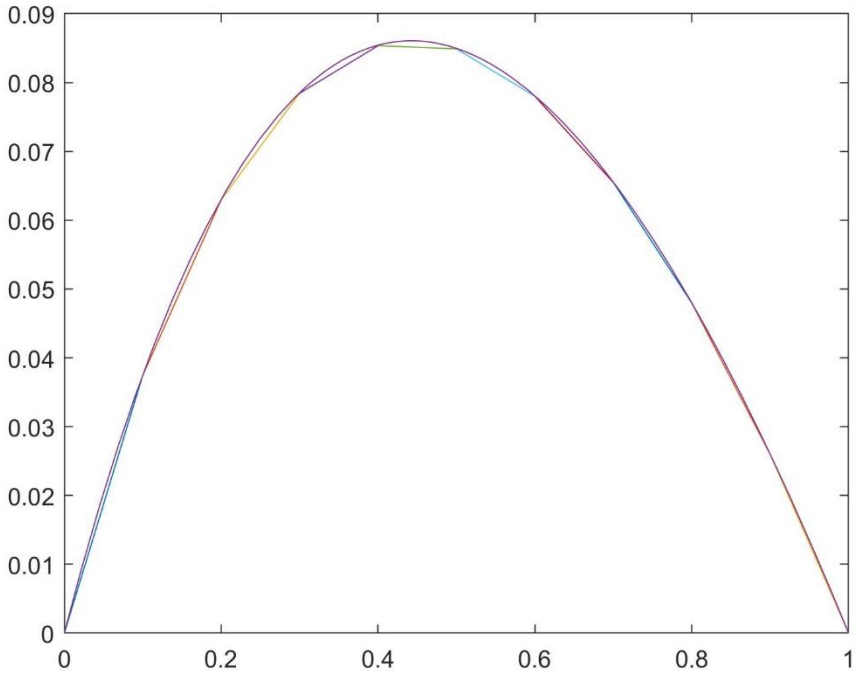
5168 Final Project

Liam Chen (chen.7976)

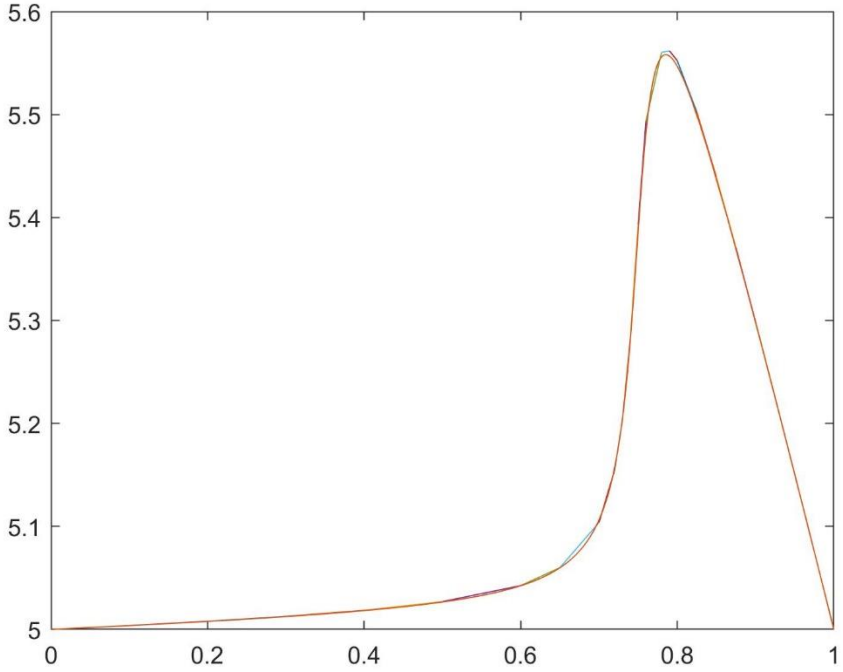
Civil Engineering Dept.

1.1

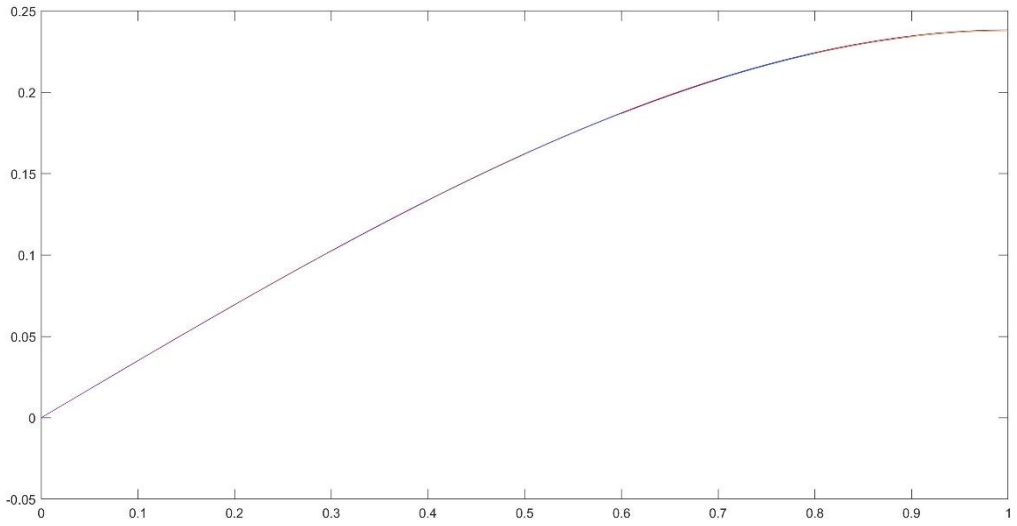
Homogenous Dirichlet:



