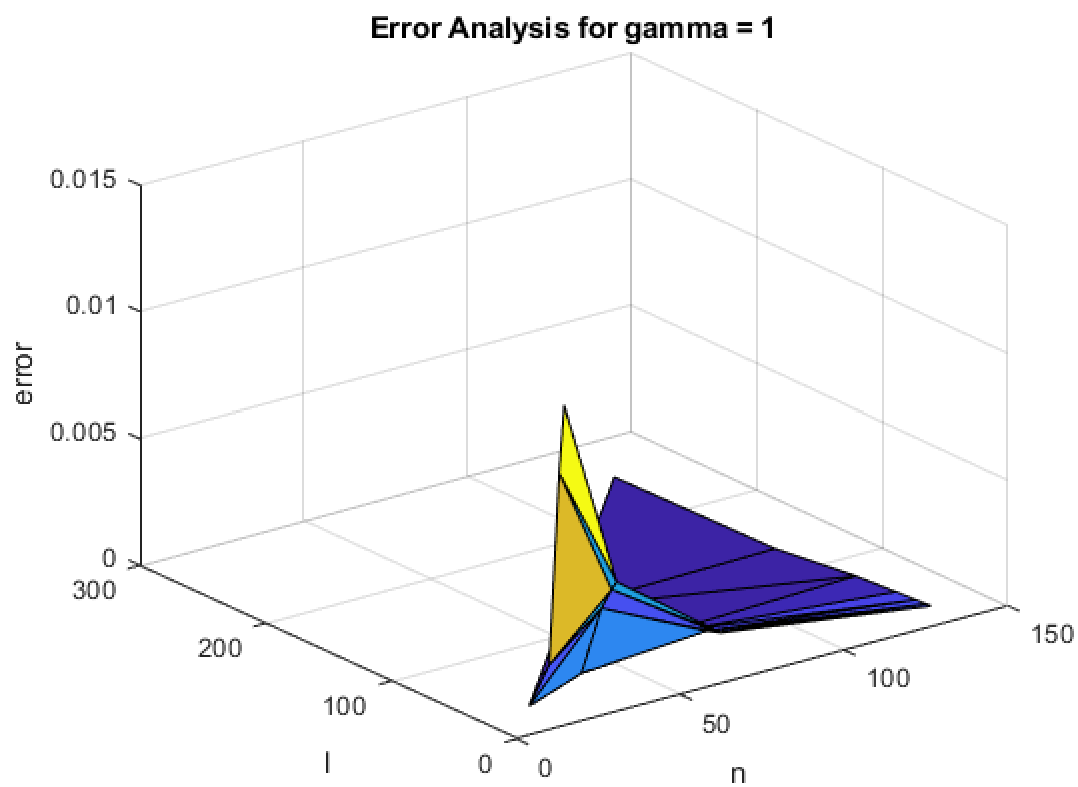
Published with MATLAB® R2019b

```
A=randi([10 100], 100, 100);
n = 100;
x0 = A(3,:)';
M = eye(100);

if det(A) ~=0
    b  = randi([10 100],100, 1);
    maxit = 100;
    tol=1e-1;
    [sol,xs,ys,Vs,Hs] = gmres_matlab(A,b,maxit,x0, M, n);
end
diff = b - A*sol;
```

