A Report

On

Numerical Solutions to Partial Differential Equations using FEniCS

Prepared in partial fulfillment

of

Design Project

ME F377

By

Sanath Keshav
2013A4PS359G

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI

Acknowledgements

I would like to express my deepest appreciation to all those who helped me to complete this report. A special gratitude I give to my mentor Dr. P Dhanumjaya , for giving me this opportunity to work on this topic and also for his guidance and unwavering support during the whole semester. I would also like to particularly thank my classmate K Balaje for his constant support and motivation the whole time.

Contents

Chapter 1

Introduction

In this report we introduce FEM for some elliptic model problems and study the basic properties of the method. We first consider a simple one-dimensional problem and then some two=dimensional generalisations. Consider the following *model problems* -

$$\begin{aligned} &\text{1D:} \qquad -u''(x) = f(x), \quad 0 < x < 1, \quad u(0) = u(1) = 0 \\ &\text{2D:} \qquad -\Delta u(x,y) = f(x,y), \quad (x,y) \in \Omega, \quad u(x,y)|_{\partial\Omega} = 0 \end{aligned}$$

where Δ denotes the *Laplacian* operator, Ω is the domain in the (x, y) plane with the boundary $\partial\Omega$.

1.1 Preliminaries

We now introduce some notation that will be used below. We define

$$D^\alpha v = \frac{\partial^{|\alpha|} v}{\partial x_1^{\alpha_1} \partial x_2^{\alpha_2} \dots \partial x_n^{\alpha_n}} \tag{1.1}$$

where here $\alpha = (\alpha_1, \alpha_2), \alpha_i$ is a non-negative natural number and $|\alpha| = \alpha_1 + \alpha_2$. As an example, a partial derivative of order 2 can there be written as $D^\alpha v$ with $\alpha = (2, 0), \alpha = (1, 1)$ or $\alpha = (0, 2)$, which are theα with $|\alpha| = 2$.

We now define for $k = 1, 2, ...,$

$$H^k(\Omega) = \Big\{ u(x) \ \Big| \ D^\alpha u \in L^2(\Omega), \ |\alpha| \le m \Big\} \tag{1.2}$$

1.1.1 The Hilbert spaces

When giving the variational forms of the boundary value problems for partial differential equations, it is from the emathematical point of view natural and very useful to work with function spaces V that are slightly larger than the spaces of continuous functions with piecewise continuous derivatives. It is also useful to endow the spaces V with various scalar products with the scalar product related to the boundary value problem. More precisely, V will be a Hilbert space.

We now introduce some Hilbet spaces that are natural to use for variational formulations of the boundary value problems we will consider. Let us start with the one-dimensional case. If $I = (a, b)$ is an interval, we define the space of "square integrable functions" on I :

$$L^2(I) := \left\{ v : \int_I v^2(x)\ dx < \infty \right\} \tag{1.3}$$

with the norm:

$$\|v\|_{L^2} := \left(\int_I v^2\ dx \right)^{1/2}. \tag{1.4}$$

The space $H^1(I)$consists of the functions v defined on I which together with their first derivatives are square integrable, ie, belong to $L_2(I)$

$$H^1(I) = \left\{ v \;\Big|\; \int_I v^2\ dx < \infty,\ \int_I (v')^2\ dx < \infty \right\} \tag{1.5}$$

$$H^1(\Omega) = \left\{ v(x,y) \;\Big|\; v \in L^2(\Omega),\ \frac{\partial v}{\partial x} \in L^2(\Omega),\ \frac{\partial v}{\partial y} \in L^2(\Omega) \right\} \tag{1.6}$$

We equip this space with the scalar product

$$(v, w)_{H^(I)} = \int_I (vw + v'w')dx \tag{1.7}$$

$$\|v\|_{H^1} := \left(\int_I (v^2 + v'^2)\ dx \right)^{1/2}. \tag{1.8}$$

In case of boundary value problems of the form $-u'' = f$ on $I = (a, b)$ with boundary conditions $u(a) = u(b) = 0$, we shall use the space

$$H_0^1(I) = \left\{ v \in H^1(I) : v(a) = v(b) = 0 \right\} \tag{1.9}$$

with the same scalar product and norm as for $H^1(I)$.

Chapter 2

1D Elliptic Problems

In this chapter, we will consider the following boundary value problem for the poisson equation and demonstrate how the Finite Element Method can be implmented in MATLAB to compute the solutions to different Ordinary Differential Equations subjected to **Dirichlet, Neumann and Robin** Boundary Conditions.

2.1 Weak formulation

Consider the *2-Point Boundary Value Problem* in 1D,

$$\begin{aligned} -u''(x) &= f(x) \quad \text{in} \quad \Omega \equiv (0,1) && (2.1)\\ u(0) &= u(1) = 0 && (2.2)\end{aligned}$$

Now, let us look at the general procedure that is usually followed to find out the **Finite Element Solution** of the problem.

1. **Derive the Variational/Weak Formulation**: We shall now give an abstract formulation of the finite element method for elliptic problems. This makes it possible to give a unified treatment of many problemsin mathematics and physics. Considering the Galerkin Approach, we construct the weak formulation of our **2-Point BVP** by multiplying both sides of 2.1 with a **test function** v satisfying $v(0) = v(1) = 0$ and integrating it over the domain $\Omega \equiv (0,1)$.

$$-\int_0^1 u''v\ dx = \int_0^1 fv\ dx \tag{2.3}$$

$$-\ u'v\ \Big|_0^1 + \int_0^1 u'v'\ dx = \int_0^1 fv\ dx \tag{2.4}$$

As we have assumed our test function space to follow $v(0) = v(1) = 0$, we have,

$$\int_0^1 u'v' \, dx = \int_0^1 fv \, dx \tag{2.5}$$

Equation 2.5 is known as the **Weak formulation** of the problem.

2. **Discretization of the problem:** For a 1D Problem, the domain is discretized into n subintervals $[x_{i-1}, x_i]$ of equal length, where

$$h = 1/n, \qquad x_i = ih, \qquad i = 1, 2, \ldots, n \tag{2.6}$$

We should keep in mind that equal subintervals have been considered only for simplicity sake but, FEM is perfectly capable of handling an unstructured discretization.

3. **Construct the basis functions on Master element:** Next we choose the set of basis functions depending on the problem. Let us denote them as,

$$\{\phi_i\}_{i=1,2,\ldots,n} \tag{2.7}$$

More about basis functions are discussed in the following sections.