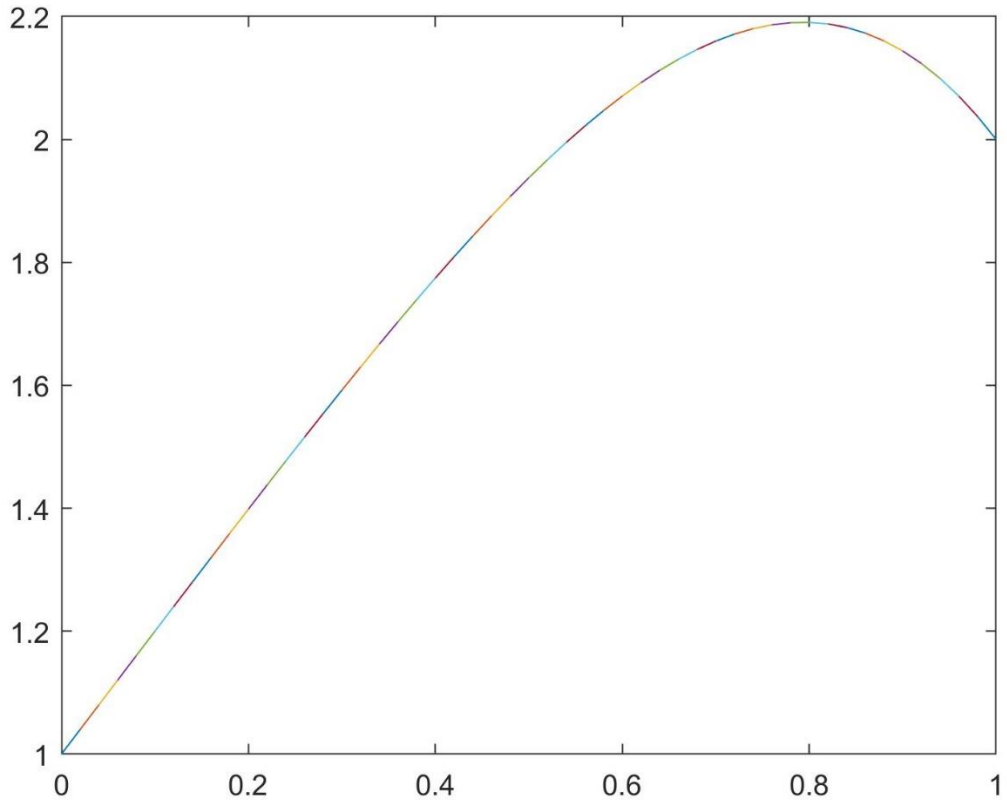
Inhomogenous Dirichlet:



Homogenous Mixed:



Inhomogenous Mixed (I plotted this one by itself because you can't tell the difference between the FEM and the exact solution on the same plot):



1.2

a.)

1.2

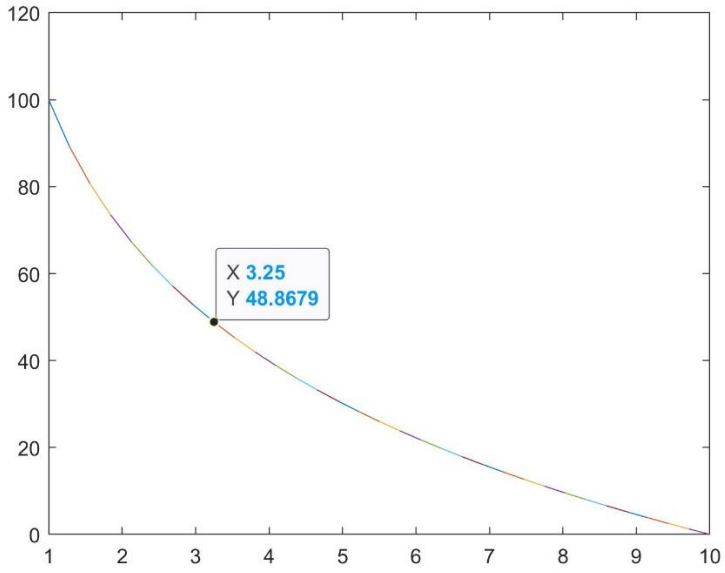
a.) strong form: $\frac{d}{dr}\left(-kr\frac{du}{dr}\right) = 0$, Ω: $1<r<10$, $u \in C^0(\Omega)$

$K = 1.7$, BC: $u(1) = 100$, $u(10) = 0$

Weak form: $\int_1^{10}\left(-kr\frac{du}{dr}\frac{dv}{dr}\right)dr = 0 \quad \forall v \in H_0^1$ where

$H_0^1 = \{g : g(1) = 100, g(10) = 0$ and $\int_1^{10} E\left(\frac{dg}{dr}\right)^2 dr < \infty\}$

b.)



Mesh 4 (left) gives an absolute error of |u(3.25)-48.8679|=0.0562 < 0.1

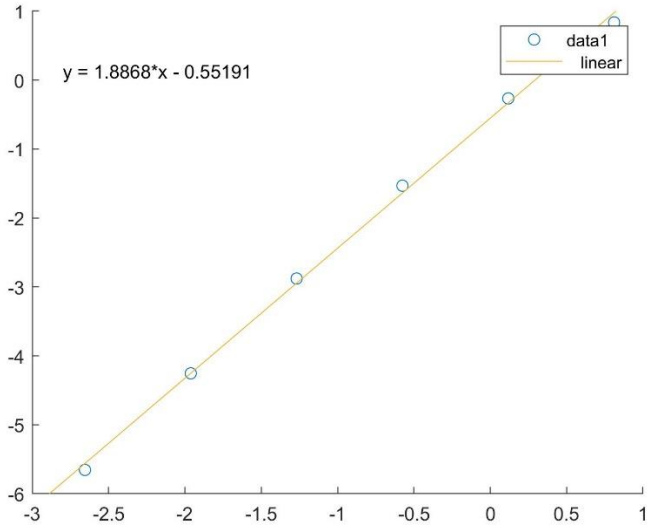
Mesh 3 gave an absolute error of |u(3.25)-49.0278|=0.216 > 0.1

So, Mesh 4 is the first mesh that gives the desired accuracy with 32 elements in the mesh.

c.)