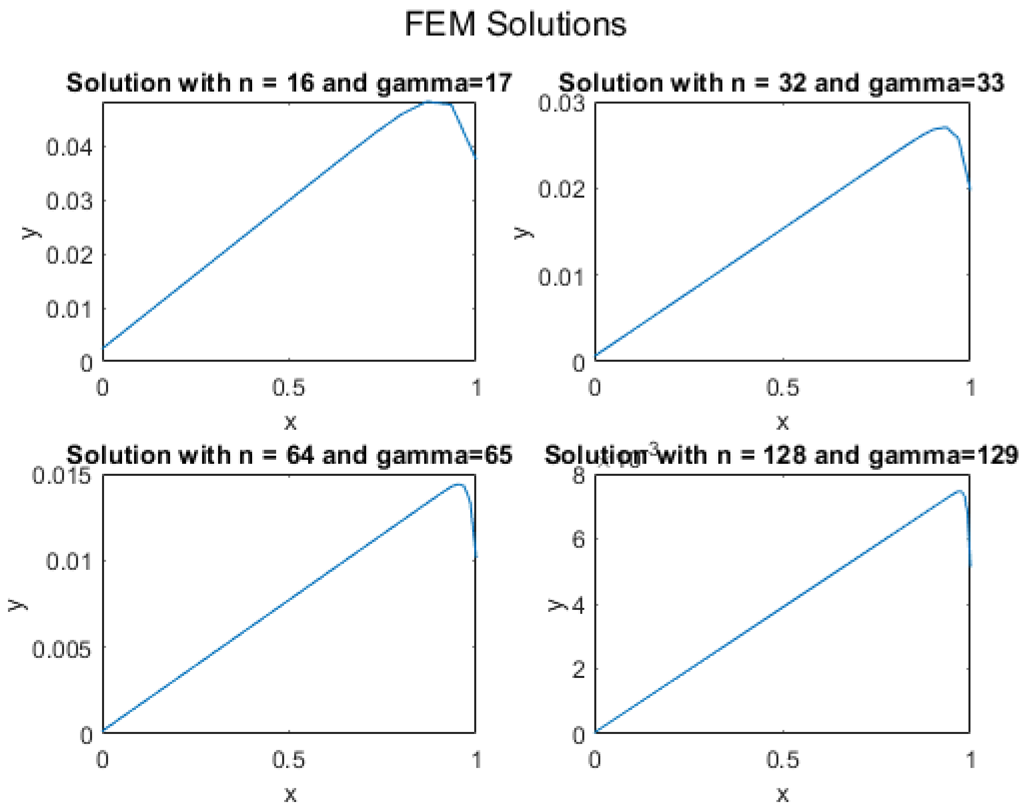
Published with MATLAB® R2019b

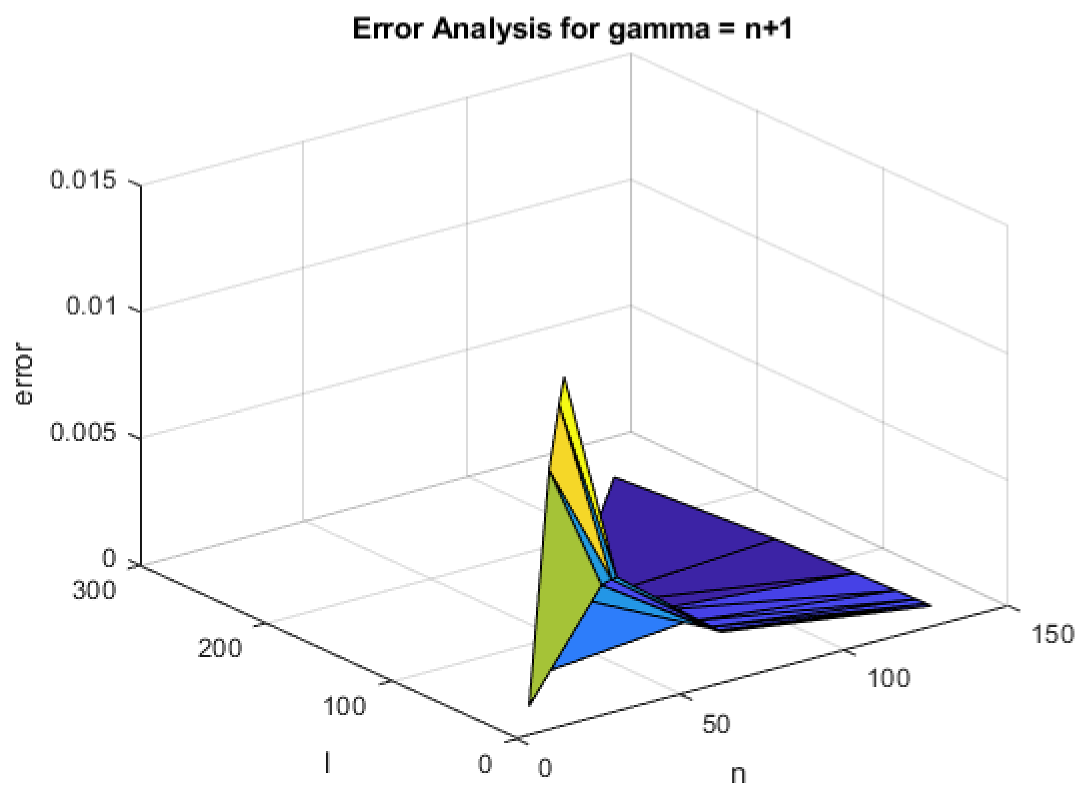
```
%gamma = 1
figure(1)
count = 1;
errMat = [];%[n l error]
sgtitle('FEM Solutions')
for n = [16,32,64,128]
    l = 2;
    error = 1;
    [A,b] = popMatrices(n,1);
    x0 = zeros(n,1);
    M = eye(n);
    while error > 10^(-6)
        [sol,xs,ys,Vs,Hs] = gmres_matlab(A,b,l,x0, M, n);
        res = b - A*sol;
        error = norm(res)/n;
        l = l * 2;
        errMat(end+1,:) = [n l error];
    end

    x = linspace(0,1,n);
    subplot(2,2,count)
    plot(x,sol)
    title(sprintf('Solution with n = %d and gamma=1',n))
    xlabel('x')
    ylabel('y')
    count = count + 1;
end
figure(2)
x = errMat(:,1);
y = errMat(:,2);
z = errMat(:,3);
dt = delaunayTriangulation(x,y) ;
tri = dt.ConnectivityList ;
trisurf(tri,x,y,z)
title('Error Analysis for gamma = 1')
xlabel('n')
ylabel('l')
zlabel('error')
```


