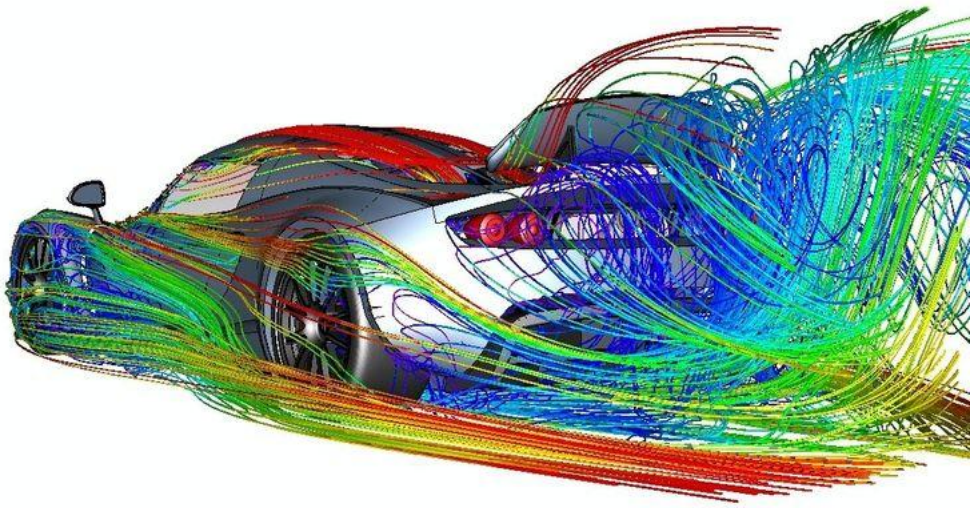
Partial Diffrential Equations: Continuity and Momentum differential equations in fluid flow in 1-D using Finite Element Method LaTeX Report

By
Aashutosh Trivedi
under the guidance of
Prof. Sanat Tiwari

November 30, 2018



Abstract

In this report, the solution of a partial differential equation using 2-D triangular element is shown. In particular a convection-diffusion equation is solved as a boundarylayer problem. Codes for 2-D mesh generation, equation for elemental nodes and assembling matrices written in matlab is also attached with the report.

1 Introduction

The description of the laws of physics for space and time-dependent problems are usually expressed in terms of partial differential equations (PDEs). For the vast majority of geometries and problems, these PDEs cannot be solved with analytical methods. Instead, an approximation of the equations can be constructed, typically based upon different types of discretizations. These discretization methods approximate the PDEs with numerical model equations, which can be solved using numerical methods. The solution to the numerical model equations are, in turn, an approximation of the real solution to the PDEs. FEM is one such technique for computing the approximate solution of a partial differential equation. There are other ways as well. Let's start by discussing it further. Let us say, we are given a PDE as a BVP and we need to solve it. We can have many possible cases, some of them are listed below:

1. Analytical solution, best but not always available.
2. FDM: Finite Difference Methods.
3. FEM: Finite Element Methods.
4. FVM: Finite Volume Methods.
5. MOM:method of moments.
6. Approximate solutions. There is always an error. The difference between the exact value and the approximate value is called the residual, we will denote it by R. For instance, lets suppose I have to solve the following ODE:

$$\frac{\mathrm{d}\hat{\phi}}{\mathrm{d}x} + \hat{\phi}(x) = 0$$

if my approximate solution is ϕ(x) = 1 + $a_1 x + a_2 x^2$, where a_1 and a_2 must be determined to get the ϕ as close to the exact solution $\hat{\phi}$ as I can. There are further methods for solving approximate PDE like Ritz Goodmans' method,etc.

The method of our concern is Weighted residual method:
In this method we determine the parameters a_1, a_2, a_3... by asking the residue to follow general set of equations

$$\int_D R w_i(x) dx = 0$$

where w_i with i = 1,2,3,...n are n arbritary functions called weighing functions. Some usual choices of w_i are given by:

- Collocation method
- Sub-domain method
- Least-Squares method
- Galerkin method

2 Basic Steps in Solving any Finite Element Problem

The basic procedure for solving partial differential equations (PDE) is as follows:

1. **Discretize the physical domain into finite elements :** Divide the solution into smaller regions that we call elements. The elements contain inside a certain number of points we call nodes. There are lots of shapes the elements can have. From segments of lines, triangles, squares, etc, to curved elements. It depends on problem, which element to choose. For instance, for a 1D problem, like a cylindrical rod with radial symmetry, the most simple is to take the elements as linear segments with two nodes per segment and discretize it. For a 2D problem, the most simple can be using triangular elements.

