MoRed Lab 1: The Proper Orthogonal Decomposition

Mohammad Shahid

August 23, 2020

Abstract

A numerical study was conducted to analyze the efficacy of the Proper Orthogonal Decomposition (POD) model reduction technique to approximate the solution of the full order one dimensional transient heat conduction problem. The problem was formulated using the finite element method. Several runs of the full order model were taken with and without source term to investigate the effect of various parameters on the solution of the problem. Then a run for 5 seconds of the full order finite element model was taken in order to extract the reduced basis using the given 'pod' function. Then result of the whole 40 seconds process, simulated using the reduced model, was compared to the full order finite element solution and the L2 norm error was plotted as a function of space. The result show that POD approximation captures the dynamics of the system quite well with a significantly reduced number of basis.

Introduction

Proper Orthogonal Decomposition (POD) is an efficient method which can extract characteristic samples from a large data set and can combine with interpolation method to reconstruct complex problems by establishing the reduced-order model. The reduced order model can substantially reduce the computational time by reducing the number of degrees of freedom. Hence, POD has become one of the hot-spots in the field of numerical calculation of flows and heat transfer, and it has been widely used to analyze practical problems.

POD is based on the numerical solution to construct a linear system rather than directly linearize the control equations with a good processing power for both linear and non-linear problems, so it is continuously studied and applied in many fields, such as Singular Value Analysis and Simple Identification, statistics, image processing, turbulent drag-reduction, etc. Time dependent matrix is calculated instead of the spatial covariance matrix to solve the basis function, which saves the storage space and computing time greatly. The POD has been widely used in the study of heat and mass transfer.

Problem Description

The heat conduction equation is a parabolic partial differential equation (PDE) which has been used extensively in modeling of many engineering problems such as heat transfer by

extended surfaces, cooling of reactor rods in nuclear reactors, etc. In statistics, the heat equation is connected with the study of Brownian motion via the Fokker-Planck equation. In the special case of heat conduction in an isotropic and homogeneous material in one dimensional space, the equation can be written as:

$$\rho c_p \frac{\partial \mathbf{u}}{\partial t} - k \frac{\partial^2 \mathbf{u}}{\partial x^2} = \mathbf{s} \tag{1}$$

where, $\mathbf{u} \equiv u(x,t)$ is the temperature as a function of space and time, ρ, c_p, k are the material properties, namely, density, specific heat capacity and the thermal conductivity, respectively, and $\mathbf{s} \equiv s(x,t)$ is the heat source as a function of space and time. In the present numerical experiments the following two forms of boundary conditions were applied at the two ends of the bar of length L:

1. Dirichlet Boundary Condition: $u(x_L,t) = u_L(t), \;\; u(x_R,t) = u_R(t)$
2. Neumann Boundary Condition: $\frac{\partial u}{\partial t}\big|_{x_L,t} = -q_L(t), \;\; \frac{\partial u}{\partial t}\big|_{x_R,t} = -q_R(t)$

where subscript 'L' refers to the Left end and 'R' refers to the right end of the bar. The solution was initialized with a uniform value of temperature at all nodes
$u(x,0) = u_0(x)$. In the present study u_0 is taken to be zero.

Numerical Methodology

The governing equation (1) was discretized using the finite element method (FEM). Let us introduce the weak formulation of the given problem (1) as follows:

$$\int_0^L \big[\rho c_p \frac{\partial u(x,t)}{\partial t} - k \frac{\partial^2 u(x,t)}{\partial x^2}\big] v(x) dx = \int_0^L s(x,t) v(x) dx \tag{2}$$

where $v(x)$ is a test function of our choice in a certain vector space.
Assuming that N nodes are used to discretize the domain. After introducing the finite element approximation one can arrive at the following sets of equations:

$$\mathbf{C}\dot{\mathbf{u}}(t) + \mathbf{K}\mathbf{u}(t) = \mathbf{f}(t) \tag{3}$$

where, $\mathbf{C}, \mathbf{K} \in^{N \times N}$ and $\mathbf{u}, \mathbf{f} \in^{N}$ and the right hand side of the equation (3) takes into account the source term and the Neumann boundary conditions:

$$\mathbf{f}(t) = \mathbf{M}\mathbf{s}(t) + \mathbf{q}(t)$$
$$\text{with } \mathbf{q}(t) = [\; q_L \; 0 \ldots 0 \; q_R \;]^T$$

The time integration was implemented in the code in such a way that depending on the value of a parameter θ, the scheme will switch to:

- First Order backward Euler (Explicit) for $\theta = 0$

- First Order forward Euler (Implicit) for $\theta = 1$
- Crank-Nicolson method (Implicit) for $\theta = 0.5$

Assuming that at time k the solution is known, the solution at time $k+1 (t_{k+1} = t_k + \Delta t)$ is computed as:

$$\mathbf{u}_{k+1} = \mathbf{G}^{-1}\mathbf{b} \; with \; \begin{cases} \mathbf{G} = \mathbf{C} + \theta \Delta t \mathbf{K} \\ \mathbf{b} = (\mathbf{C} - (1-\theta)\Delta t \mathbf{K})\mathbf{u}_k + \Delta t \hat{\mathbf{f}} \end{cases} \tag{4}$$

and $\hat{\mathbf{f}} = \mathbf{f}_k + \theta(\mathbf{f}_{k+1} - \mathbf{f}_k)$.

Exercise 1

A script 'Ex1.m' was created which executes the function HT1D_FEM_SS11.m. Five different runs were taken to explore the effect of various parameters on the solution. Table 1 gives a short description of the various cases considered.

Case	ρ	k	c_p	Boundary Conditions	$s(x,t)$	Scheme
1	1	1	40	$q_L = 0$	0	Forward Euler
2	1	1	40	$q_L = 1,\ q_R = 0$	0	Forward Euler
3	1	1	80	$q_L = 0$	0	Forward Euler

Table 1: Brief description of various cases

In all the cases the length of the bar is taken as 1 and the simulations were run for 40 seconds.

Case1: Neumann BC on Left end of the bar and $C_p = 40$

In this case we impose a Neumann BC on the left end of the bar by specifying the derivative of temperature with respect to length as $q_L = -\frac{\partial u}{\partial x}\Big|_L = 0$.

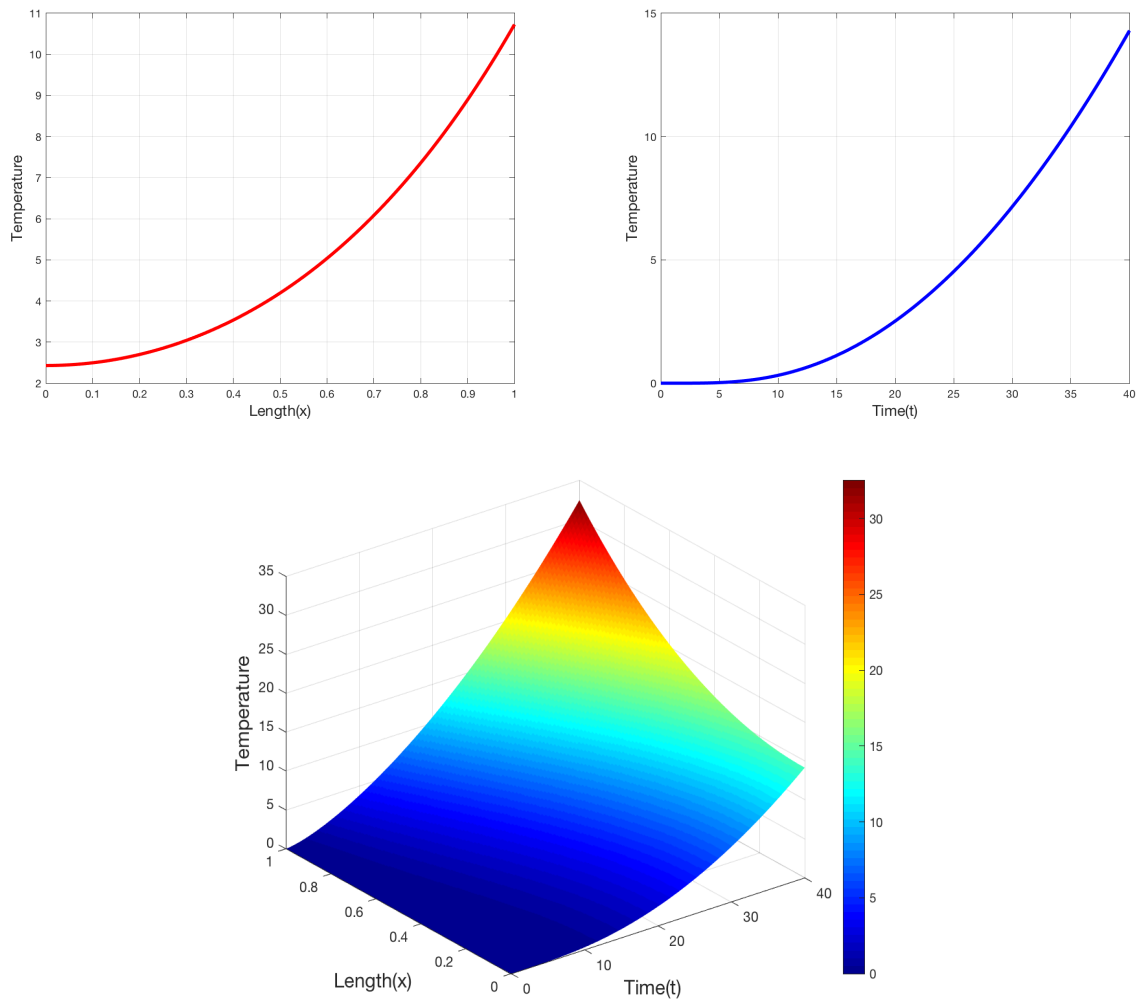


Figure 1: Variation of Temperature with Length and Time

The solution agress well with the prescribed boundary conditions. In order to test the nature of the solution, several other cases were run with different sets of boundary conditions.

Case2: Neumann BC on Left and Right end of the bar and $C_p = 40$

In this case we impose Neumann BC on the left and end of the bar by specifying the derivative of temperature with respect to length as $q_L = -\frac{\partial u}{\partial x}\Big|_L = 1$ and $q_R = -\frac{\partial u}{\partial x}\Big|_R = 0$.

