Universität Stuttgart

A PGD-BASED PRECONDITIONER FOR SCALER ELLIPTIC PROBLEMS

UPC

Master's Thesis

Submitted by
Hasini Garikapati

August 29 , 2014

Master's thesis at:	Laboratori de Calcul Numeric (LaCan), Department of Applied Mathematics III (MA3), Universitat Politècnica de Catalunya, Barcelona
Supervisors:	Dr. Marco Discacciati, Dr .Antonio Heurta (UPC, BarcelonaTech)
Examiner:	Dr. Bernard Haasdonk Institute of Applied Analysis and Numerical Simulation, University of Stuttgart

I hereby certify that I have prepared this thesis independently, and that only those sources, aids and advisors that are duly noted herein have been used and/or consulted.

Barcelona, August 29

Hasini Garikapati

Acknowledgements

I extend my gratitude to Dr. Antonio Heurta for providing me the opportunity to work at LaCan and being my supervisor. I express profound sense of gratitude to Dr. Marco Discacciati for being my supervisor. I thank him for providing me indispensable assistance throughout the course of my thesis work. With his support, the project was approached in a systematic manner which lead to better understanding of the work. I thank him for his excellent guidance, clarity of vision, and soothing presence in moments of panic. Also, I am highly indebted to Dr. Bernard Haasdonk for being my examiner and also giving me his critical comments at every stage of work.

Abstract

The Proper Generalized Decomposition Method (PGD) is a powerful model reduction technique, based on separated representations. In particular, the solution is sought as a finite sum of terms, each one involving the product of functions of each coordinate. The solution is then calculated by means of a sequence of one dimensional problems. Thus, the PGD is able to circumvent the curse of dimensionality and makes possible the efficient solution of models defined in multidimensional spaces.

The solution of large linear systems of the form $Au = b$ where A = [a_{ij}] is an n x n matrix and b is a right hand side vector, is central to many numerical simulations in science and engineering and is often most time-consuming and expensive part of a computation. Although direct methods are robust , they scale poorly with problem size in terms of operation counts and memory requirements. While iterative methods require fewer storage and often require fewer operations than direct methods (especially when an approximate solution of relatively low accuracy is sought), they do not have the reliability of direct methods. In some applications, iterative methods often fail and preconditioning is necessary, though not always sufficient, to attain convergence in a reasonable amount of time.

In the present thesis, scalar elliptic boundary value problems are considered. The problem is discretized by employing Galerkin Finite Elements, which generate the linear system of equations. The linear system is solved using an iterative method and a suitable preconditioner is characterized to improve the convergence properties of the method. In the present thesis, the PGD method is used to characterize the preconditoner. The algebraic residual is computed, from which the residual finite elements function is defined. For this, the PGD procedure is applied to compute the solution of the ¨Preconditioning Problem¨. This PGD preconditioner is implemened to various ranges of cases from a simple case (constant source term) to complex, such as non constant source, non constant diffusion terms .

Nomenclature

A	Matrix
b	Vector
C_n	Preconditioner
N_j	Shape Functions
N	Number of enrichment steps
w	Weight Function
K	Diffusion Coefficient
$X_n(x)$	Basis function of *x* of the nth enrichment step
$Y_n(y)$	Basis function of *y* of the nth enrichment step
M_x	1-Dimensional Mass Matrix in *x* direction
M_y	1-Dimensional Mass Matrix in *y* direction
N_{max}	Maximum Basis functions used
$F_n(x)$	Normalized basis function of *x* of the nth enrichment step
$G_n(y)$	Normalized basis function of *y* of the nth enrichment step

Greek Letters

Ω	Domain
Ω_e	Elemental Domain
ϵ	Tolerance used for iterative steps
ϵ_n	Tolerance used for enrichment step

λ	Coefficient of Normalization
α	Coefficient defined in the ODE
β	Coefficient defined in the ODE
γ	Coefficient defined in the ODE
δ	Coefficient defined in the ODE

Acronyms

FEM	Finite Element Method
PGD	Proper Generalized Decomposition
CG	Conjugate Gradient
PC	Preconditioner
PCG	Preconditioned Conjugate Gradient
ODE	Ordinary Differential Equation
PDE	Partial Differential Equation

Subscripts & Superscripts

x	Functions or coefficients that are dependent on *x*
y	Functions or coefficients that are dependent on *y*
n	Enrichment step
ij	I th coordinate/node in x direction and jth coordinate/node in y direction
p	Represents the iterative step
max	Maximum value

Table of Contents

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Introduction

This chapter describes the motivation of the thesis. The basic idea of the thesis is presented. Further, it has the outline of the content of all the chapters of this report.

1.1 Motivation

The need to solve large linear system of equations are generated in most of the scientific problems where mathematical models are used. The systems are generated from discretization of differential equations, optimization problems, etc. The solution of large linear systems of the form $Ax=b$ where A = [a_{ij}] is a n x n matrix and b is a right hand side vector, is central to many numerical simulations in science and engineering and is often most time-consuming and expensive part of a computations.

When the discretization of the original problem is done using a method such as Finite Elements, Finite Differences, Finite Volumes, etc., the matrix that is generated from the system of equations is large and sparse. There are direct methods such as Gaussian Elimination. However, when one wants to solve very large systems of equations, the computational complexity increases the size of the problem. Iterative methods for solving general, large sparse linear systems have been gaining popularity in many areas of scientific computing. This is because one can take advantage of "sparsity" to design iterative methods that can be quite economical and faster than the direct solution methods.

In some applications, iterative methods often fail and preconditioning is necessary, though not always sufficient, to attain convergence in a reasonable number of iterations. The general idea underlying any preconditioning procedure for an iterative solvers is to convert the following system $Ax=b$ in such a way that an equivalent system [5]

$$\hat{A}\hat{x}=\hat{b} \tag{1.1}$$

for which the iterative method converges faster. A standard approach is to use a non singular matrix C_n and rewrite the system as

$$C_n^{-1} Ax = C_n^{-1} b \tag{1.2}$$

The preconditioned C_n needs to be chosen in such that $C_n^{-1} A = \hat{A}$ where $\hat{A}$ is better conditioned and ideally $\hat{A}$ is the identity matrix. A iterative method computes successive approximate of the solution { $x^0, x^1, \ldots, x^n$ } at respective steps. In practice, the iterative process is stopped when $\|x^n - x\| < \epsilon$ where ϵ is a fixed tolerance and $\|.\|$ is any convenient vector norm. However, since the exact solution is obviously not available, it is necessary to introduce suitable stopping criteria to monitor the convergence of the iteration according to the problem (Discussed in section 3.3)

Iterative schemes of the form

$$x^{n+1} = x^n - C_n^{-1}(Ax^n - b) \tag{1.3}$$

are common where C_n denotes a suitable preconditioner which may change at each iteration to enhance the convergence properties.

The Proper Generalized Decomposition (PGD) is is a powerful model reduction technique, based on separated representations. The PGD builds on successive enrichment strategy, a numerical approximation of the unknown fields in a separated form involving a *priori* unknown function. The computational complexity of PGD scales linearly with the dimension in space wherein the model is defined, which is in contrast with the conventional methods wherein the complexity is scaled exponentially. In particular, the solution is sought as a finite sum of terms, each one involving the product of functions of each coordinate. For instance, the material parameters and boundary conditions appearing in a particular mathematical model can be regarded as extra-coordinates of the problem in addition to the usual coordinates such as space and time. The solution is then calculated by means of a sequence of one dimensional problems. Thus, the PGD is able to circumvent the curse of dimensionality and makes possible the efficient solution of models defined in multidimensional spaces. [2]

In the present thesis, scaler elliptic boundary value problems are considered. The problem is discretized by employing Galerkin Finite Elements, which generate the linear system of equations. The linear system is then solved using an iterative method and the preconditioner is characterized using the PGD method in order to improve the convergence properties. The main idea behind the thesis is – First , to study and implement the PGD method. Second, to use it to characterize the preconditioner for the standard iterative methods to solve the linear system of equations

1.2 Contents of Thesis

The present thesis involves implementation of the PGD-based preconditioner for scaler elliptic problems. Chapter 2 presents the outline of the mathematical equations of the finite element method that lead to the linear system of equations. In the chapter 3, the idea and the implementation of the PGD, and the convergence criteria that is considered for the PGD method are discussed. In the chapter 4 , the way the PGD based preconditioner is coupled with the scaler elliptic problems (with the diffusion term) is discussed. Chapter 5 presents the results and analysis of the present technique to solve the linear system of equations is described and compared with the convergence properties of the finite element method and conjugate gradient method without preconditioner. Finally, in the last chapter, a brief description of conclusions and scope for future research works are discussed.

Chapter 2

Basics of Finite Element Method

In chapter 2, the basics of Finite Element Method that are used in the implementation of the PGD method and later the two-dimensional FEM which is used for solving the elliptic equations pertaining to the present thesis are briefly described.

2.1 Galerkin Finite Element Method

The Finite Element Method (FEM) has emerged as one of the most powerful Numerical methods to find approximate solutions to the boundary value problems for the differential equations. Such a method uses a spatial discretization and a weighted residual formulation to transform the governing PDE (strong form) into an integral equation (weak form) that upon variational treatment yields to the solution of a system of matrix equations. One of the most successful in application of the Standard FEM formulation are based upon the Galerkin formulation of the method of weighted residuals. The reason for this success is that, when applied to problems governed by self-adjoint elliptic or parabolic partial differential equations, the Galerkin finite element method leads to symmetric stiffness matrices. [7]

The following are few of the compact notations used in the report

$$(u,v)=\int_{\Omega} u.v\, d\Omega \qquad (2.1)$$
$$a(u,v)=\int_{\Omega} \nabla u : \nabla v\, d\Omega$$

2.2 Strong and weak form of the Problem

To illustrate the strong and the weak problem, the following Poisson equation is considered.

$$-\Delta u = f \quad \in \Omega \tag{2.2}$$

For the sake of simplicity, only the Dirichlet conditions are considered

$$u = u_D \quad on\,\Gamma \tag{2.3}$$

The strong form of a boundary value problem comprises of the differential equation of the problem along with the boundary conditions. So in the present case the PDE (2.2) along with boundary conditions (2.3) constitutes the strong form.

The first step in a weighted residual formulation leading to the finite element discretization of our model problem consists of formulating a weak (or variational) form of the boundary value problem. This is achieved by multiplying the governing equation (2.2) by the weighting function w and integrating over the computational domain Ω [7]

$$-\int_\Omega w \Delta u \, d\Omega = \int_\Omega wf \, d\Omega \tag{2.4}$$

Applying divergence theorem to the left hand side of the equation

$$\begin{aligned} -\int_\Omega w \Delta u \, d\Omega &= -\int_\Omega (\nabla.(w\nabla u) - \nabla w.\nabla u) \, d\Omega \\ &= \int_\Omega \nabla w.\nabla u \, d\Omega - \int_\Gamma w(n.\nabla u) \, d\Gamma \end{aligned} \tag{2.5}$$

The test function $w=0 \;\; on \;\; \Gamma_D$ **.** It vanishes on the Dirichlet portion of the boundary and taking into account the Neumann boundary condition the following is obtained which is the weak form of the problem

$$\int_\Omega \nabla w.\nabla u \, d\Omega = \int_\Omega wf \, d\Omega \tag{2.6}$$

2.3 Discretization of the Problem

Now that the weak form of the problem is available, the next step is to discretize the weak form by using the Galerkin Finite Element Method. A suitable mesh is used to subdivide the computational domain Ω into element domain Ω_e . In practice, for every element shape functions are defined using a transformation from a reference element.

In one-dimension, Piecewise Lagrange Polynomials are used. For instance, the piecewise linear function in 1-D which are used in the present thesis

$$N_j(x)=\begin{cases}\dfrac{x-x_{j-1}}{x_j-x_{j-1}}, & \text{if } x_{j-1}\leqslant x<x_j\\ \dfrac{x_{j+1}-x}{x_{j+1}-x_j}, & \text{if } x_j\leqslant x<x_{j+1}\\ 0, & \text{otherwise}\end{cases} \tag{2.7}$$

on the mesh $x_0<x_{1.}......<x_N$. And in two-dimensions, meshes generally consist of triangles or quadrilaterals. In the present thesis the quadrilateral elements are used. The following is the normalized reference element of the quadrilateral.

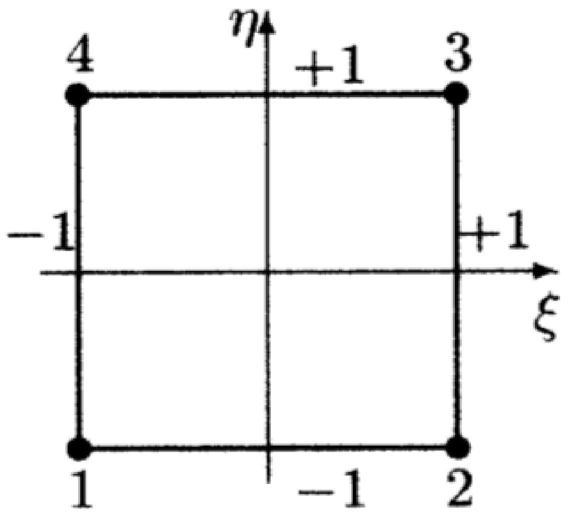


Fig 2.1 normalized reference element

The following are the element shape functions

$$\begin{aligned}N_1&=\frac{1}{4}(1-\xi)(1-\eta)\\ N_2&=\frac{1}{4}(1+\xi)(1-\eta)\\ N_3&=\frac{1}{4}(1+\xi)(1+\eta)\\ N_4&=\frac{1}{4}(1-\xi)(1+\eta)\end{aligned} \tag{2.8}$$

2.4 Finite Element System of Equation

The assembly of the element contributions to the discrete weak form into the complete system results in a matrix equation of the form

$$\hat{K}\,u=\hat{f} \tag{2.9}$$

where $\hat{K}\,,\hat{f}$ represent stiffness matrix , mass matrix and force vector respectively. The components of these critical integrals are defined as follows. (for the 1- dimension system which is used in the thesis)

1) Components of Stiffness Matrix are

$$\hat{K}_{ij}=\int_0^L \left(\frac{dN_i}{dx}\right)\left(\frac{dN_j}{dx}\right)dx \tag{2.10}$$

2) Components of force Matrix are

$$\hat{f}_i=\int_0^L f(x)\,N_i\,dx \tag{2.11}$$

There is one more matrix used in the thesis that is encountered in Finite Elements, is Mass Matrix. It is defined as follows

3) Components of Mass Matrix are

$$\hat{M}_{ij}=\int_0^L N_i N_j\,dx \tag{2.12}$$

These matrix when assembled result in the system of equations which are solved either by direct or iterative methods. In the present thesis, the focus is on solving these system of equations which are generated after discretization of the scaler elliptic problems, by using an iterative method of which the preconditioner is characterized by the Proper Generalized Decomposition (PGD) method. This is discussed in detail in the subsequent chapters.