4. **Construct the system of equations** First, write down the approximate *Finite Element Solution* $u_h(x)$ as a linear combination of the basis functions ϕ_i,

$$u_h(x) = \sum_{i=1}^{n} \xi_i \phi_i(x) \tag{2.8}$$

Using this in the weak formulation 2.5, we have,

$$\begin{aligned} \int_0^1 u_h' v' \, dx &= \int_0^1 \left(\sum_{i=1}^{n} \xi_i \phi_i' \right) v' \, dx & (2.9) \\ &= \sum_{i=1}^{n} \xi_i \int_0^1 \phi_i' v' \, dx = \int_0^1 fv \, dx & (2.10) \end{aligned}$$

Now substituting $v = \phi_j, \ j = 1, 2, \ldots, n$ in 2.10, we have a system of linear equations,

$$\begin{aligned} [A]\{\xi\} &= \{b\} & (2.11) \\ A_{ij} &= \int_0^1 \phi_i' \phi_j' \, dx & (2.12) \\ b_j &= \int_0^1 f \phi_j dx & (2.13) \end{aligned}$$

5. **Solve the system** 2.11 to obtain ξ_i and get $u_h(x) = \sum_{i=1}^{n} \xi_i \phi_i(x)$

6. Proceed to carry out the **error analysis**.

2.1.1 Analysis of FEM

For analysis, we consider the 1D equivalent of the model problem,

$$\begin{aligned} a(u,v) \;&:=\; \int_0^1 u'v'\,dx \\ &=\; \int_0^1 fv\,dx \quad := \quad (f,v) \end{aligned} \tag{2.14}$$

Next we present some of the interesting properties of the **abstract form ??**

1. $a(u,v)$ is symmetric. $a(u,v) = a(v,u)$
2. $a(u,v)$ is continuous/bounded, i.e, there exists a constant $\gamma > 0$, such that
$$|a(v,w)| \leq \gamma \|v\|_V \|w\|_V \quad \forall v, w \in V$$
3. $a(u,v)$ is coercive/V-Elliptic, i.e, there exists a constant $\alpha > 0$, such that
$$a(v,v) \geq \alpha \|v\|_V^2 \quad \forall v \in V \tag{2.15}$$
4. (f,v) is bounded. i.e, there exists a constant $\Lambda > 0$ such that
$$|(f,v)| \leq \Lambda \|v\|_V \quad \forall v \in V \tag{2.16}$$

An important consequence of the properties is the **uniqueness** of the solution to the variational problem which is given by the **Lax-Milgram Theorem**. Now that we have addressed the issue of uniqueness of the solution, we now look at the basis functions for the model problem.

Since the space $V = H_0^1(\Omega)$ is an infinite dimensional space, searching for the finite element solution is an impossible task for the computer. So we "approximate" the Infinite Dimensional space by a Finite Dimensional space $V_h \subset V$. Let,

$$V_h = span\{\phi_1, \phi_2, \ldots, \phi_n\}$$

where ϕ_i are the basis functions for the finite dimensional space V_h. Hence any function in V_h can be represented as,

$$v_h(x) = \sum_{i=1}^{n} \xi_i \phi_i(x)$$

2.1.2 Basis functions

A common choice for the basis functions for our **Model Problem in 1D** are the linear Lagrange basis functions/Hat functions.

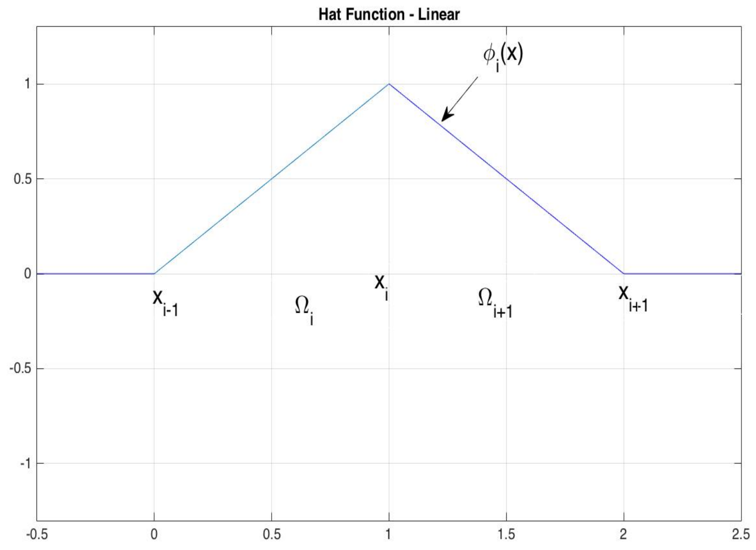


Figure 2.1: Hat Function

$$\phi_i(x) = \begin{cases} \frac{x - x_{i-1}}{h} & x_{i-1} \le x \le x_i \\ \frac{x_{i+1} - x}{h} & x_i \le x \le x_{i+1} \\ 0 & otherwise \end{cases} \tag{2.17}$$

Observe that

$$\phi_i(x_j) = \delta_{ij}, \quad \text{where} \tag{2.18}$$

$$\delta_{ij} = \begin{cases} 1, & i = j \\ 0, & i \ne j \end{cases} \tag{2.19}$$

at nodes x_j. The expresssion 2.17 can be obtained by assuming a linear equation of the form $\phi_i(x) = a + b\ x$ and using the Kronecker Delta Property 2.18. The function defined above is said to have a **Compact Support** in Ω which means that the function is non-zero only at certain points in the domain. This property is reflected in the **sparse nature** of the Global Stiffness Matrix.

To find out the solution at each nodes in the domain, we construct stiffness matrix K^e and load vector f^e at the element level and then assemble them to obtain the global system and solve it. So at the element level, we have the system of equations

$$[K^e]\{u^e\} = \{f^e\} \tag{2.20}$$

The local numbering of the linear basis functions in an element Ω^e is shown in Figure 1.2(a). Let ψ_1^e and ψ_2^e be the basis function in an element Ω^e. If the element size is h_e, then for the model problem in 1D, we have,

$$K^e = \left[\int_0^{h_e} \psi_i'^e \psi_j'^e \, dx\right]_{i,j=1,2} = \begin{bmatrix} \frac{1}{h_e} & -\frac{1}{h_e} \\ -\frac{1}{h_e} & \frac{1}{h_e} \end{bmatrix} \tag{2.21}$$

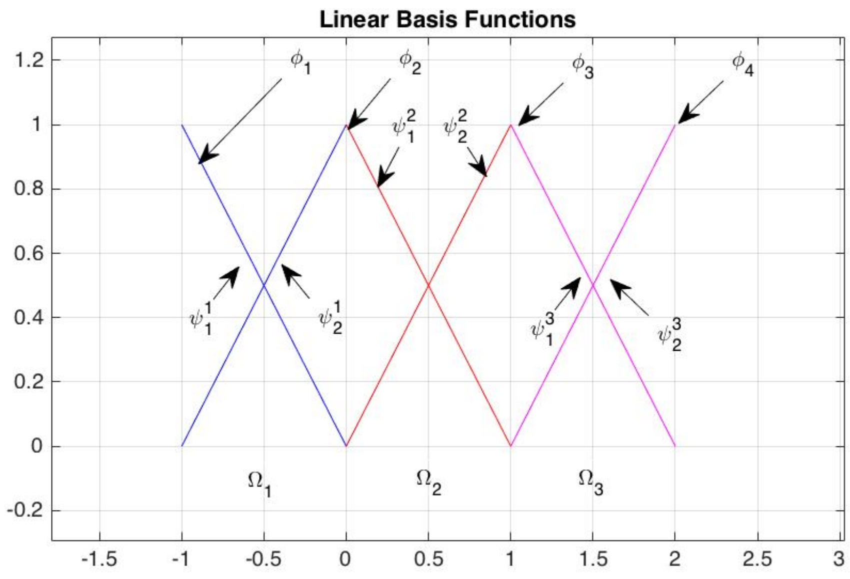


Figure 2.2: Basis Functions

2.1.3 Assesmbly

Upon adding the contributions of different element for any node x_i, we obtain the global stiffness matrix. This process is known as **Assembly** of the local siffness matrices. If $h_1, h_2, \ldots, h_n$ are the element sizes, then the **global stiffness matrix A** can easily be constructed as follows.