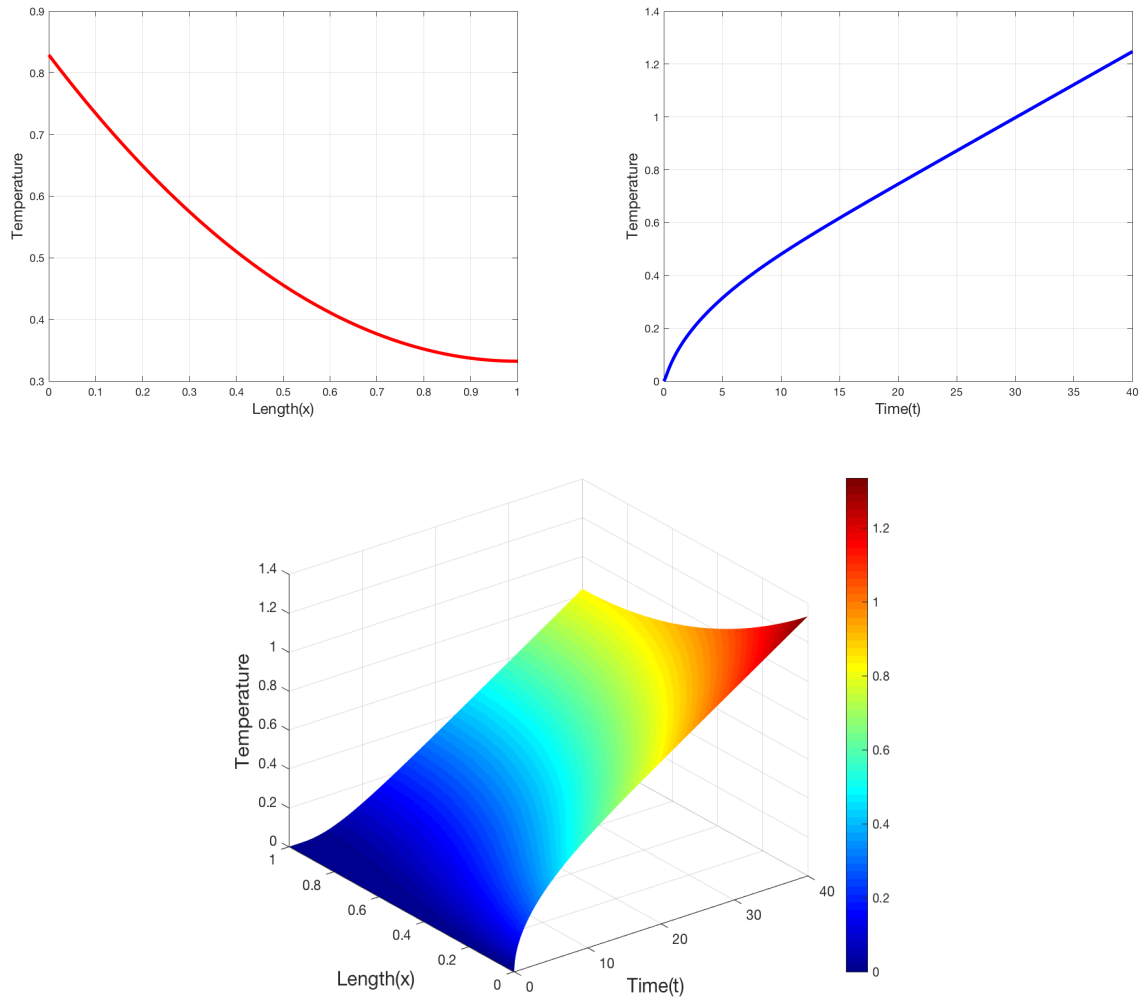


Figure 2: Variation of Temperature with Length and Time

Case3: Neumann BC on Left end of the bar and $C_p = 80$

In this case we impose Neumann BC on the left and end of the bar by specifying the derivative of temperature with respect to length as $q_L = -\frac{\partial u}{\partial x}\Big|_L = 1$ and took $C_p = 80$.

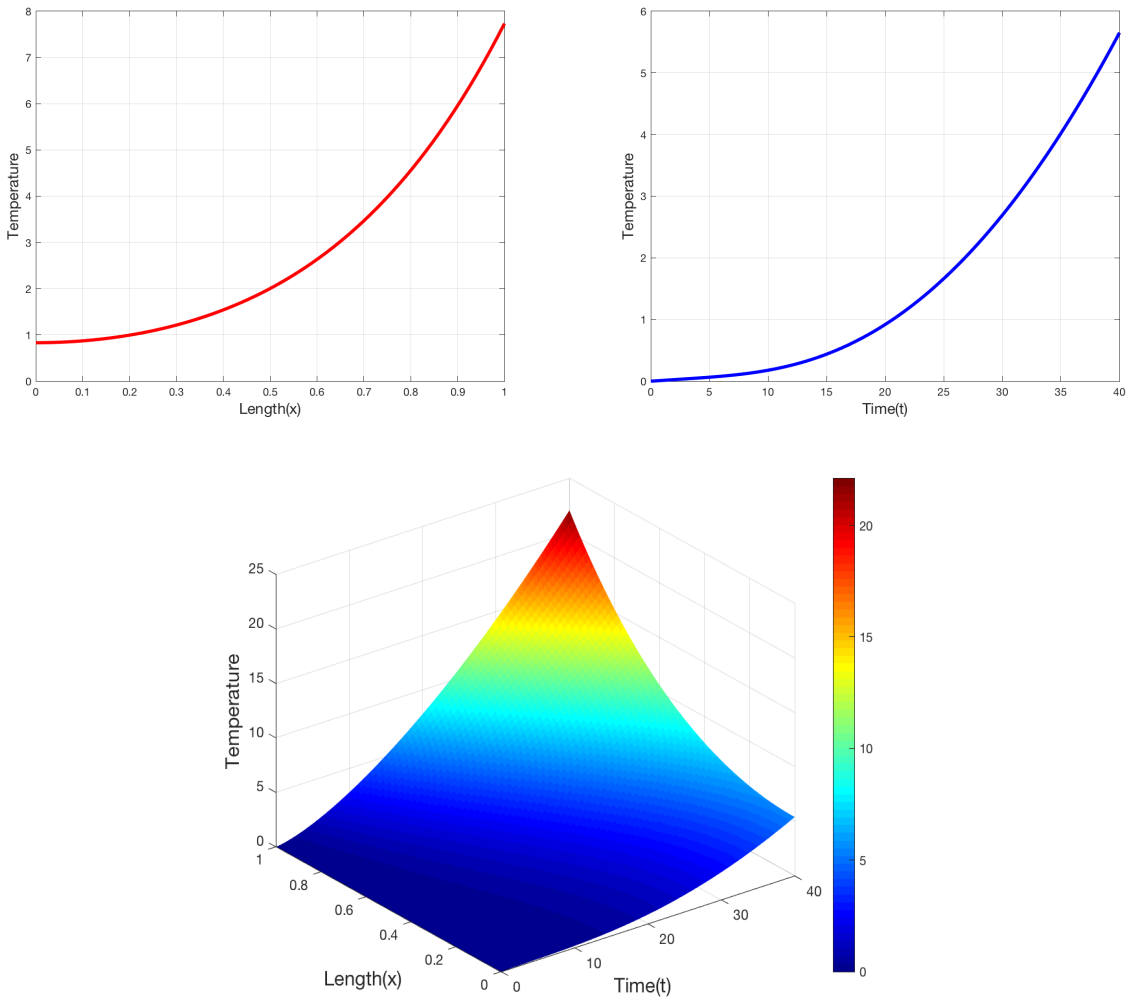


Figure 3: Variation of Temperature with Length and Time

As compared to the first case, we don't find any significant change in the solution by decreasing the value of thermal diffusivity (i.e. by increasing the value of c_p).

Exercise 2

The heating process originated by the flux shown in figure (4) is simulated for 40 seconds using the Implicit Euler time integration with `dt=0.08` seconds. The domain of unitary length is discretized using 51 nodes. The material values were taken as `k=1`, `rho=1` and `Cp=50`, where the symbols have there usual meanings. The simulation was performed and the result was saved as `Full_Solution`. The temperature distribution and variation of temperature with time is plotted in figure (5).

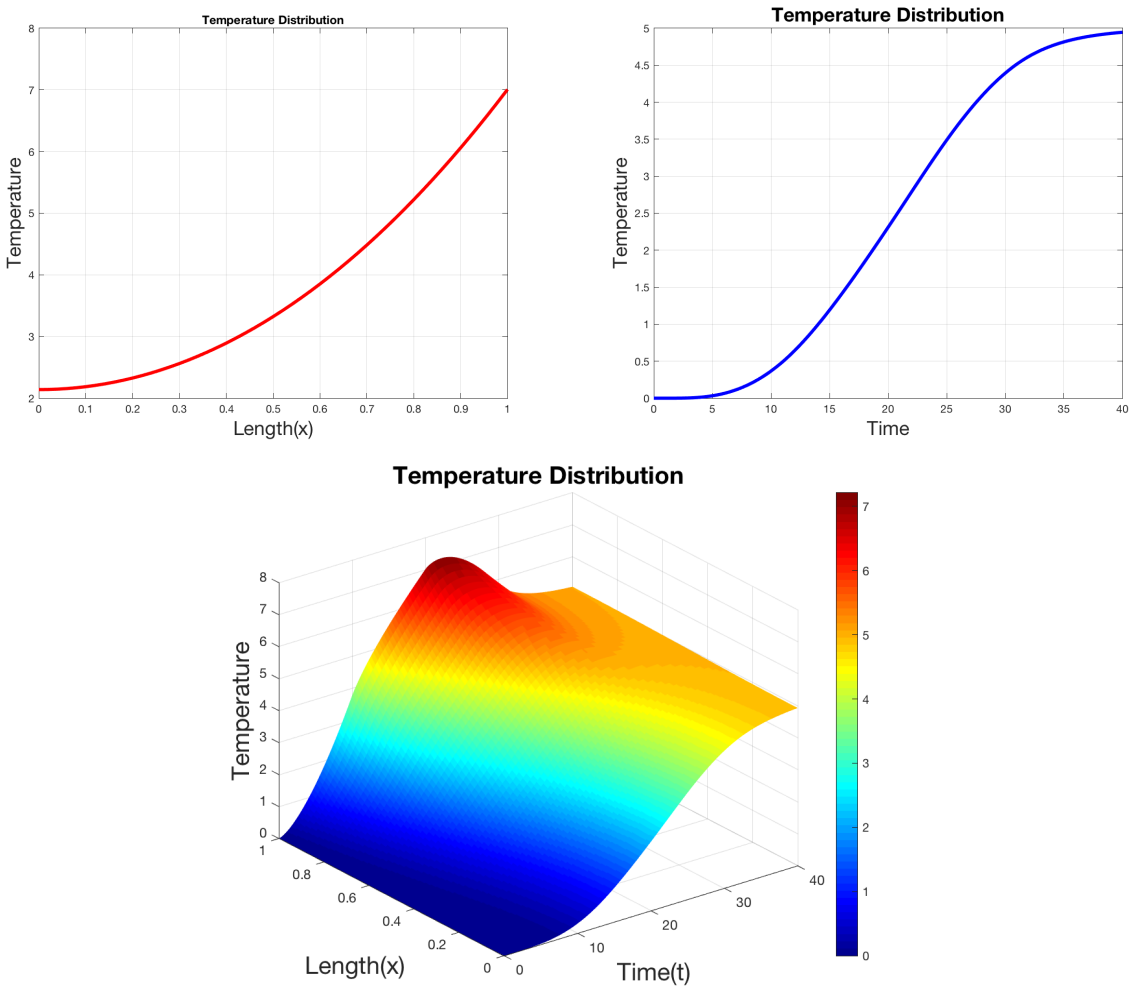


Figure 4: Variation of Temperature with Length and Time

Exercise 3

In order to apply Proper Orthogonal Decomposition for further reducing the computation time, the previous case was ran again for `T=5` seconds to get an initial sample solution for computing reduced basis. The initial solution profiles are plotted in figure (6).