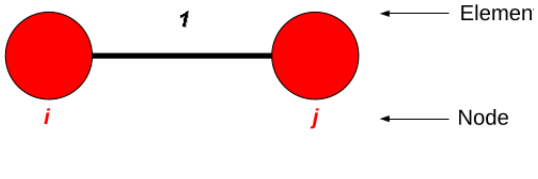


Figure 1: 1-D element

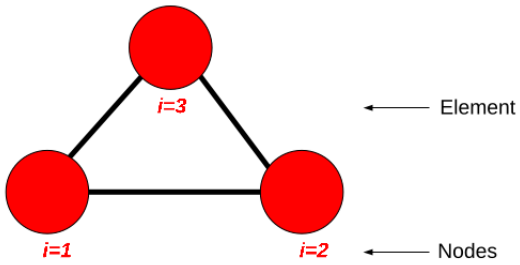


Figure 2: 2-D element

2. **Rewrite the PDE in its weak formulation by selecting the type of trial function to use, andin turn the shape functions:** The second point is selecting an appropriate trial function which is used to approximate the value of the unknown variable. This trial function can be any function but generally we prefer to use a polynomial function. At this point I will define two terms P_k and Q_k

$$P_k = \sum_{0 \leq \alpha_1+\alpha_2+\alpha_3,... \leq k} C_{\alpha_1\alpha_2\alpha_3\alpha_4} x_1^{\alpha_1} x_2^{\alpha_2} x_3^{\alpha_3} x_4^{\alpha_4} ...$$

$$Q_k = \sum_{0 \leq \alpha_1,\alpha_2,\alpha_3,... \leq k} C_{\alpha_1\alpha_2\alpha_3\alpha_4} x_1^{\alpha_1} x_2^{\alpha_2} x_3^{\alpha_3} x_4^{\alpha_4} ...$$

We define $P_k(k \geq 0)$ as the space of polynomials of degree less than or equal to k in variables $x_1, ..., x_d, and Q_k$ as the space of polynomials of degree less than or equal to k in each variable $x_1, ..., x_d$. The trial function is then taken for every element and the value at each node is calculated. This resuts in n_p number of equation, where n_p is the number of nodes in a given element.
Lets suppose that we have a trial function as ϕ. Now solving the above equations, value of our trial function ϕ for a given position x inside an element can be written as a function of the values of ϕ at the N nodes of the element $([\phi_1, \phi_2, ..., \phi_N])$, i.e.

$$\phi(x) = [\phi][N] = [\phi_1\, \phi_2\, \phi_3\, \phi_4\, ...][N_1\, N_2\, N_3\, N_4\, ...]$$

where $N_i(x)$ is called as shape function of node i.

3. **Calculate element matrices for each finite element:** Given the PDE we want to solve, now we must find a system of algebraic equations for each element "e" such that by solving it we get the values of ϕ at the position of nodes of the element "e" $([\phi_1, \phi_2, ..., \phi_N] = [\phi_e])$, i.e., you must find for each element "e" the matrix $[K]_e$ and the vector $[f]_e$ such that,

$$[K]_e.[\phi]_e = [f]_e$$

This is one of the more tricky parts of the FEM. There are different ways of getting the matrix $[K]_e$, which we will see later.

4. **Assemble element matrices to form a global linear system:** After calculating the equation matrix for all element "e" in the sample space, we have to assemble all the equations to form a global matrix. Now, one certain problem which comes in this step is the multiplicity of the nodes, i.e one node can be found in more than one element, which has to be taken care of! In the end if we have M global nodes in the system, then we must have to build a global stiffness matrix [K] of size MxM and a global vector [f] of size M such that FEM problem reduces to solving the following matrix equation:

$$[K].[\phi] = [f]$$

where $[\phi] = [\phi_1\, \phi_2 \ldots \phi_M\,]$ is the approximate value of the function $\hat{\phi}$ at the global node points.