FEM Solutions
Solution with n = 16 and gamma=1
Solution with n = 32 and gamma=1
Solution with n = 64 and gamma=1
Solution with n = 128 and gamma=1
x
y


Error Analysis for gamma = 1
error
l
n

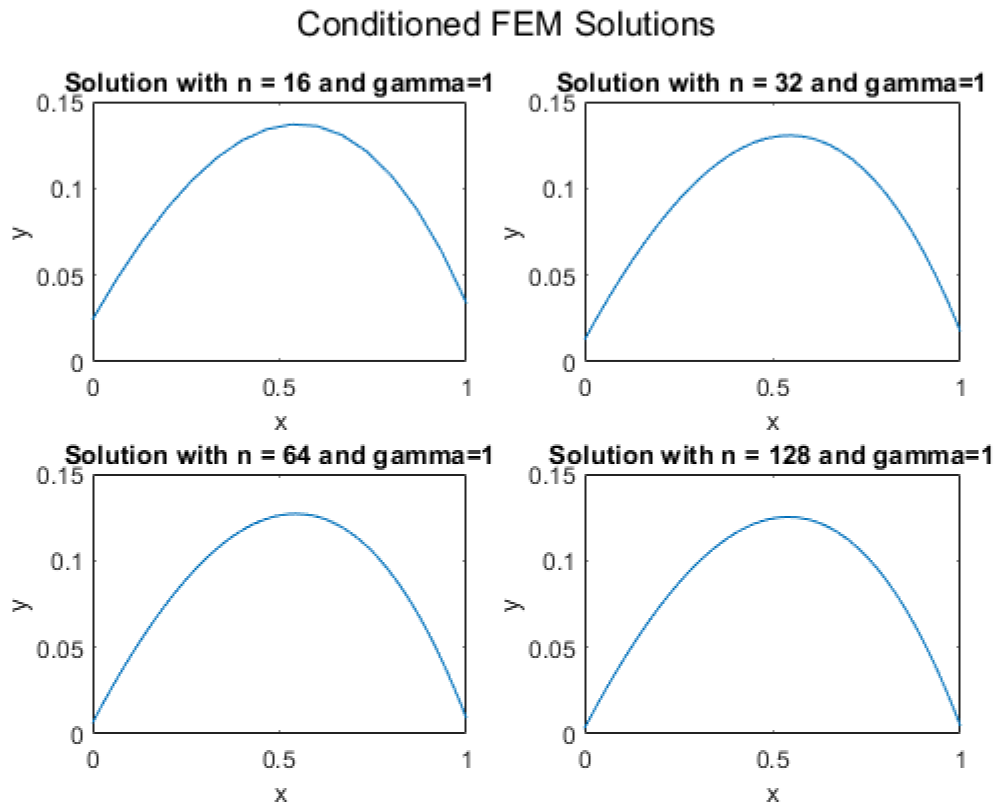
```
%gamma = n + 1
figure(3)
count = 1;
errMat = [];%[n l error]
sgtitle('FEM Solutions')
for n = [16,32,64,128]
    l = 2;
    error = 1;
    [A,b] = popMatrices(n,n+1);
    x0 = zeros(n,1);
    M = eye(n);
    while error > 10^(-6)
        [sol,xs,ys,Vs,Hs] = gmres_matlab(A,b,l,x0, M, n);
        res = b - A*sol;
        error = norm(res)/n;
        l = l * 2;
        errMat(end+1,:) = [n l error];
    end

    x = linspace(0,1,n);
    subplot(2,2,count)
    plot(x,sol)
    title(sprintf('Solution with n = %d and gamma=%d',n,n+1))
    xlabel('x')
    ylabel('y')
    count = count + 1;
end
figure(4)
x = errMat(:,1);
y = errMat(:,2);
z = errMat(:,3);
dt = delaunayTriangulation(x,y) ;
tri = dt.ConnectivityList ;
trisurf(tri,x,y,z)
title('Error Analysis for gamma = n+1')
xlabel('n')
ylabel('l')
zlabel('error')
```

