Deflation of the Finite Pointset Method

Sara Pålsson

Master's thesis
2014:E37

Lund University

Faculty of Engineering
Centre for Mathematical Sciences
Numerical Analysis

Master's Thesis

Deflation of the Finite Pointset Method

June 2014

Author:
Sara Pålsson

Supervisors:
Dr. Jörg Kuhnert
Prof. Gustaf Söderlind

Fraunhofer ITWM

Lunds universitet
Lunds Tekniska Högskola

Technische Universität Kaiserslautern

Abstract

In this thesis a deflation method for the Finite Pointset Method (FPM) is presented. FPM is a particle method based on Lagrangian coordinates to solve problems in fluid dynamics. A strong formulation of the occuring differential equations is produced by FPM, and the linear system of equations obtained by an implicit approach is solved by an iterative method such as BiCGSTAB. To improve the convergence rate of BiCGSTAB, the computational domain is divided into a number of deflation cells and a projection between the deflated domain and the original domain is constructed with the help of different ansatz functions, either constant, linear or quadratic. Also, the Moore-Penrose pseudoinverse of the projection is computed. Applying the projection and restriction to the linear FPM system, a deflated system is obtained which can easily be solved with a direct method. The deflated solution is then projected onto the full domain.

The deflation is tested for a number of test cases in one and two dimensions. Constant ansatz functions provide acceptable results for Dirichlet problems, but give big errors when deflating problems with mixed boundary conditions. Linear ansatz functions provide good approximations which converge to the exact solution as the number of deflation cells increases. Quadratic ansatz functions provide deflated solutions as good as the exact solutions for all test cases but are computationally expensive. The BiCGSTAB convergence rate is improved when using a deflated solution as initial guess, compared to using the zero-vector. The size of the improvement varies between which ansatz functions are used.

Overall, the proposed method provides an increased convergence rate in the BiCGSTAB algorithm for FPM. However, the computational effort in the deflation process should also be taken into account.

Populärvetenskaplig sammanfattning

Deflation av Finite Pointset metoden

Sara Pålsson
Handledare: Dr. Jörg Kuhnert[1],
Prof. Gustaf Söderlind[2]

Introduktion

Att simulera olika sorters flöden är ett stort område som täcker allt från hur man kan förutsäga väder till hur man förbättrar motorer. Flödesberäkningar görs i två steg, det första är att bygga matematiska modeller som beskriver flödet man är intresserad av. Dessa modeller består oftast av ett antal ekvationer som beskriver hur flödesinformation såsom tryck och hastighet ändras över tid beroende på placering i tillämpningens geometri. Det andra steget är att försöka lösa dessa ekvationer. Det gör man oftast genom att diskretisera dem, det vill säga placera ut punkter över geometrin där man utför beräkningarna. Man får då ett stort system av ekvationer att lösa. För att göra detta finns många effektiva algoritmer utvecklade. Ju fler punkter man vill använda, d.v.s. ju högre precision man vill att ens lösning ska ha, desto längre tid tar det.

Ett exempel på en metod för att beräkna flöden är Finite Pointset Method (FPM). I FPM fyller man sin geometri med punkter som beter sig som flödespartiklar och förflyttar sig med flödet, det vill säga lite som att följa vattendroppar i en flod. Lösningen tas fram med en vedertagen metod vid namn BiCGSTAB. Denna metod startas med en initialgissning, som här är noll. I många fall är detta en effektiv metod, men för vissa fall tar det väldigt lång tid att komma fram till ett bra resultat. Målet med det här examensarbetet är därför att utveckla en snabb metod för att få fram en grov gissning av lösningen som kan ersätta noll som initialgissning.

Metod

I ett första steg delas flödesgeometrin upp i ett givet antal delar, så kallade deflationsceller, som man räknar på istället för de ursprungliga punkterna. Man kan likna det vid att man istället för att räkna på varje vattendroppe räknar på en deciliter vatten i taget. Det ger ett mycket mindre system att lösa än det man började med och går snabbt att lösa med dagens metoder. Utifrån lösningen av det mindre systemet använder vi olika testfunktioner, såsom konstanta och linjära, för att överföra den till alla punkterna. I det konstanta fallet innebär detta att alla punkter i en deflationscell får samma värde. Att använda en linjär testfunktion innebär att man i varje deflationscell skapar en rak linje utifrån lösningen av det mindre systemet, som sedan används för att bestämma värdet på alla punkter i cellen. Således skapas över hela geometrin en gissning på vilka värden punkterna borde ha. Denna gissning används slutligen som initialgissning i BiCGSTAB. Detta är så kallad deflation.