5. **Implement the boundary conditions by modifying the global linear system** This step is performed before evaluating ϕ in above step. After we have valuee of $[\phi]$ at global nodes, we can calculate its value at any point in the domain.

3 Convection-Diffusion equation using FEM

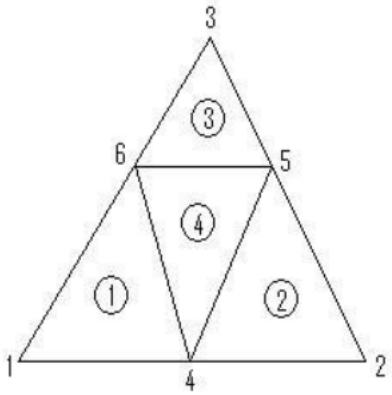
In this section we will see the practical implementation of FEM by solving convection-diffusion equation.

$$\Delta u + +v.\nabla u = s$$

Convection-diffusion equations are very similar to time independent navier-stokes equations. As stated earlier any PDE can be solved using FEM by following particular steps stated above. We'll first discretise the domain into N_e number of elements, then rewriting the equation in its weak form, followed by evaluation of elemental matrix. These matrices will then be assembled into a global matrix to solve for the value of function at nodes.

3.1 FEM mesh generation

For simplicity, we assume that the physical domain Ω is a polygon, which can be discretized into N_E triangular elements each called T_h . Here we describe a simple algorithm: implementation starting with a very coarse mesh, then generating different levels of finer meshes by uniform subdivision, i.e., each triangle is subdivided into four smaller triangles.



To proceed with the mesh generation, first we generate the nodal coordinates (x_i, y_i),
$1 \leq \mathrm{i} \leq N_G$ where N_G denotes the total number of nodes in the mesh. Then we need a connectivity matrix $conn(i, j)$ to describe relation between local nodes and global nodes.For a linear triangular grid, $conn(i, j)$, j = 1, 2, 3, i = 1, , N_E , denotes the global label of the jth node of the ith element, where N_E is the total number of elements.
Now, in order to implement boundary conditions, we need to know which nodes are on the boundary, and what types of boundary conditions are there. In our implementation, we introduce a global boundary condition indicator gbc(i, j), $1 \leq i \leq N_G, 1 \leq j \leq 2$ which can directly be obtained by $elf(i, j)$ In order to generate gbc(i,j) we use the local boundary element using elf(i,j), which is zero for inner nodes and one for outer nodes.

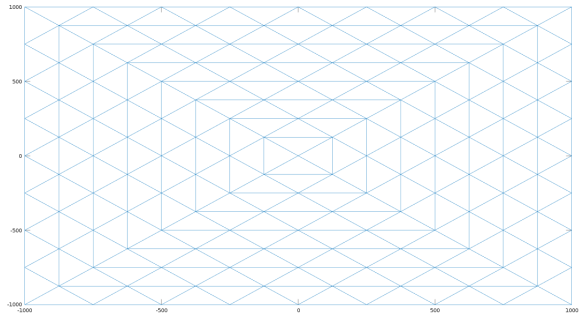


Figure 3: Mesh generation using triangular elements by refinement of coarse element

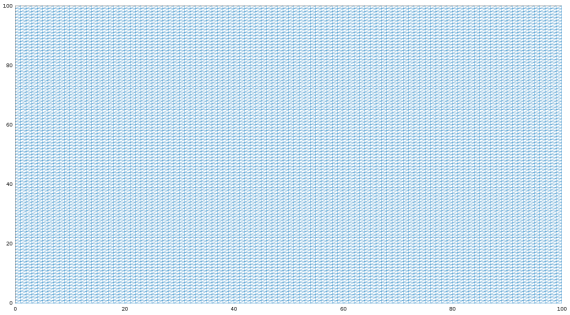


Figure 4: Mesh generation using square element where L = H = 100, dx = 1, dy =1

3.2 Forming FEM equations

The weak formulation is given by

$$(\nabla u_h, \nabla \psi_h) + (v.\nabla u_h, \psi_h) = (\nabla s, \psi_h)$$

u is approximated for an element as

$$u(x,y) = u_h^E(x,y) = \sum_{j=1}^{3} u_j^E \psi_j^E$$

where u_j^E is the value of u_h at j^th node of the element E, and ψ_h is Lagrange's interpolation function which satisfies