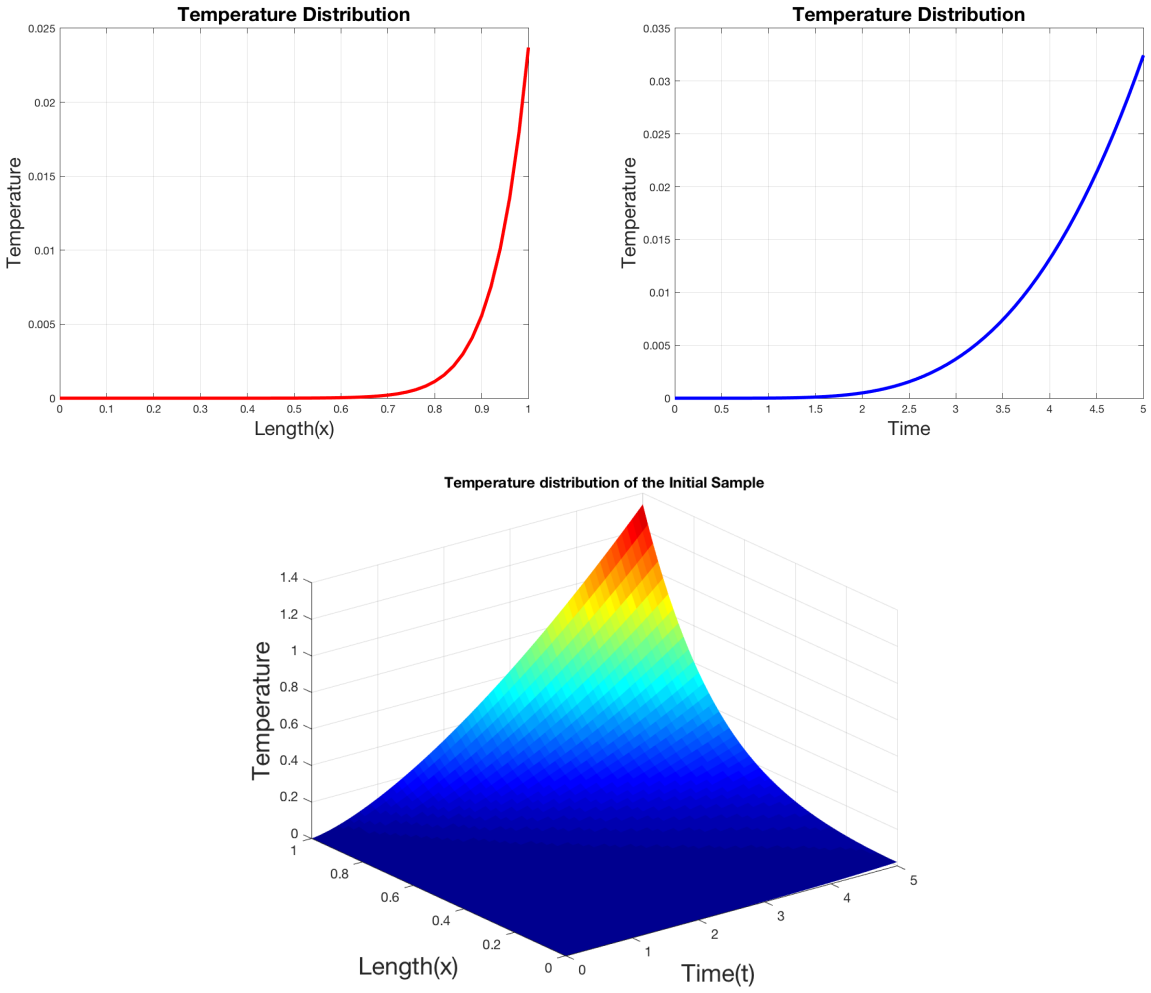


Figure 5: Variation of Temperature with Length and Time for Initial Sample

Exercise 4

The function `pod` was completed to perform the `POD` model order reduction. It computes the left (space) singular vectors and the singular vectors.

Exercise 5

In this exercise the reduced basis are computed from the initial sample solution of the full order model. The the function `POD` was used to simulate the 40 sec of the process. The solution obtained using the POD function is presented below.

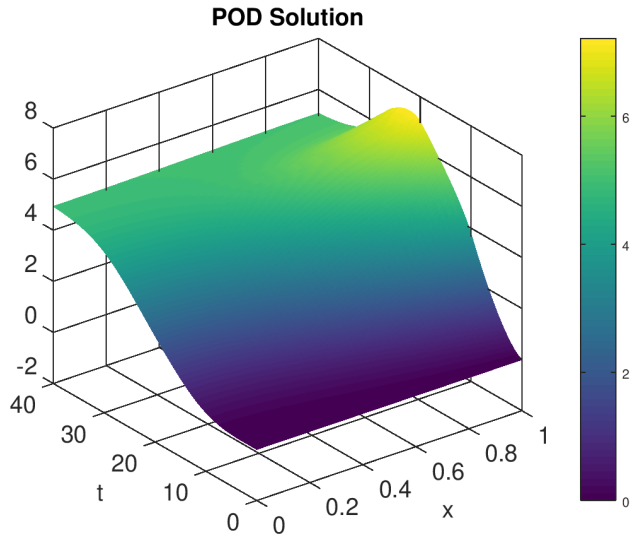


Figure 6: POD Solution for whole 40 sec process

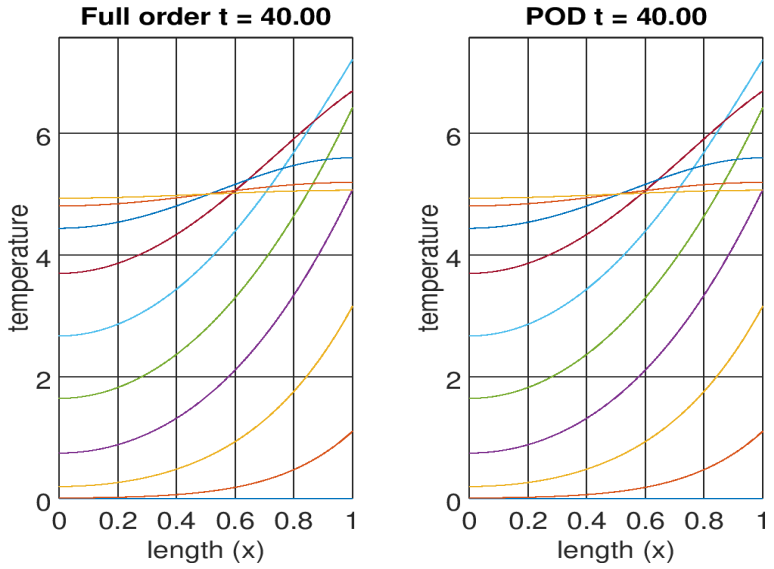


Figure 7: Comparison between Full Order and POD solution

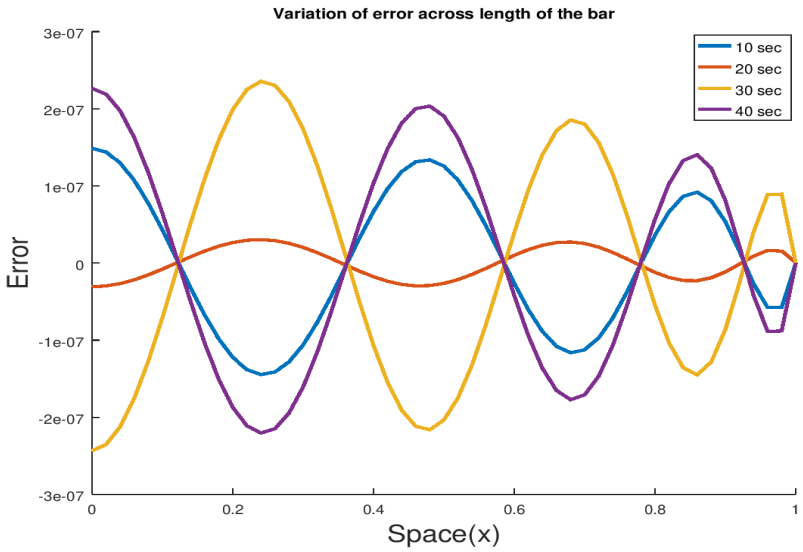


Figure 8: Error distribution across the length of the bar at different time instants

From figure (8), we can conclude that the difference between the POD solution and the full order solution is quite small, of the order of 10^{-7}. The error between the exact solution and the POD solution increases as the solution evolves with time. We can clearly see that upto 20 seconds the error is negligible as the solution is still near the instant where the reduced basis are calculated. The error will be more pronounced if the solution is allowed to run for more time. The wavy character of the error distribution may be attributed to the poor sampling of the initial solution.

MoRed Lab 3: Space-Time Representations with Proper Generalized Decomposition

Mohammad Shahid

August 23, 2020

Abstract

A numerical study was conducted to analyze the efficacy of the Proper Generaliized Decomposition (PGD) model reduction technique to approximate the solution of the full order one dimensional transient heat conduction problem. The problem was formulated using the finite difference method. The process was simulated using the reduced model for 0.1 seconds and the results obtained from the PGD solution is compared to the exact solution and the effect of threshold on L2 norm error is presented. The result show that PGD solution agrees well with exact solution with an accuracy of about 2%.

Introduction

We begin with the following one-dimensional transient heat transfer equation for computing the temperature field $u(x,t)$ in the space time domain $\Omega = \Omega_x \times \Omega t = (0,L) \times (0,T)$.

$$\rho c_p \frac{\partial u}{\partial t} - k\frac{\partial^2 u}{\partial x^2} = f(x,t) \tag{1}$$

with homogeneous initial and boundary conditions

$$u(x_L,t) = 0, \qquad u(x_R,t) = 0, \quad \& \quad u(x,0) = 0$$

The material properties density (ρ), specific heat capacity (c_p) and thermal conductivity (k) are assumed to be constant.
The weighted residual form of equation (1) reads:

$$\int_{\Omega_x \times \Omega_t} u^* \left(\rho c_p \frac{\partial u}{\partial t} - k\frac{\partial^2 u}{\partial x^2} - f \right) dxdt = 0 \tag{2}$$

for all suitable test functions u^*. The objective is to obtain a PGD approximate solution in the separated form by computing each term of the expansion at each step of an enrichment process, until a suitable stopping criteria is met.

$$u(x,t) = \sum_{i=1}^{N} \mathbf{X}_i(\mathbf{x}).\mathbf{T}_i(\mathbf{t}) + \mathbf{S}.\mathbf{T} \tag{3}$$

Substituting equation () in equation () and assuming the test function u^* as $u^* = \mathbf{T}^*.\mathbf{S} + \mathbf{S}^*.\mathbf{T}$, to get the weak form:

$$\int_0^T \int_0^L (\mathbf{T}^*.\mathbf{S}+\mathbf{S}^*.\mathbf{T})\left\{\rho c_p \frac{\partial \mathbf{T}}{\partial t}.\mathbf{S} - k\Delta\mathbf{S}.\mathbf{T} - \left(\sum_{i=1}^{M} k\Delta\mathbf{F}_1^i.\mathbf{F}_2^i - \mathbf{F}_1^i.\frac{\partial \mathbf{F}_2^i}{\partial t}\right) - \sum_{j=1}^{N} \mathbf{f}_1^j.\mathbf{f}_2^j\right\} dx\, dt = 0 \tag{4}$$

The above equation can be split in to two separate boundary value problems, one which governs the temperature distribution is space and one governing the temporal evolution of the temperature field.

Space Problem:

$$[\alpha_1(\mathbf{T})\mathbf{I}_x - \beta_1(\mathbf{T})\mathbf{K}_x]\mathbf{S} = \sum_{j=1}^{N} \gamma_1^j(\mathbf{T}, \mathbf{f}_2^j)\mathbf{f}_1^j - \sum_{i=1}^{M} [\alpha_1^i(\mathbf{T}, \mathbf{F}_2^i)\mathbf{I}_x - \beta_1(\mathbf{T}, \mathbf{F}_2^i)\mathbf{K}_x]\mathbf{F}_1^i \tag{5}$$

where,

$$\alpha_1(\mathbf{T}) = \int_0^T \mathbf{T}.\frac{\partial \mathbf{T}}{\partial t}\, dt \qquad \beta_1(\mathbf{T}) = \int_0^T \mathbf{T}.\mathbf{T}\, dt \qquad \gamma_1^j(\mathbf{T}, \mathbf{f}_2^j)\, dt$$

$$\alpha_1^i(\mathbf{T}, \mathbf{F}_2^i) = \int_0^T \mathbf{T}.\frac{\partial \mathbf{F}_2^i}{\partial t}\, dt \qquad \beta_1^i(\mathbf{T}, \mathbf{F}_2^i) = \int_0^T \mathbf{T}.\mathbf{F}_2^i\, dt$$

Time Problem:

$$[\alpha_2(\mathbf{S})\mathbf{G}_t - \beta_2(\mathbf{S})\mathbf{I}_t]\mathbf{T} = \sum_{j=1}^{N} \gamma_2^j(\mathbf{S}, \mathbf{f}_1^j)\mathbf{f}_2^j - \sum_{i=1}^{M} [\alpha_2^i(\mathbf{S}, \mathbf{F}_1^i)\mathbf{G}_t - \beta_2^i(\mathbf{S}, \mathbf{F}_1^i)\mathbf{I}_t]\mathbf{F}_2^i \tag{6}$$

where,

$$\alpha_2(\mathbf{S}) = \int_0^L \mathbf{S}.\mathbf{S}\, dx \qquad \beta_2(\mathbf{S}) = \int_0^L \mathbf{S}.\Delta\mathbf{S}\, dx \qquad \gamma_2^j(\mathbf{S}, \mathbf{f}_1^j) = \int_0^L \mathbf{S}\mathbf{f}_1^j\, dx$$

$$\alpha_2^i(\mathbf{S}, \mathbf{F}_1^i) = \int_0^L \mathbf{S}.\mathbf{F}_1^i\, dx \qquad \beta_2^i(\mathbf{S}, \mathbf{F}_1^i) = \int_0^L \mathbf{S}.\Delta\mathbf{F}_1^i\, dx$$

In that notation, $\mathbf{F}_1^i$ is the i-th column of $\mathbf{F}_1$ and $\mathbf{F}_2^i$ is the i-th column of F_2. Analogously, $\mathbf{I}_1^j$ is the j-th column of f_1 and f_2^j is the j-th column of f_2. Due to essential non-linearity in the nature of the two equations arising due to the enrichment terms, a non-linear iteration is required. The iteration process is continued until a suitable stagnation criteria is satisfied and after obtaining the converged values of $\mathbf{S}$ and $\mathbf{T}$, the enrichment process is continued until required tolerance.