Chapter 3

Implementation of the PGD

This chapter describes the idea and the implementation of the PGD method. In the present thesis, the PGD method is implemented using alternate direction strategy. The simplest two dimensional elliptic PDE, Poisson equation is considered. First, the simplest case where the source term is constant is developed. Then later, the method is extended to the non-constant source term and with non-constant diffusion term. The results of all these cases are demonstrated in this chapter. Further the convergence criteria which is used in the method is illustrated.

3.1 Separated Representation

The PGD method implemented is based on separated representation of the solution.[3]. The solution is taken as a sum of finite sum of terms, each one involving the product of functions. For instance, the problem of D dimensions, the solution *u* is written as the sum of N finite terms in the following way

$$u(x_{1,\cdot}\dots x_D)=\sum_{j=1}^{j=N}\prod_{k=1}^{k=D}T_{kj}(x_k) \tag{3.1}$$

where T_{kj} is the *j*th basis function, which only depends on the *k*th coordinate.[4]. In the present thesis only 2 Dimensional problems are considered. For instance, consider the solution of the Poisson equation.

$$-\Delta u(x,y)=f(x,y) \tag{3.2}$$

in the two dimensional rectangular domain $\Omega=\Omega_x\times\Omega_y$. The PGD approximate solution is obtained in a separated form

$$u(x,y)=\sum_{i=1}^{N} X_i(x).Y_i(y) \tag{3.3}$$

where N is total number of enrichment steps. Each term of the expansion is computed one at a time, enriching the PGD approximation until a suitable convergence criterion

is reached. The separated form is progressively constructed. At each enrichment step n ($n \geqslant 1$), the first n-1 terms of the PGD approximation is already computed which is in the form $u^{n-1}(x,y)=\sum_{i=1}^{n-1} X_i(x).Y_i(y)$

Now, the next term n, $X_n(x).Y_n(y)$ is computed which would further enrich the PGD solution [2]

$$u^n(x,y)=u^{n-1}(x,y)+X_n(x).Y_n(y)=\sum_{i=1}^{n} X_i(x).Y_i(y) \tag{3.4}$$

The functions X_n, Y_n are obtained at the enrichment step n. To arrive at the functions at each n enrichment step, a suitable iterative scheme is employed. Thus, the index p is used to denote a particular iteration (3.5)

$$u^{n,p}(x,y)=u^{n-1}(x,y)+X_n^p(x).Y_n^p(y) \tag{3.5}$$

Let the functions at each iteration be denoted by $R_n^p(x), S_n^p(y)$ in place of $X_n^p(x), Y_n^p(y)$ respectively. The iterations at each enrichment step proceed until reaching a fixed point tolerance.(3.6)

$$\frac{\|R_n^p(x).S_n^p(y)-R_n^{p-1}.S_n^{p-1}(y)\|}{\|R_n^p(x).S_n^p(y)\|}<\epsilon \tag{3.6}$$

where $\|.\|$ is the L^2-norm and ϵ is the tolerance. a The iterative scheme stops at the point when the condition (3.6) is satisfied. At the end of the iterative step, the last obtained value is assigned to enrichment step.

$$R_n^p(x)\rightarrow X_n(x);\, S_n^p(y)\rightarrow Y_n(y)$$

As the solution is enriched the magnitude of the basis functions obtained is either quite low or high. In order to avoid working with high or low order numbers for the basis functions, the normalized form of the solution is obtained at each step. This is done by introducing a coefficient λ The approximate solution can be restored by multiplying the coefficient with the normalized function. (3.9)

$$u^n(x,y)=\sum_{i=1}^{n}\lambda_i F_i(x)G_i(y) \tag{3.7}$$

where F and G are the normalized functions of x and y respectively and λ_i is the coefficient at the corresponding step. Henceforth, $R_n^p(x), S_n^p(y)$ and $F_n(x), G_n(y)$ notations are used in derivation of the method for different cases.

3.1.1 Normalization of the functions.

As discussed in the previous section, the functions $R_n^p(x), S_n^p(y)$ represent the functions obtained at the end of the iterative step, after which they are normalized and for the further computations the normalized functions are used which are represented by $F_n(x), G_n(y)$ .The way the functions are normalized by following

$$\begin{aligned} F_n(x) &= \frac{R_n^p(x)}{\sqrt{(R_n^p(x) M_x R_n^P(x)^T)}} \\ G_n(y) &= \frac{S_n^p(y)}{\sqrt{(S_n^p(y) M_y S_n^P(y)^T)}} \end{aligned} \tag{3.8}$$

where M_x, M_y represent one-dimensional mass matrices in *x, y* directions respectively. However, in practice, one of the function is assumed to be normalized function. for instance, $S_n^0(y)$ at the beginning of the iterative process is assumed to one in the present case or any other function whose norm is one. in that way, the $S_n^p(y)$ which is obtained at the end of the iterative process is already a normalized function which is $G_n(y)$. While, the function $R_n^p(y)$ is normalized by using (3.9). The coefficient λ is computed in the following way

$$\lambda_n=\sqrt{(R_n^p(x) M_x R_n^{pT})} \tag{3.9}$$

3.2 Alternate Direction Strategy for Constant Source Term

As discussed earlier (3.7), an iterative scheme is employed to get the refined solution at each enrichment step. In the present thesis, the iterative scheme used is Alternate Direction Strategy. Basically, here first $R_n^p(x)$ is computed from $S_n^{p-1}(y)$, and then $S_n^p(y)$ is computed from $R_n^p(x)$. Thus, the functions of x, y are computed alternately from one to another. For this, an initial arbitrary guess $S_n^0(y)$ is specified at the start of the each iterative process. The non-linear iterations proceed until reaching a fixed point within a defined tolerance (3.7).

First, the PGD is implemented for a constant source term *f* over the domain Ω for the equation 3.2, in the Poisson equation in a two- dimensional rectangular domain with homogeneous Dirichlet Boundary conditions for the unknown field *u(x,y)*, which vanishes at the domain boundary Γ. With a test function u^*, the weighted residual is in the following way

$$\int_{\Omega_x \times \Omega_y} u^* . (\Delta u + f) dx . dy \tag{3.10}$$

As discussed above $R_n^p(x)$ is computed from $S_n^{p-1}(y)$. In this case the PGD approximation is as follows (similar to (3.6)).

$$u^{n,p}(x, y) = \sum_{i=1}^{n-1} \lambda_i F_i(x) . G_i(y) + R_n^p(x) . S_n^{p-1}(y) \tag{3.11}$$

In the above, all the functions are unknown except $R_n^p(x)$. The choice of the weight function that is made (3.10) is

$$u^*(x, y) = R_n^*(x) . S_n^{p-1}(y), \tag{3.12}$$

which is the Galerkin weighted residual formulation.

Now, substituting (3.12), (3.11) into the (3.10) , the following is obtained. For simplification purpose , $R_n^p(x)$ is written as R_n^p , $F_i(x)$ is written as F_i , $R_n^*(x)$ as R_n^* and similarly for the y functions.

$$\begin{aligned} & -\int_{\Omega_x \times \Omega_y} R_n^* . S_n^{p-1} . \left(\frac{d^2 R_n^p}{dx^2} . S_n^{p-1} + R_n^p \frac{d^2 S_n^{p-1}}{dy^2}\right) dx . dy \\ & = \int_{\Omega_x \times \Omega_y} R_n^* . S_n^{p-1} . \sum_{i=1}^{n-1} \lambda_i \left(\frac{d^2 F_i}{dx^2} . G_i + F_i \frac{d^2 G_i}{dy^2}\right) dx . dy + \int_{\Omega_x \times \Omega_y} R_n^* . S_n^{p-1} . f \, dx . dy \end{aligned} \tag{3.13}$$

Note that in the above equation , all the functions concerned with *R, F* are solely functions of *x.* Similarly, all the functions concerned with *S, G* are solely functions of *y.* Thus the above equation, the integral for each of the terms can be split into Ω_x and Ω_y . All the terms that are dependent on *x* (and independent of y) are collected under the integral x and similarly for integral y. In the above equation all the terms which are functions of *y* are known. Thus, the following one- dimensional integrals of *y* can be computed over the domain Ω_y

$$\begin{cases} \alpha^x = \int_{\Omega_y} (S_n^{p-1}(y))^2 dy \\ \beta^x = \int_{\Omega_y} S_n^{p-1}(y) . \frac{d^2 S_n^{p-1}(y)}{dy^2} dy \\ \gamma_i^x = \int_{\Omega_y} S_n^{p-1}(y) . \lambda_i G_i(y) dy \\ \delta_i^x = \int_{\Omega_y} S_n^{p-1}(y) . \lambda_i \frac{d^2 G_i(y)}{dy^2} dy \\ \xi^x = \int_{\Omega_y} S_n^{p-1}(y) . f \, dy \end{cases} \tag{3.14}$$

The above values are computed and substituted in the equation (3.13) which becomes

$$\begin{aligned} & -\int_{\Omega_x} R_n^* . (\alpha_{x.} \frac{d^2 R_n^p}{dx^2} + \beta_{x.} R_n^p) dx \\ & = \int_{\Omega_x} R_n^* . \sum_{i=1}^{n-1} (\gamma_i^{x.} \frac{d^2 F_i}{dx^2} + \delta_i^x F_i) dx \ + \int_{\Omega_x} R_n^* . \xi^x dx \end{aligned} \tag{3.15}$$

The above is the weighted residual form of the one-dimensional problem defined over the domain Ω_x . The corresponding strong formulation (3.15) of the above can be returned to the following

$$-(\alpha_{x.} \frac{d^2 R_n^p}{dx^2} + \beta_{x.} R_n^p) dx = \sum_{i=1}^{n-1} (\gamma_i^{x.} \frac{d^2 F_i}{dx^2} + \delta_i^x F_i) dx \ + \ \xi^x dx \tag{3.16}$$

Thus, the two-dimensional Poisson equation is converted to a one-dimensional ordinary differential equation. This can be solved using any suitable ODE solving process. In the present thesis , *ode45* is employed to solve the equation. Thus, the ODE is solved for $R_n^p(x)$. After solving for R_n^p from S_n^{p-1} , the next step of the iteration is to proceed to solve the $S_n^p(y)$ from $R_n^p(x)$.

The same procedure is followed even for the next step , except for a slight change in the PGD approximation described in the equation 3.11. Now, the PGD approximation is the following

$$u^{n,p}(x,y)=\sum_{i=1}^{n-1}\lambda_i F_i(x).G_i(y)+R_n^p(x).S_n^p(y) \tag{3.17}$$

In the above approximate all the terms except $S_n^p(y)$ is known. The choice of the weight function in this case is made accordingly.

$$u^*(x,y)=R_n^p(x).S_n^*(y) \tag{3.18}$$

Now, again substituting (3.17) , (3.18) in (3.10), the following equation is obtained.

$$\begin{aligned}&-\int_{\Omega_x\times\Omega_y} R_n^p.S_n^*.(\frac{d^2R_n^p}{dx^2}.S_n^p+R_n^p\frac{d^2S_n^p}{dy^2})dx.dy\\&=\int_{\Omega_x\times\Omega_y} R_n^p.S_n^*.\sum_{i=1}^{n-1}\lambda_i(\frac{d^2F_i}{dx^2}.G_i+F_i\frac{d^2G_i}{dy^2})dx.dy+\int_{\Omega_x\times\Omega_y} R_n^p.S_n^*.f\,dx.dy\end{aligned} \tag{3.19}$$

Note that the functions are either solely dependent on *x* or solely dependent on *y.* This makes the integral to be split separately into the Ω_x and Ω_y .And all the functions of *x* are known and the integrals of them over Ω_x can be computed

$$\begin{cases}\alpha^y=\int_{\Omega_x}(R_n^p(x))^2dx\\ \beta^y=\int_{\Omega_x}R_n^p(x).\frac{d^2R_n^p(x)}{dx^2}dx\\ \gamma_i^y=\int_{\Omega_x}R_n^p(x).\lambda_iF_i(x)dx\\ \delta_i^y=\int_{\Omega_x}R_n^p(x).\lambda_i\frac{d^2F_i(x)}{dx^2}dx\\ \xi^y=\int_{\Omega_x}R_n^p(x).f\,dx\end{cases} \tag{3.20}$$

The above values are computed and substituted in the equation (3.19) which becomes

$$\begin{aligned}&-\int_{\Omega_y} S_n^* . (\alpha_{y.} \frac{d^2 S_n^p}{dy^2} + \beta_{y.} S_n^p) dy \\ &= \int_{\Omega_y} S_n^* . \sum_{i=1}^{n-1} (\gamma_i^{y.} \frac{d^2 G_i}{dy^2} + \delta_i^y G_i) dy \; + \int_{\Omega_y} S_n^* . \xi^y dy\end{aligned} \tag{3.21}$$

The above is the weighted residual form of the one-dimensional problem defined over the domain Ω_y . The corresponding strong formulation (3.21) of the above can be returned to the following

$$-(\alpha_{y.} \frac{d^2 S_n^p}{dy^2} + \beta_{y.} S_n^p) dy \; = \; \sum_{i=1}^{n-1} (\gamma_i^{y.} \frac{d^2 G_i}{dy^2} + \delta_i^y G_i) dy \; + \; \xi^y dy \tag{3.22}$$

As it can be seen, once again a ordinary differential equation is obtained. Thus, the two-dimensional Poisson equation is converted into two one-dimensional ordinary differential equations thus making it computationally cheaper. And also, this is the major advantage of the PGD method. It would convert any dimensional problem into the corresponding number of one – dimensional problems. This is the reason, for the with the PGD method the computational cost increase linearly with the increase in the number of dimensions unlike the conventional methods wherein increase in the number of dimensions increase the computational cost exponentially.

The properties of the PGD are studied for the present case with the help of an example. The Poisson equation is considered with the source term *f* which is set to the value one ,with a two dimensional domain $\Omega = \Omega_x \times \Omega_y$ = (-1,1) x (-1,1). This case is tested with many different step sizes, tolerances, initial approximations to study the method. The two tolerances that are defined in the method are the following :

Iteration Tolerance : This is the tolerance that is defined for the iterative step. The ϵ defined in the equation 3.7

Enrichment Tolerance : This is the tolerance that is defined for the enrichment step. The criteria that is considered for this tolerance is computed by residual *Re.*

$$\mathrm{Re} = \sqrt{\left(\int_{\Omega} (\Delta u + f)^2\right)} < \epsilon_n \tag{3.23}$$

If the above condition is satisfied the enrichment process itself stops and yields a solution. The way the enrichment tolerance is implemented is explained in more detail in the section 3.5 of this chapter.

3.2.1 Results and Analysis

The method is implemented for the simple case where the source term is constant following the procedure defined. Many tests are performed by varying the input parameters such as tolerances ,step size (mesh size) , initial approximation to study the method. The tests of few cases are illustrated below. Some of the plots that lead to the analysis are shown as and when required.