$$A = \begin{bmatrix} \frac{1}{h_1} & -\frac{1}{h_1} & 0 & \ldots & 0 & 0 & 0 \\ -\frac{1}{h_1} & \frac{1}{h_1}+\frac{1}{h_2} & -\frac{1}{h_2} & \ldots & 0 & 0 & 0 \\ 0 & -\frac{1}{h_2} & \frac{1}{h_2}+\frac{1}{h_3} & \ldots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & -\frac{1}{h_{n-1}} & \frac{1}{h_{n-1}}+\frac{1}{h_n} & -\frac{1}{h_n} \\ 0 & 0 & 0 & \ldots & 0 & -\frac{1}{h_n} & \frac{1}{h_n} \end{bmatrix} \tag{2.22}$$

and the **global load vector b** is,

$$b = \left\{ \begin{array}{c} \int_0^{h_1} f\psi_1^1 dx \\ \\ \int_0^{h_1} f\psi_2^1 dx + \int_{h_1}^{h_2} f\psi_1^2 dx \\ \\ \int_{h_1}^{h_2} f\psi_2^2 dx + \int_{h_2}^{h_3} f\psi_1^3 dx \\ \\ \vdots \\ \\ \int_{h_{n-2}}^{h_{n-1}} f\psi_2^{n-1} dx + \int_{h_{n-1}}^{h_n} f\psi_1^n dx \\ \\ \int_{h_{n-1}}^{h_n} f\psi_2^n dx \end{array} \right\} \tag{2.23}$$

The next step is to apply the boundary conditions to the system $A\xi = b$ by setting the values accordingly in the Stiffness Matrix and Load Vector.

When we make the approximation by choosing u_h from $V_h \subset V$, we introduce errors in the solution. If $u(x)$ denotes the exact solution and $u_h(x)$ denotes the Finite Element solution, we mention the following **error estimates** when a linear basis function is used. If $\|.\|_a$ denotes the energy norm, $\|.\|_\infty$ denotes the pointwise inifinity norm and $\|.\|_{H^m}$ denotes the H^m norm we have the following error estimates

$$\|u - u_h\|_a \leq Ch\|u''\|_\infty \tag{2.24}$$

$$\|u - u_h\|_\infty \leq Ch^2\|u''\|_\infty \tag{2.25}$$

$$\|u - u_h\|_{H^1} \leq Ch\|u''\|_\infty \tag{2.26}$$

Result 2.25 shows that the Finite Element Method is second order accurate when linear basis is used.

2.2 Numerical Examples

Example 2.1 *Consider the following 2-Point Boundary Value Problem,*

$$-u'' = x(3+x)e^x, \quad 0 < x < 1 \tag{2.27}$$
$$u(0) = u(1) = 0 \tag{2.28}$$

The exact solution is given by

$$u(x) = x(1-x)e^x \tag{2.29}$$

We wish to find the solution to this problem using Finite Element Method.

The weak form for the problem is given by 2.5 with $f(x) = 1$.

Assuming uniform grid spacing - h while discretizing the domain into n elements

$$0 = x_0 < x_2 < \cdots < x_n = 1 \tag{2.30}$$
$$h = \frac{x_n - x_0}{n} \tag{2.31}$$
$$x_i = i\ h \tag{2.32}$$

we have the finite dimensional weak form,

$$\textit{Find}\ u_h \in V_h \subset V = H_0^1(0,1),\ \textit{such that} \tag{2.33}$$
$$\int_0^1 u_h' v_h'\ dx = \int_0^1 f v_h\ dx \quad \forall v_h \in V_h \tag{2.34}$$

When we construct the element stiffness matrix, K^e, we get

$$K^e = \begin{bmatrix} \frac{1}{h} & -\frac{1}{h} \\ -\frac{1}{h} & \frac{1}{h} \end{bmatrix} \tag{2.35}$$

and after assembly (Ref. 2.22, 2.23) we obtain the system of linear equations.

$$A\xi = b \tag{2.36}$$

and we solve the system of equations after accounting for the boundary conditions. The solution plots for different number of elements are shown in Figure,

The infinity norm of the error in the Finite Element Solution was found out to be in the order of $\approx 10^{-16}$ which is very close to the machine error $\approx 2.2 \times 10^{-16}$