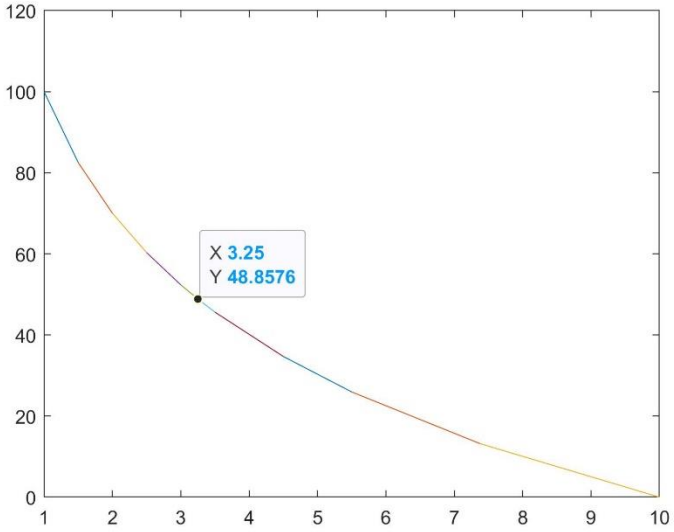
Absolute/Log error of meshes:

	Mesh 1	Mesh 2	Mesh 3	Mesh 4	Mesh 5	Mesh 6
Absolute	2.3102	0.7667	0.2161	0.0562	0.0142	0.0035
Log	0.8373	-0.2657	-1.5320	-2.8788	-4.2545	-5.6550



Average rate of convergence = 1.8868

d.)



Unstructured Mesh

The unstructured mesh gives an absolute error of |u(3.25)-48.8576|=0.0459

This is less than the absolute error given by Mesh 4 (0.0562). This is interesting, given that the unstructured mesh uses only 10 elements while Mesh 4 uses 32 elements. This is interesting, as it shows the benefits of using unstructured meshes over simply increasing the number of elements (and the finite-element method in general).

The accuracy at r=3.25 actually increases with the use of the unstructured mesh, because although there less elements overall, there is a higher "concentration" of elements near the point of interest.

e.)

Using the formulas given, we calculate the exact heat flux to be **73.8301**.

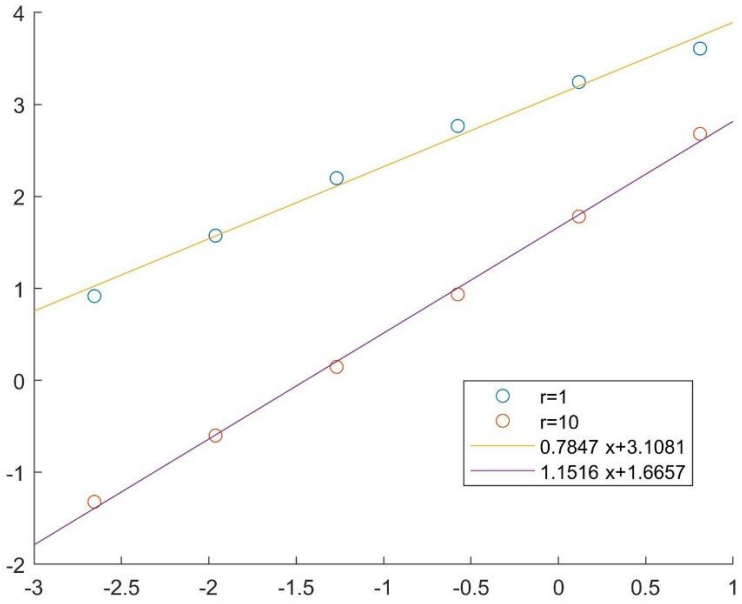
To calculate the heat flux using the FEM solution, this code was used:

```
flux_1 = -1.7*1*(100-u0(2))/(1-MESH.Points(2))
flux_10 = -1.7*10*(0-u0(nNodes-1))/(MESH.Points(nNodes)-
MESH.Points(nNodes-1))
```

u_h'(r) was calculated by determining the slope of the first/last piecewise element.

The first mesh that gave the desired accuracy (under 10% error) was Mesh 5, which had a relative error of (73.8301-69.0043)/73.8301 = **0.0654** at r = 1
and a relative error of (73.8301-74.3791)/73.8301 = **-0.0074** at r = 10.

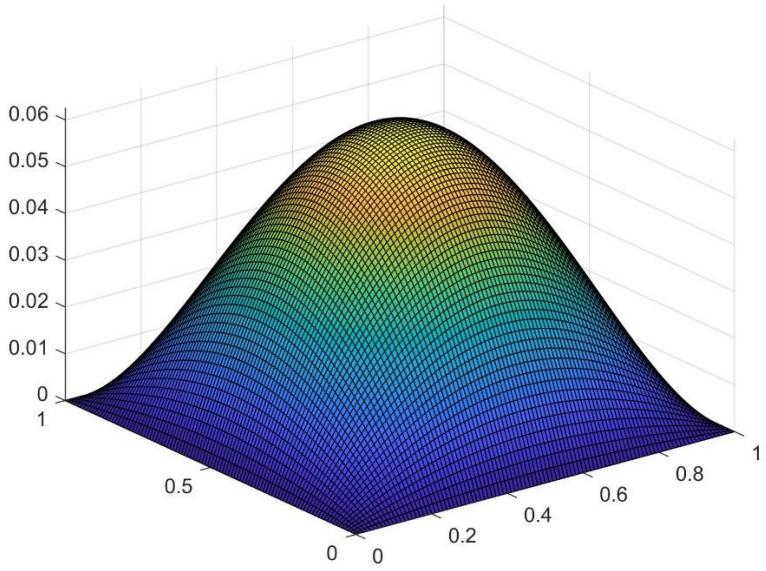
g.)