Exercise

The function `HT1D_PGD_XT` was completed. It computes space-time separated representations of the transient heat equation. After completing the function, a script was created to compare

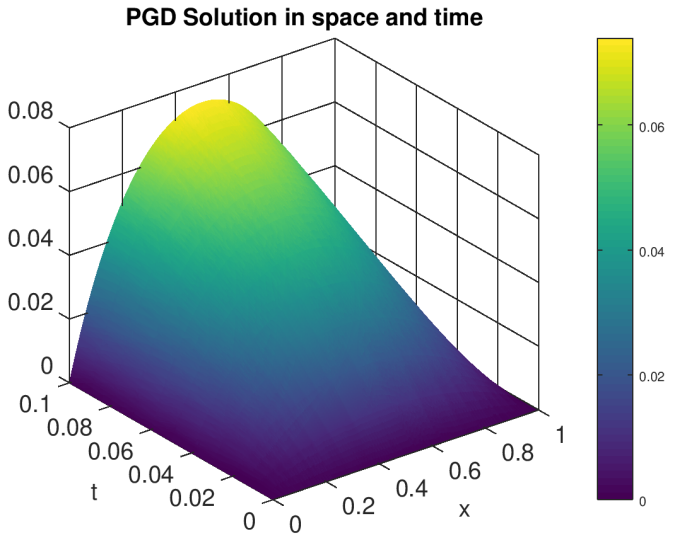
the solution obtained using the function `HT1D_PGD_XT` with the exact solution. The problem is stated as:

$$\frac{\partial u}{\partial t} - \frac{\partial^2 u}{\partial x^2} = 1 \tag{7}$$

with homogeneous initial and boundary conditions

$$u(x_L, t) = 0, \qquad u(x_R, t) = 0, \quad \& \quad u(x, 0) = 0$$

The material properties ρ, c_p and k are taken as unity The length of the domain L is taken as 1 and the solution is run for $T = 0.1$ seconds. The results obtained from the PGD solution are presented below:



.height.height

Figure 1: Surface plot of the PGD solution

The variation of error with the threshold (terms enrichment criteria) is plotted in figure 2.

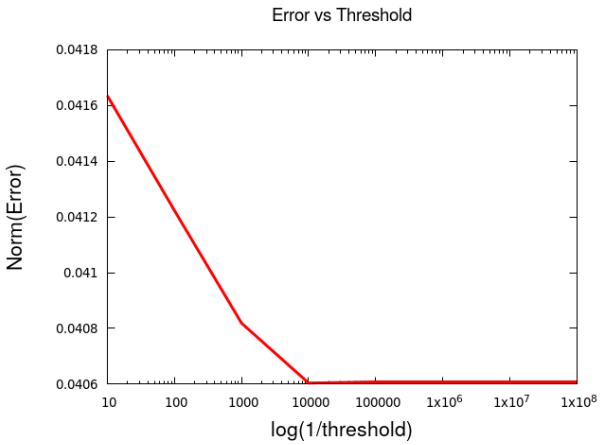


Figure 2: Variation of error with threshold

The value of the norm of the error decreases as the value of the threshold for exiting the enrichment loop is decreased. Decreasing the value of threshold below 10^{-5} doesn't affect the value of the norm of the error. As we are decreasing the value of the threshold, more terms are being added to the enrich the PGD solution and hence, increasing the accuracy of the solution.

The evolution of error with time is plotted in figure 3.

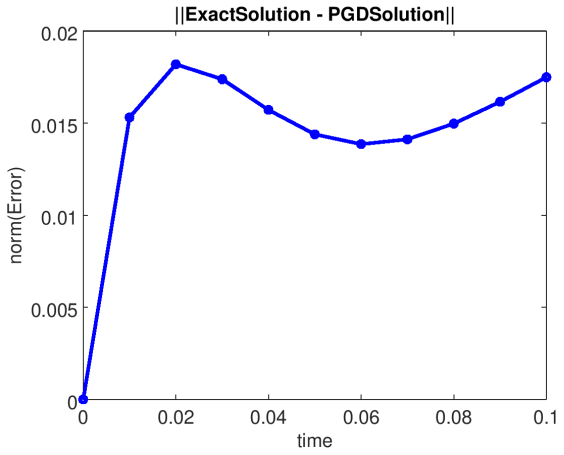


Figure 3: Evolution of error with time

From figure 3, we can conclude that the approximated solution obtained using the PGD model reduction technique is accurate up to 1.8% at t=0.1s.

Model Reduction Lab-4
PGD seperated representations in cartesian domains and Code Generalization

Mohammad Shahid

August 23, 2020

Contents

Abstract

This report we are going to use the same approach which was implemented in lab-3. This approach is called Proper Generalized Decomposition;in the previous lab we use this method in order to solve the transition heat equation which depended on two separate variable space(x) and time(t). In this time we work on solving Poisson Equation which depend on two space variable(x,y). An approximate PGD formulation to the last lab was used in the separation of our new variables in order to built the numerical solution. For the second part we try to generalize the PGD code used in MATLAB, therefore this method can be applied to any problem with appropriate tensor structure like for example we can use PGD solver to construct the solution of any problem which depends on two or more separate variables like in case of 3D domain (x,y,z).

1 Introduction

As we mention in the abstract a PGD method is going to be used in order to solve the given Poisson equation. So by definition, PGD makes use of the separation of variable, allowing to reduce the computational complexity of cartesian domains such as plates. PGD can also be applied to reduce the computational complexity by computing in- plane=out-plane separated representation and it could be applied to non-cartesian domain. In the first part of the report we consider a Poisson equation in a 2D rectangular domain with homogeneous dirichlet boundary condition on four edges. PGD is applied to compute the solution as a succession of 1D problems (One defined in the x-coordinate and other defined in the y-coordinate). Also in this report we are asked to test different source term with several modes. In addition to this part we will end up by comparing PGD solution with a given exact solution by defining two kind of error one is PGD error which depends on the number of enrichment and the discretization method which arises from different discretization technique such as finite difference. In the second part we are asked to use a general function that compute separated representation of any problem with appropriate tensor structure using PGD method.

2 Part A

2.1 Exercise 1

At first, we were assigned to complete the PGD function and test it on the given Poisson equation. At this time, we have two options in order to solve the equation, one is been introduced before in lab3 and the second one is by starting with a priori known modes of the solution.

2.1.1 Problem Setup

$$\nabla^2 u = f(x,y) \quad \text{in }]0,2[\times]0,1[\tag{1}$$

$$u = 0 \quad \text{on boundaries} \tag{2}$$

2.1.2 PGD Formulation

$$\alpha_1(S)I_x R + \beta_1(S)K_x R = \gamma_1(S,f_2)f_1 - (\sum_{i=1}^{M} \alpha_1^i(S,F_2^i)I_x F_1^i + \beta_1^i(S,F_2^i)K_x F_1^i) \tag{3}$$

$$\alpha_2(R)I_y S + \beta_2(R)K_y S = \gamma_2(R,f_1)f_2 - (\sum_{i=1}^{M} \alpha_2^i(R,F_1^i)I_y F_2^i + \beta_2^i(R,F_1^i)K_y F_2^i) \tag{4}$$

2.1.3 Matlab Codes

In order to deal with the Poisson equation, we were asked to use PGD method which is built by using separation variable way in order to simplify the equation and solve it in 2 given spaces (x,y) separately. So the coefficients in PGD formulation were obtained by decoupling $u_{solution}$ into two variables (u=R(x)S(y)). Then the same procedure which were taken in the previous lab was done by getting the weak formulation and plugging the finite element discretization to end up with having all the necessary coefficients.

```
%Discrete operators along each dimension
%Identity matrix long each dimension
Ix = speye(Nx);
Iy = speye(Ny);%modified
%Finite Differences second order differentiation matrices
Kx = spdiags([ones(Nx,1)  -2*ones(Nx,1)  ones(Nx,1)],[-1 0 1],Nx,Nx)/hx^2;%modified
Ky = spdiags([ones(Ny,1)  -2*ones(Ny,1)  ones(Ny,1)],[-1 0 1],Ny,Ny)/hy^2;
```

Figure 1: Filling Identity and Differentiation Matrices

In figure 1, it is shown that it is just building the discrete operators along each dimensions, in case K_x, it is been gotten by using FEM discretization along x-axis.

```
%LHS coefficients
alpha_1 = trapz(y,S.*(Ky*S));
beta_1  = trapz(y,S.^2);%modified
%Construction of the bx
gamma_1 = trapz(y,S.*f2);
bx = gamma_1*f1;
%In case this is not the first enrichment, previous terms are
%to the bx
if (term>1)
    %bx coefficients
    if nargin==8
      OPTION 1
    alpha_1_i = trapz(y,bsxfun(@times,S,Ky*F2));
    beta_1_i = trapz(y,bsxfun(@times,S,F2));
    bx = bx - ((Ix*F1)*alpha_1_i' + (Kx*F1)*beta_1_i');%modif
```

Figure 2: Coefficients Computed in x-direction

```
%LHS coefficients
alpha_2 = trapz(x,R.*(Kx*R));%modified
beta_2  = trapz(x,R.^2);
%Construction of the by
gamma_2 = trapz(x,R.*f1);%modified
by = gamma_2*f2;
%In case this is not the first enrichment, previous terms are add
%to the by
if (term>1)
    %by coefficients
      OPTION 1
    if nargin==8
    alpha_2_i = trapz(x,bsxfun(@times,R,Kx*F1));%modified
    beta_2_i = trapz(x,bsxfun(@times,R,F1));
    by = by -((Iy*F2)*alpha_2_i' + (Ky*F2)*beta_2_i');%modified
```

Figure 3: Coefficients Computed in y-direction

As we mentioned before, our purpose is to find all the coefficients in order to find the numerical solution using PGD method. The above figures represent the computing part of those coefficients after plugging the weak formulation into the Poisson equation while using separation method to solve in each direction.

2.2 Exercise 2

At this time, we were asked to test the exact solution file which was given and compare it to the numerical solution that we got after completing the first part.

```
%Exercise2: to test the exact solution
%Nex=50;
L=2; H=1; dx = 0.02; dy = 0.02;
Max_fp_iter=200;
epsilon=1.e-5; epsilon_tilde=1.e-7;
```

Figure 4: Criteria Problem

```
%x = linspace(0,L,L/dx+1); y = linspace(0,H,H/dy+1);
      x = linspace(0,L,N(j)); y = linspace(0,H,N(j));
      Nx = numel(x); Ny = numel(y);
```

Figure 5: Creating Domain

```
Nex = 200;
U = ExactSolution(x,y,Nex);
Uexact=U';
```

Figure 6: Test Function

In the part, we created a script in order to call the exact given solution function and to test it. We define the domain (x,y) since this function takes those arguments, plus we gave a random value N for number of terms.