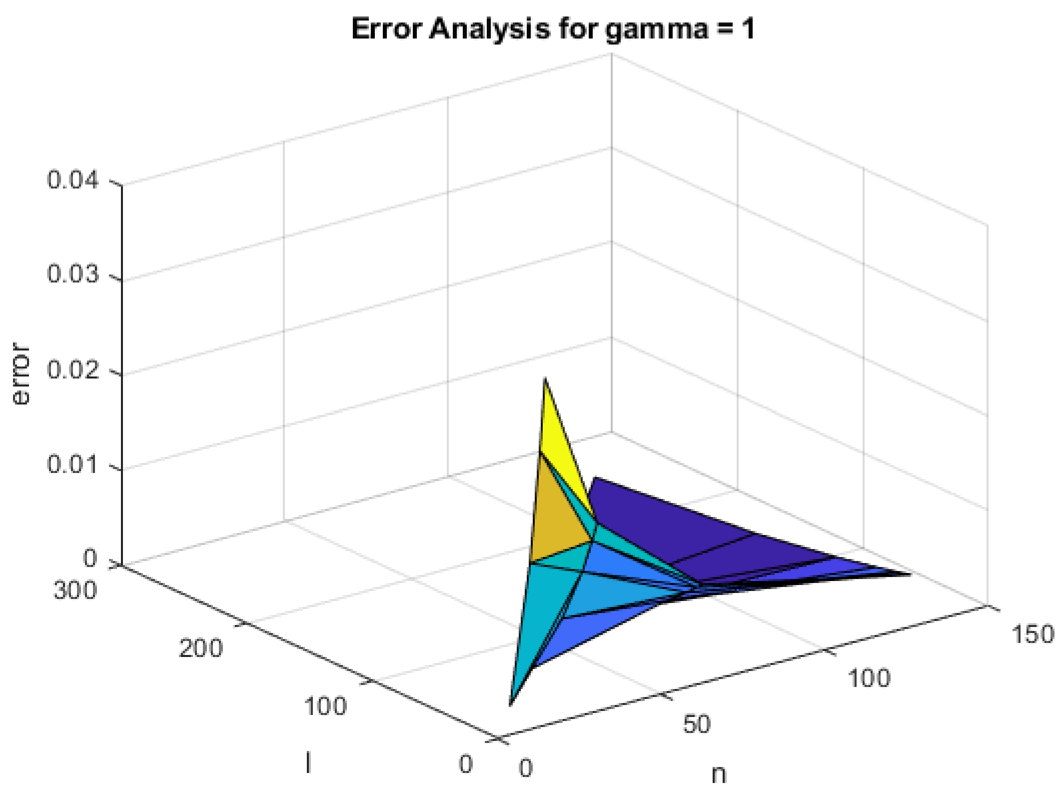

FEM Solutions
Solution with n = 16 and gamma=17
Solution with n = 32 and gamma=33
Solution with n = 64 and gamma=65
Solution with n = 128 and gamma=129
x
y