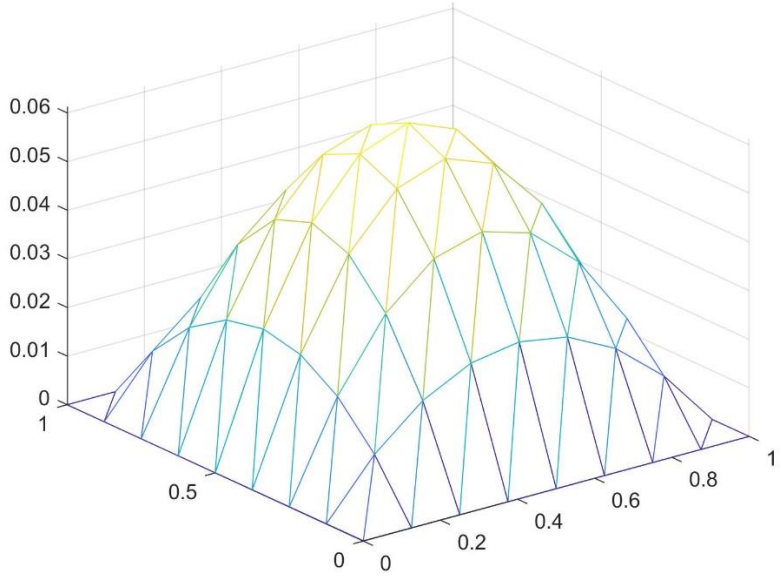
The rates of convergence are **0.7847** and **1.1516** for r = 1 and r = 10, respectively. These are both significantly smaller than the rate of convergence obtained in part c.

2.1

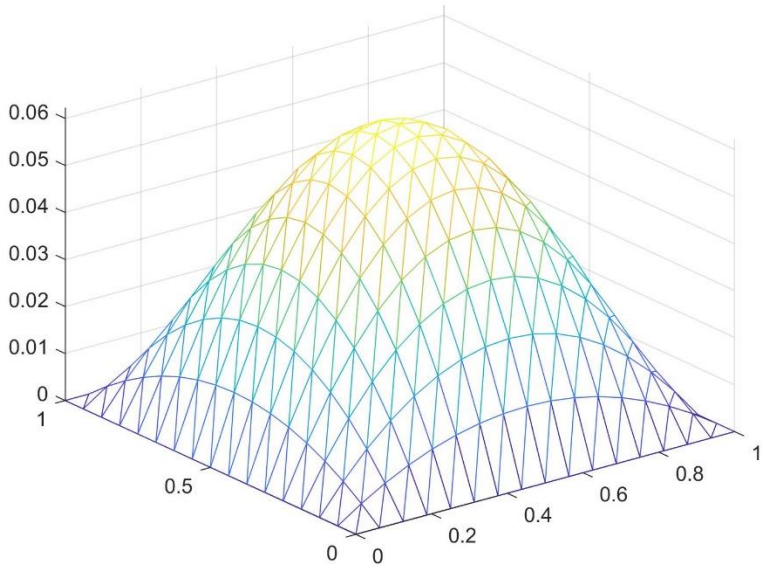
a.)



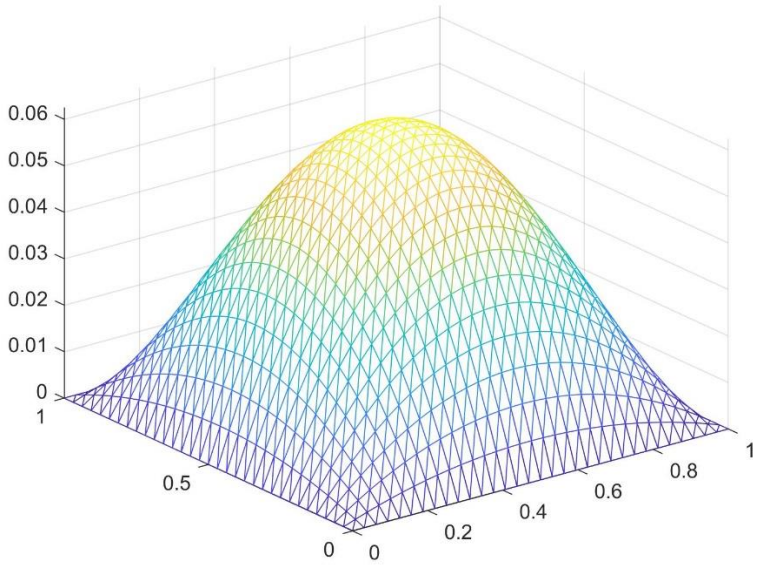
Exact solution



Mesh 1


0.06
0.05
0.04
0.03
0.02
0.01
0
1
0.5
0
0
0.2
0.4
0.6
0.8
1

Mesh 2


0.06
0.05
0.04
0.03
0.02
0.01
0
1
0.5
0
0
0.2
0.4
0.6
0.8
1

Mesh 3

b.) and c.)

L2 error norm	Mesh 1	Mesh 2	Mesh 3
Error	0.00145	0.00037	9.2385e-05

0.00145 /0.00047 = 3.919 factor error decrease from Mesh 1 to Mesh 2

Rate of convergence = [log(0.00037)-log(0.00145)]/[log(0.002)-log(0.0078)] = 1.004

0.00037/9.2385e-05 = 4.005 factor error decrease from Mesh 2 to Mesh 3

Rate of convergence = [log(9.2385e-05)-log(0.00037)]/[log(4.8828e-4)-log(0.002)] = 0.984

L_inf error norm	Mesh 1	Mesh 2	Mesh 3
Error	0.0023	0.00057	0.000143

0.0023/0.00057 = 4.035 factor error decrease from Mesh 1 to Mesh 2

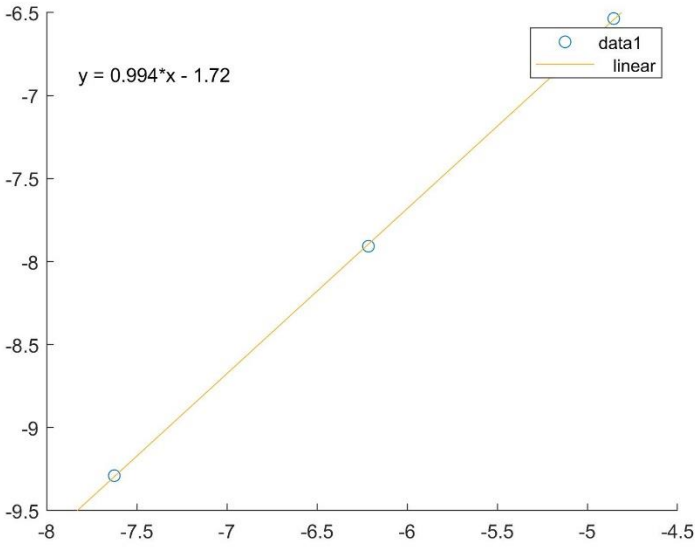
Rate of convergence = [log(0.00057)-log(0.0023)]/[log(0.002)-log(0.0078)] = 1.025