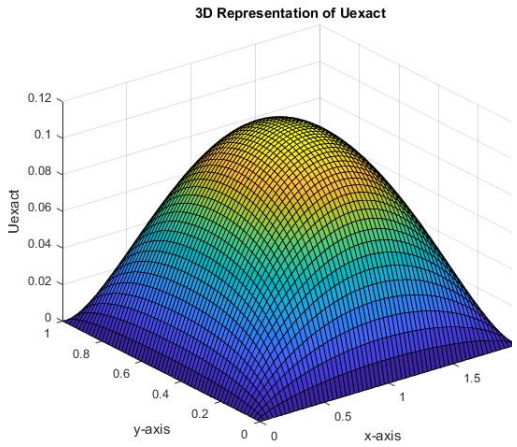


Figure 7: 3D Distribution of Uexact

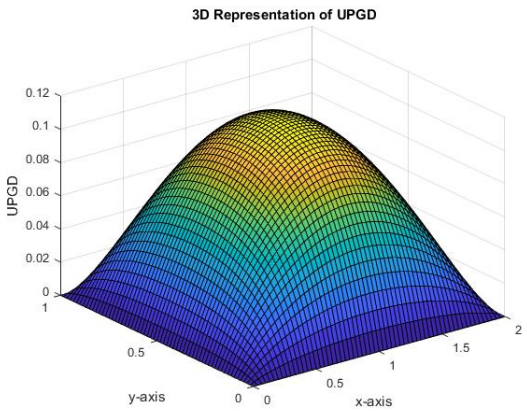


Figure 8: 3D Distribution of UPGD

After running the script, we obtained the above figures which represents the 3D distribution of both analytical and numerical solutions(OPTION1). We can notice that there is a slightly difference between them. As we said before there were two options to compute numerical solutions, one depends on increasing the number of enrichment. The second way of computed numerical solution was depending on priori known modes of the solution ($F1_0$ and $F2_0$) given in the lab. The following figure represents the second option numerical solution:

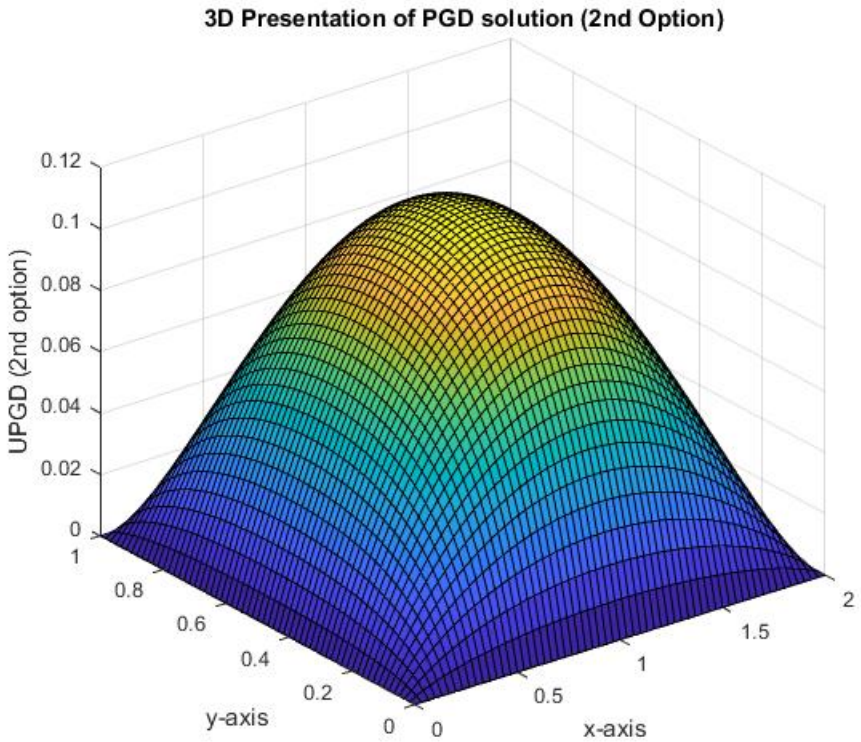


Figure 9: 2nd Option Of PGD Solution

Here we can observe there is no obvious difference between the two options of solving the numerical solutions. So the error which we are going to compute later can be taken from U1 or U2 compared to Uexact.

2.3 Exercise 3

In this part we were asked to compute the approximation error committed by the PGD approximation whereas there were two kind of errors, one is called the PGD error and the second one is called discretization error which depends on separated discretization technique.
The error was computed using the following formula:

$$e_N^M = \frac{||u_N^{ex} - u_N||_\infty}{||u_N^{ex}||_\infty} \tag{5}$$

So we obtained the following figures:

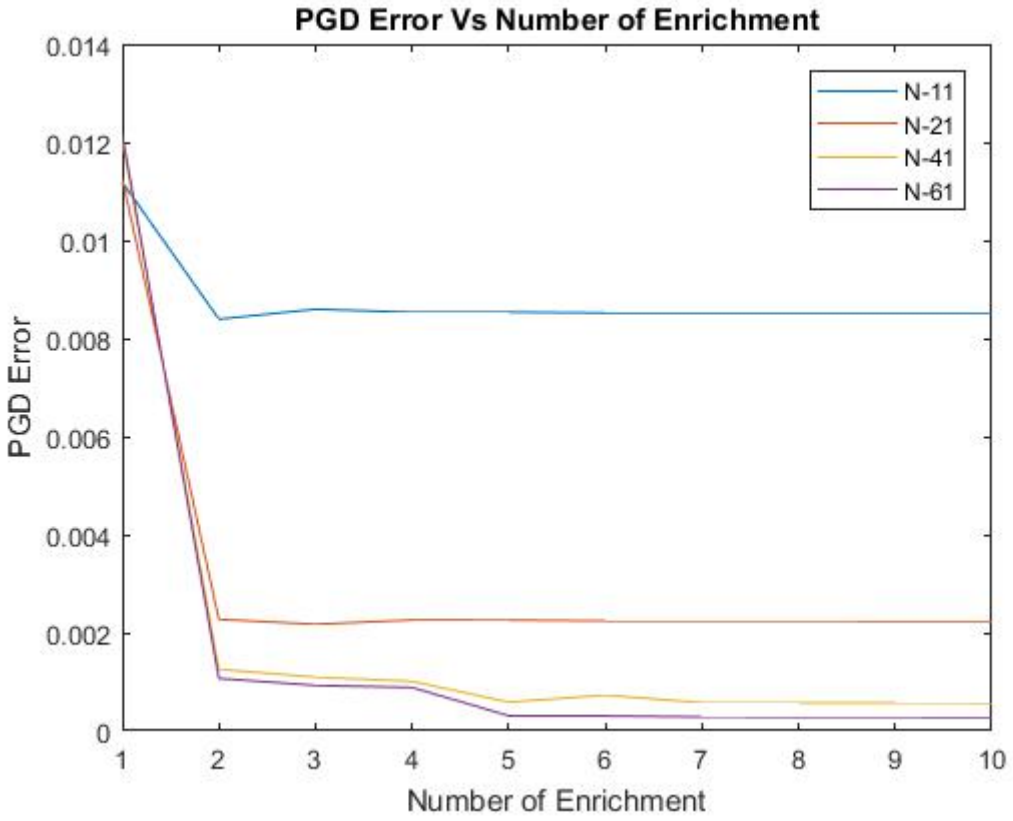


Figure 10: Error Evaluation Vs Number of Enrichment With variation of different meshes

In figure 10, we can analysis the variation error. First for all different number of meshes the error will start approximately at same point between 0.011 and 0.012, then it decreases dramatically while the number of meshes increases by noticing N=11 has the highest error, from that we can deduce that the more the increasing the number of meshes the more the error decreasing; this can be come from discretization error. Also we notice that the error is affected by the number of enrichment which comes from the use of PGD method, as we can see, the error decreases while the number of enrichment increases; this can be called the PGD error.

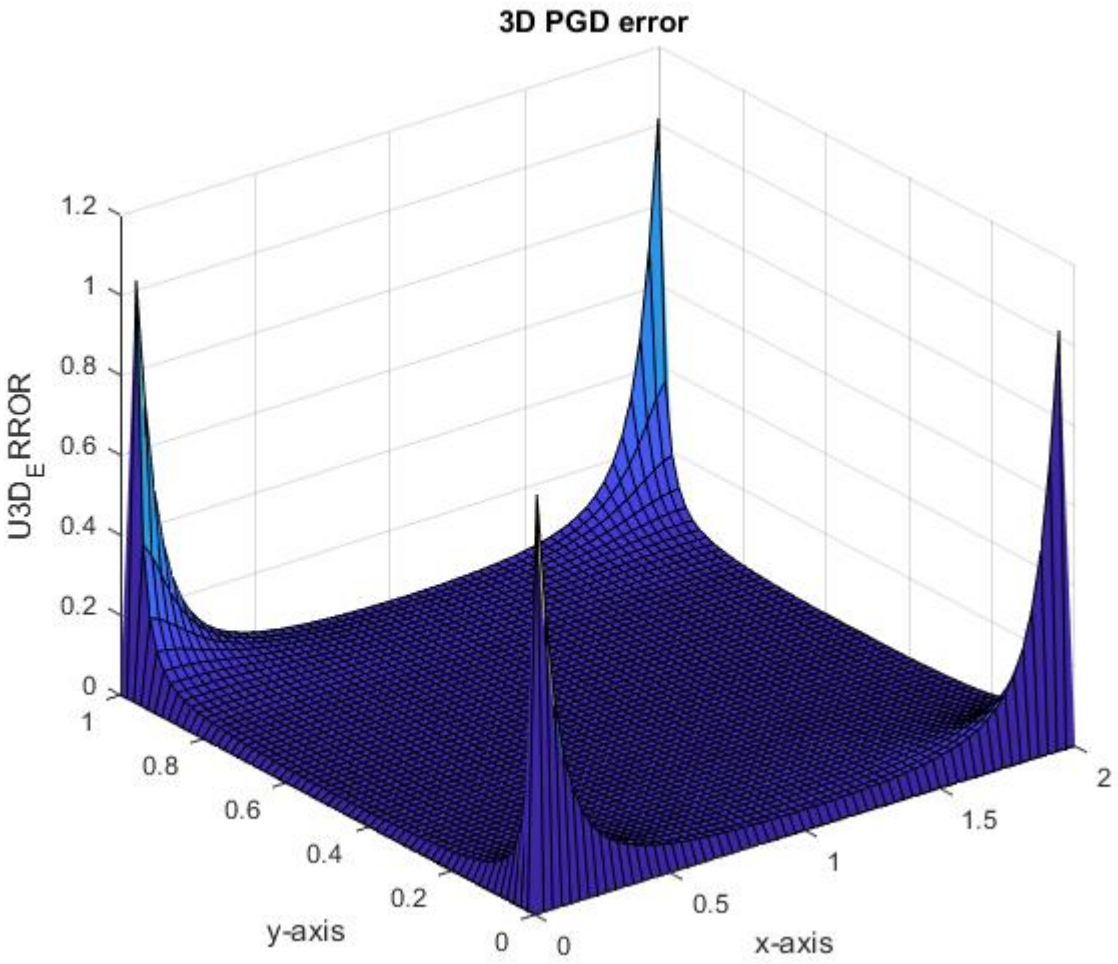


Figure 11: 3D-Error Evaluation

2.4 Exercise 4

In this exercise we were asked to modify the code while changing the source term, so the following formula must be checked:

$$\nabla^2 u = -(x^2 - y^2) \quad \text{in }]-1,1[^2 \tag{6}$$

$$u = 0 \quad \text{on boundaries} \tag{7}$$

After that we re-run the code after modifying the terms which depend on f_1 and f_2. Getting figure:

```
f1=-[x(:).^2 -ones(Nx,1)] ;
f2=[y(:).^2 ones(Ny,1)];
```

Figure 12: New Source Term Implementation

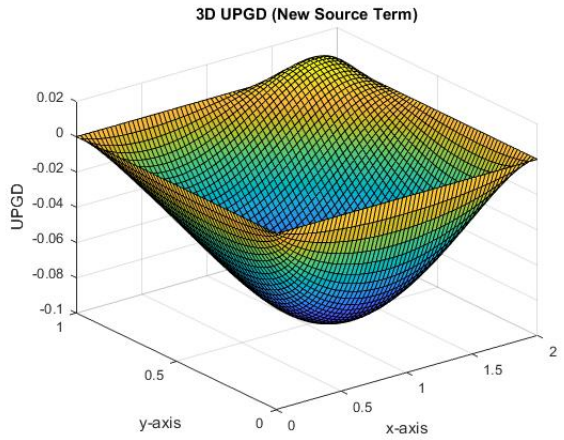


Figure 13: 3D new Source PGD solution

3 Part B

3.1 Exercise 1

In this exercise we are assigned to make use of the general function called easy PGD function which allow us to apply it using PGD method to any class of problem with appropriate tensor structure. This easy PGD function is tested on the known problem which was done in Exercise 1 of Lab-3 and the results obtained are thus compared.