The solution obtained for the case where $f=1$ and for the following input parameter

Iteration Tolerance	$\epsilon=10^{-6}$
Enrichment Tolerance	$\epsilon_n=10^{-3}$
Step size in the x	$h_x=0.04$
Step size in the y	$h_y=0.04$

Note : Initial approximation is assumed to be one at all the nodal points for all the cases

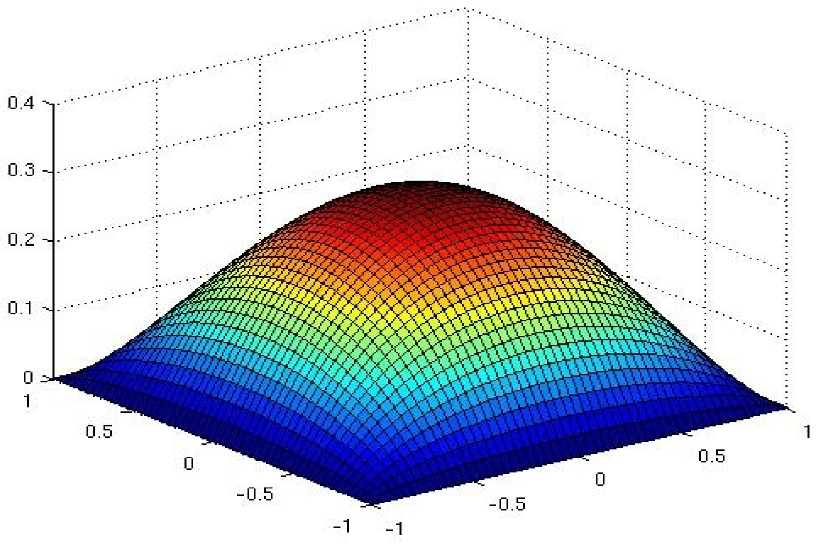
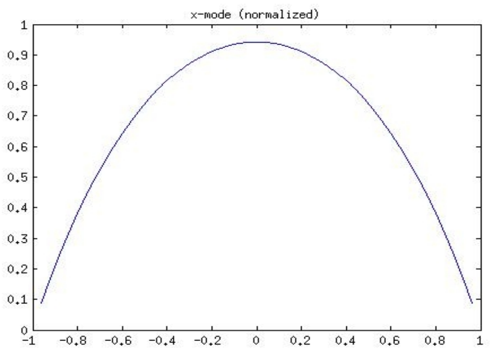
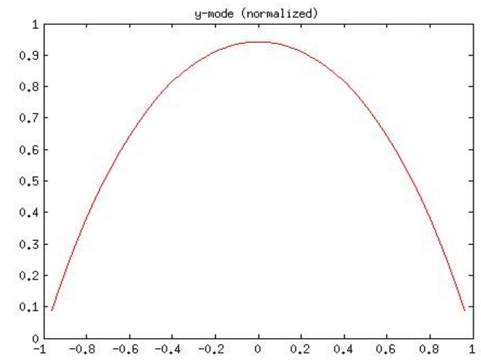


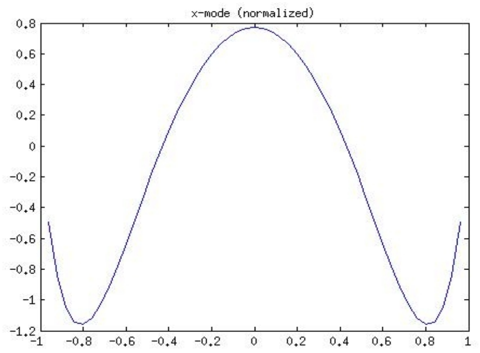
fig:3.1 Solution when f=1



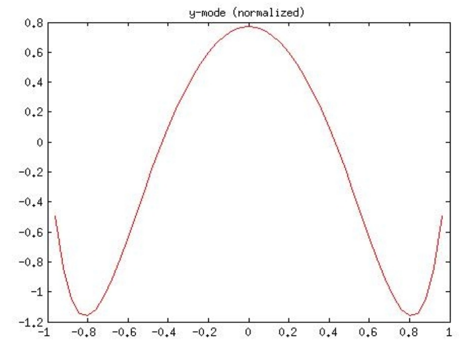
$F_1(x)$



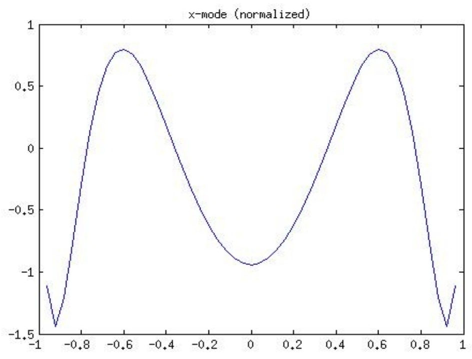
$G_1(y)$



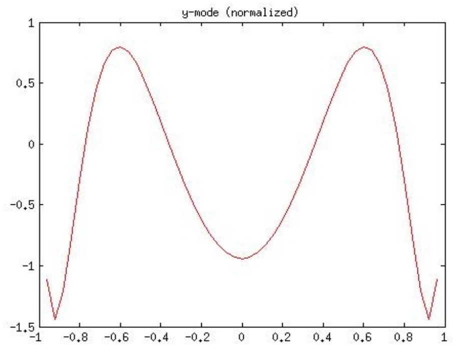
$F_2(x)$



$G_2(y)$



$F_3(x)$



$G_3(y)$

fig 3.2: x and y basis functions at each enrichment step

The final solution obtained in the PGD is method as discussed earlier is the sum of the product of the x & y functions obtained at each enrichment steps. The above are the x & y modes of the above case.

The following table shows the values obtained at each enrichment step

Enrichment Step (i)	Number of iterations taken (p)	Iteration tolerance reached	Enrichment tolerance reached	Lamba λ_i
1	5	5.845139e-09	-	3.293934e-01
2	7	2.115641e-07	8.975210e-03	2.956375e-03
3	11	4.106346e-07	5.150767e-04	1.696628e-04

Table 3.1 : Parameters obtained at each enrichment step

The following are some of the key points that can be noted from the tests

1. It can be observed that the enrichment tolerance decrease with each step , which means that the method is going to converge.

2. The number of iterations taken per each enrichment step increases. This can be explained with the help of the plots. It can be seen from the plots that the frequency of each mode for both x and y functions increase as the enrichment step progresses.

3. The coefficient λ_i decreases with the increase in the enrichment step.

3.3 Implementation for Non-constant Source Term

In the present section, the implementation of the PGD method is extended when the source term f is a non-constant source term is discussed. [2],[4]

$$f(x,y)=\sum_{j=1}^{K} F_j^x(x).\,F_j^y(y) \tag{3.24}$$

For instance, consider the function x^2-y^2 , it can be written in the form of $f(x,y)=\sum_{j=1}^{2} F_j^x(x).\,F_j^y(y)$ where ,

$$F_1^x(x)=x^2, F_2^x(x)=-1, F_1^y(y)=1, F_2^y(y)=y^2$$

Thus, most of the functions can be written in this separated form. Even in this case, the alternate direction strategy is used to solve for the functions at each enrichment step. The Poisson equation with non constant source term *f* (written as 3.24) in a two- dimensional rectangular domain with homogeneous Dirichlet Boundary conditions for the unknown field *u(x,y)* , which vanishes at the domain boundary Γ is considered. With a test function u^* , the weighted residual is in the following way

$$\int_{\Omega_x\times\Omega_y} u^*.(\Delta u+\sum_{j=1}^{K} F_j^x(x).F_j^y(y))dx.dy \tag{3.25}$$

As discussed in the section 3.2, $R_n^p(x)$ is computed from $S_n^{p-1}(y)$. The PGD approximation is as follows (similar to (3.6)).

$$u^{n,p}(x,y)=\sum_{i=1}^{n-1}\lambda_i F_i(x).G_i(y)+R_n^p(x).S_n^{p-1}(y) \tag{3.26}$$

In the above, all the functions are unknown except $R_n^p(x)$. The choice of the weight function that is made is

$$u^*(x,y)=R_n^*(x).S_n^{p-1}(y), \tag{3.27}$$

which is the Galerkin weighted residual formulation.

Now, substituting (3.26), (3.27) into the (3.28) , the following is obtained. For simplification purpose , $R_n^p(x)$ is written as R_n^p , $F_i(x)$ is written as F_i , $R_n^*(x)$ as R_n^* and similarly for the y functions.

$$\begin{aligned}
-\int_{\Omega_x\times\Omega_y} R_n^*.S_n^{p-1}.(\frac{d^2R_n^p}{dx^2}.S_n^{p-1}+R_n^p\frac{d^2S_n^{p-1}}{dy^2})dx.dy \\
=\int_{\Omega_x\times\Omega_y} R_n^*.S_n^{p-1}.\sum_{i=1}^{n-1}\lambda_i(\frac{d^2F_i}{dx^2}.G_i+F_i\frac{d^2G_i}{dy^2})dx.dy \\
+\int_{\Omega_x\times\Omega_y} R_n^*.S_n^{p-1}\sum_{j=1}^{K}F_j^x(x).F_j^y(y)dx.dy
\end{aligned} \tag{3.28}$$

Just like in the case of constant source term (section 3.2), each term of the integral can be split separately under Ω_x and Ω_y .Then the above equation becomes

$$-\int_{\Omega_x} R_n^*.(\alpha_{x.}\frac{d^2 R_n^p}{dx^2}+\beta_{x.} R_n^p)dx$$

$$=\int_{\Omega_x} R_n^*.\sum_{i=1}^{n-1}(\gamma_i^{x.}\frac{d^2 F_i}{dx^2}+\delta_i^x F_i)dx \;+\; \int_{\Omega_x} R_n^*.(\sum_{j=1}^{K}\xi_j^x.F_j^x(x))dx \tag{3.29}$$

where the coefficients are the following

$$\begin{cases} \alpha^x=\int_{\Omega_y}(S_n^{p-1}(y))^2 dy \\ \beta^x=\int_{\Omega_y} S_n^{p-1}(y).\frac{d^2 S_n^{p-1}(y)}{dy^2}dy \\ \gamma_i^x=\int_{\Omega_y} S_n^{p-1}(y).\lambda_i G_i(y)dy \\ \delta_i^x=\int_{\Omega_y} S_n^{p-1}(y).\lambda_i\frac{d^2 G_i(y)}{dy^2}dy \\ \xi_j^x=\int_{\Omega_y} S_n^{p-1}(y).F_j^y dy \end{cases} \tag{3.30}$$

It can be seen that the coefficients are all functions of *y* which are all known. Thus, it can be solved either using ODE technique (like in the section 3.2) or by the finite element method which is explained in the section to obtain the $R_n^p(x)$.

The same procedure is followed even for the next step , except for a slight change in the PGD approximation described in the equation 3.27. Now, the PGD approximation is the following

$$u^{n,p}(x,y)=\sum_{i=1}^{n-1}\lambda_i F_i(x).G_i(y)+R_n^p(x).S_n^p(y) \tag{3.31}$$

In the above approximate all the terms except $S_n^p(y)$ is known. The choice of the weight function in this case is made accordingly.

$$u^*(x,y)=R_n^p(x).S_n^*(y) \tag{3.32}$$

Now , again substituting (3.17) , (3.18) in (3.10), the following equation is obtained.

$$
\begin{aligned}
&-\int_{\Omega_x\times\Omega_y} R_n^p . S_n^* . \left(\frac{d^2 R_n^p}{dx^2} . S_n^p + R_n^p \frac{d^2 S_n^p}{dy^2}\right) dx . dy \\
&= \int_{\Omega_x\times\Omega_y} R_n^p . S_n^* . \sum_{i=1}^{n-1} \lambda_i \left(\frac{d^2 F_i}{dx^2} . G_i + F_i \frac{d^2 G_i}{dy^2}\right) dx . dy \\
&+ \int_{\Omega_x\times\Omega_y} R_n^p . S_n^* . \sum_{j=1}^{K} F_j^x(x) . F_j^y(y)\, dx . dy
\end{aligned}
\tag{3.33}
$$

Note that the functions are either solely dependent on *x* or solely dependent on *y.* This makes the integral to be split separately into the Ω_x and Ω_y And all the functions of *x* are known and the integrals of them over Ω_x can be computed

$$
\left\{
\begin{aligned}
\alpha^y &= \int_{\Omega_x} (R_n^p(x))^2\, dx \\
\beta^y &= \int_{\Omega_x} R_n^p(x) . \frac{d^2 R_n^p(x)}{dx^2}\, dx \\
\gamma_i^y &= \int_{\Omega_x} R_n^p(x) . \lambda_i F_i(x)\, dx \\
\delta_i^y &= \int_{\Omega_x} R_n^p(x) . \lambda_i \frac{d^2 F_i(x)}{dx^2}\, dx \\
\xi_j^y &= \int_{\Omega_x} R_n^p(x) . F_j^x(x)\, dx
\end{aligned}
\right.
\tag{3.34}
$$

The above values are computed and substituted in the equation (3.19) after which it becomes

$$
\begin{aligned}
&-\int_{\Omega_y} S_n^* . \left(\alpha_{y.} \frac{d^2 S_n^p}{dy^2} + \beta_{y.} S_n^p\right) dy \\
&= \int_{\Omega_y} S_n^* . \sum_{i=1}^{n-1} \left(\gamma_i^{y.} \frac{d^2 G_i}{dy^2} + \delta_i^y G_i\right) dy \; + \; \int_{\Omega_y} S_n^* . \left(\sum_{j=1}^{K} \xi_j^{y.} F_j^y(y)\right) dy
\end{aligned}
\tag{3.35}
$$

Thus, the two-dimensional Poisson equation is reduced to two one-dimensional equations which can be solved by using finite element methods.

3.3.1 Numerical Results

The numerical results of the few of the cases are discussed.

Case 1 :The first test case is defined in the domain Ω = (-1,1) x (-1,1), with the source term $f(x,y)=\cos(2\pi x)\sin(2\pi y)$. The solution took 2 iterations to reach the enrichment tolerance of 10^{-6} . The solution consists of two functions in the x-coordinate and other two functions in the y-coordinate. The modes which are the normalized functions along with the solution are represented in the following figures. The λ values obtained are $\lambda_1=1.14\times10^{-2};\lambda_2=9.74\times10^{-16}$.