0.00057/0.000143 = 3.986 factor error decrease from Mesh 2 to Mesh 3

Rate of convergence = [log(0.000143)-log(0.00057)]/[log(4.8828e-4)-log(0.002)] = 0.981

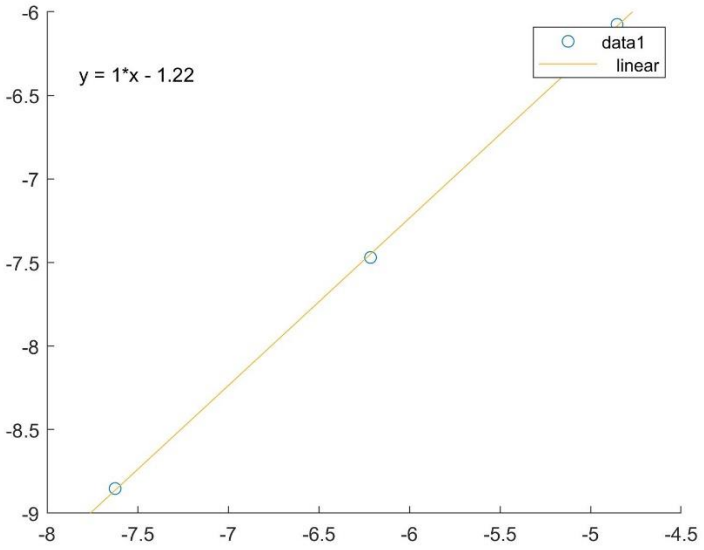
Since we are halving the element size with each refinement, we expect the error to theoretically decrease by a factor of Ch^2/(C(h/2)^2) = 4. This very closely matches with our actual error decrease.

d.)



log(L2 error) vs log(h)

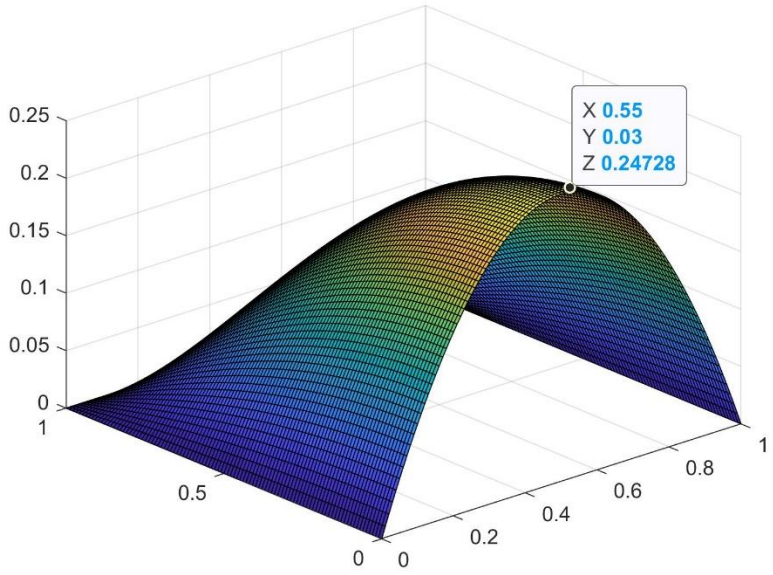
Average rate of convergence = 0.994



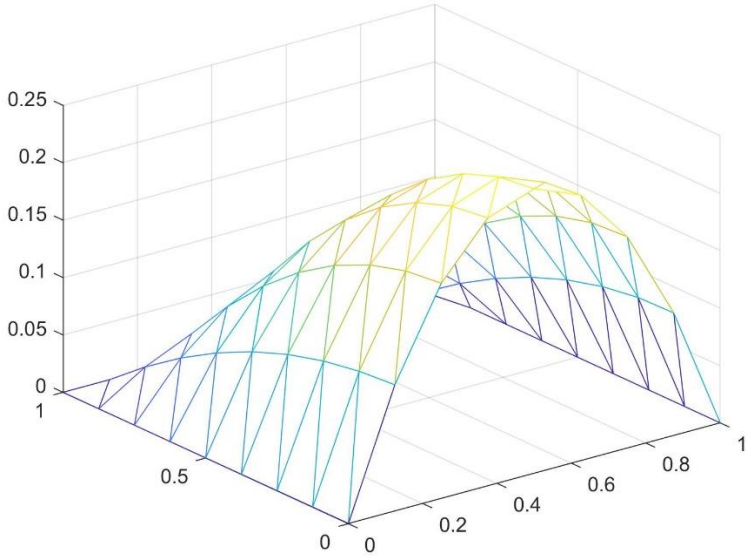
log(L_inf error) vs log(h)

Average rate of convergence = 1.000

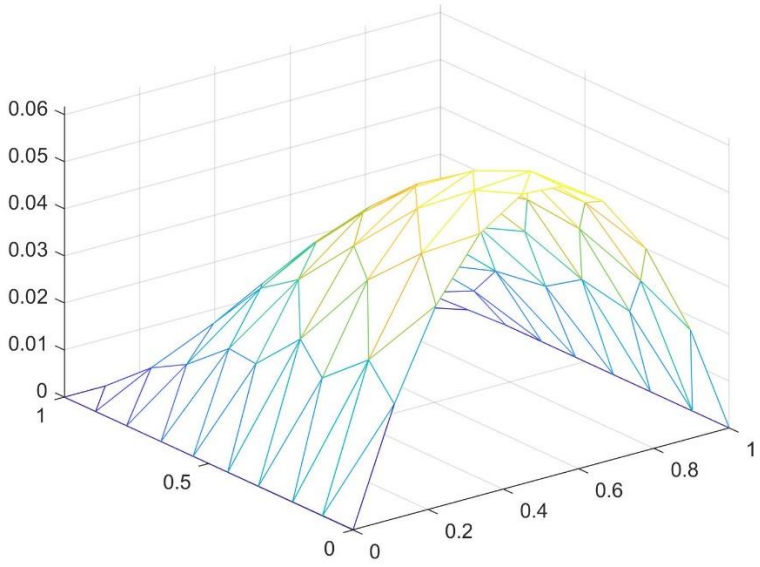
e.)