3.1.1 Problem Setup

$$\nabla^2 u = -1 \quad \text{in }]0,2[\times]0,1[\tag{8}$$

$$u = 0 \quad \text{on boundaries} \tag{9}$$

3.1.2 EASY PGD Formulation

$$Au = b \tag{10}$$

$$\text{with} \quad A = \sum_{K=1}^{N_A} A_1^K \times ..... \times A_D^K \quad \text{and} \quad b = \sum_{K=1}^{N_b} b_1^K \times ..... \times b_D^K \tag{11}$$

3.1.3 Result Comparison

In this part, we are going two compare between the solution which were computed in lab4-a and lab4-b; the solution in 4a was calculated using the PGD method while also applying the finite difference method while in 4b, we calculated the numerical solution by assembling the finite element matrices.

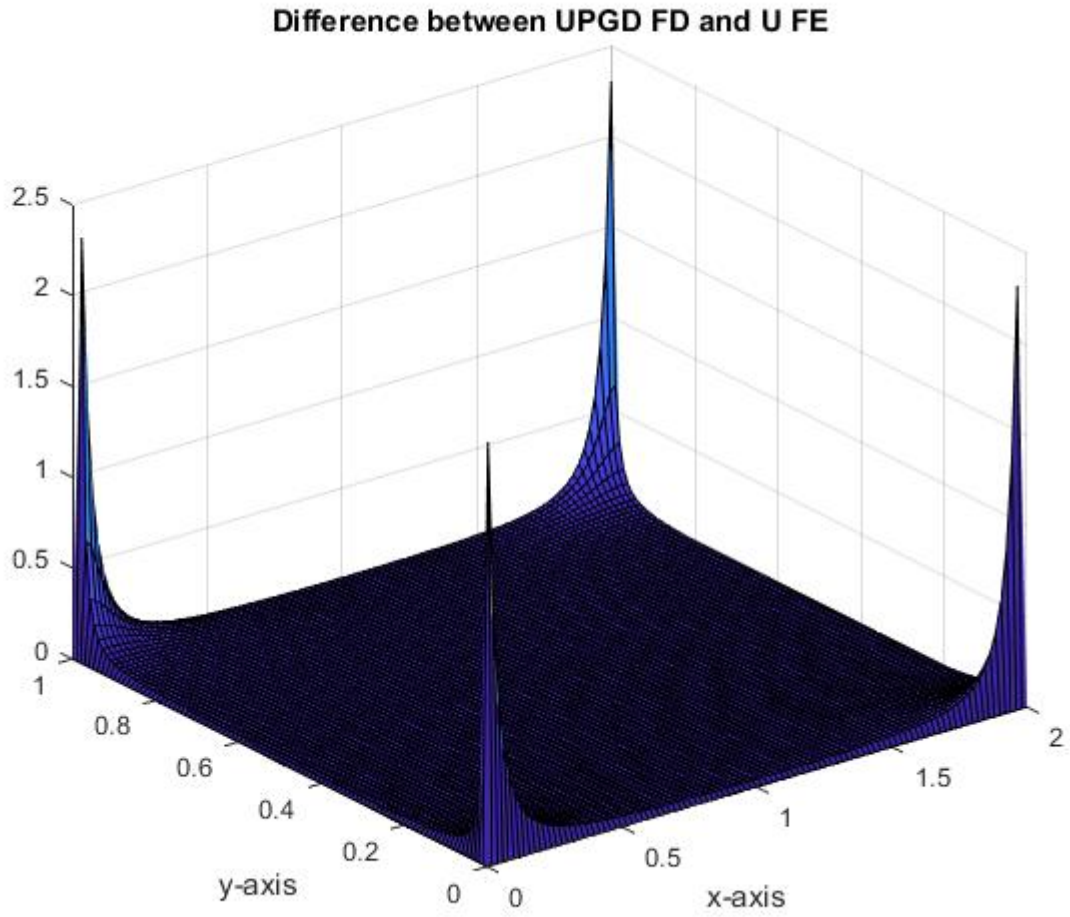


Figure 14: Difference between the UPGD-FD and UFE

So in the figure14, we plot a 3D difference in percentage between the solution taken in first part and the solution obtained in second part.We can notice the maximum percentage reaches 2.5% which should be considered slightly low. It is also noticed that maximum difference is occured on the borders of the 2D-domain.

3.2 Exercise 2

In this exercise the easyPGD code is further modified in regard to implent alternating directions stagnation criterion and this modified code is further tested on the problem mention above for the first two mode and the results obtained for convergence are thus presented. Following are the stagnation criteria to implement alternating directions algorithm.

$$||(u_1 \times ..... \times u_D)_{current} - (u_1 \times ..... \times u_D)_{previous}||_{L^2} < \epsilon_1 \tag{12}$$

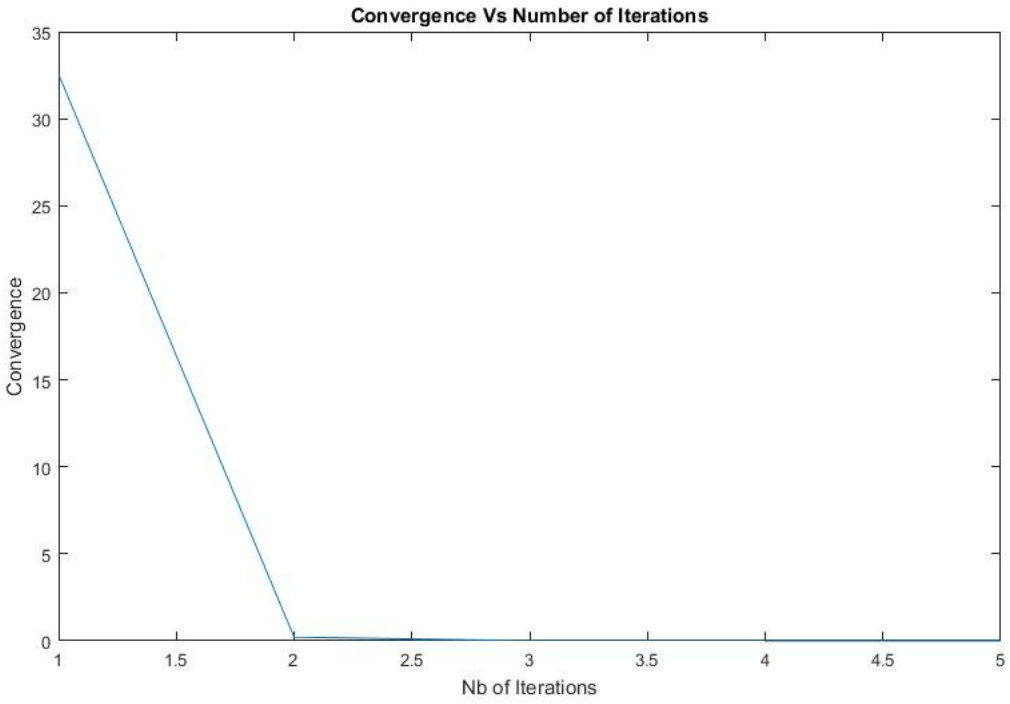


Figure 15: Convergence Vs Number of Iterations

In the figure 15, the plot shows that the convergence is decreasing while the number of iterations is increasing till it reaches approximately near zero.

3.3 Exercise 3

In this exercise the code is further modified in regard with the convergence criterion, this time using global convergence criterion which is based on residual norm defined as:

$$\frac{||b - Au||_{L^2}}{||b||_{L^2}} < \epsilon_2 \tag{13}$$

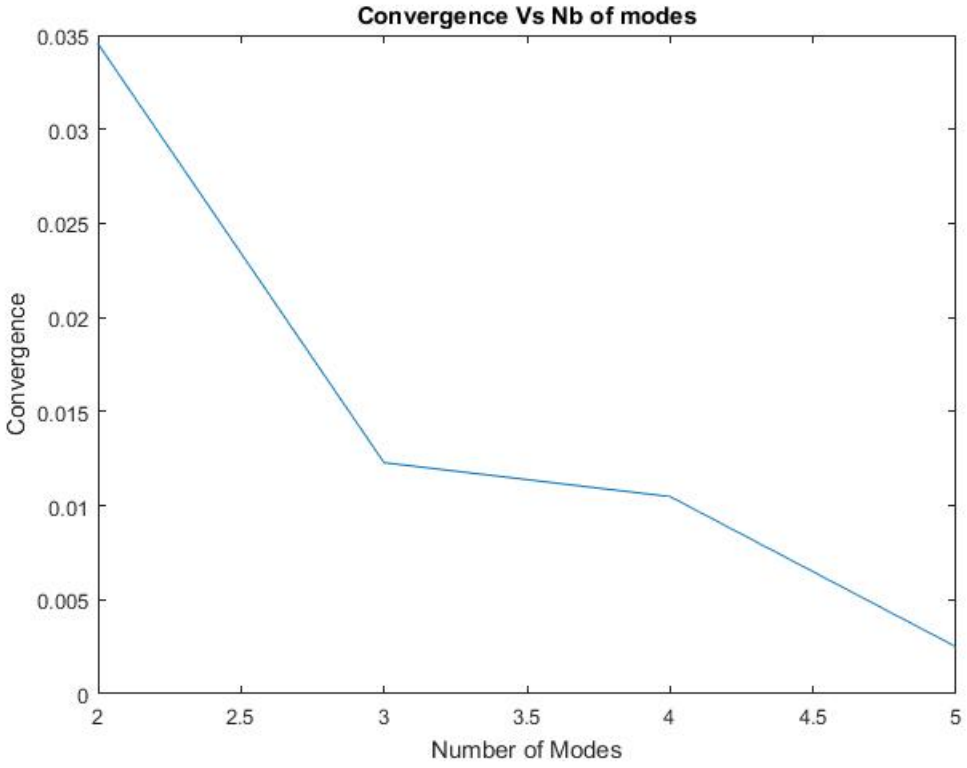


Figure 16: Convergence Vs Number of Modes

As we can notice from the figure above, the convergence is decreasing while increasing the number of modes which should be considered as logic interpolation, due to the fact that the more you add in modes the more you get near to the exact solution.

MSc Computational Mechanics

MODEL REDUCTION

PGD Separated Representation of Vector Field Problems
LAB 5

Mohammad Shahid

Supervisor:
Jose Vicente Aguado

Contents

1 Introduction

1.1 Introduction

In this report we are introducing a technique of handling a vector field problems, which operates by constructing a separated representation of each one of the vector field components. Although both the scalar as well as the vector class of problems are conceptually easy to analyze while the implementation of vector field problems are quite challenging. Nevertheless, we have successfully implemented this technique on a simple problem of considering a two dimensional displacement field as a class of elasticity problem by using the general PGD code.

In the exercise 1 of this report to follow, at first we have identified the left hand side of the terms which appears during the weak formulation of the plane stress elasticity problems thus enable us to find the number of rows and column of cell array AA.In the exercise 2 we have computed the stiffness matrix AA. Then in exercise 3 we have imposed boundary conditions on bottom edge (vertical component of displacement is zero) and left edge (horizontal component of displacement is zero) and also imposed a constant body force. In exercise 4, following the exercise 3, a numerical simulation is performed using easy PGD code at node size of 21X21 for the following fixed values $E = 10^4$, $v = 0.3$, Number of mode equal to 20 and Number of iterations equal to 10 and the results obtained for a displacement field are thus plotted. The plot for deformed shape using a scale factor of $5X10^3$ is also compared and found to be in a good agreement with those given in figures 2a,4a. For a given condition as mention in exercise 4 once again a simulation is carried out for exercise 5 for varying body force and the results obtained for displacement fields are thus plotted.