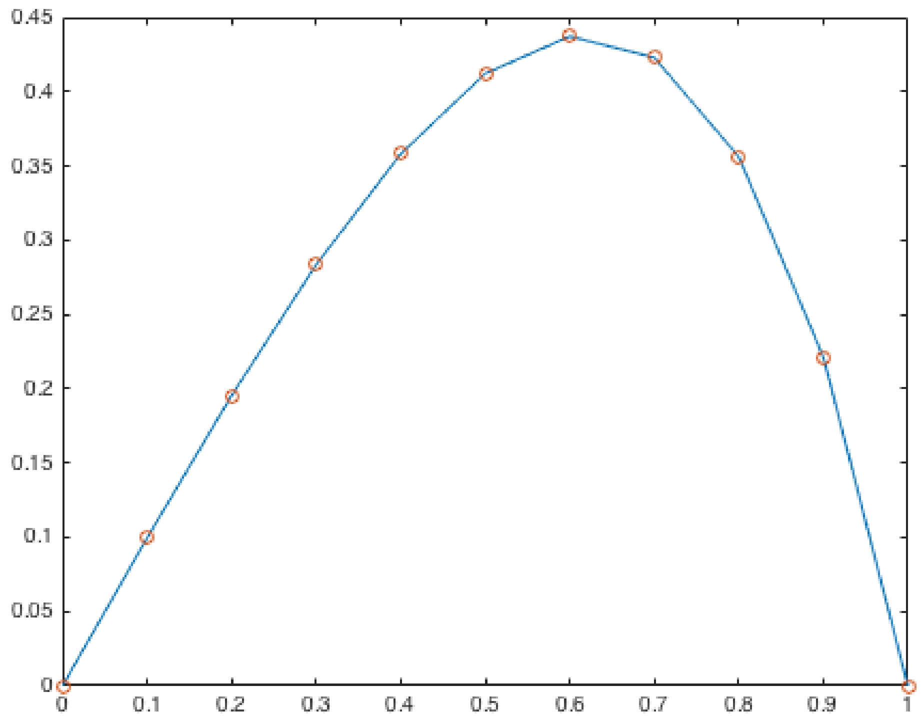


Figure 2.3: $NumericalSolution$

Example 2.2 *Consider the following 2-Point Boundary Value Problem,*

$$-u'' + u = 12x + 3x^2 - 2x^3 - 6, \quad 0 < x < 1 \tag{2.37}$$

$$u(0) = 0 \tag{2.38}$$

$$u'(1) = 0 \tag{2.39}$$

The weak form for this problem is given by,

Find $u_h \in V_h \subset V = \{u \mid u \in H^1(0,1),\ u(0) = 0\}$, *such that*

$$-u_h' v_h\Big|_0^1 + \int_0^1 (u_h' v_h' + u_h v_h)\ dx = \int_0^1 f v_h\ dx \qquad \forall v_h \in V_h$$

$$\implies \int_0^1 (u_h' v_h' + u_h v_h)\ dx = \int_0^1 f v_h\ dx + u_h'(1) v_h(1) \qquad \forall v_h \in V_h \tag{2.40}$$

$$\implies \int_0^1 (u_h' v_h' + u_h v_h)\ dx = \int_0^1 f v_h\ dx \qquad \forall v_h \in V_h \tag{2.41}$$

The Neumann Boundary Condition is accounted in the assembled RHS by adding the value of $u'(1)$ in the last entry of b.

Hence the final system of equations become,

$$A\xi = b + \vec{f_n} \tag{2.42}$$

The order of convergence observed for different elements are tabulated below,

Element Type	*Order of convergence observed*
Linear	≈ 2
Quadratic	≈ 4
Cubic	≈ 5

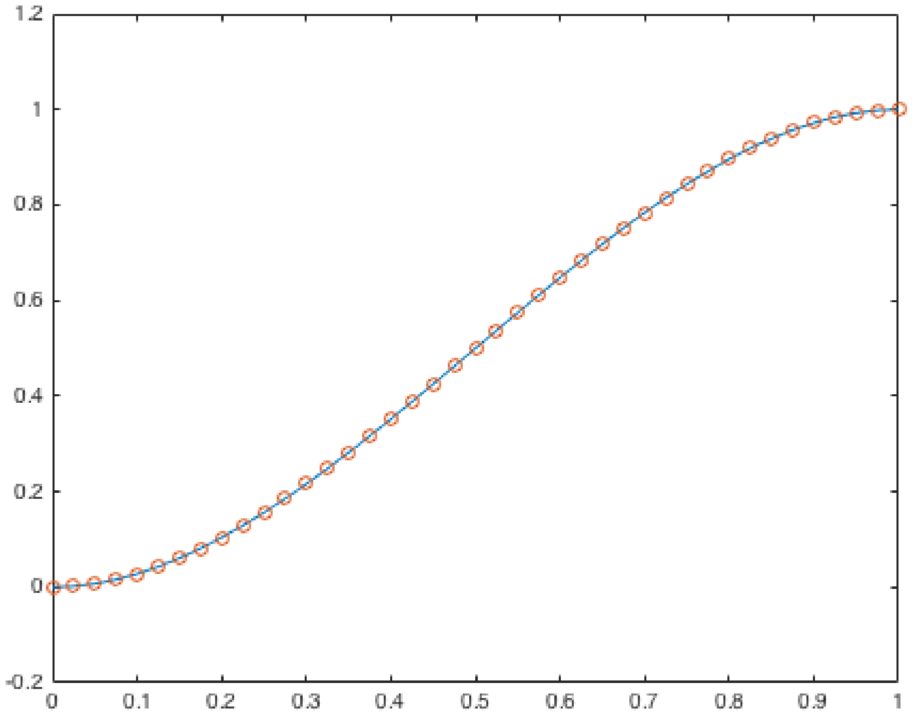


Figure 2.4: $n = 40$

Example 2.3 *Consider the following 2-Point Boundary Value Problem,*

$$\begin{aligned} -u'' + u &= x^2 + x - 2, \quad 0 < x < 1 && (2.43) \\ u(0) &= 0 && (2.44) \\ u(1) + u'(1) &= 5 && (2.45) \end{aligned}$$

The weak form for this problem is given by,

$$\begin{aligned} &\text{Find } u_h \in V_h \subset V = \{u \mid u \in H^1(0,1),\ u(0) = 0\},\ \text{such that} \\ &-u_h' v_h\Big|_0^1 + \int_0^1 (u_h' v_h' + u_h v_h)\ dx = \int_0^1 f v_h\ dx \qquad \forall v_h \in V_h \\ &\implies \int_0^1 (u_h' v_h' + u_h v_h)\ dx = \int_0^1 f v_h\ dx + u_h'(1) v_h(1) \qquad \forall v_h \in V_h \quad (2.46) \\ &\implies \int_0^1 (u_h' v_h' + u_h v_h)\ dx = \int_0^1 f v_h\ dx + (5 - u_h(1))\ v_h(1) \qquad \forall v_h \in V_h \quad (2.47) \end{aligned}$$

The Robin Boundary Condition is accounted in the assembled R.H.S by adding the value of $u'(1) + u(1) = 5$ in the last entry of b, and since $u_h(1)$ appears in the R.H.S, the value must be accounted for in the assembled global stiffness matrix.

Hence the final system of equations become,

$$\begin{bmatrix} A_{11} & A_{12} & \dots & A_{1n} \\ \vdots & \vdots & \vdots & \vdots \\ A_{n1} & A_{n2} & \dots & A_{nn} \end{bmatrix} \begin{Bmatrix} \xi_1 \\ \vdots \\ \xi_n \end{Bmatrix} = \begin{Bmatrix} b_1 \\ \vdots \\ b_n \end{Bmatrix} + \begin{Bmatrix} 0 \\ \vdots \\ 5 - \xi_n \end{Bmatrix} \tag{2.48}$$

which in-turn becomes,

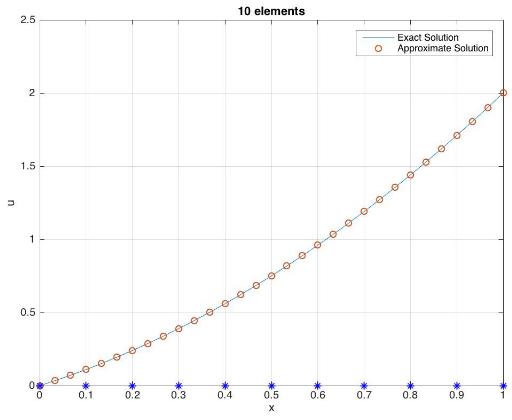
$$\begin{bmatrix} A_{11} & A_{12} & \dots & A_{1n} \\ \vdots & \vdots & \vdots & \vdots \\ A_{n1} & A_{n2} & \dots & A_{nn} + 1 \end{bmatrix} \begin{Bmatrix} \xi_1 \\ \vdots \\ \xi_n \end{Bmatrix} = \begin{Bmatrix} b_1 \\ \vdots \\ b_n \end{Bmatrix} + \begin{Bmatrix} 0 \\ \vdots \\ 5 \end{Bmatrix} \tag{2.49}$$