$$\psi_j^E(x_i, y_i) = \delta_{ij}$$

Taking this over each element and substituting the source function by its basis function we obtain the diffusion, convection stiffness matrix and mass matrix.
Diffusion Matrix:

$$A_{ij} = \int_D \nabla \psi_j^E . \nabla \psi_i^E dydx$$

Convection Matrix:

$$B_{ij} = \int_D (v.\nabla \psi_j^E) \psi_i^E dydx$$

Mass Matrix

$$M_{ij} = \int_D \psi_j^E \psi_i^E dydx$$

Summing up all the elements, we come up we a set of linear equations given by:

$$\sum_{N=1}^{N_E} (A_{ij} + B_{ij}) u_j = \sum_{N=1}^{N_E} M_{ij} s_j$$

3.3 Calculation of Element Matrices

The calculaltion of element matrices is often computed on a reference element.The map- ping from an arbitrary triangular element with vertices $(x_i, y_i), 1 \leq i \leq 3$, to a reference element with vertices

$$(\eta_1, \epsilon_1) = (0,0)\ (\eta_2, \epsilon_2) = (1,0)\ (\eta_3, \epsilon_3) = (0,1)$$

is given by

$$x = x_1 + (x_2 - x_1)\eta + (x_3 - x_1)\epsilon = \sum_{j=1}^{3} x_j \hat{\psi}_j(\eta, \epsilon)$$

$$y = y_1 + (y_2 - y_1)\eta + (y_3 - y_1)\epsilon = \sum_{j=1}^{3} y_j \hat{\psi}_j(\eta, \epsilon)$$

where
$\hat{\psi}_1(\eta, \epsilon) = 1 - \eta - \epsilon$
$\hat{\psi}_2(\eta, \epsilon) = \eta$
$\hat{\psi}_3(\eta, \epsilon) = \epsilon$

Using this maping we arrive at the modified value of the matrices as integral in domains of $(d\eta, d\epsilon)$ which upon solving gives the value of $A_{ij}, B_{i,j}$ and M_{ij} in terms of $x_1, x_2, x_3, y_1, y_2 \, and \, y_3$

3.4 Assembly and implementation of bondary conditions:

To obtain the global coefficient matrix by assembling the contributions from each element coefficient matrix, we just need to loop through all elements in the mesh, find the corresponding global nodal label for each local node, and put them in the right locations of the global coefficient matrix.

3.5 Two Equations using FEM

First equation is:

$$\Delta u \; + \; + v.\nabla u \; = \; s$$

which will be solved by using P_1 basis function.

Second equation is:

$$-\nabla u + u = f(x, y)$$

which will be solved by using Q_1 basis function.

3.6 FEM solution for P_1 element

The solution for convection-diffusion equation using s is done by using matlab coded. Details of the code:
gen_P1grid.m : This generates the grid mesh for P1 type basis function. Domain is a square with variable size.
locA.m, locB.m, locM.m : Generates the diffusion, convection and mass matrices.
scpt.m: Solves the system using the above functions.
SRC.m : Value of the source function s. scpt.m : Script for usinf the above functions to evaluate the value of function.