Utvärdering

För att undersöka hur väl metoden fungerar har den testats på ett antal olika fall som kan uppkomma i flödesberäkningar, i både en och två dimensioner. Extra intresserant är att räkna ut trycket i långa utsträckta geometrier, som t.ex. rör, eftersom det är där den ursprungliga metoden med noll som initialgissning tar längst tid. Det visar sig att det för konstanta testfunktioner knappt lönar sig att använda deflationsmetoden, medan linjära testfunktioner i de flesta fall förbättrar beräkningstiden rejält - från upp till 10 000 beräkningar för att nå lösningen till en eller två.

[1]Fraunhofer ITWM, Kaiserslautern, Tyskland
[2]Lunds Tekniska Högskola, LTH

Acknowledgements

This is a master's thesis in Engineering Mathematcs at Lund University, and Mathematics International at Technische Universität Kaiserslautern. It has been carried out from January to June in 2014 at Fraunhofer ITWM in Kaiserslautern, Germany.

First, I wish to thank Fraunhofer ITWM and especially the department of grid-free methods for welcoming me and for all help and support they have given me. Many thanks to Dr. Jörg Kuhnert and Dr. Isabel Ostermann for supervising me and giving me direction. Also, I want to thank Dr. Elisa Röhrig for introducing me to Fraunhofer in the first place. Second, thank you to Prof. Gustaf Söderlind at Lund University who has given me support and encouragement along the way.

Last but not least, a big thank you to my family and friends who have helped me, listened when I needed them to and provided me with many good ideas and suggestions. Without you this would not have been possible.

Abbreviations and Notations

Abbreviations

BiCGSTAB	Biconjugate Gradient Stabilised Method
CG	Conjugate Gradient Method
FDM	Finite Difference Method
FEM	Finite Element Method
FPM	Finite Pointset Method
FVM	Finite Volume Method
GMRES	Generalised Minimal Residual Method
PDE	Partial Differential Equation
SPH	Smoothed Particle Hydrodynamics
SOR	Successive Over-Relaxation Method

Notations

Δ	Laplace operator
Ω	Computational domain
$\delta\Omega$	Boundary of computational domain
$\delta\Omega_D$	Boundary with Dirichlet boundary condition
$\delta\Omega_N$	Boundary with Neumann boundary condition
N	Number of particles / grid points
h	Smoothing length
η	Constant in FPM for the Gaussian weight function
$A^\dagger$	Moore-Penrose pseudoinverse of A
A^*	Hermitian transpose of A
M	Number of deflation subdomains

Contents

Abstract . . . iii
Acknowledgements . . . vii

Abbreviations and Notations **ix**

Table of Contents **xi**

1 Introduction **1**
1.1 Aim of Thesis . . . 1
1.2 Related Work . . . 2
1.3 Overview . . . 3

2 Approach **5**
2.1 Theory . . . 5
2.1.1 Finite Difference Method . . . 6
2.1.2 Finite Pointset Method . . . 6
2.1.3 Pseudoinverses . . . 8
2.2 Method . . . 9
2.2.1 Construction of P and $P^{\dagger}$ . . . 10
2.2.2 Varieties . . . 13
2.2.3 Evaluation . . . 13

3 Results **15**
3.1 Results in 1D . . . 15
3.1.1 Constant Approximations . . . 15
3.1.2 Linear Approximations . . . 19
3.1.3 Quadratic Approximations . . . 23
3.2 Results in 2D . . . 28
3.2.1 Constant Approximations . . . 28
3.2.2 Linear Approximations . . . 32

4 Discussion and Conclusions **37**
4.1 Discussion . . . 37
4.1.1 Constant Approximations . . . 37
4.1.2 Linear Approximation . . . 38
4.1.3 Quadratic Approximation . . . 38
4.1.4 Comparisons . . . 38
4.1.5 Future Work . . . 39
4.2 Conclusions . . . 39