2 Exercise1

The weak form of the plane-stress elasticity problem in a body is as follows:

$$\int_\Omega \epsilon^{*t}\, D\, \epsilon\, d\Omega = \int_\Omega u^{*t} b\, d\Omega \tag{1}$$

where

$$\epsilon = \begin{bmatrix} \epsilon_x \\ \epsilon_y \\ \gamma_{xy} \end{bmatrix} \quad \epsilon_x = \frac{\partial u}{\partial x} \quad \epsilon_y = \frac{\partial v}{\partial y} \quad \gamma_{xy} = \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}$$

$$D = \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix} \quad b = \begin{bmatrix} b_x \\ b_y \end{bmatrix}$$

where b is the body force , E is the Young's modulus, ν is the poisson coefficient.

$$\int_\Omega \epsilon^{*t}\, D\, \epsilon\, d\Omega = \begin{bmatrix} \epsilon_x & \epsilon_y & \gamma_{xy} \end{bmatrix} * \frac{E}{1-\nu^2} \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix} * \begin{bmatrix} \epsilon_x \\ \epsilon_y \\ \gamma_{xy} \end{bmatrix}$$

$$\int_\Omega \epsilon^{*t}\, D\, \epsilon\, d\Omega = \frac{E}{1-\nu^2} * \int_\Omega \left[\epsilon_x^*\epsilon_x + \nu\epsilon_x^*\epsilon_y + \nu\epsilon_y^*\epsilon_x + \epsilon_y^*\epsilon_y + \tfrac{1-\nu}{2}\gamma_{xy}^*\gamma_{xy}\right] d\Omega$$

$$= \int_\Omega \left[\frac{\partial u^*}{\partial x}\frac{\partial u}{\partial x} + \nu\frac{\partial u^*}{\partial x}\frac{\partial v}{\partial y} + \nu\frac{\partial v^*}{\partial y}\frac{\partial u}{\partial x} + \frac{\partial v^*}{\partial y}\frac{\partial v}{\partial y} + (\tfrac{1-\nu}{2})\left(\frac{\partial u^*}{\partial y}\frac{\partial u}{\partial y} + \nu\frac{\partial u^*}{\partial y}\frac{\partial v}{\partial x} + \nu\frac{\partial v^*}{\partial x}\frac{\partial u}{\partial y} + \frac{\partial v^*}{\partial x}\frac{\partial v}{\partial x}\right)\right] d\Omega \quad (2)$$

Hence,the matrix AA has 2 rows and 8 columns. Two rows indicate the dimensions of the space and each column represents each term in the above expansion.

First term : $\frac{\partial u^*}{\partial x}\frac{\partial u}{\partial x}$

$$\begin{aligned}\frac{\partial u^*}{\partial x}\frac{\partial u}{\partial x} &= (X_u^t K_x X_u) \otimes (Y_u^t M_y Y_u) \\ &= \begin{bmatrix} X_u^t & X_v^t \end{bmatrix} \begin{bmatrix} K_x & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} X_u^t \\ X_v^t \end{bmatrix} \otimes \begin{bmatrix} Y_u^t & Y_v^t \end{bmatrix} \begin{bmatrix} M_y & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} Y_u^t \\ Y_v^t \end{bmatrix}\end{aligned}$$

where $K_x = \int \frac{\partial N_x}{\partial x}\frac{\partial N_x}{\partial x}dx$, $M_y = \int N_y * N_y dy$

Second term : $\frac{\partial u^*}{\partial x}\frac{\partial v}{\partial y}$

$$\begin{aligned}\frac{\partial u^*}{\partial x}\frac{\partial v}{\partial y} &= (X_u^t C_x X_u) \otimes (Y_v^t C_y Y_v) \\ &= \begin{bmatrix} X_u^t & X_v^t \end{bmatrix} \begin{bmatrix} 0 & C_x \\ 0 & 0 \end{bmatrix} \begin{bmatrix} X_u^t \\ X_v^t \end{bmatrix} \otimes \begin{bmatrix} Y_u^t & Y_v^t \end{bmatrix} \begin{bmatrix} 0 & C_y \\ 0 & 0 \end{bmatrix} \begin{bmatrix} Y_u^t \\ Y_v^t \end{bmatrix}\end{aligned}$$

where $C_x = \int \frac{\partial N_x}{\partial x} N dx$

Third term : $\frac{\partial v^*}{\partial y}\frac{\partial u}{\partial x}$

$$\begin{aligned}\frac{\partial v^*}{\partial y}\frac{\partial u}{\partial x} &= (X_v^t C_x X_v) \otimes (Y_u^t C_y Y_u) \\ &= \begin{bmatrix} X_u^t & X_v^t \end{bmatrix} \begin{bmatrix} 0 & 0 \\ C_x & 0 \end{bmatrix} \begin{bmatrix} X_u^t \\ X_v^t \end{bmatrix} \otimes \begin{bmatrix} Y_u^t & Y_v^t \end{bmatrix} \begin{bmatrix} 0 & 0 \\ C_y & 0 \end{bmatrix} \begin{bmatrix} Y_u^t \\ Y_v^t \end{bmatrix}\end{aligned}$$

Fourth term : $\frac{\partial v^*}{\partial y}\frac{\partial v}{\partial y}$

$$\begin{aligned}\frac{\partial v^*}{\partial y}\frac{\partial v}{\partial y} &= (X_v^t M_x X_v) \otimes (Y_v^t K_y Y_v) \\ &= \begin{bmatrix} X_u^t & X_v^t \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & K_x \end{bmatrix} \begin{bmatrix} X_u^t \\ X_v^t \end{bmatrix} \otimes \begin{bmatrix} Y_u^t & Y_v^t \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & K_y \end{bmatrix} \begin{bmatrix} Y_u^t \\ Y_v^t \end{bmatrix}\end{aligned}$$

Fifth term : $\frac{\partial u^*}{\partial y}\frac{\partial u}{\partial y}$

$$\begin{aligned}\frac{\partial u^*}{\partial y}\frac{\partial u}{\partial y} &= (X_u^t M_x X_u) \otimes (Y_u^t K_y Y_u) \\ &= \begin{bmatrix} X_u^t & X_v^t \end{bmatrix} \begin{bmatrix} M_x & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} X_u^t \\ X_v^t \end{bmatrix} \otimes \begin{bmatrix} Y_u^t & Y_v^t \end{bmatrix} \begin{bmatrix} K_y & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} Y_u^t \\ Y_v^t \end{bmatrix}\end{aligned}$$

Sixth term : $\frac{\partial u^*}{\partial y}\frac{\partial v}{\partial x}$

$$\begin{aligned}\frac{\partial u^*}{\partial y}\frac{\partial v}{\partial x} &= (X_u^t C_x X_u) \otimes (Y_v^t C_y Y_v) \\ &= \begin{bmatrix} X_u^t & X_v^t \end{bmatrix} \begin{bmatrix} 0 & C_x \\ 0 & 0 \end{bmatrix} \begin{bmatrix} X_u^t \\ X_v^t \end{bmatrix} \otimes \begin{bmatrix} Y_u^t & Y_v^t \end{bmatrix} \begin{bmatrix} 0 & C_y \\ 0 & 0 \end{bmatrix} \begin{bmatrix} Y_u^t \\ Y_v^t \end{bmatrix}\end{aligned}$$

Seventh term : $\frac{\partial v^*}{\partial x}\frac{\partial u}{\partial y}$

$$\begin{aligned}\frac{\partial v^*}{\partial x}\frac{\partial u}{\partial y} &= (X_v^t C_x X_v) \otimes (Y_u^t C_y Y_u) \\ &= \begin{bmatrix} X_u^t & X_v^t \end{bmatrix} \begin{bmatrix} 0 & 0 \\ C_x & 0 \end{bmatrix} \begin{bmatrix} X_u^t \\ X_v^t \end{bmatrix} \otimes \begin{bmatrix} Y_u^t & Y_v^t \end{bmatrix} \begin{bmatrix} 0 & 0 \\ C_y & 0 \end{bmatrix} \begin{bmatrix} Y_u^t \\ Y_v^t \end{bmatrix}\end{aligned}$$

Eighth term : $\frac{\partial v^*}{\partial x}\frac{\partial v}{\partial x}$

$$\begin{aligned}\frac{\partial v^*}{\partial x}\frac{\partial v}{\partial x} &= (X_v^t K_x X_v) \otimes (Y_v^t M_y Y_v) \\ &= \begin{bmatrix} X_u^t & X_v^t \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & K_x \end{bmatrix} \begin{bmatrix} X_u^t \\ X_v^t \end{bmatrix} \otimes \begin{bmatrix} Y_u^t & Y_v^t \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & M_y \end{bmatrix} \begin{bmatrix} Y_u^t \\ Y_v^t \end{bmatrix}\end{aligned}$$

3 Exercise2

$$AA \begin{bmatrix} u(x,y) \\ v(x,y) \end{bmatrix} = BB$$

The entries of the stiffness matrix,AA are defined based on the above theory in the program as follows:

3.1 Stiffness Matrix: AA

```
% Constants
E = 10^4;
v = 0.3;
C1 = E/(1-v^2);
C2 = (1-v)/2;
%Adding AA, BB, GG, Nmodes, Niter, Boards
AA = cell(2,8);
% First Term
AA{1,1} = [C1*FEM_mat_1D(x,1,1) zeros(Ny);zeros(Ny) zeros(Ny)];
AA{2,1} = [FEM_mat_1D(y,0,0) zeros(Ny);zeros(Ny) zeros(Ny)];
% Second Term
AA{1,2} = [zeros(Nx) v*C1*FEM_mat_1D(x,1,0);zeros(Nx) zeros(Nx)];
AA{2,2} = [zeros(Ny) FEM_mat_1D(y,0,1);zeros(Ny) zeros(Ny)];
%Third Term
AA{1,3} = [zeros(Nx) zeros(Nx);v*C1*FEM_mat_1D(x,0,1) zeros(Nx)];
AA{2,3} = [zeros(Ny) zeros(Ny);FEM_mat_1D(y,1,0) zeros(Ny)];
%Fourth Term
AA{1,4} = [zeros(Nx) zeros(Nx);zeros(Nx) C1*FEM_mat_1D(x,0,0)];
AA{2,4} = [zeros(Ny) zeros(Ny);zeros(Ny) FEM_mat_1D(y,1,1)];
%Fifth Term
AA{1,5} = [C2*C1*FEM_mat_1D(x,0,0) zeros(Nx);zeros(Nx) zeros(Nx)];
AA{2,5} = [FEM_mat_1D(y,1,1) zeros(Ny);zeros(Ny) zeros(Ny)];
%Sixth Term
AA{1,6} = [zeros(Nx) C2*C1*FEM_mat_1D(x,0,1);zeros(Nx) zeros(Nx)];
AA{2,6} = [zeros(Ny) FEM_mat_1D(y,1,0);zeros(Ny) zeros(Ny)];
%Seventh Term
AA{1,7} = [zeros(Nx) zeros(Nx);C2*C1*FEM_mat_1D(x,1,0) zeros(Nx)];
AA{2,7} = [zeros(Ny) zeros(Ny);FEM_mat_1D(y,0,1) zeros(Ny)];
%Eigth Term
AA{1,8} = [zeros(Nx) zeros(Nx);zeros(Nx) C2*C1*FEM_mat_1D(x,1,1)];
AA{2,8} = [zeros(Ny) zeros(Ny);zeros(Ny) FEM_mat_1D(y,0,0)];
```

4 Exercise3

4.1 Boundary Conditions (Bords) and Body Force (BB)

As per the given problem symmetry is considered on the square domain by applying the boundary conditions on both bottom and left edges.Therefore in the bottom edge vertical component of the displacement is considered zero and in the left edge horizontal component of displacement is considered zero. Since the two dimensional system is reduced to system of two 1 dimensional equations. The boundary condition on the bottom edge with vertical component zero can be expressed applied at one nodal point. Similarly the left edge horizontal component zero can be expressed at one nodal point.Hence the following boundary conditions are taken.

```
%Bords : Boundary Conditions
Bords{1}=1;
Bords{2}=Ny+1;

%Body Force
f11 = FEM_mat_1D(x,0,0)*ones(Nx,1);
% Can also be expressed as follows
% f11 = (dx/2)*[1 ;(ones(Nx-2,1)+1);1]
f21 = FEM_mat_1D(y,0,0)*ones(Ny,1);
f12 = FEM_mat_1D(x,0,0)*ones(Nx,1);
f22 = FEM_mat_1D(y,0,0)*ones(Ny,1);

BB{1,1} = [f11 zeros(Nx,1);zeros(Nx,1) f12];
BB{2,1} = [f21 zeros(Ny,1);zeros(Ny,1) f22];
```