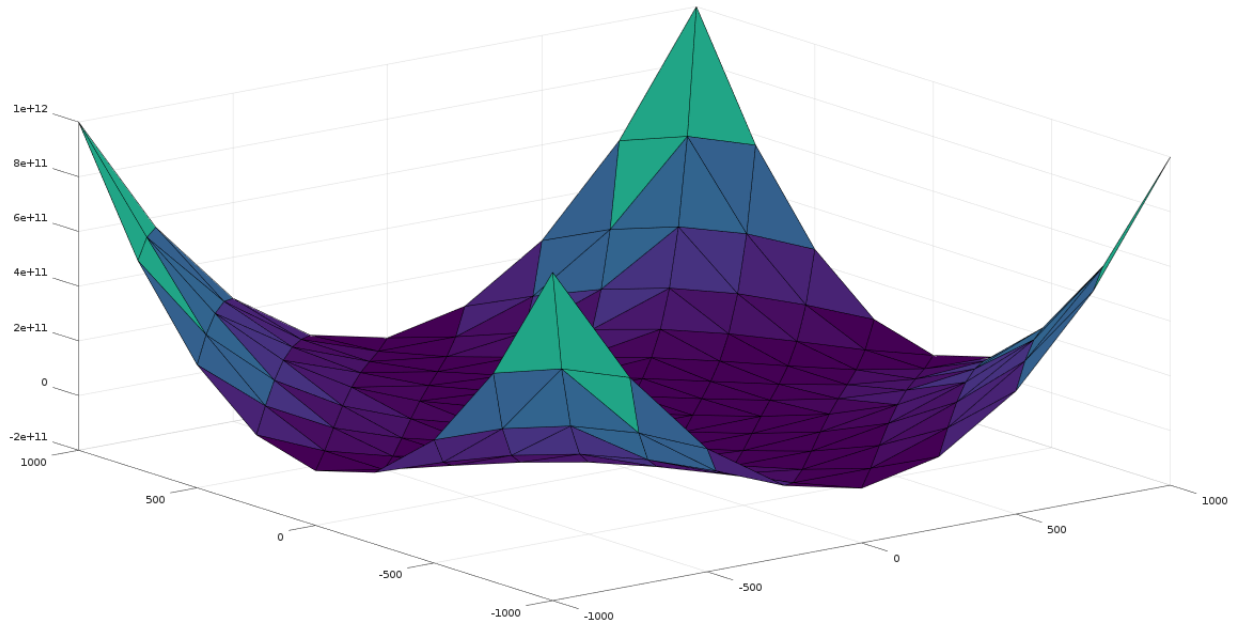


Figure 5: Using P1 element and source function as some complicated s(x,y), here analytically we get x^2y^2 in first equation

3.7 FEM solution for Q_1 element

Another equation which I solved is $-\nabla u + u = f(x, y)$ using Q_1 element. Details of the code:
gen_Q1grid.m : This generates the grid mesh for Q1 type basis function. Domain is a square with variable size. Here the method of generation of domain is square with height and length along with no. of elements in both x,y mentioned rather than generating finer meshes using coarse mesh as in above case. elemA.m : Generation of element A matrix. SRC.m : Value of the source function s