The plots showing the solution curves for a Cubic Element is shown in Figure 2.5. The error in the Finite Element solution is in the order of $\approx 10^{-13}$

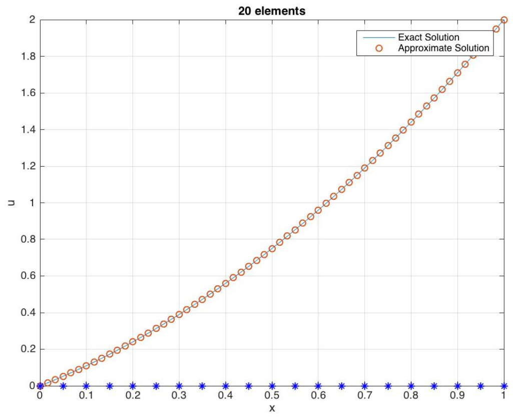
2.3 Non-Linear Problem

Next we consider solving a non-linear problem using Finite Element Method. We obtain a non-linear system of equations,

$$K(U)\ U = F \quad \text{or} \quad KU = F(U) \tag{2.50}$$



(a) $n = 10$



(b) $n = 20$

Figure 2.5: Solution plots using cubic elements - Example 2.3

So, we use the **Newton Raphson** iterations to solve the system of equations. The procedure is to first construct the system of equations at the element level,

$$K^e(u^e)\ u^e = f^e \tag{2.51}$$

and compute the elemental Jacobian Matrix J^e. Next we assemble the elemental Jacobian and the elemental load vector to the global level J and F and use Newton Raphson Iterations,

1. Define $error = 100$ and tolerance $tol = 10^{-10}$ and Initial Guess U_{old}.
2. Compute elemental Jacobian Matrix and Load Vector J^e and F^e
3. while $error > tol$:

```
    Assemble:                      $J^e$ and $F^e \implies J\ and\ F$
    Apply Boundary Conditions
    Newton Raphson Iteration:      $U_{new} = U_{old} - (J^{-1}F)(U_{old})$
    Set                            $error = \|U_{new} - U_{old}\|$
    Do                             $U_{old} = U_{new}$
```

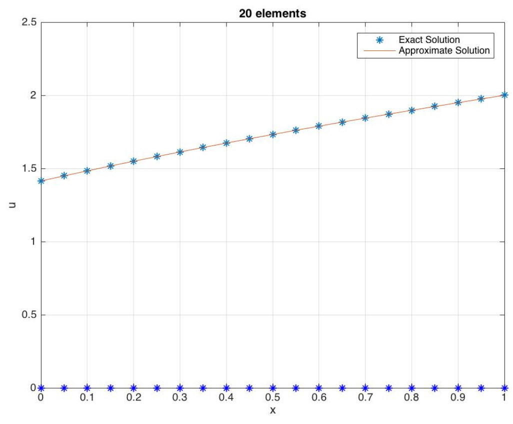
Example 2.4 *Consider the following 2-Point Boundary Value Problem,*

$$-u'' + (u')^3 = 0, \quad 0 < x < 1 \tag{2.52}$$

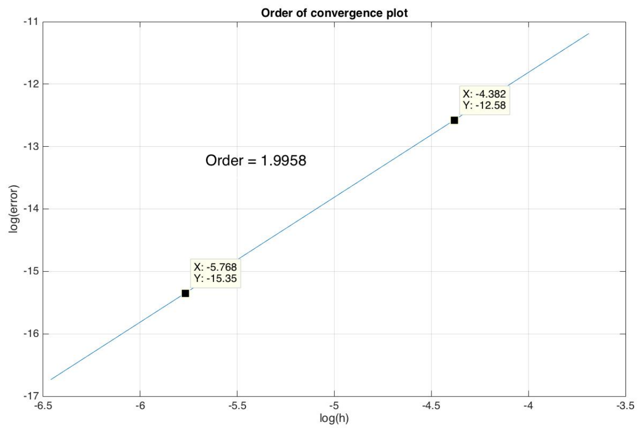
$$(u + u')(0) = 3/\sqrt{2} \tag{2.53}$$

$$u'(1) = 0.5 \tag{2.54}$$

The exact and the approximate solution are shown in Figure 2.6. The order of convergence is found to be 2, when linear basis functions are used.



(a) $n = 20$



(b) Order of convergence ≈ 2

Figure 2.6: Non Linear Problem - Example 2.4

Chapter 3

2D Elliptic problems in FEniCS

The FEniCS Project is a collaborative project for the development of innovative concepts and tools for automated scientific computing, with a particular focus on automated solution of differential equations by finite element methods. FEniCS has an extensive list of features for automated, efficient solution of differential equations, including automated solution of variational problems, automated error control and adaptivity, a comprehensive library of finite elements, high performance linear algebra and many more.

This chapter discusses how to solve a Boundary Value Problem, in particular 2D problems using FreeFEM++.

3.1 Model Problem

Consider the following model problem,

$$\begin{aligned} -\Delta u &= f, \quad \text{in } \Omega && (3.1)\\ u &= z \quad \text{on } \Gamma_1 && (3.2)\\ \frac{\partial u}{\partial n} &= \nabla u.n = 0 \quad \text{on } \Gamma_2 && (3.3) \end{aligned}$$

The FEniCS code that solves the problem is given below.

```
        -Laplace(u) = f on the unit square.
        u = u0 on the boundary.
        u0 = u = 1 + x^2 + 2y^2, f = -6.

        from dolfin import *
        mesh = UnitSquare(6, 4)
        V = FunctionSpace(mesh, 'Lagrange', 1)

        u0 = Expression('1 + x[0]*x[0] + 2*x[1]*x[1]')
```

```python

def u0_boundary(x, on_boundary):
return on_boundary

bc = DirichletBC(V, u0, u0_boundary)

u = TrialFunction(V)
v = TestFunction(V)
f = Constant(-6.0)
a = inner(nabla_grad(u), nabla_grad(v))*dx
L = f*v*dx

u = Function(V)
solve(a == L, u, bc)

plot(u)
plot(mesh)

interactive()
```

Let us first look at some important commands and syntax that are necessary to understand further problems.

```python
from dolfin import *
```

This line imports the key classes UnitSquare, FunctionSpace, Function, and so forth, from the DOLFIN library. All FEniCS programs for solving PDEs by the finite element method normally start with this line. DOLFIN is a software library with efficient and convenient C++ classes for finite element computing, and dolfin is a Python package providing access to this C++ library from Python programs.

```python
mesh = UnitSquare(6, 4)
```

The UnitSquare command defines a uniform finite element mesh over the unit square. The mesh consists of cells, which are triangles with straight sides. The parameters 6 and 4 tell that the square is first divided into 64 rectangles, and then each rectangle is divided into two triangles. Other types of mesh generation will be discussed in the later part of this chapter.

```python
V = FunctionSpace(mesh, 'Lagrange', 1)
```

The functionspace command defines a function space V with 3 arguments. The first argument is the mesh. The second argument reflects the type of element, while the third argument is the degree of the basis functions on the element.