Bibliography **41**

A Further Results for FPM **43**

B Results for FDM **55**

Chapter 1

Introduction

The modelling of fluids and fluid flows is an immense area which covers everything from the simulation of airbag deployment to the analysis of combustion engines. Computational fluid dynamics consists of two main parts. The first is to model the flows, obtaining equations that describe the physical phenomena. The second is to develop numerical algorithms that solve these equations. Fast, robust algorithms are needed, solving the equations by discretising them and forming linear systems. There are many different ways of discretising a partial differential equation (PDE), among the most common are the finite difference (FDM), the finite volume (FVM) and the finite element (FEM) methods. For fluid dynamics a number of particle methods have also been developed - one of them being the Finite Pointset Method (FPM).

One might ask why the notion of particle methods is interesting, with FVM and FEM being as well-developed as they are. For many problems, these two methods are all one needs but there are situations where they prove to be inefficient. Both FVM and FEM use geometrical grids that can be expensive to set up. For applications where the geometry does not change this is only done once and the method is efficient. On the other hand, with flow problems where the domain changes often, or where free-surface flows occur, the grid needs to be adapted on the fly - which is computationally expensive. In such instances, a Lagrangian particle method may be preferable. In a Lagrangian method the particles move within the fluid making the computational geometry self-adapting to flow movements.

Different discretisations generate different linear systems to solve numerically. Depending on each system's properties, e.g. symmetry, positive definiteness, sparseness and size, different methods, either direct or iterative, can be employed to find a solution. When solving a system, computational effort and convergence rate should always be taken into account. To speed up an algorithm, and to improve convergence, one can apply a number of different techniques such as preconditioners, deflation or multigrid solvers. Generally, deflation is a method to decompose the approximation space of a linear system into complementary subspaces. This gives a projected, or deflated, system of a smaller dimension which is much easier to solve than the original problem [3]. In this thesis the focus lies on improving the convergence rate and speed of FPM by deflation.

1.1 Aim of Thesis

The aim of this master's thesis is to develop a method to deflate FPM with the help of a variety of ansatz functions. The solution obtained should be able to serve as a good initial guess in the final iteration of FPM, during which the obtained discretised linear system is solved with BiCGSTAB.

The method should work for solving the Poisson equation with different sources and boundary conditions. Especially, it should be able to handle long and narrow geometries. Our method should work in a MATLAB implementation in one and two dimensions. For certain problems, e.g. the Poisson equation with mixed boundary conditions in narrow geometries, the convergence of BiCGSTAB is slow. Therefore, the change in convergence rate will be investigated.

1.2 Related Work

One of the first grid-free techniques proposed was the Smoothed Particle Hydrodynamics (SPH) which was first used to solve problems without boundaries in astrophysics [5], [8]. It uses a Lagrangian description of a fluid flow, where the mass points are carriers of all information needed. The approximation is based upon kernel estimation through an integral interpolant [10, p. 1709]. Lots of work on how this kernel should be decided has been done, and most of the resulting kernels have compact support [10]. Any desired quantity can then be interpolated as a summation over all particles' quantities multiplied with the kernel [10, p. 1710]. As the kernel has compact support, in practice this summation will only be performed over the neighbours of the particle, within a selected distance. The derivatives are approximated by differentiating the interpolant [10, p. 1712]. Originally, SPH was developed for problems without boundary conditions. Later on, ideas on how to properly add boundary conditions were explored [10, p. 1747]. Two of the more popular ideas are to modify the kernel so that it is not defined outside of the boundary of the domain, or to use boundary potentials that neutralise the errors at the boundary [7, p. 19-21].

Later, an extension of SPH was presented in [7], called General SPH. This method especially tries to perform better approximations of the boundary elements. Instead of using symmetric smoothing kernels as in SPH, this method uses (weighted) least squares approximations to obtain the quantities on each particle. The boundary conditions are incorporated by the placing of boundary particles which may move with the fluid along the boundary, considering the characteristic curves of the equations. This method has been investigated as an iterative method approximating first and second order derivatives, when solving compressible viscous flows [17]. Furthermore, it solves the Poisson equation with weighted least squares iteratively in [19]. When extending Chorin's projection method to this grid-free framework, we can solve incompressible flows by solving at least one Poisson equation per time-step, which gives the FPM [18]. As can be seen in [20], it has also been used to model two-phase flows. An overview of FPM can be found in Section 2.1.2. There, we describe how the approximated operator stencils are obtained. Finally, the linear system set up by FPM is solved by an appropriate solver. Which solver is used depends on the properties of the approximated system, but usually the BiCGSTAB method, Gauss-Seidel or SOR are used [4], [20].