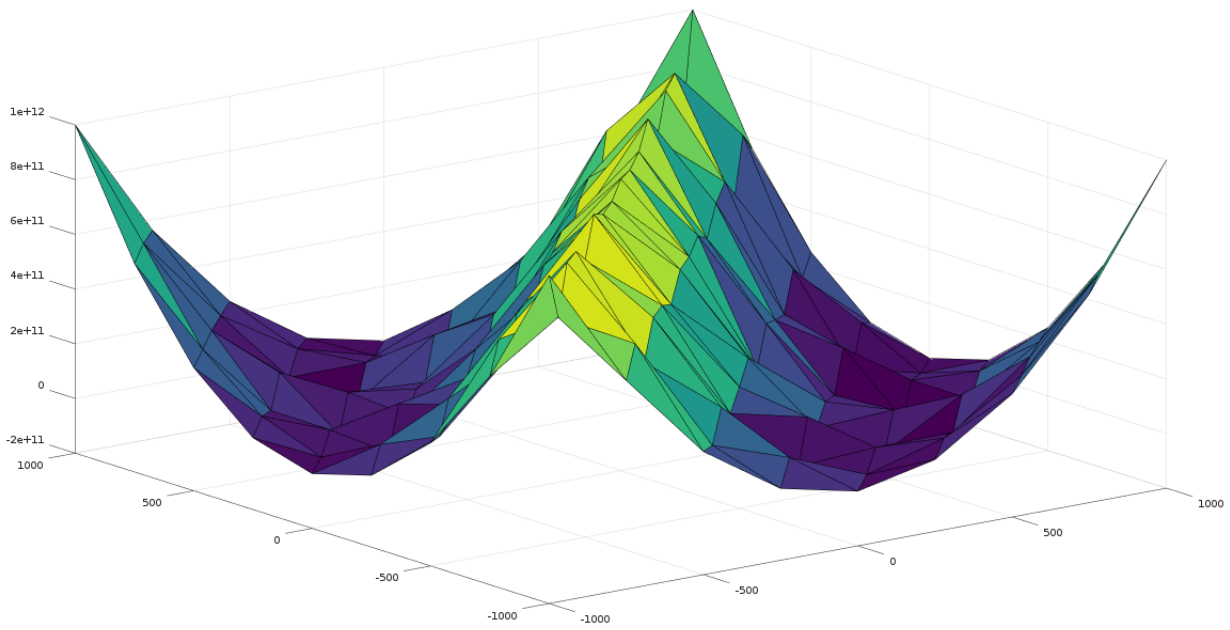


Figure 6: Using P1 element and source function as s(x,y) = 0 in first equation

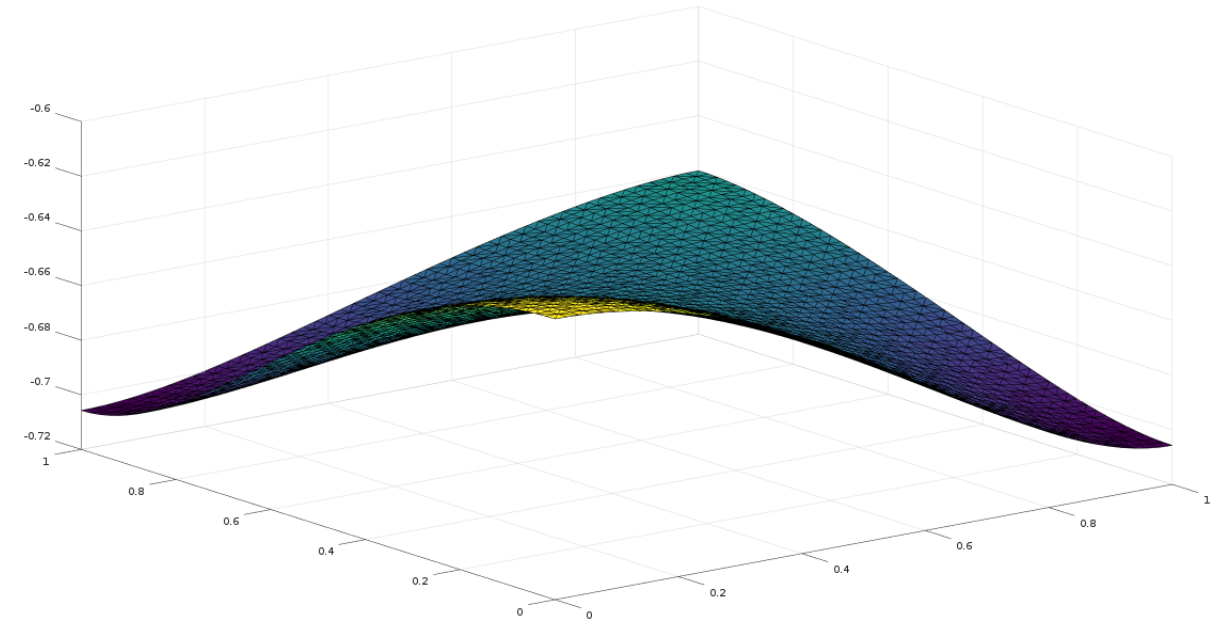


Figure 7: Using Q_1 element and a complicated source function s(x,y) in second equation