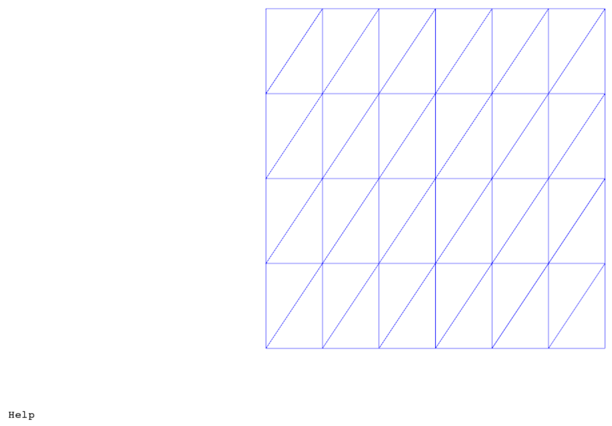


Figure 3.1: FEniCS Mesh

```
        u = TrialFunction(V)
        v = TestFunction(V)
```

These commands define the finite dimensional trial and test spaces.

```
        bc = DirichletBC(V, u0, u0_boundary)
```

The next step is to specify the boundary condition on the boundary.

```
        f = Constant(-6.0)
```

When **f** is constant over the domain, **f** can be more efficiently represented as a Constant object.

We input the weak form of the problem

$$\int_{\Omega} \nabla u_h . \nabla v_h \; d\Omega - \int_{\Omega} f v_h \; d\Omega = 0 \tag{3.4}$$

with the boundary condtions. The **solve** command solves the problem yielding u_h.

```
        a = inner(nabla_grad(u), nabla_grad(v))*dx
        L = f*v*dx
        u = Function(V)
        solve(a == L, u, bc)
```

Note that we first defined the variable u as a TrialFunction and used it to represent the unknown in the form a. Thereafter, we redefined u to be a Function object representing the solution, i.e., the computed finite element function u.

```
        plot(u)
        plot(mesh)
        interactive()
```

This is the simplest way of quickly looking at u and the mesh. The interactive() call is necessary for the plot to remain on the screen.

3.2 Meshing in Fenics

In this section we will discuss a couple of standard mesh generation techniques.

3.2.1 Built-in Mesh generation tools

DOLFIN has a few tools for creating various types of meshes over domains with simple shape: **UnitInterval, UnitSquare, UnitCube, Interval, Rectangle, Box, UnitCircle, and UnitSphere**.

```
        # 1D domains
        mesh = UnitInterval(20)      # 20 cells, 21 vertices
        mesh = Interval(20, -1, 1)   # domain [-1,1]

        # 2D domains (6x10 divisions, 120 cells, 77 vertices)
        mesh = UnitSquare(6, 10)  # 'right' diagonal is default
        # The diagonals can be right, left or crossed
        mesh = UnitSquare(6, 10, 'left')
        mesh = UnitSquare(6, 10, 'crossed')

        # Domain [0,3]x[0,2] with 6x10 divisions and left diagonals
        mesh = Rectangle(0, 0, 3, 2, 6, 10, 'left')

        # 6x10x5 boxes in the unit cube, each box gets 6 tetrahedra:
        mesh = UnitCube(6, 10, 5)

        # Domain [-1,1]x[-1,0]x[-1,2] with 6x10x5 divisions
        mesh = Box(-1, -1, -1, 1, 0, 2, 6, 10, 5)

        # 10 divisions in radial directions
        mesh = UnitCircle(10)
        mesh = UnitSphere(10)
```

3.2.2 Other techniques

Other standard techniques for importing/generating meshes into FEniCS are reading **.XML or .OFF** files, using the Mesheditor, using a few **CGAL** functions or using **dolfin-convert**.
XML and OFF files are formats which are useful for saving and communicating meshes, not for generating them. **CGAL** stands for Computational Geometry Algorithms Library. Some of the few commands in **CGAL** are **CircleMesh, EllipseMesh, SphereMesh, EllipsoidMesh, PolyhedralMeshGenerator, Add ,Subtract** etc

3.3 Linear Numerical examples

```
        -Laplace(u) = f on the unit square.
        u = u0 on the boundary.
        u0 = 1 + x^2 + 2y^2, f = -6.
```

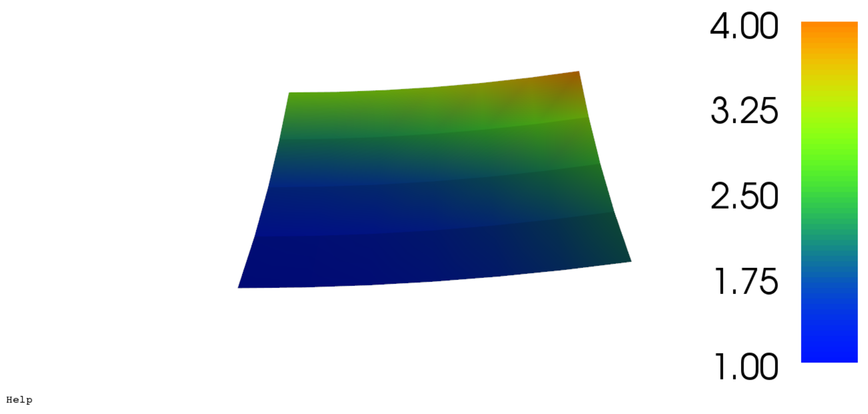


Figure 3.2: FEniCS Solution

Order of convergence data for Linear largrange element

```
        Error norm based on infinity norm (of dofs)
        h=1.25E-01 E=3.75E-02 r=1.88
        h=6.25E-02 E=9.57E-03 r=1.97
        h=3.12E-02 E=2.41E-03 r=1.99
        h=1.56E-02 E=6.02E-04 r=2.00
        h=7.81E-03 E=1.51E-04 r=2.00
        h=3.79E-03 E=3.54E-05 r=2.00
```

Dirichlet Neumann Problem

```
- div grad u(x, y) = f(x, y)
on the unit square with source f given by
f(x, y) = 10*exp(-((x - 0.5)^2 + (y - 0.5)^2) / 0.02)
and boundary conditions given by
u(x, y) = 0          for x = 0 or x = 1
du/dn(x, y) = sin(5*x) for y = 0 or y = 1
```