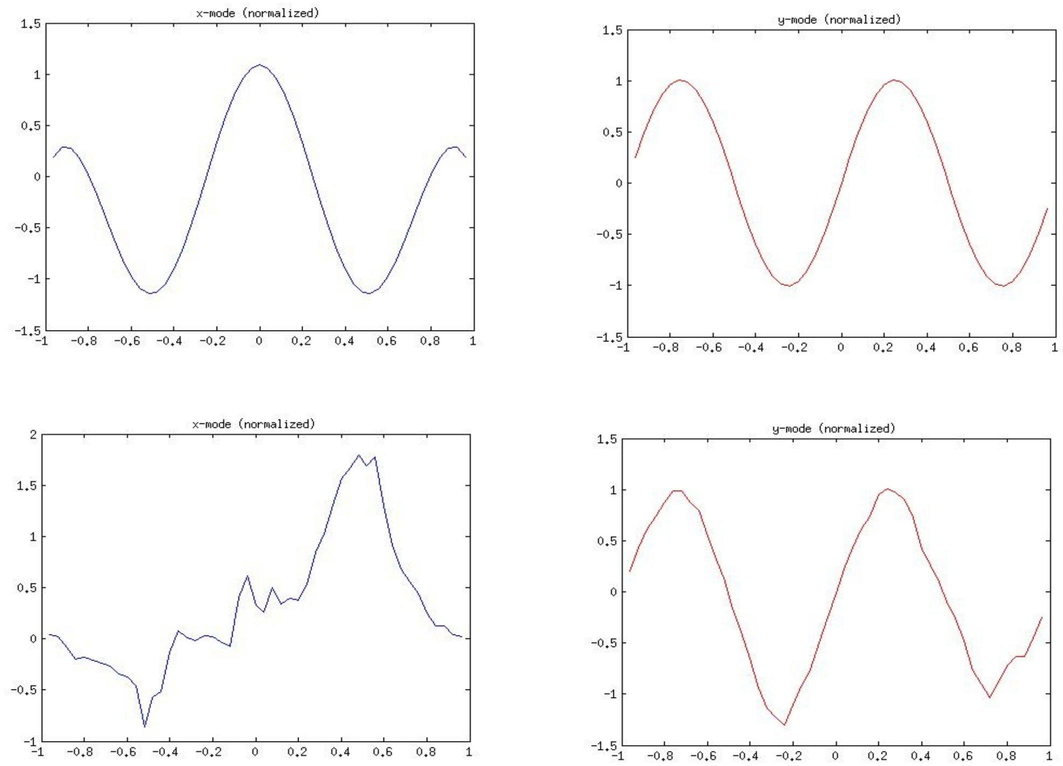


fig 3.3: Modes of the case 1($F_1(x)$, $G_1(x)$,$F_2(x)$,$G_2(y)$)

The following is the corresponding computed solution.

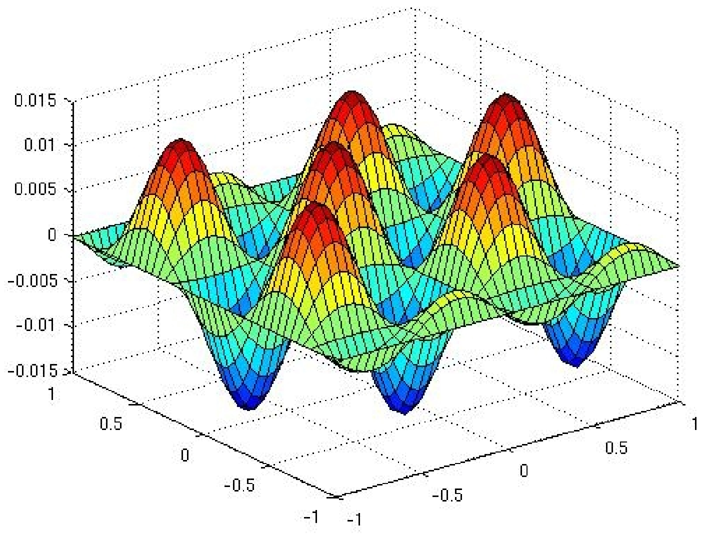


fig 3.4: Computed solution of the case 1

Case 2 :The second test case is defined in the domain Ω = (-1,1) x (-1,1), with the source term $f(x,y)=x^2-y^2$. The solution takes 10 iterations to converge to the tolerance of 10^{-5} . The modes which are the normalized functions along with the solution are represented in the following figures.

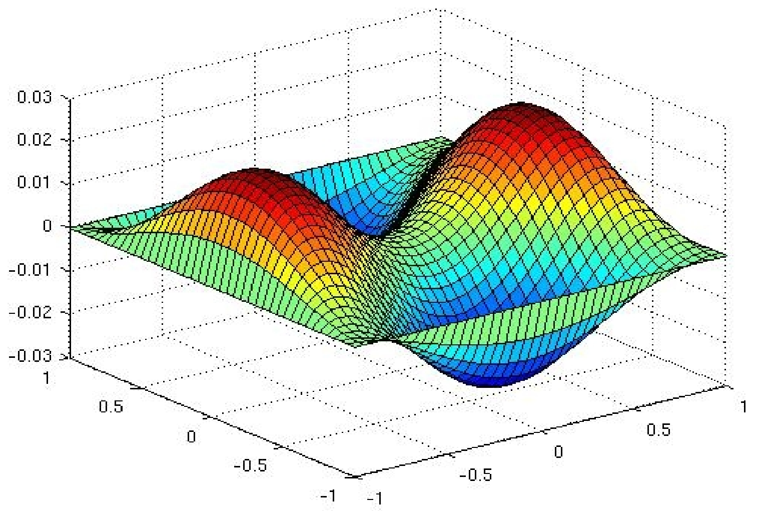
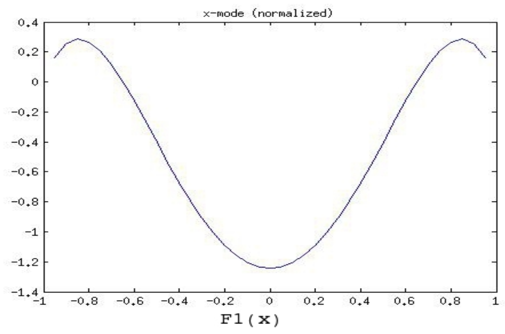
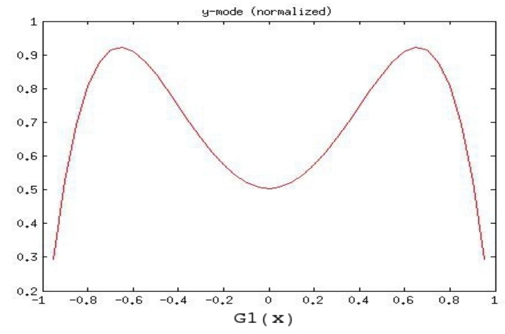
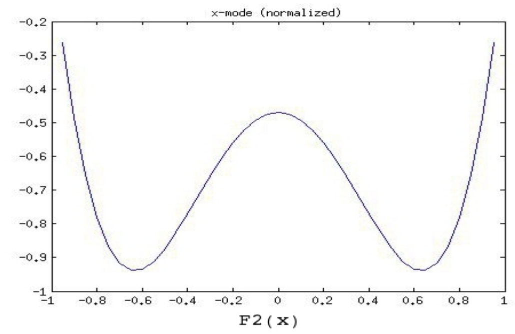


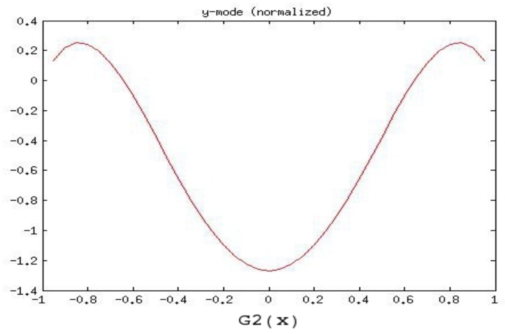
fig 3.5: Computed solution of the case 2

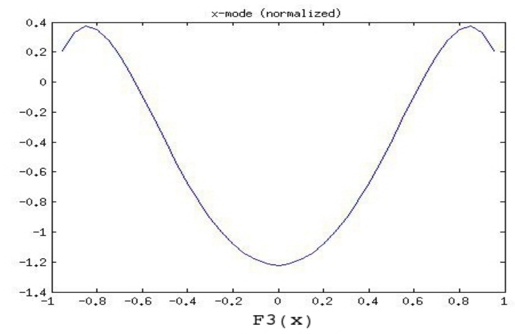
The corresponding plots of the basis functions at each enrichment step are:

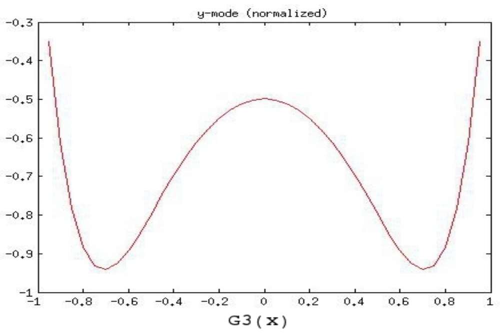


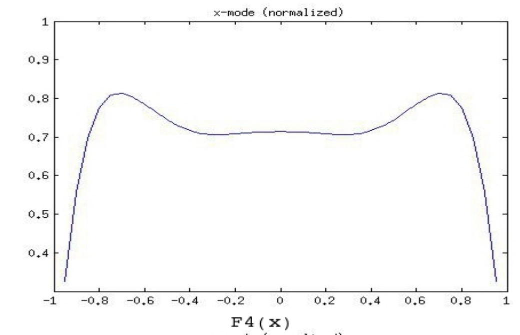


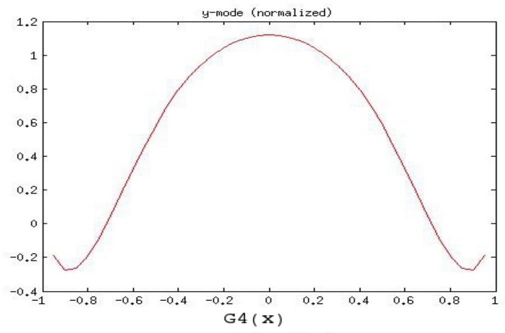


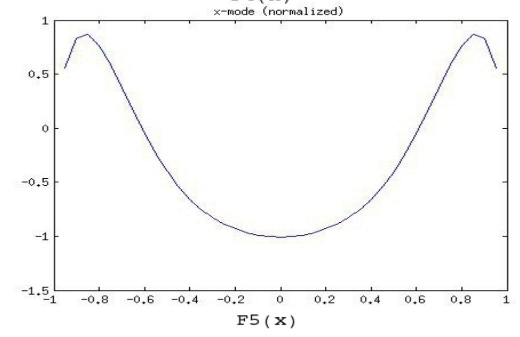


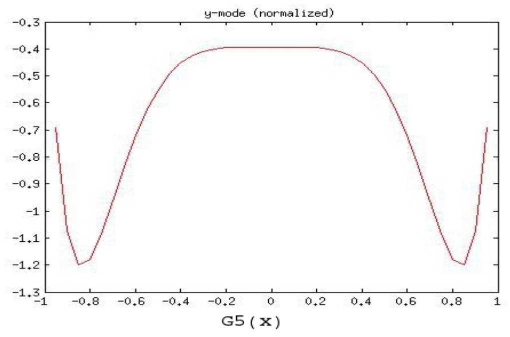












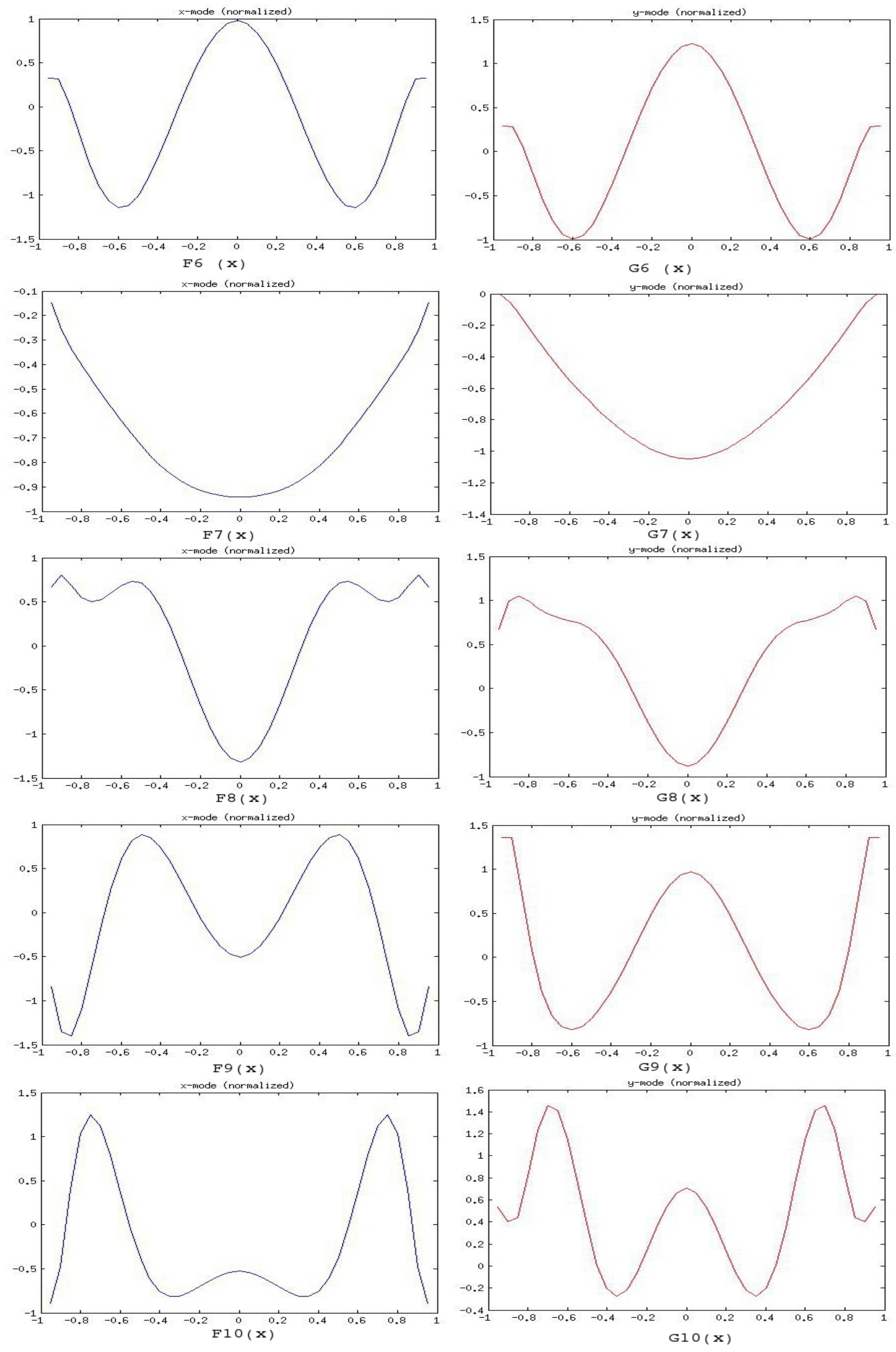
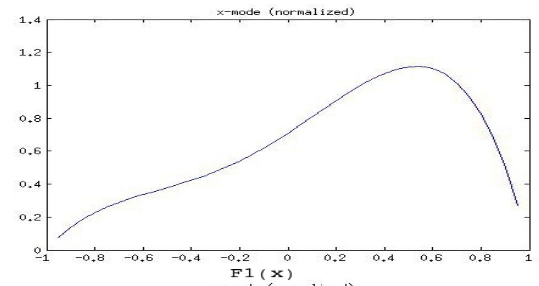
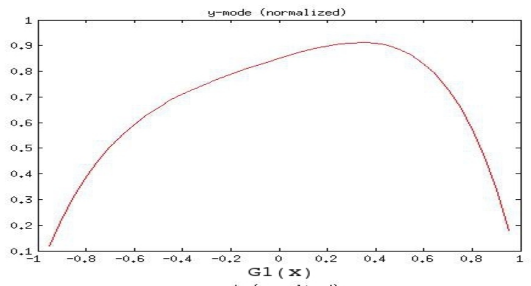
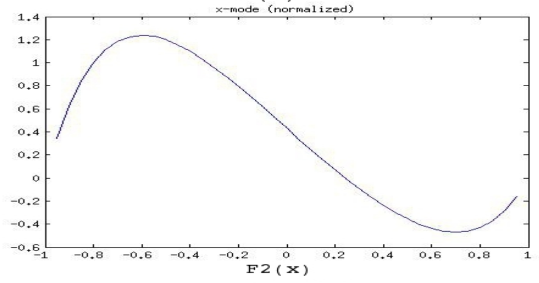