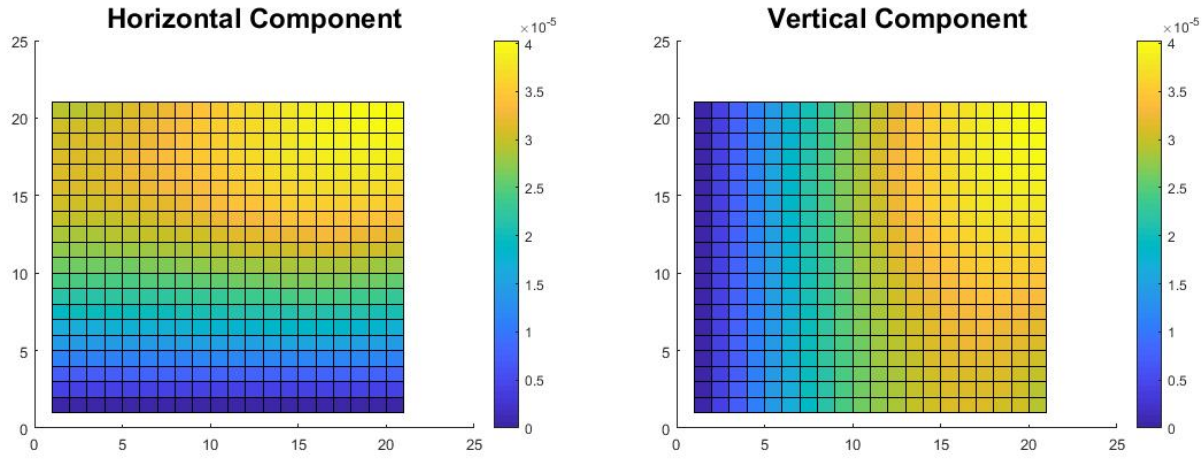
$$(Bodyforce) \quad BB = \int_{\Omega} u^{*t} b \, d\Omega = \begin{pmatrix} \frac{dx}{2} \begin{pmatrix} 1 \\ 2 \\ . \\ . \\ 2 \\ 1 \end{pmatrix} & \begin{pmatrix} 0 \\ 0 \\ . \\ . \\ 0 \\ 0 \end{pmatrix} \\ \begin{pmatrix} 0 \\ 0 \\ . \\ . \\ 0 \\ 0 \end{pmatrix} & \frac{dy}{2} \begin{pmatrix} 1 \\ 2 \\ . \\ . \\ 2 \\ 1 \end{pmatrix} \end{pmatrix}$$

where $b_x = 1; b_y = 1$

5 Exercise4

The horizontal and vertical component of the displacement are plotted in the figures 1a, 1b.

5.1 Horizontal and Vertical Components of Displacement field

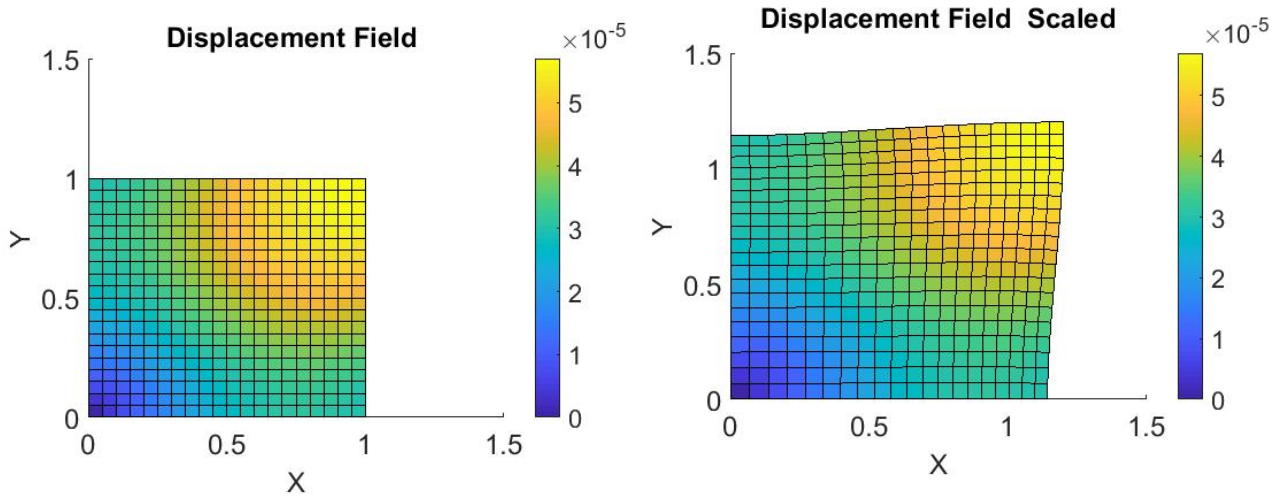


(a) Horizontal Component of the Displacement (b) Vertical Component of the Displacement

Figure 1: Displacement Component Fields

5.2 Magnitude of Displacement field: Deformed and Undeformed Shape

The magnitude of the displacement field are plotted in figure 2a and the magnitude of displacement field on the deformed shape with a scale factor of $5 * 10^3$ is plotted in figure 2b. The maximum displacement value obtained is equal to $5.69 * 10^{-}5$.



(a) Magnitude of Displacement Field

(b) Magnitude of Displacement Field on Deformed Shape:Scale factor of 5000

Figure 2: Magnitude of Displacement Field

6 Exercise5

6.1 Boundary Conditions (Bords) and Non linear Body Force (BB)

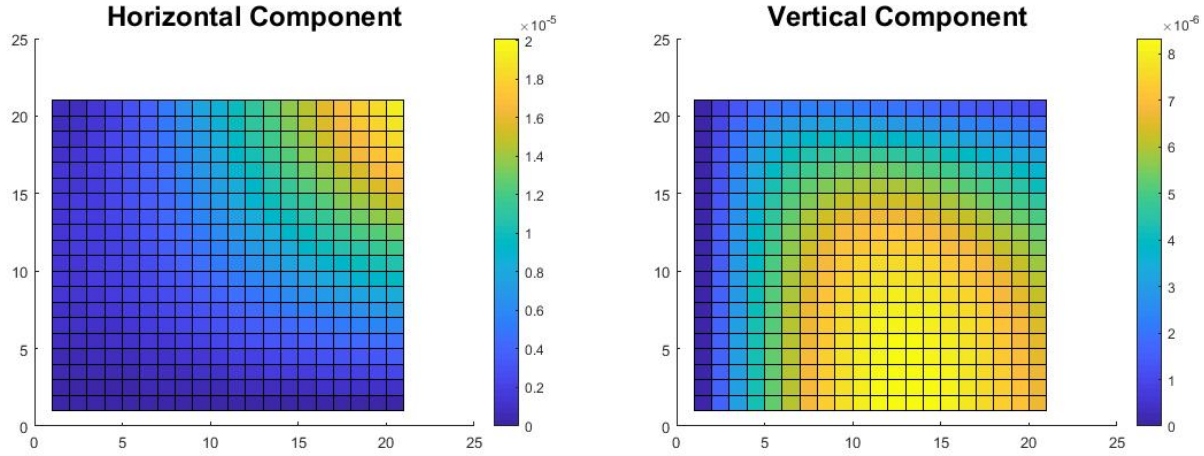
The above code is modified to compute the system with the non-linear body force, BB
$b_x = x^2 y \quad ; b_y = (y-1)^2$.

```
%Bords : Boundary Conditions
Bords{1}=1;
Bords{2}=Ny+1;

%Body Force
f11 = FEM_mat_1D(x,0,0)*(x.^2)';
% Can also be expressed as follows
% f11 = (dx/2)*[1 ;(ones(Nx-2,1)+1);1].*(x.^2)'
f21 = FEM_mat_1D(y,0,0)*y';
f12 = FEM_mat_1D(x,0,0)*ones(Nx,1);
f22 = FEM_mat_1D(y,0,0)*((y'-ones(Ny,1)).^2);

BB{1,1} = [f11 zeros(Nx,1);zeros(Nx,1) f12];
BB{2,1} = [f21 zeros(Ny,1);zeros(Ny,1) f22];
```

$$(Bodyforce)\ BB = \int_\Omega u^{*t} b \, d\Omega = \begin{pmatrix} \frac{dx}{2} \begin{pmatrix} 1 \\ 2 \\ . \\ . \\ 2 \\ 1 \end{pmatrix} x^2 & \begin{pmatrix} 0 \\ 0 \\ . \\ . \\ 0 \\ 0 \end{pmatrix} y \\ \begin{pmatrix} 0 \\ 0 \\ . \\ . \\ 0 \\ 0 \end{pmatrix} & \frac{dy}{2} \begin{pmatrix} 1 \\ 2 \\ . \\ . \\ 2 \\ 1 \end{pmatrix} (y-1)^2 \end{pmatrix}$$

The horizontal and vertical component of the displacement are plotted in the figures 3a, 3b.

6.2 Horizontal and Vertical Components of Displacement field

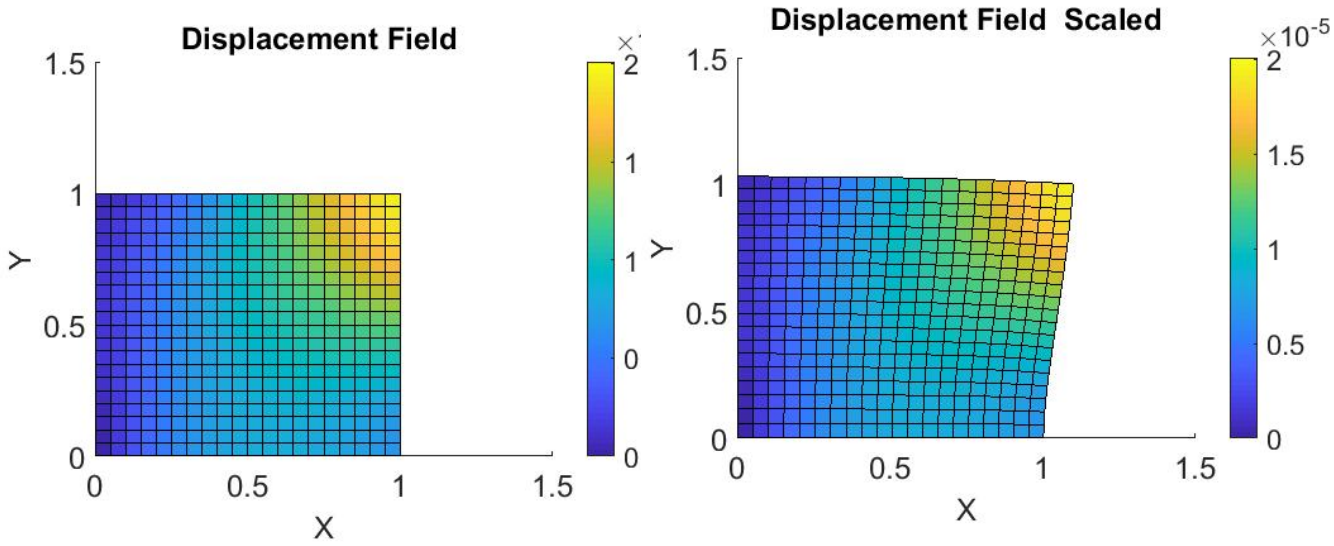


(a) Horizontal Component of the Displacement (b) Vertical Component of the Displacement

Figure 3: Displacement Component Fields

6.3 Magnitude of Displacement field: Deformed and Undeformed Shape

The magnitude of the displacement field are plotted in figure 4a and the magnitude of displacement field on the deformed shape with a scale factor of $5 * 10^3$ is plotted in figure 4b.The maximum displacement value obtained is equal to $2.01 * 10^{-}5$.



(a) Magnitude of Displacement Field

(b) Magnitude of Displacement Field on Deformed Shape:Scale factor of 5000

Figure 4: Magnitude of Displacement Field