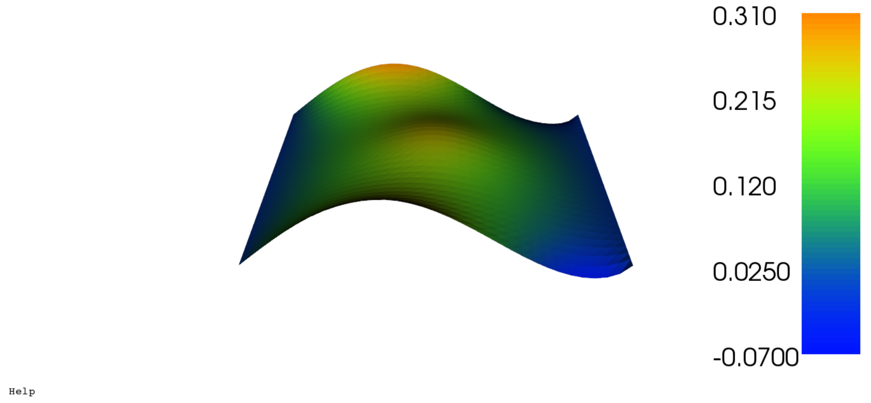


Figure 3.3: FEniCS Solution

The order of convergence data for cubic Lagrange element

```
Error norm based on infinity norm (of dofs)
h=5.00E-02 E=7.36E-06 r=3.97588
h=2.50E-02 E=4.62E-07 r=3.99406
h=1.25E-02 E=2.89E-08 r=3.99884
h=6.25E-03 E=1.78E-09 r=4.01965
```

3.4 Nonlinear Problems

Now we shall address how to solve nonlinear PDEs in FEniCS. Our sample PDE for implementation is taken as a nonlinear Poisson equation:

$$-\nabla \cdot (q(u)\nabla u) = f\,. \tag{3.5}$$

The coefficient $q(u)$ makes the equation nonlinear (unless $q(u)$ is constant in u).
The variational formulation of our model problem reads: Find $u \in V$ such that

$$F(u; v) = 0 \quad \forall v \in \hat{V}, \tag{3.6}$$

where

$$F(u; v) = \int_\Omega q(u)\nabla u \cdot \nabla v \,\mathrm{d}x, \tag{3.7}$$

and

$$\begin{aligned}\hat{V} &= \{v \in H^1(\Omega) : v = 0 \text{ on } x_0 = 0 \text{ and } x_0 = 1\},\\ V &= \{v \in H^1(\Omega) : v = 0 \text{ on } x_0 = 0 \text{ and } v = 1 \text{ on } x_0 = 1\}\,.\end{aligned} \tag{3.8}$$

The discrete problem arises as usual by restricting V and $\hat{V}$ to a pair of discrete spaces. The discrete nonlinear problem is then wirtten as: find $u \in V$ such that

$$F(u; v) = 0 \quad \forall v \in \hat{V}, \tag{3.9}$$

3.4.1 Picard Iteration

Picard iteration is an easy way of handling nonlinear PDEs: we simply use a known, previous solution in the nonlinear terms so that these terms become linear in the unknown u. The strategy is also known as the method of successive substitutions. For our particular problem, we use a known, previous solution in the coefficient $q(u)$. More precisely, given a solution u_k from iteration k, we seek a new (hopefully improved) solution u_{k+1} in iteration $k+1$ such that u_{k+1} solves the linear problem,

$$\nabla \cdot \left(q(u^k)\nabla u^{k+1}\right) = 0, \quad k = 0, 1, \ldots \tag{3.10}$$

The iterations require an initial guess u^0. The hope is that $uk \to u$ as $k \to \infty$, and that u^{k+1} is sufficiently close to the exact solution u of the discrete problem after just a few iterations.

3.4.2 Newton Method at the Algebraic Level

After having discretized our nonlinear PDE problem, we may use Newtons method to solve the system of nonlinear algebraic equations. From the continuous variational problem, the discrete version results in a system of equations for the unknown parameters $U_1, , U_N$.

In order to calculate the solution FEniCS has an inbuilt function to carry out Jacobian calculation and finally solving it by Newton Raphson iteration.

Note the important feature in Newtons method that the previous solution u_k replaces u in the formulas when computing the matrix $\partial F_i / \partial U_j$ and vector F_i for the linear system in each Newton iteration.

Chapter 4

Time Dependent Problems in FEniCS

Many researchers will prefer to code the solution strategy of the nonlinear problem themselves and experiment with various combinations of strategies in difficult problems. Time-dependent problems are somewhat similar in this respect: we have to add a time discretization scheme, which is often quite simple, making it natural to explicitly code the details of the scheme so that the programmer has full control. We shall explain how easily this is accomplished through examples.

4.1 Diffusion Problem and its Discretization

Let our problem be defined as,

$$\begin{aligned} \frac{\partial u}{\partial t} &= \nabla^2 u + f \text{ in } \Omega, \text{ for } t > 0, \\ u &= u_0 \text{ on } \partial\Omega, \text{ for } t > 0, \\ u &= I \text{ at } t = 0\,. \end{aligned} \tag{4.1}$$

Here, u varies with space and time, e.g., $u = u(x, y, t)$ if the spatial domain Ω is two-dimensional. The source function f and the boundary values $u0$ may also vary with space and time. The initial condition I is a function of space only.
The time-derivative can be approximated by a finite difference. For simplicity and stability reasons we choose a simple backward difference:

$$\frac{\partial}{\partial t} u^k \approx \frac{u^k - u^{k-1}}{\Delta t}, \tag{4.2}$$

where Δt is the time discretization parameter.

Inserting this approximation in the PDE yields

$$\frac{u^k - u^{k-1}}{\Delta t} = \nabla^2 u^k + f^k \,. \tag{4.3}$$

We use a finite element method to solve the time-discrete equations which still have spatial differential operators. This requires turning the equations into weak forms. As usual, we multiply by a test function $v \in V$ and integrate second-derivatives by parts. Introducing the symbol u for u^k , the resulting weak form can be conveniently written in the standard notation: $a_0(u, v) = L_0(v)$ for the initial step and $a(u, v) = L(v)$ for a general step, where

$$\begin{aligned}
a_0(u, v) &= \int_\Omega uv \,\mathrm{d}x, \\
L_0(v) &= \int_\Omega Iv \,\mathrm{d}x, \\
a(u, v) &= \int_\Omega (uv + \Delta t \nabla u \cdot \nabla v) \,\mathrm{d}x, \\
L(v) &= \int_\Omega \left(u^{k-1} + \Delta t f^k\right) v \,\mathrm{d}x \,.
\end{aligned} \tag{4.4}$$

The continuous variational problem is to find $u^0 \in V$ such that $a_0(u^0, v) = L_0(v)$ holds for all $v \in \hat{V}$, and then find $u^k \in V$ such that $a(u^k, v) = L(v)$ for all $v \in \hat{V}$, $k = 1, 2,$

4.2 Numerical Example

```
        du/dt -Laplace(u) = f on the unit square.
        u = u0 on the boundary.
        u = I at t=0
```