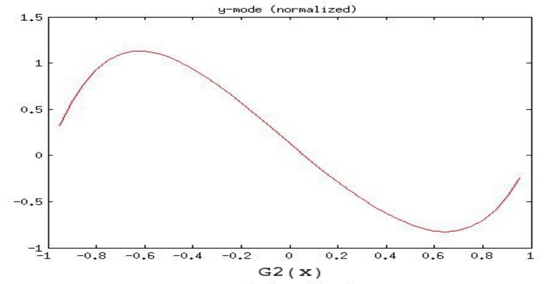
fig 3.6: Modes of the case 2

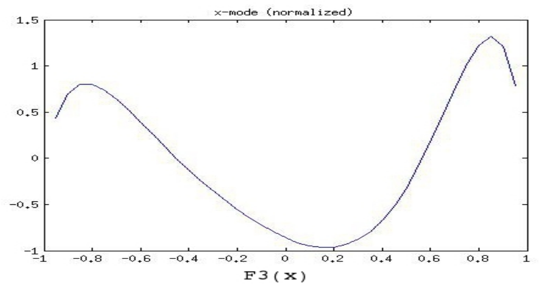
Case 3 :The third test case is defined in the domain Ω = (-1,1) x (-1,1), with the source term $f(x, y)=2x^2+x+y^2-0.2y+3xy$. The solution takes 39 iterations to converge to the enrichment tolerance of 10^{-5} .The first 9 normalized functions along with the computed solution are presented in the following figures

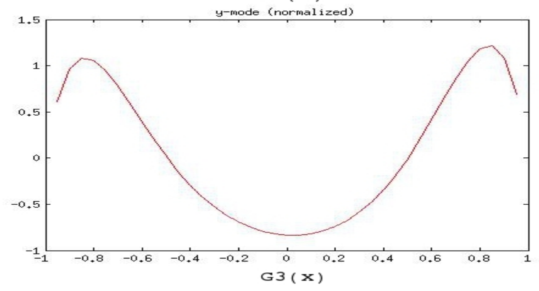


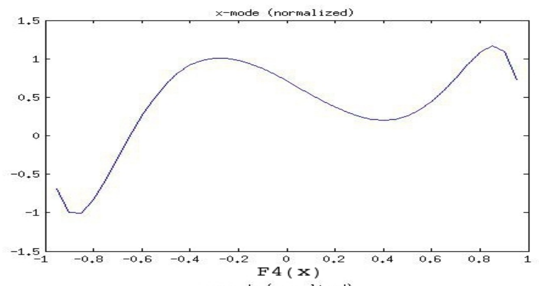


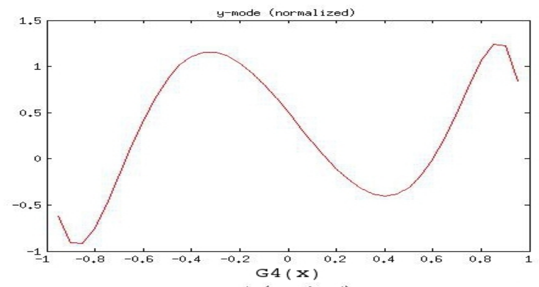


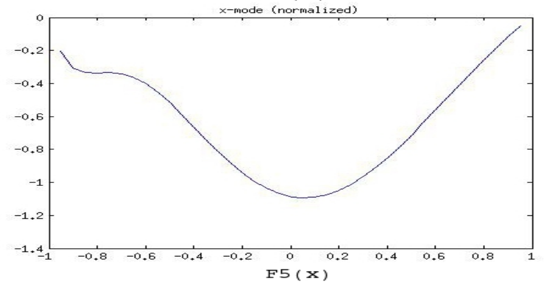


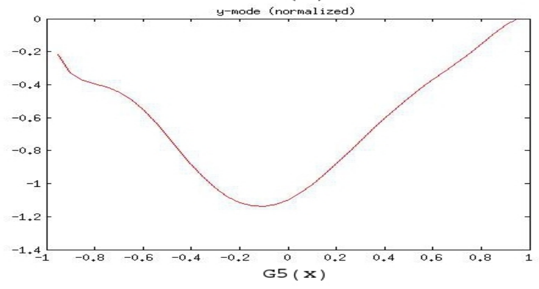












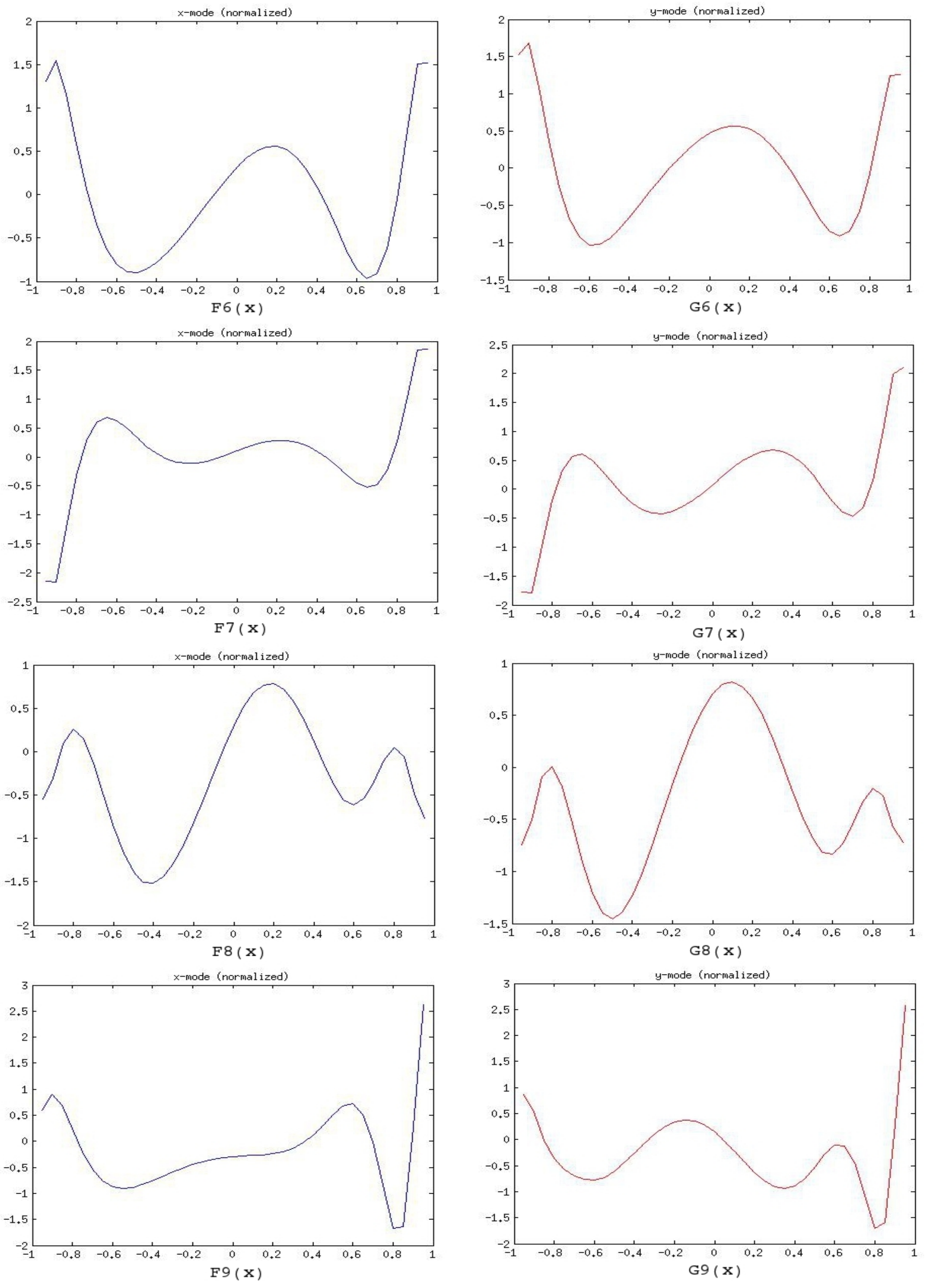


fig 3.7: First 9 Modes of the case 3

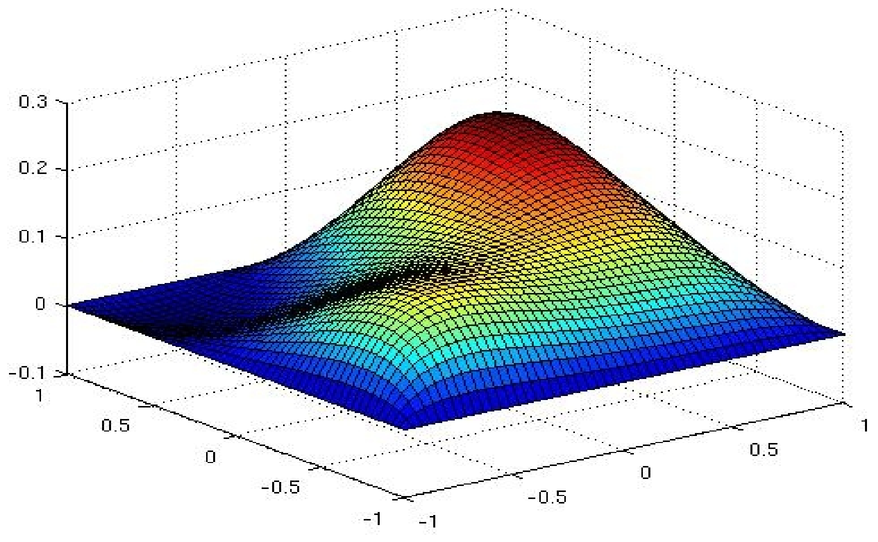


fig 3.8: Computed solution of the case3

Case 4 :The fourth test case is defined in the domain Ω = (-1,1) x (-1,1), with the source term $f(x,y)=2(2-x^2-y^2)$. The solution takes 3 iterations to converge to the tolerance of 10^{-5} . The normalized functions along with the computed solution are presented in the following figures

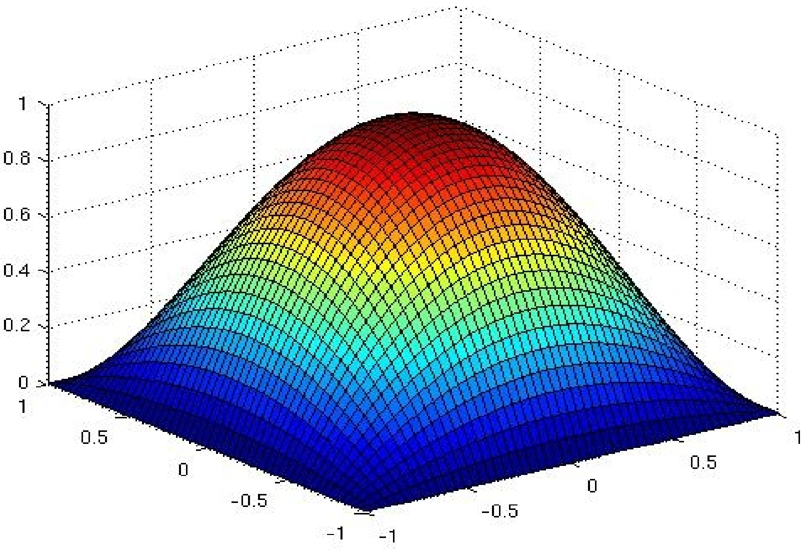


fig 3.9: Computed solution of the case 4

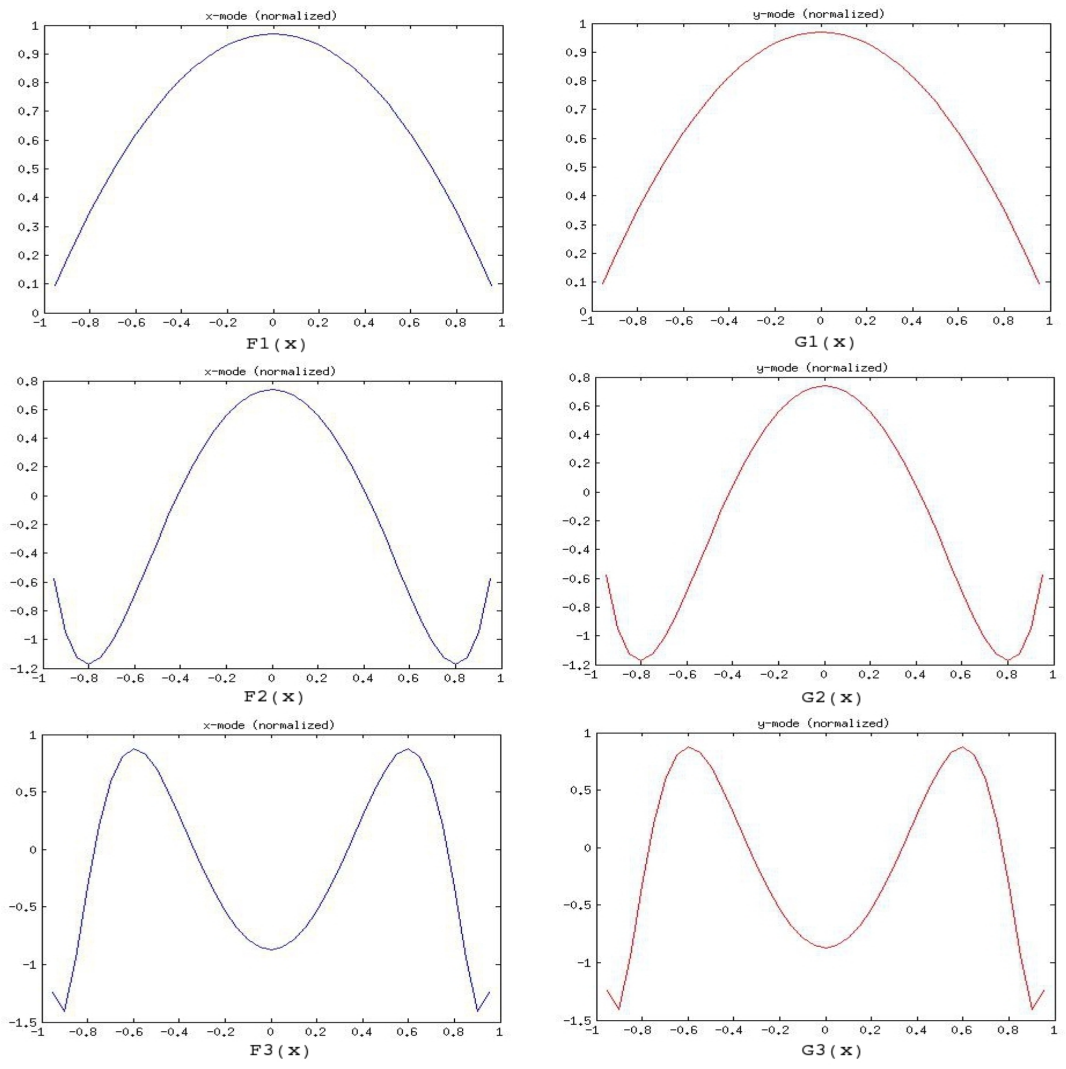


fig:3.10 Modes of case 4

3.4 Implementation for Diffusion Term

Now, the PGD method is further extended to the equation 3.2 with a diffusion coefficient k. The equation would then look like the following

$$-\nabla.(k\nabla u)=f(x,y) \tag{3.36}$$

in the two dimensional rectangular domain $\Omega=\Omega_x\times\Omega_y$. The PGD approximate solution is obtained in the same separated form as previous $u^n(x,y)=\sum_{i=1}^{n}\lambda_i F_i(x)G_i(y)$