Exact solution



Structured mesh



Unstructured mesh

f.)

Structured Mesh:

L_2 error norm:0.0032739

L_inf error norm:0.0053067

Unstructured Mesh:

L_2 error norm:0.0026194

L_inf error norm:0.0043014

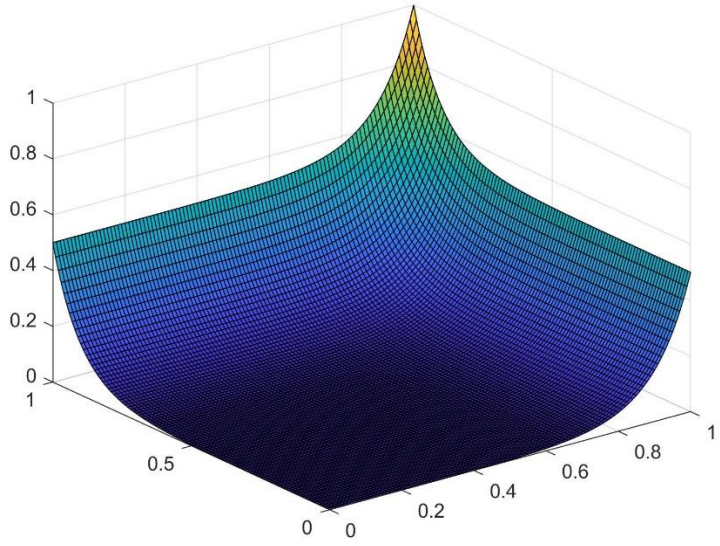
I found the error norms for the unstructured/structured mesh to be very similar.

Personally, I expected the unstructured mesh to return much smaller error norms, but I can see why there is such a small difference. My explanation would be that in certain cases, there is little benefit to using unstructured meshes because the solution is computationally “interesting” throughout its entire domain.

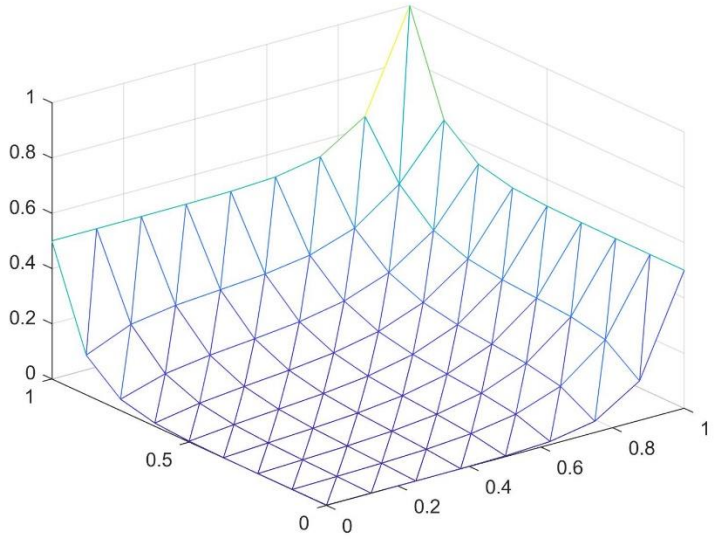
More rigorously, the unstructured mesh in this example does not use significantly smaller elements. Both the L2 norm and the L_inf norm are dependent on element size (L2, L_inf < Ch^2). Therefore, mathematically, there is no reason to expect a lower error norm simply by using an unstructured mesh instead of a structured mesh if there is no change in element size.

2.2

a.)