Error Analysis for gamma = n+1
error
l
n

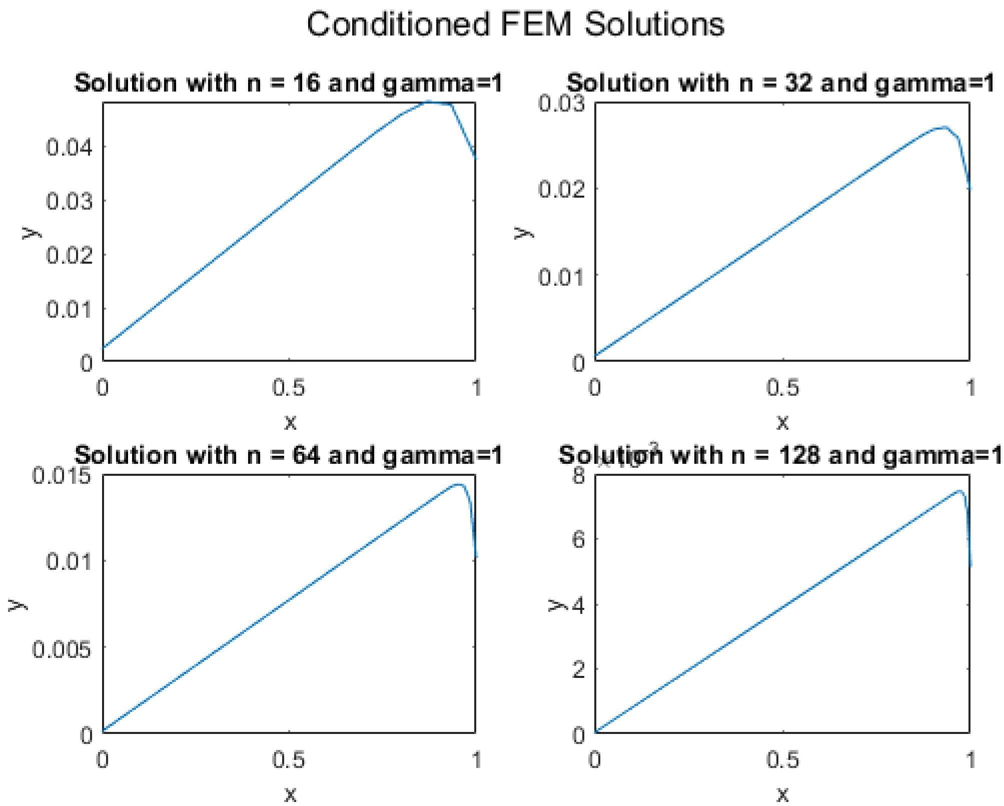
Published with MATLAB® R2019b

```
%gamma = 1
figure(1)
count = 1;
errMat = [];%[n l error]
sgtitle('Conditioned FEM Solutions')
for n = [16,32,64,128]
    l = 2;
    error = 1;
    [A,b] = popMatrices(n,1);
    x0 = zeros(n,1);
    M = eye(n);
    [L,~] = ilu(sparse(A));
    while error > 10^(-6)
        [sol,xs,ys,Vs,Hs] = gmres_matlab(inv(L)*A,inv(L)*b,l,x0, M,
 n);
        res = b - A*sol;
        error = norm(res)/n;
        l = l * 2;
        errMat(end+1,:) = [n l error];
    end

    x = linspace(0,1,n);
    subplot(2,2,count)
    plot(x,sol)
    title(sprintf('Solution with n = %d and gamma=1',n))
    xlabel('x')
    ylabel('y')
    count = count + 1;
end
figure(2)
x = errMat(:,1);
y = errMat(:,2);
z = errMat(:,3);
dt = delaunayTriangulation(x,y) ;
tri = dt.ConnectivityList ;
trisurf(tri,x,y,z)
title('Error Analysis for gamma = 1')
xlabel('n')
ylabel('l')
zlabel('error')
```