For the sake of simplicity, like in the case of non-constant source term, the diffusion term is written in the following separated way

$$k = k_x(x) . k_y(y) \tag{3.37}$$

The idea here is to split the integrals of the domain into *x & y* like in the previous cases. The Poisson equation with non constant source term *f* (written as 3.24) and with a diffusion coefficient in a two- dimensional rectangular domain with homogeneous Dirichlet Boundary conditions for the unknown field *u(x,y)* , which vanishes at the domain boundary Γ is considered. With a test function u^* , the weighted residual is in the following way

$$\int_{\Omega_x \times \Omega_y} u^* . (\nabla . (k \nabla u) + \sum_{j=1}^{K} F_j^x(x) . F_j^y(y)) dx . dy \tag{3.38}$$

As discussed in the section 3.2, $R_n^p(x)$ is computed from $S_n^{p-1}(y)$. The PGD approximation is as follows (similar to (3.6)).

$$u^{n,p}(x, y) = \sum_{i=1}^{n-1} \lambda_i F_i(x) . G_i(y) + R_n^p(x) . S_n^{p-1}(y) \tag{3.39}$$

In the above, all the functions are unknown except $R_n^p(x)$. The choice of the weight function that is made is

$$u^*(x, y) = R_n^*(x) . S_n^{p-1}(y), \tag{3.40}$$

which is the Galerkin weighted residual formulation. Now, substituting (3.40), (3.39) into the (3.38) , the following is obtained. Note that $k_x(x)$ is written as k_x and $k_y(y)$ as k_y for the sake of simplicity.

$$\begin{aligned} &\int_{\Omega_x} \left(\frac{dR_n^*}{dx} \frac{dR_n^p}{dx} k_x \alpha^x + R_n^* R_n^p k_x \beta^x \right) dx \\ &= -\int_{\Omega_x} \sum_{i=1}^{n-1} \lambda_i \left(\frac{dR_n{}^*}{dx} \frac{dF_i}{dx} k_x \gamma_i^x + R_n^* F_i k_x \delta_x \right) . dx + \int_{\Omega_x} R_n^* \sum_{j=1}^{K} \xi_j^x F_j(x) . dx \end{aligned} \tag{3.41}$$

Like in the previous sections, the integral is split into corresponding *x , y* domains. In the above equation the constants are given by

$$\left\{\begin{aligned} \alpha^x &= \int_{\Omega_y} k_y(y)(S_n^{p-1}(y))^2\,dy \\ \beta^x &= \int_{\Omega_y} k_y(y)\frac{d^2 S_n^{p-1}(y)}{dy^2}\,dy \\ \gamma_i^x &= \int_{\Omega_y} k_y(y) S_n^{p-1}(y).\lambda_i G_i(y)\,dy \\ \delta_i^x &= \int_{\Omega_y} k_y(y).\frac{dS_n^{p-1}(y)}{dy}.\lambda_i\frac{d\,G_i(y)}{dy}\,dy \\ \xi_j^x &= \int_{\Omega_y} S_n^{p-1}(y).F_j^y\,dy \end{aligned}\right. \tag{3.42}$$

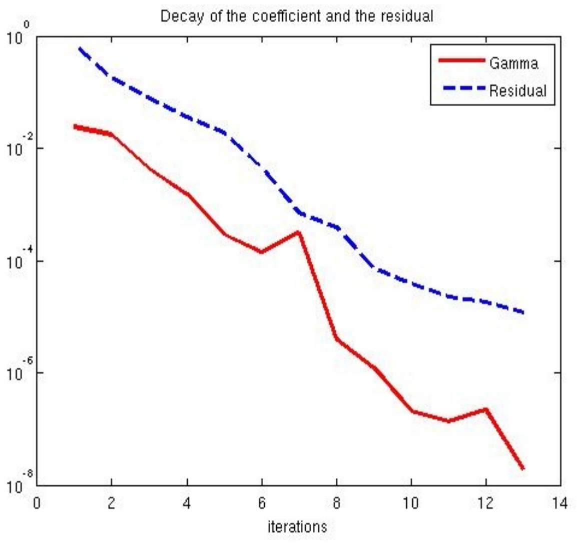
The equation 3.41 is solved using 1-D finite elements. (Discussed in chapter 2) The following are computed using the finite elements. The numerical tests that are done using the diffusion term are discussed in chapter 5

3.5 Convergence Criteria

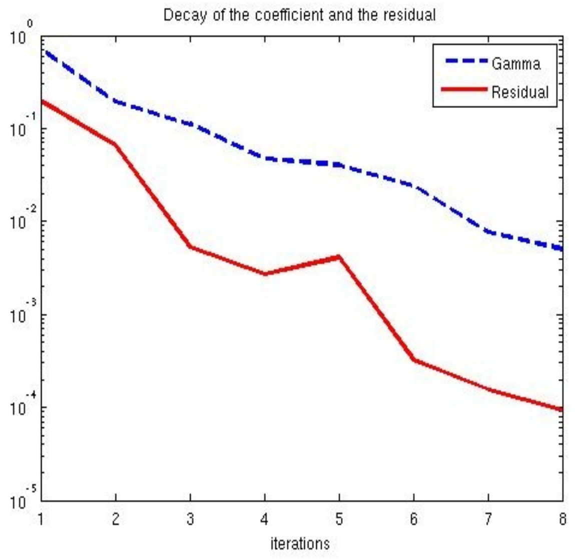
The convergence criteria for the enrichment step is already discussed in the section 3.2 , which is the $\mathrm{Re}=\sqrt{(\int_{\Omega}(\Delta u+f)^2)}<\epsilon_n$ (3.23). But computation of the residual (Re) at each step is computationally expensive. So, in order to decrease the computational cost the following criteria is implemented.[2]

$$\frac{\lambda_{n+1}}{max(\lambda_{1,}..\lambda_n)}<\epsilon_n \tag{3.43}$$

It is checked that the order of the tolerance obtained by calculation of the residual and above equation (3.43) are the same. From the numerical tests presented (in this section 3.2.1 & 3.3.1), it can be observed that the way value of coefficient λ_n is decayed as the steps are progressed. Also it is observed that the magnitude of the λ and the enrichment tolerance (table 3.1) are the same after each enrichment step . Thus, in implementation the (3.43) is used as convergence criteria as it is computationally much cheaper than to compute the residual at each step . The following are the plots which show the decay of the coefficient λ over the iterations with the value of the residual calculated as per (3.23)



case 3



case 2

fig 3.11 Decay of the coefficient and residual

Since the decay shows the same behavior and the order of the magnitude of the residual (3.23) and the criteria shown in (3.43) is the same, the latter is used in the implementation as it is computationally much economical.

Chapter 4

The PGD method for preconditioning

This chapter describes the idea and the implementation of the based preconditioner for the scaler elliptic problems. In the present thesis, scaler elliptic boundary value problems are considered. The problem is discretized by employing Galerkin Finite Elements, which generate the linear system of equations. The linear system is solved using an iterative method and a suitable preconditioner is employed to improve the convergence properties of the method. The PGD method is used to characterize the preconditioner. The algebraic residual is computed, from which the residual finite elements function is defined. For this, the PGD procedure is applied to compute the solution of the ¨Preconditioning Problem¨.

4.1 Preconditioning

The basic problem with many iterative schemes compared to the direct solvers is the lack of robustness. While some of the iterative methods which are well founded in the theory suffer from slow convergence. Both the convergence properties and the robustness can be improved by using the preconditioning. Preconditioning is a procedure of an application of a transformation, called the preconditioner, that conditions a given problem into a form that is more suitable for numerical solution. Further, the preconditioned problem is solved using a iterative scheme. [5]. Let the preconditioner be $C_{n,}$. The preconditioned C_n needs to be chosen in such that

$$C_n^{-1} Ax = C_n^{-1} b \tag{4.1}$$

In general a good preconditioner satisfies the following three conditions.

1. The computation of C_n is fast and computationally cheap.
2. Let the preconditioned residual be z_n, then the computation of $C_n z_n$ be easy and computationally as effective as possible
3. The spectrum of $C_n^{-1} A$ is much better clustered. This would lead to a

faster convergence.

Note that, C_n is associated to a symmetric, continuous and coercive problem if the original problem (here A) itself is a symmetric continuous and coercive. Thus, it can be used in the framework of much more effective iterative methods such as preconditioned Conjugate Gradient (PCG) method.

4.2 Preconditioned Conjugate Gradient Method

The Conjugate Gradient (CG) method works for works very well on matrices that are well-conditioned. However, in real applications, most matrices are ill-conditioned, reducing the efficiency of the method. So, the Preconditioned Conjugate Gradient (PCG) is method which can be used for ill-conditioned matrices also effectively. The PCG method is one of the most effective tools for solving the large sparse symmetric positive- definite systems.

The PCG algorithm, applied to the system $Ax = b$, starts with an initial guess of the solution x_0 , with an initial residual r_0 , and with an initial search direction that is equal to the initial residual: $p_0 = r_0$. The idea behind the CG method is that the residual $r_k = b - Ax_k$ is orthogonal to the Krylov subspace generated by b, and therefore each residual is perpendicular to all the previous residuals. The residual r_n is computed at each step n. The solution of the next step is found using a search direction that is only a linear combination of the previous search directions and the current residual. One of the main property that distinguishes CG algorithm from the other iterative methods is that the solution is reached in at most n steps for the system $Ax = b$ where A is an n x n symmetric positive definite matrix. For, instance the following is the CG algorithm for a 2 x 2 matrix.

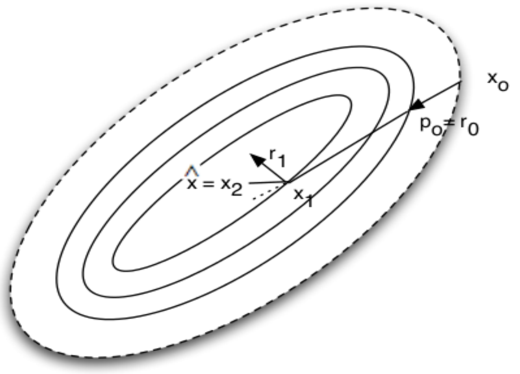


Fig: 4.1 CG algorithm for a 2 x *2 matrix*

As discussed in section 4.1, preconditioning an important technique used to develop an efficient CG method solver and help for the faster convergence of the method.

The following shows the algorithm of the PCG

ALGORITHM 4.1

```
1. Compute  $r_0 = b - Ax_0$ , $z_0 = C_0^{-1} r_0$, and  $p_0 = z_0$
2. For j = 0, 1...........until the convergence Do
3.         $\hat{\alpha}_j = (r_j, z_j)/(Ap_j, p_j)$
4.         $x_{j+1} = x_j + \hat{\alpha}_j p_j$
5.         $r_{j+1} = r_j - \hat{\alpha}_j A p_j$
6.         $z_{j+1} = C_j^{-1} r_{j+1}$
7.         $\hat{\beta}_j = (r_{j+1}, z_{j+1})/(r_j, z_j)$
8.         $p_{j+1} = z_{j+1} + \hat{\beta}_j p_j$
9. EndDo
```

Note that p_j is the conjugate gradient directions , r_j is the residual at each step, z_j is the preconditioned residual.[6]

4.3 The PGD as preconditioner for PCG

In this section, the way the PGD method is applied to compute the preconditioner for the PCG is explained. Consider the following elliptic boundary value problem

$$\begin{aligned} -\Delta u &= f \quad \text{in}\,\Omega \\ u &= 0 \quad on\,\partial\Omega \end{aligned} \tag{4.1}$$

where Ω is an open bounded domain, $\Omega \subset \mathbb{R}^2$. The weak form of the problem is given by $u \in V$ such that

$$a(u,v) = l(v) \quad \forall V \tag{4.2}$$

where $V = H_0^1(\Omega), a(u,v) = \int_\Omega \nabla u . \nabla v \, d\Omega$ is a continuous and coercive bilinear form on $V \times V$, and $l(v) = \int_\Omega fv \, d\Omega$ is a linear continuous functional on V .

The weak problem (4.3) is well-posed. The weak form of the problem is approximated to a discrete form using Galerkin Finite Elements, where $u_h \in V_h$ such that

$$a(u_h, v_h) = l(v_h) \quad \forall\, v_h \in V_h \tag{4.4}$$

where $V_h \subset V$ is a suitable finite dimensional space. The basis functions for the space V_h is introduced, $\{N_i(x, y)\}_{i=1,2,\dots,N_h}$ .In this case, a piecewise polynomial functions defined on a quadrilateral mesh is defined. Now the continuous problems produces a linear system of equations.

$$\mathrm{Au} = \mathrm{b} \tag{4.5}$$

The above linear system is solved using an iterative method. A suitable preconditioner is characterized to improve the convergence properties. In order to explain the implementation for the sake of simplicity, preconditioned Richardson method is used (in place of which PCG is employed later). The $\mathbf{u}^0$, the initial value is approximated. For the n+1 th step the u^{n+1} is computed in the following way

$$u^{n+1} = u^n - C_n^{-1}(Au^n - b) \tag{4.6}$$

where C_n denotes a suitable preconditioner which changes at the each iteration to improve the convergence properties. As explained in the section 4.1, the preconditioner C_n the is best if is in a such a way that $C_n^{-1} A = I$ so that the preconditioned residual z^n that is computed at each step is as cheap as possible

$$C_n z^n = Au^n - b \tag{4.7}$$

Now, in the present thesis, this preconditioner C_n is characterized using the PGD method. The following are the steps involved.