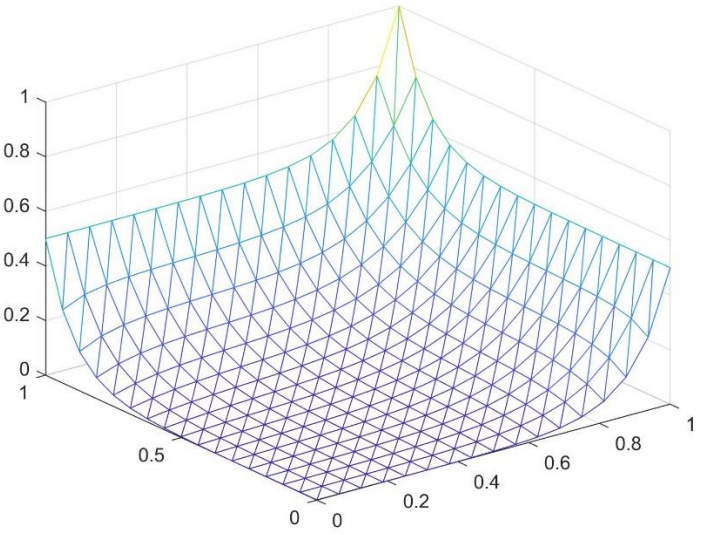


Exact solution



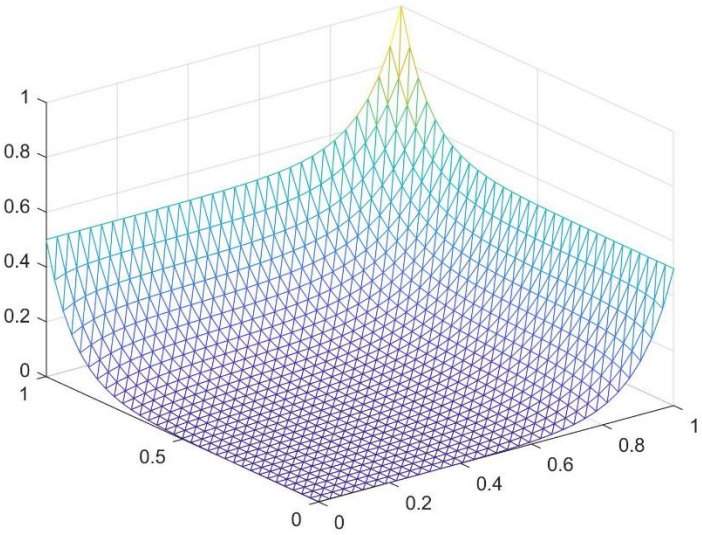
Mesh 1

Computational time: about 4 seconds



Mesh 2

Computational time: about 8 seconds



Mesh 3

Computational time: about 14 seconds

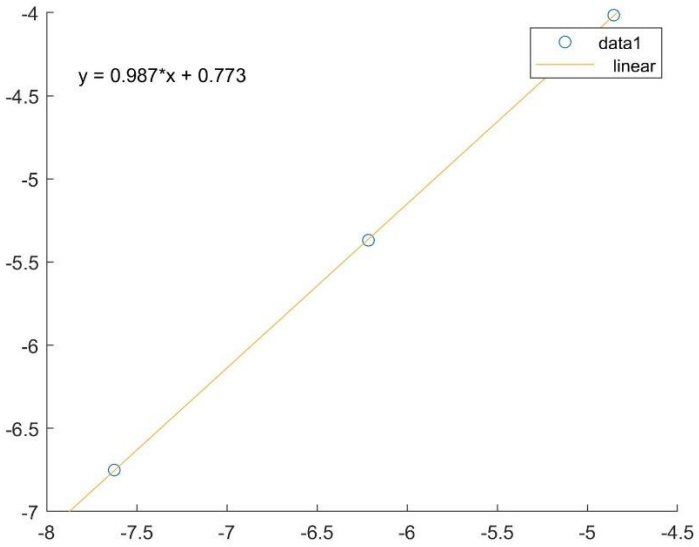
b.)

Although the problem solution does have a sharp edge, I do not observe any oscillations of any sort in any of the solutions. Since this problem has no advective term, you would not expect to see any oscillations.

c.)

	Mesh 1	Mesh 2	Mesh 3
L2 error norm	0.0180	0.0047	0.0012
L_inf error norm	0.0825	0.0267	0.0077

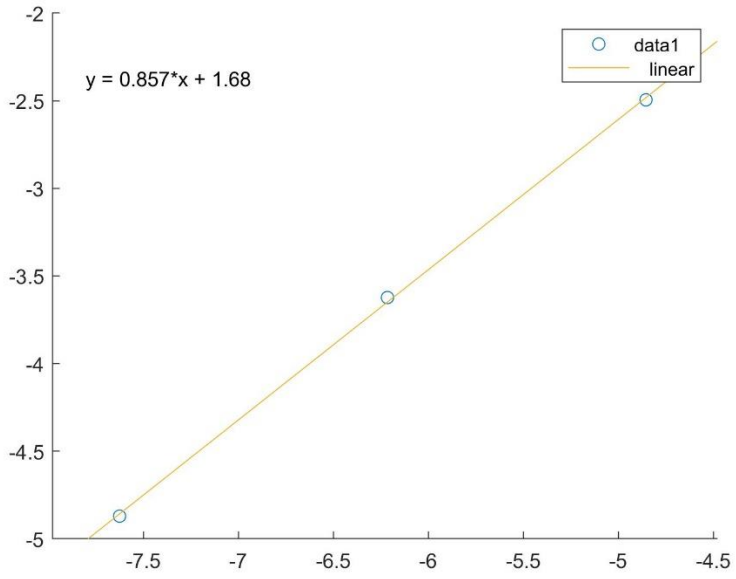
d.)



Log(L2) vs log(h)

This gives a rate of convergence of **0.987**.

Compared to the ROC of 2.1(d), we see that it is very similar.

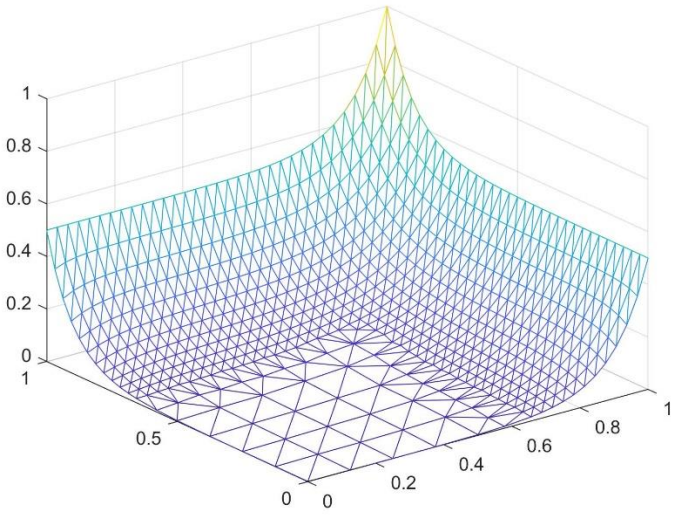


Log(L_inf) vs log(h)

Again omitting the Mesh 3 error data, we calculate an approximate rate of convergence of **0.857**.

This is somewhat smaller that the ROC from 2.1(d).

e.)



Unstructured mesh

L_2 error norm:0.0011767

L_inf error norm:0.0076672

Although the unstructured mesh took about half (~5.4 seconds) as long as Mesh 3 (~8.4 seconds), they had almost the same L2 and L_inf errors. From my observations, this is because in the exact solution, there is a region that is nearly flat without much variation. The unstructured mesh takes advantage of this, setting fewer elements around that region because the results will differ very slightly for those elements. Furthermore, in the region where there is significant variation (the sharp edge), the unstructured mesh uses a higher concentration of smaller elements. On the other hand, Mesh 3 simply uses a very fine uniform mesh across the entire domain, including over the flat region. Overall, the unstructured mesh only uses 1286 elements compared to mesh 3, with uses 2048 elements. Despite this, we see that they result in similarly accurate solutions.