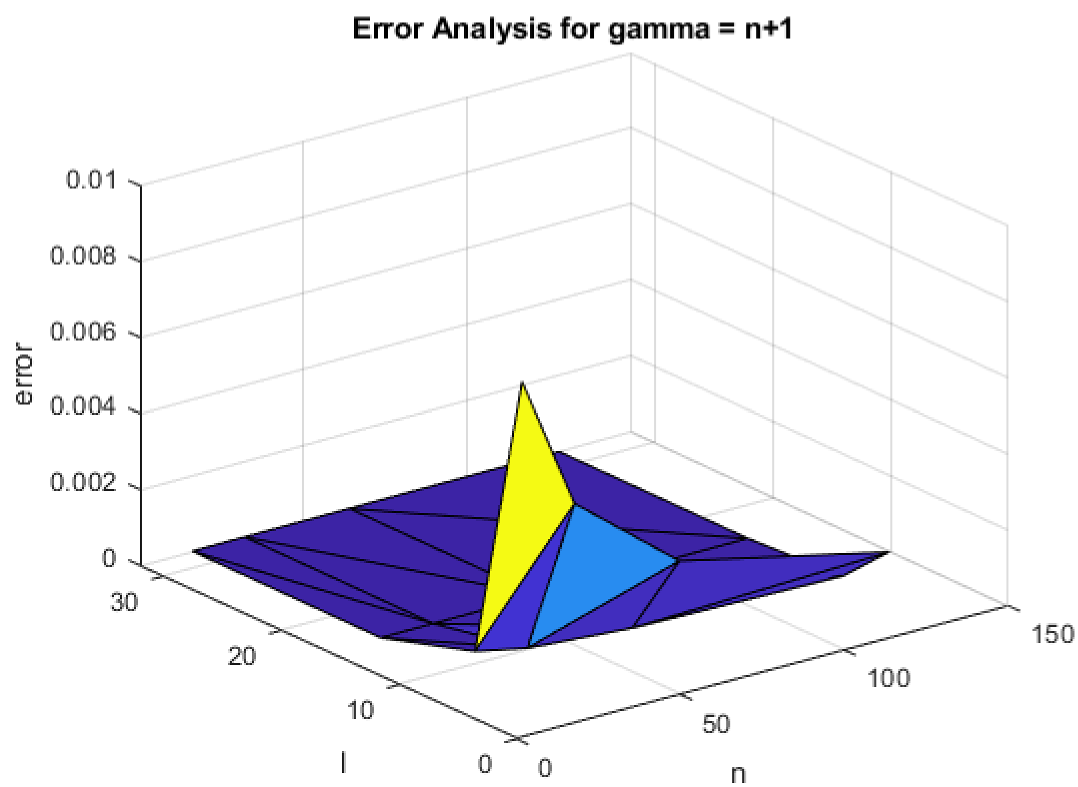

Conditioned FEM Solutions
Solution with n = 16 and gamma=1
Solution with n = 32 and gamma=1
Solution with n = 64 and gamma=1
Solution with n = 128 and gamma=1
0.15
0.1
0.05
0
0.5
1
x
y


Error Analysis for gamma = 1
0.04
0.03
0.02
0.01
0
error
300
200
100
0
l
0
50
100
150
n

```
%gamma = n + 1
figure(3)
count = 1;
errMat = [];%[n l error]
sgtitle('Conditioned FEM Solutions')
for n = [16,32,64,128]
    l = 2;
    error = 1;
    [A,b] = popMatrices(n,n+1);
    x0 = zeros(n,1);
    M = eye(n);
    [L,~] = ilu(sparse(A));
    while error > 10^(-6)
        [sol,xs,ys,Vs,Hs] = gmres_matlab(inv(L)*A,inv(L)*b,l,x0, M,
 n);
        res = b - A*sol;
        error = norm(res)/n;
        l = l * 2;
        errMat(end+1,:) = [n l error];
    end

    x = linspace(0,1,n);
    subplot(2,2,count)
    plot(x,sol)
    title(sprintf('Solution with n = %d and gamma=1',n))
    xlabel('x')
    ylabel('y')
    count = count + 1;
end
figure(4)
x = errMat(:,1);
y = errMat(:,2);
z = errMat(:,3);
dt = delaunayTriangulation(x,y) ;
tri = dt.ConnectivityList ;
trisurf(tri,x,y,z)
title('Error Analysis for gamma = n+1')
xlabel('n')
ylabel('l')
zlabel('error')

%The conditioned method converges MUCH quicker than the nonconditioned
%method. This is because the condition number is greatly improved when
 we
%left multiply by the triangular matrix L on both sides of the
 equation.
```



Conditioned FEM Solutions
Solution with n = 16 and gamma=1
Solution with n = 32 and gamma=1
Solution with n = 64 and gamma=1
Solution with n = 128 and gamma=1
x
y



Error Analysis for gamma = n+1
error
l
n

Published with MATLAB® R2019b