1. Computation of the Finite Elements Residual

As discussed earlier, say the approximate solution is $\mathbf{u}^n$, of the algebraic solution u is available at the step *n*. (where $n \geqslant 1$). The algebraic residual is given by

$$r^n = Au^n - b \qquad (r^n \in \mathbb{R}^{N_h}) \tag{4.8}$$

The above step can be expressed in the variational terms as follows

$$\int_{\Omega} r_h^n v_h \, d\Omega = a(u_h^n, v_h) - l(v_h) \qquad \forall v_h \in V_h \tag{4.9}$$

where $r_h^n \in V_h$ is the residual finite elements function. Indeed, the components of the vector r_h^n are defined as

$$r_i^n = \int_{\Omega} r_h^n N_i(x, y) \, d\Omega \qquad \forall i = 1, 2, \ldots.., N_h \tag{4.10}$$

The idea here is to write the residual at each node of the finite element mesh that is defined in 4.4. In this way the residual at all the elements is obtained.

Let *M* be the finite elements mass matrix, which is defined as follows at a nodal coordinate (*i, j*)

$$M_{ij} = \int_{\Omega} N_i(x, y) N_j(x, y) \, d\Omega \qquad i, j = 1, 2, \ldots. N_h \tag{4.11}$$

Then, the vector $\hat{r}^n$ is computed as the solution of the linear system (where M is the corresponding mass matrix)

$$M \hat{r}^n = r^n \tag{4.12}$$

This contains the nodal values of the residual finite elements function r_h^n in the nodes of the finite element grid and it can be written as

$$r_h^n(x, y) = \sum_{i=1}^{N_h} \hat{r}_i^n N_i(x, y) \tag{4.13}$$

If $\mathbb{Q}_r$ elements are used, on the generic element Ω_e of the computational mesh, then it can be written as

$$r_h^n(x, y)_{\Omega_e} = \sum_{i=1}^{n_e} \hat{r}_i^n N_i^{loc}(x) N_i^{loc}(y) \tag{4.14}$$

2. Projection of the residual on the PGD mesh

In the first step, the residual is obtained at each nodal point of the mesh. Now even the PGD method has a mesh defined. Note that the domain Ω is same for the both. The residual of the finite element mesh has to be projected onto the PGD mesh by using appropriate extrapolation technique. Note that in the present implementation the mesh used for the FEM and PGD is the same. So there is no need for this step.

3. PRECONDITIONING

After r_h^n is obtained, which is the residual at each of the points in the PGD mesh. Note, in the present case it comes from the step 1. Now the PGD is procedure is applied as described in the chapter 3 to compute the solution of the following "preconditioned problem".

$$\begin{aligned} -\Delta z^n_{PGD,m} &= r_h^n \quad \text{in}\,\Omega \\ z^n_{PGD,m} &= 0 \quad \text{on}\,\partial\Omega \end{aligned} \tag{4.15}$$

In the weak form :

$$\int_\Omega \nabla z^n_{PGD,m} . \nabla v_{PGD}\, d\Omega = \int_\Omega r_h^n v_{PGD}\, d\Omega \qquad \forall v_{PGD} \in V_{PGD} \tag{4.16}$$

where n denotes the number of basis functions (or the enrichment steps) that can be fixed a-priori and that can vary at each iteration *n* of the Richardson method (4.6). V_{PGD} is the space which is

$$V_{PGD} = span\{F_x^1(x)F_y^1(y), ..., F_x^m(x)F_y^m(y)\} \tag{4.17}$$

where $F_x^i(x)$ and $F_y^i(y)$ $(i=1,...,m)$ are the PGD basis functions that are computed using the procedure illustrated in the chapter 3. The number of basis functions that are needed to be computed has to be decided. This is discussed in the next section

4. PROJECTION OF PRECONDITIONED RESIDUAL ON THE FE SPACE

The function $z^n_{PGD,m}$ belongs to V_{PGD} and it is defined in Ω .It represents the preconditioned residual function in the space V_{PGD} . To obtain the preconditioned residual in the finite element space V_h , function $z^n_{PGD,m}$ needs to be reflected back using appropriate interpolation function. More precisely, this can be defined as follows

$$z_h^n \in V_h \qquad z_h^n = I_h z^n_{PGD,m} \tag{4.18}$$

where I_h is the interpolation operator such that

$$z_h^n(x_i) = z^n_{PGD,m}(x_i) \quad \forall x_i \, \textit{node of the finite element mesh} \tag{4.19}$$

Finally , $z^n \in \mathbb{R}^{N_h}$ be the vector of the nodal values of z_h^n . Note, in the present case since the mesh of the PGD and the FEM is the same this step is skipped as in this case $z_h^n = z_{PGD,m}^n$.

5. UPDATING THE SOLUTION

The final step involves updating the solution with the preconditioned residual obtained. This is computed in the following way

$$u^{n+1} = u^n - z^n \tag{4.20}$$

Computing the preconditioned residual or, equivalently, applying the preconditioner C_n in (4.7) corresponds to the step 3 of the procedure described above. To highlight the dependence on the PGD procedure and on the number of basis functions that one should compute, this is written as $C_{n,m}$. In the next section the number of basis functions that need to be considered is discussed.

4.4 The Basis Functions of the PGD

The solution from the PGD method is in the form (3.9) $\sum_{i=1}^{n} \lambda_i F_i(x) G_i(y)$ where *n* is the number of basis functions or the enrichment steps. The tolerance that can be reached depends upon the number of basis functions that are considered. Also the PGD error is a function of number of grid points for different number of basis functions. Numerical tests are conducted to reach the optimal number of basis functions. [2] . The tests are performed with different number of grid points and by varying the number of basis functions. This is done to optimize and get an idea of the behavior with different basis functions. There is need to optimize because as the number of basis functions are increased the computational cost increase, while on the other hand if the basis functions are too less the tolerance that can be reached is limited. To strike a balance between them and to come to a conclusion regarding the effect of tolerance reached the following tests are performed.The tests illustrated below, for different number of grid points (say it is N_x) and the number of grid points in they direction N_y , each with different set of base functions. The flags in the table denote the following

0 PCG converged to the desired tolerance TOL (tolerance) within MAXIT (maximum number of)iterations

1 PCG iterated MAXIT times but did not converge.

2 preconditioner M was ill-conditioned.

3 PCG stagnated (two consecutive iterates were the same).

4 one of the scalar quantities calculated during PCG became too small or too large to continue computing.

Note in the present case the desired tolerance is taken as 10^{-8} and the maximum number of iterations is 250.

For the source function $f(x,y)=2x^2+x+y^2-0.2y+3xy$ following are the plots for different grid values.

For $N_x = N_y = 20$

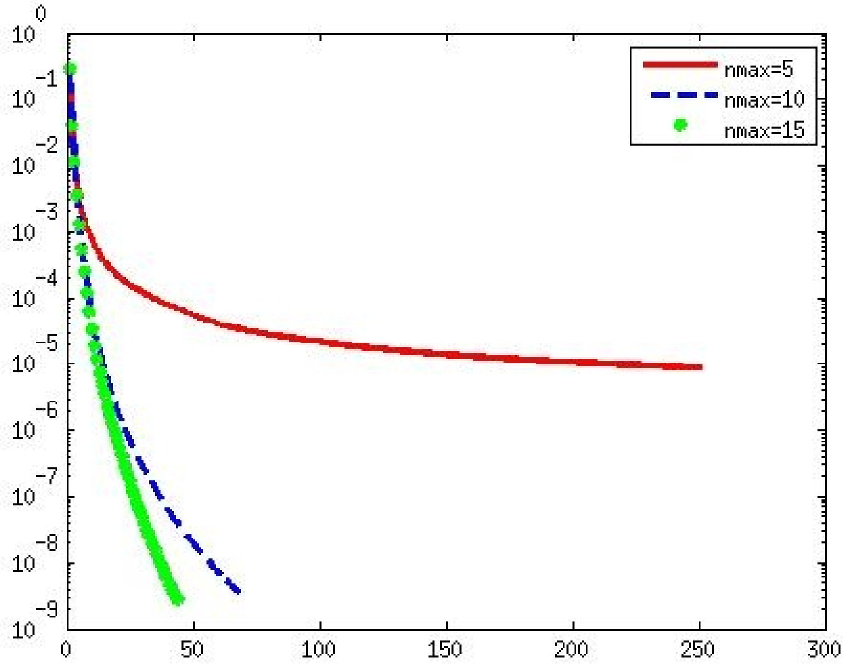


fig 4.2 log(residual) vs number of iterations for grid points in x and y 20x20

Number of basis functions (Nmax)	Number of iteration taken	Flag
5	250	1
10	70	0
12	43	0

Table 4.1 Number of iterations taken for the basis functions for 20x20

It can be seen from the plot and the table that if the number of basis functions is limited to 5, the method doesn't converge. After reaching the tolerance of

10^{-4} the residual doesn't decrease even with the increase in the number of iterations. It can be concluded that the contribution from the basis functions is limited with the number of basis functions.

An another case is considered where the source term is $f(x,y)=2(2-x^2-y^2)$

For N_x = N_y = 40 the following is the plot with different number of basis functions

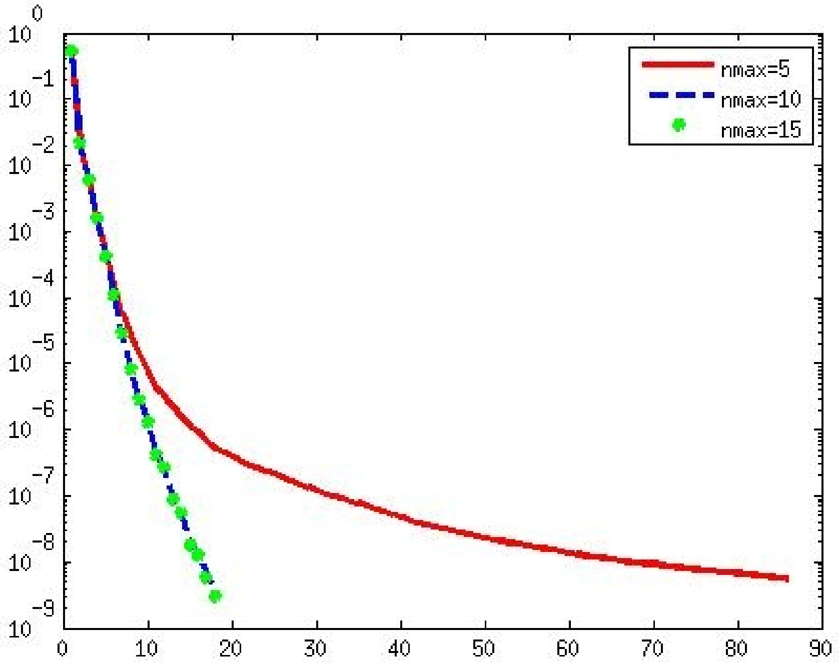


fig 4.3 log(residual) vs number of iterations for grid points in x and y 40x40

Number of basis functions (Nmax)	Number of iteration taken	Flag
5	85	0
10	17	0
12	17	0

*Table 4.2 Number of iterations taken for the basis functions for grid points 40*x40

It can be seen in the plot that for the maximum basis functions 10 and 12 the

convergence shows the same behavior (**the plots coincide**), whereas when N_{max} is 5, the convergence flattens and thus takes more number of iterations, although the desired tolerance is reached.
For the same source term and increasing the number of grid points to 80, following is the plot obtained between residual vector and number of iterations.

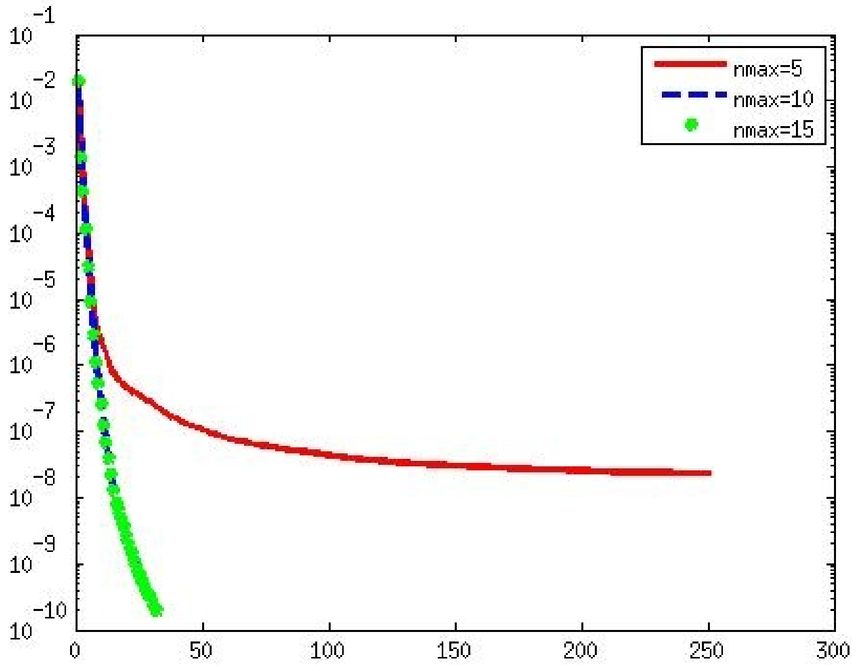


fig 4.4 log(residual) vs number of iterations for grid points in x and y 80x80

Number of basis functions (Nmax)	Number of iteration taken	Flag
5	250	1
10	31	0
12	31	0

*Table 4.3 Number of iterations taken for the basis functions for grid points 80*x80

Just like the previous plot, the convergence behavior for N_{max} equals 10 and 12 is the same. But it can be seen that for N_{max} equal to 5, the convergence is not reached even after reaching the maximum number of iterations. It can be also noted that the n number of iterations taken to reach the convergence has increased when there is an increase in the grid points.

4.4.1 Conclusions

The following conclusions can be drawn from the tests conducted with varying the number of basis functions or enrichment steps in the PGD and the grid points used.

1. The number of iterations taken increase with the increase in number of the grid points. This is expected as the number of grid points are increased the computations and iterations required has to be increased for the same tolerance.

2. There is a limitation on the tolerance reached for a fixed number of basis functions. For a lower tolerances, the number of basis functions that need to be used should be increased. This can be explained with the behavior of the PGD method. In the PGD method, the higher the mode, the frequency of the plot is increased. (Section 3.3.1 illustrated in all the cases). Thus, for smaller errors the number of modes or the basis functions that should be considered ought to be increased.

3. If the number of basis functions that are considered are too low, for certain accuracy the tolerance is reached at a very slow pace. It can be seen in the above cases when N_{max} is taken as 5.

4. As the number of basis functions are increased the convergence is faster and faster. Like for instance when N_{max} is 5 and 10, the difference in convergence rate is substantial.