This problem emphasizes the benefits of using an unstructured mesh over a structured mesh in certain cases.

Appendices (for partial credit just in case my calculations are wrong):

```
%main
psi = polyLagrange2D(p);
[K,F]=element2D(psi,KofXY,BofXY,FofXY,MESH);
[K,F]=enforce2DBCs(K,F,KofXY,boundaryValues,boundaryNodes);
u=K\F;
trimesh(MESH.ConnectivityList,MESH.Points(:,1),MESH.Points(:,2),u);
%FEM solution



figure(2);
```

```
%hold on;
x=0:0.01:1;
y=0:0.01:1;
[x,y]=meshgrid(x,y);

%2.1 Dirichlet exact solution

%u_exact=x.*y.*(1-x).*(1-y);


%2.1 Mixed exact solution

u_exact=(x.^2-x).*(y.^2-1);


%2.2 exact solution (Inhomogenous Dirichlet)

%u_exact=(cosh(10*x)+cosh(10*y))/(2*cosh(10));

surf(x,y,u_exact);
%-------------------------------------------------------------------------
-----


%error
%u_function=(@(x,y) (x*y*(1-x)*(1-y))); %2.1 Dirichlet exact
%u_function=(@(x,y) (x^2-x)*(y^2-1)); %2.1 Mixed exact
u_function=(@(x,y) (cosh(10*x)+cosh(10*y))/(2*cosh(10))); %2.2 exact

L_2=0; L_inf=0;
q=quadtriangle(2,'Domain',[0 0; 0 1; 1 0],'Type','nonproduct');
polyvals = zeros(3,3);

for k=1:3
    for i=1:3

polyvals(k,i)=bipolyval(psi(i).fun,[q.Points(k,1),q.Points(k,2)]);
    end
end

avg =[];

for i=1:nElems
    currElement = MESH.ConnectivityList(i,:);
    one = currElement(1); two = currElement(2); three =
currElement(3); %local node numbers

    x=[MESH.Points(one,1) MESH.Points(two,1) MESH.Points(three,1)];
    y=[MESH.Points(one,2) MESH.Points(two,2) MESH.Points(three,2)];

    u_coeffs=[u(one) u(two) u(three)];
```

```
    element_area = polyarea([x(1) x(2) x(3)], [y(1) y(2) y(3)]);
    avg = [avg element_area];

    for j=1:3


        u_h=dot(u_coeffs,polyvals(j,:));
        x_j=dot(x,polyvals(j,:));
        y_j=dot(y,polyvals(j,:));
        %disp(u_h);

        u_xy=u_function(x_j,y_j);

        L_2=L_2+q.Weights(j)*2*element_area*(u_xy-u_h)^2;

        L_inf=max(L_inf,abs(u_xy-u_h));
        %disp(L_2);

    end
end

L_2=sqrt(L_2);
```

```
function [K,F] = element2D(psi,KofXY,BofXY,FofXY,MESH)

fe = zeros(length(psi),1);
degree = length(psi)-1;

%q = quadtriangle(3);
q=quadtriangle(2,'Domain',[0 0; 0 1; 1 0],'Type','nonproduct');

nElems = length(MESH.ConnectivityList);
nNodes = length(MESH.Points);
ke = zeros(length(psi));
K=zeros(nNodes);
F=zeros(nNodes,1);

for n = 1:nElems

    currElement = MESH.ConnectivityList(n,:);
    one = currElement(1); two = currElement(2); three =
currElement(3); %local node numbers
    one_xy = MESH.Points(one,:); two_xy = MESH.Points(two,:); three_xy
= MESH.Points(three,:); %global coords of local nodes
    element_area = polyarea([one_xy(1) two_xy(1) three_xy(1)],
[one_xy(2) two_xy(2) three_xy(2)]);
    y_31 = three_xy(2)-one_xy(2); y_21 = two_xy(2)-one_xy(2);
    x_31 = three_xy(1)-one_xy(1); x_21 = two_xy(1)-one_xy(1);
    %disp(element_area)
```

```
    for i = 1:length(psi)

        for j = 1:i


ke(i,j)=(1/(4*element_area))*KofXY(n)*((y_31*psi(i).xider(2,2)-
y_21*psi(i).etader(2,2))*(y_31*psi(j).xider(2,2)-
y_21*psi(j).etader(2,2))...
                +(x_31*psi(i).xider(2,2)-
x_21*psi(i).etader(2,2))*(x_31*psi(j).xider(2,2)-
x_21*psi(j).etader(2,2)))...

+2*element_area*BofXY(n)*dot(q.Weights,bipolyval(psi(i).fun,q.Points).
*bipolyval(psi(j).fun,q.Points));

            ke(j,i) = ke(i,j);

            K(currElement(i),currElement(j)) =
K(currElement(i),currElement(j))+ke(i,j);

K(currElement(j),currElement(i))=K(currElement(i),currElement(j));
        end


fe(i)=(2)*element_area*FofXY(n)*dot(q.Weights,bipolyval(psi(i).fun,q.P
oints));

        F(currElement(i)) = F(currElement(i))+fe(i);

    end
end
```

```
function [K,F] = enforce2DBCs(K,F,KofXY,boundaryValues,boundaryNodes)
numDirichlet = length(boundaryNodes('Dirichlet'));
nNodes = length(K);
dirichletValues = boundaryValues('Dirichlet');
dirichletNodes = boundaryNodes('Dirichlet');

for currDirichlet = 1:numDirichlet
    for j =1:nNodes %columns
        F(j)=F(j)-
K(j,dirichletNodes(currDirichlet))*dirichletValues(currDirichlet);
        K(j,dirichletNodes(currDirichlet))=0;
    end
end

for currDirichlet = 1:numDirichlet
    K(dirichletNodes(currDirichlet),:)=zeros(1,nNodes);

    K(dirichletNodes(currDirichlet),dirichletNodes(currDirichlet))=1;
```

```
    F(dirichletNodes(currDirichlet))=dirichletValues(currDirichlet);
    %disp(K(currDirichlet,:));

end

end
```