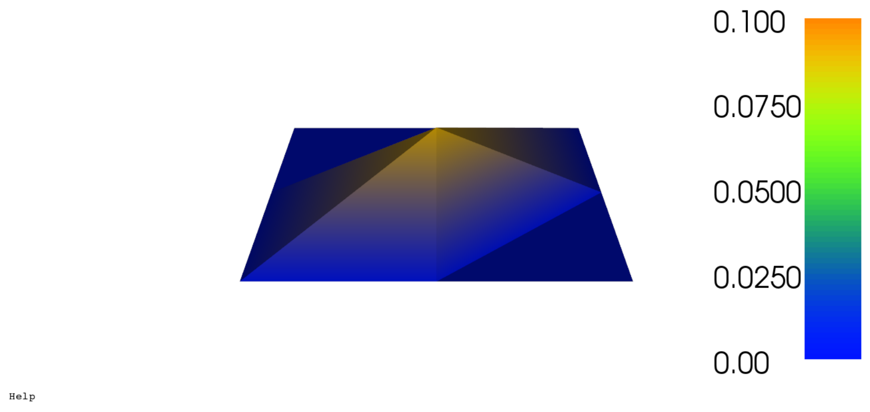


Figure 4.1: $2 * 2$ elements

Order of convergence data

```
Max error in Run 1 is 0.05900472985458862785
Max error in Run 2 is 0.02119882894487590264
Max error in Run 3 is 0.00550957587228301238
Max error in Run 4 is 0.00110141248956971416
Max error in Run 5 is 0.00005668741721423509
Order of convergence
1.47684603606
1.94397140098
2.32258639099
4.2801825243
```

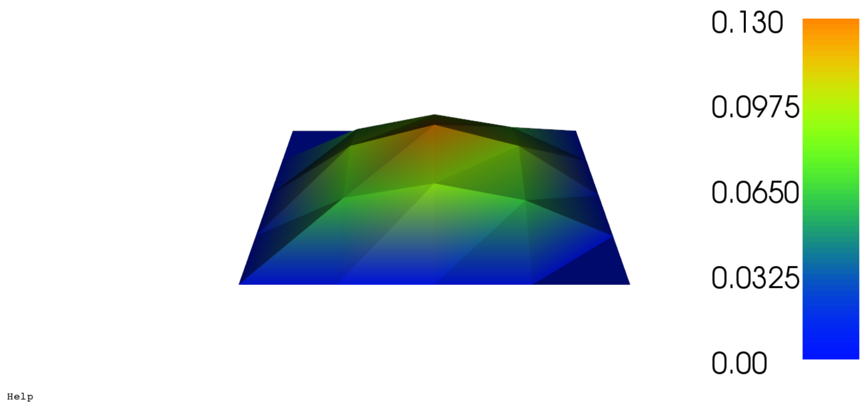


Figure 4.2: $4 * 4$ elements

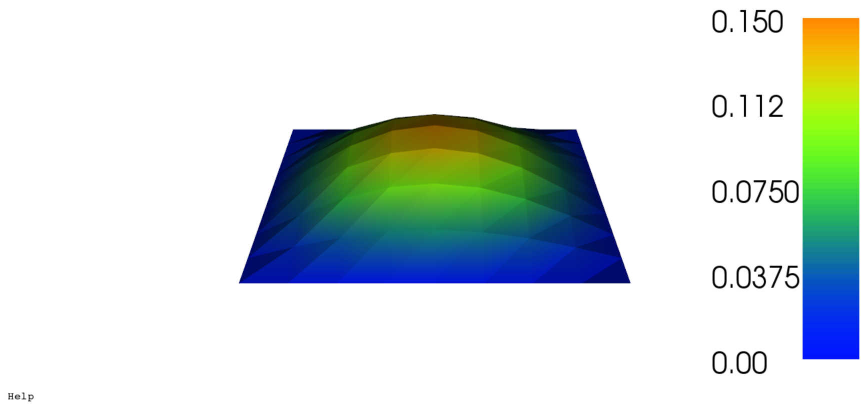


Figure 4.3: $8 * 8$ elements

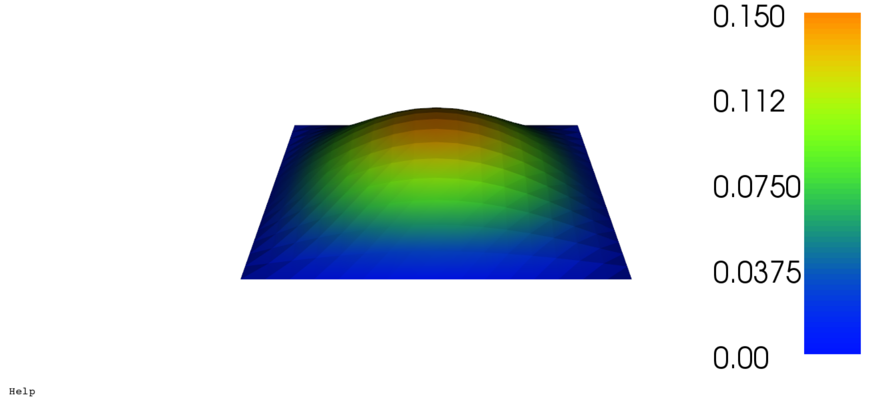


Figure 4.4: $16 * 16$ elements

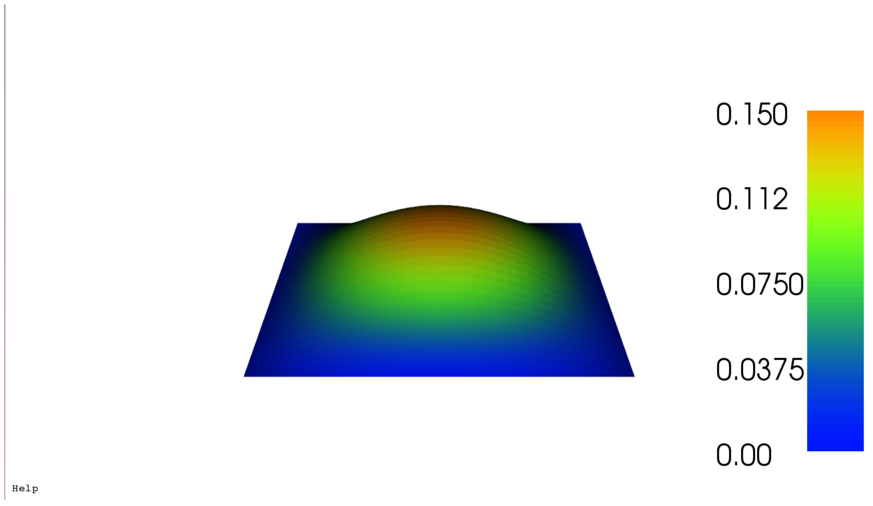


Figure 4.5: $32 * 32$ elements