5. In order to draw a balance between the computational cost and efficiency the maximum number of basis functions that need to taken should be restricted. In the present case, for instance, the difference between the case when N_{max} equals 10 and 12 is not much for the tolerance of 10^{-8}. And therefore in all the computations in later chapter the N_{max} is restricted to 10.

Chapter 5

Numerical Tests

In this chapter the tests and results of the various test cases are presented. Further, the convergence plots are presented with and without preconditioner for comparison

5.1 Tests and Results

The numerical tests for various cases using the PCG method with the PGD preconditioner algorithm that is developed in the thesis are carried out. The results for different cases are compared with the PCG method without the preconditioner, PCG method with the PGD preconditioner and the PGD method (latter two are developed as part of the thesis). All the tests are conducted for the a tolerance of 10^{-8} and for different mesh sizes

5.1.1 With Constant Source Term

The following is the case where the source term is constant

Case 1: f =1

Method	Number of iteration taken for different mesh sizes		
	20x20	40x40	80x80
PCG without preconditioner	25	52	105
PCG with the PGD preconditioner	22	31	43

Table 5.1 Number of iterations taken for constant source term

5.1.2 With Non -Constant Source Term

The following are the cases when source term is non-constant Source Term

Source Term	Method	Number of iterations Taken		
		20x20	40x40	80x80
$f(x,y)=\cos(2\pi x)\sin(2\pi y)$	PCG without preconditioner	10	20	40
	PCG with the PGD preconditioner	10	13	13
$f(x,y)=x^2-y^2$	PCG without preconditioner	17	36	72
	PCG with the PGD preconditioner	16	20	31
$f(x,y)=2x^2+x+y^2-0.2y+3xy$	PCG without preconditioner	43	43	176
	PCG with the PGD preconditioner	42	42	150
$f(x,y)=2(2-x^2-y^2)$	PCG without preconditioner	23	23	92
	PCG with the PGD preconditioner	17	17	23

Table 5.2 Number of iterations taken for Non constant source term

5.1.3 With Diffusion Term

The following are the cases when source term is kept constant (in the present case *f = 1)* and diffusion term is varied.

Diffusion Term	Method	Number of iterations Taken		
		20x20	40x40	80x80
K =0.01	PCG without preconditioner	25	52	105
	PCG with the PGD preconditioner	22	23	43
K =0.001	PCG without preconditioner	25	52	105
	PCG with the PGD preconditioner	22	23	43

Table 5.3 Number of iterations taken with Diffusion term

5.2 Convergence Behavior

The following are the convergence plots of for various cases. The semi logarithmic plots are between the residual vector and number of iterations. Basically the comparison here is made between PCG with and without the preconditioner. These plots give an idea about the way the convergence is obtained. Depending upon the way the convergence is obtained in the with respect to the number of iterations the behavior of the method can be concluded.

The same cases that are taken in the section 3 & section 4 are considered here.

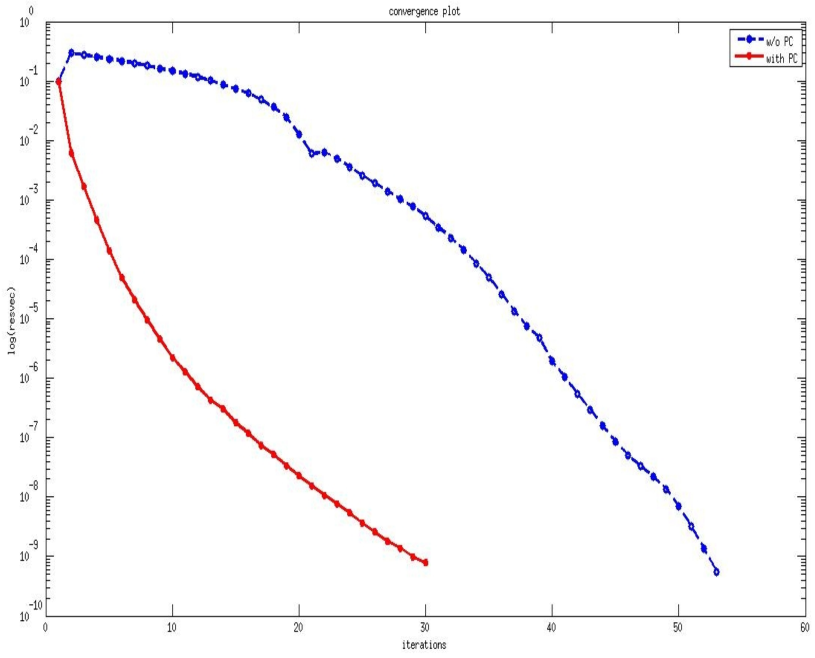


Fig 5.1 Convergence plot for Source Term f=1

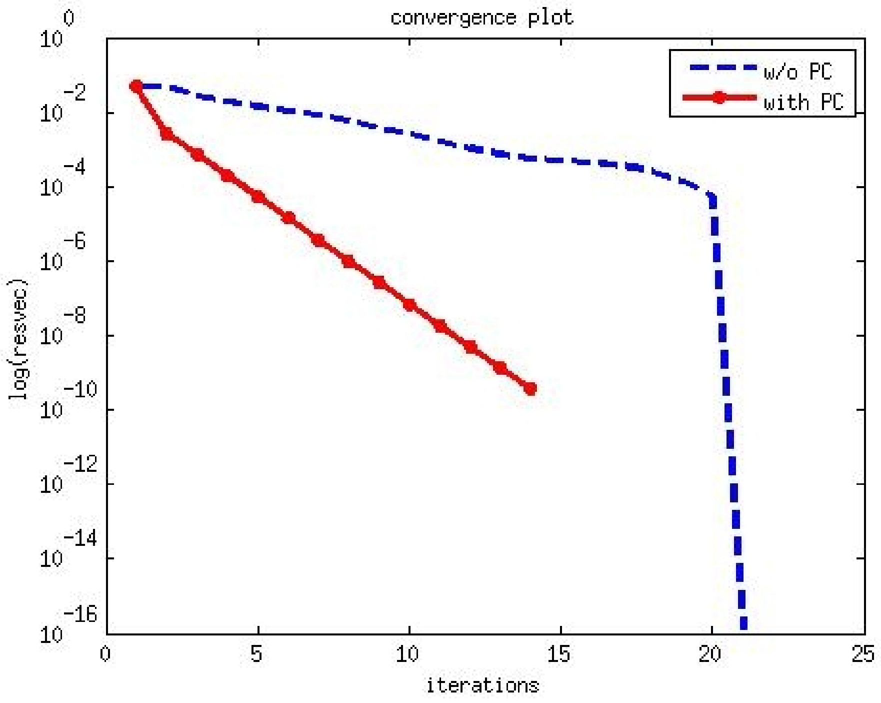


Fig 5.2 Convergence Plot for Source Term f=cos(2pix)sin(2piy)

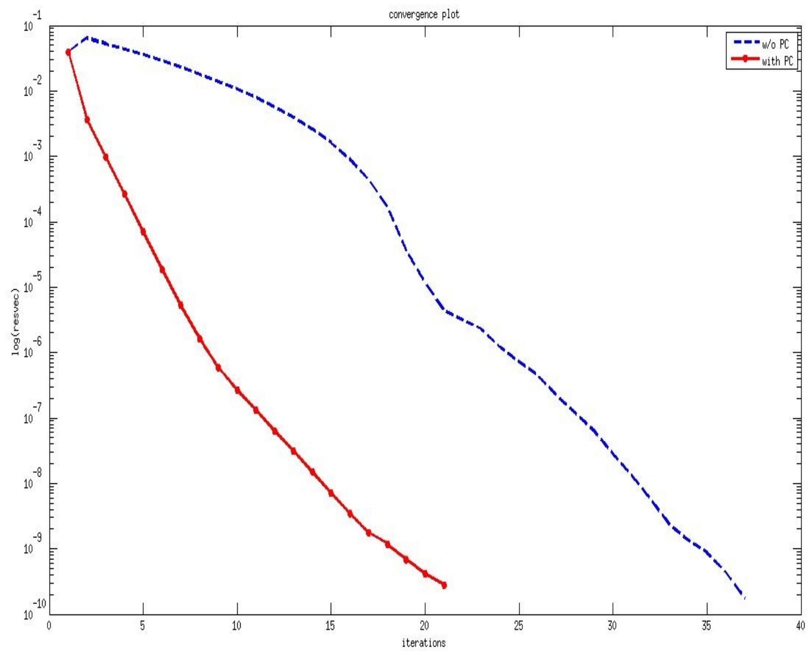


Fig 5.3 Convergence plot Source Term $f=x^2-y^2$

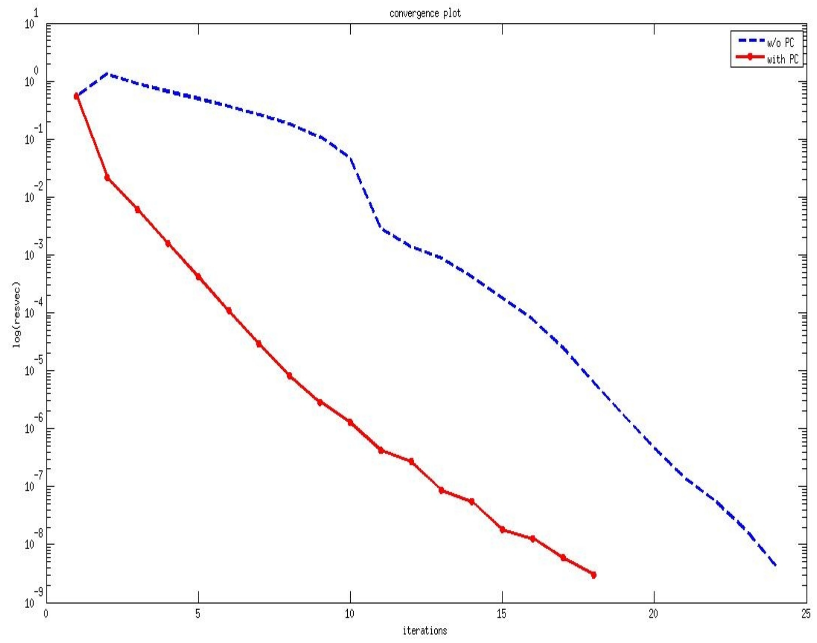


Fig 5.4 Convergence plot Source Term $f=2(2-x^2-y^2)$

The conclusions from the tables and plots are discussed in the next chapter.

Chapter 6

Conclusions and Outlook

In this chapter ,the conclusions from the numerical tests and the outlook of the work is are presented.

6.1 Conclusions

The prime objective of the current work is to study and the implementation of the PGD method and further use it to determine the preconditioner to solve the system of equations that are generated from Finite Element Method. Particularly, the method is applied for scaler elliptic problems. In this regard, all the concerned coding is done in MATLAB and the results are cross checked by using Finite Element Method (Direct method to solve the equations).

Initially, the PGD method is implemented. In the chapter 3, the effect of varying parameters such as enrichment tolerance, iterative tolerance, mesh size, initial value (value that is assumed at the start of the enrichment process) are presented. In the 4th chapter, the effect of change in the number of basis functions on the tolerances reached, convergence rate pertaining to that are studied. Finally, for the PGD conditioner the following conclusions can be made from the Numerical tests.

It can be seen from all the tables that the number of iterations taken when the PGD preconditioner is used in comparison with the method when the preconditioner is not used are much lesser. In addition, in all the cases the convergence plot of the method when the PGD preconditioner is used is much steeper than the one without the preconditioner. The idea of the thesis is to take the advantages of both the methods. It is known, that the Finite Element method being one of the robust method and it can be used for any kind of problem. On the other hand, the PGD method is very fast, but the limitation of it being not applicable to few problems. For instance, with a source term or a diffusion term where they cannot be written in a separated form, the PGD is difficult to implement.

As the grid points are increased the number of iterations taken by the PCG without preconditioner is much greater than with the PGD preconditioner. Thus, for a finer and greater accuracy the latter is better.

In case of the diffusion term, the method is robust even with the change in the coefficient.

On a general note, it can be inferred that coupling the FEM and the PGD methods give results in lesser number of computations than the conventional methods .

6.2 Outlook

In this section few of the possibilities of future work in this line of research are outlined

1. It is observed that, as the mesh size is increased ,the number of iterations that are taken are increased. In the present implementation, the mesh size for the Finite elements and the PGD are taken the same. As discussed earlier (section 4.3) the steps involving the projection of the Finite Element residual on to the PGD mesh and after computing the preconditioned residual, projecting back them on to the Finite Element mesh are eliminated (3rd and 5th steps discussed in the implementation in the section 4.3). By using a coarser mesh for the PGD and including these projection stages the computational cost can be further decreased and also the convergence can be faster. (multi grid implementation)

2. The second possibility would be development of other method instead of Alternate Direction Strategy (section 3.2) for iterations. For instance , Newton method can be used which is faster and takes less iterations. This is discussed in the paper [2].

3. The presented method can be tested with other existing preconditioning techniques.

4. The method can be tested for cases where in the source term and diffusion terms are discontinuous

Bibliography

[1] Zienkiewicz , O. C. , 1987. *The Finite Element Method: Basic Formulation and Linear Problems* (6th ed.) .Oxford, UK : Butterworth-Heinemann.

[2] Chinesta, Francisco, Keunings, Roland, Leygue, Adrien. ,2014 .*The Proper Generalized Decomposition for Advanced Numerical Simulations.* SpringerBriefs in Applied Sciences and Technology

[3] D. Gonzalez, A. Ammar, F. Chinesta, E. Cueto, *Recent advances in the use of separated representations*. Int. J. Numer. Meth. Eng. 81/5, 637–659 (2010)

[4] A. Ammar, B. Mokdad, F. Chinesta, and R. Keunings. *A new family of solvers for some classes of multidimensional partial differential equations encountered in kinetic theory modeling of complex fluids*. J. Non-Newtonian Fluid Mech., 139:153–176, 2006.

[5] Axelsson, Owe (1996). *Iterative Solution Methods*. Cambridge University Press. p. 6722.

[6] Saad, Y (1996). *Iterative Methods for Sparse Linear Systems,* WPS .

[7] Jean Donea, Antonio Huerta, .(2003) *Finite Element Methods for Flow Problems,* John Wiley & Sons

[8] Fabien Poulhaon , Francisco Chinesta & Adrien Leygue (2012) *A first step toward a PGD-based time parallelisation strategy*, European Journal of Computational Mechanics/ Revue Européenne de Mécanique Numérique, 21:3-6, 300-311

[9] D. Ryckelynck, F. Chinesta, E. Cueto, A. Ammar (2006) *On the a priori Model Reduction: Overview and Recent Developments,* Arch. Comput. Meth. Engng. Vol. 13, 1, 91-128 (2006)

[10] Amine Ammar , Etienne Pruliere , Julien Férec , Francisco Chinesta & Elias Cueto (2009) *Coupling finite elements and reduced approximation bases*, European Journal of Computational Mechanics/Revue Européenne de Mécanique Numérique, 18:5-6, 445-463