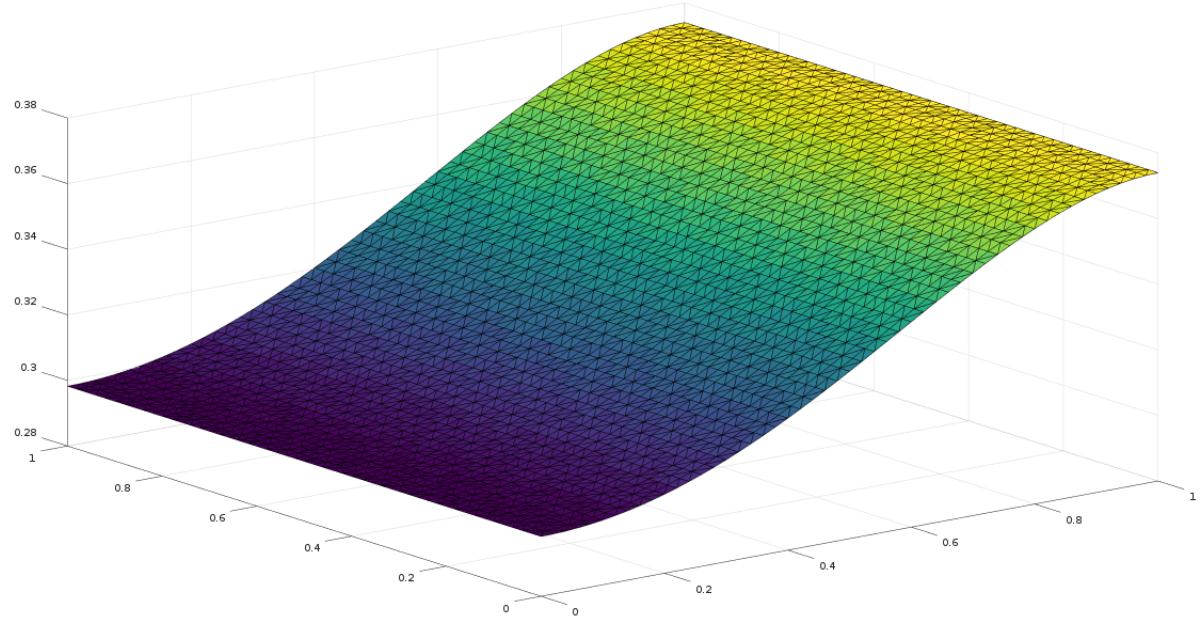


Figure 8: Using Q_1 element and source function s(x,y) = x^2 in second equation

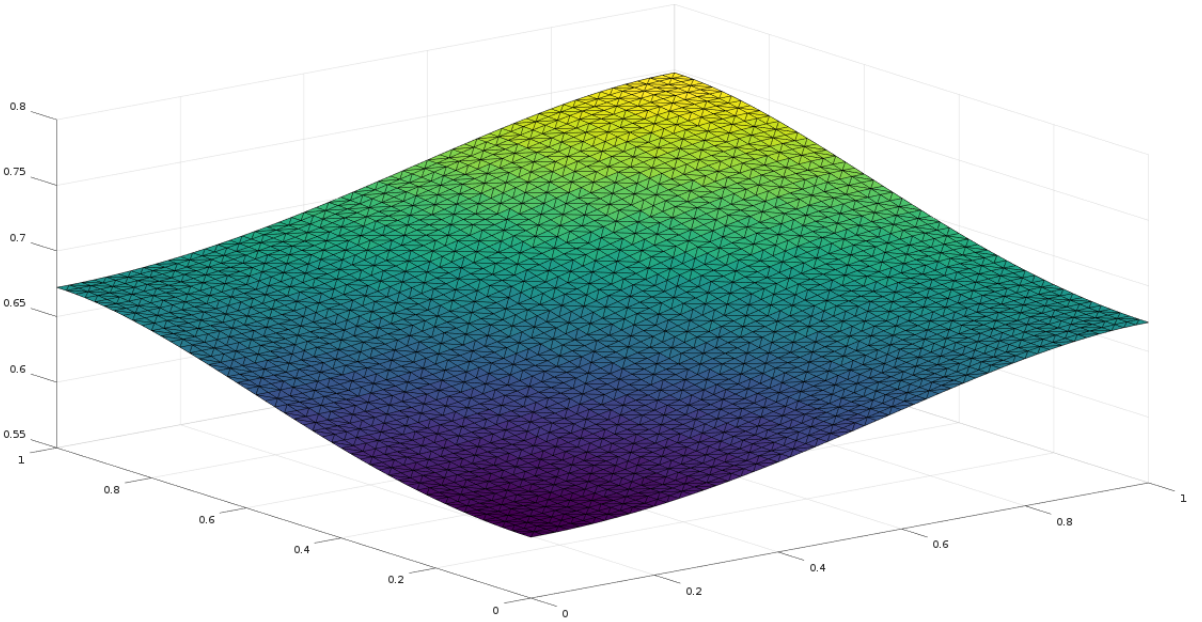


Figure 9: Using Q_1 element and source function s(x,y) = $x^2 + y^2$ in second equation