There are different approaches to speed up the convergence of FPM. One option is to consider the stencils set up by the method. A finite difference approximation of an operator, e.g. the Laplace operator, will always give a positive stencil (all neighbours will have positive weights). This is not the case with FPM. In [15], a modified FPM is presented with only positive stencils. There, positivity of the stencils is enforced using a linear minimization. The resulting linear system matrices will be sparser, thus increasing the computational speed for solving the system. This compensates for the computational effort of setting up the matrix.

Another way of speeding up the computations is to use parallel computing. This is done in [16], where the point cloud in FPM is distributed between processors. All processors are supposed to have roughly the same number of points. This distribution is done by adaptive load balancing, where each processor represents a subdomain of points. These are selected by recursive coordinate bisection, with the aim of finding bisections that yield minimal interfaces between the processors. The system is then solved using a parallel version of BiCGSTAB. It is observed that the division of the particles into subdomains is fastest if the number of subdomains (thus processors) is a power of two.

The solving of linear systems has been subject to extensive research, resulting in methods such as the conjugate gradient (CG) algorithm, BiCGSTAB etc. Often, when trying to improve convergence and speed this is the step that is altered, e.g. with preconditioners or deflating techniques. Such techniques are specific to which system one wants to solve and which algorithm one chooses to use. Most work has been done investigating the deflation of the CG algorithm. For this, the general idea is to decrease the condition number as this directly affects the convergence rate of the algorithm [14, p. 215].

An early example of how to improve the conjugate gradient algorithm with the help of deflation can be seen in [12]. There, the deflation method can be used without additional preconditioning and still generate an increase in efficiency. Moreover, it can be used together with standard preconditioners for even better results. The main focus of this study is elliptic second order problems and the deflation of slowly varying parts of the residual. Overall, this has proved to be a low cost scheme. The deflation of iterative algorithms such as the conjugate gradient algorithm has mainly been investigated for the linear systems originating from the finite element approximations. In [9] the previously mentioned deflating technique has been studied for such systems and proved to be very effective, both for the Stokes problem for incompressible fluid flows and bending of cantilever beams.

Whether using a preconditioner is preferable to deflation is investigated in [11]. There, the deflating technique for the conjugate gradient algorithm is compared to balancing preconditioning. The results show that deflating the system will generate faster results.

In [1], deflation of the BiCGSTAB algorithm is investigated. There, the authors alternate between a minimal residual projection over the right eigenvectors and cycles of BiCGSTAB. The first step is solved using GMRES-DR.

1.3 Overview

In this thesis, Chapter 2 starts with a theoretical background of numerical analysis and numerical linear algebra. Also, an overview of FPM is presented. It continues with describing the method developed for our deflation and how the different approximations are constructed. In Chapter 3 results of the deflation in 1D and 2D are presented for FPM systems with varying boundary conditions. Finally, Chapter 4 contains a discussion on the results, presents an outline for future work and concludes the thesis. In Appendix B results for the deflation of FDM are presented.

Chapter 2

Approach

In this chapter we describe the particle method used in this thesis (FPM), some mathematical theory and our deflation method. The equation to solve is the Poisson equation

$$\Delta u = f \ \text{ in } \ \Omega \in \mathbb{R}^k, \ \ k = 1, 2, 3, \tag{2.1}$$

with either Dirichlet or mixed boundary conditions.

2.1 Theory

In this section we describe the main ideas for FDM and FPM with regard to the approximation of the Laplace operator. An equation such as Eq. 2.1 appears in a multitude of problems, e.g. the pressure Poisson equation that is used for solving the Navier-Stokes equations [4]. Moreover, we mention some of the theory regarding pseudoinverses.

First of all, there are two approaches to regard flow problems. The first is the Eulerian way, where one regards the flow from a global perspective and the coordinates remain fixed in that frame of reference. The other is the Lagrangian way, where one regards the flow from "within", and the coordinates travel along with the fluid particles [13]. The FDM as presented in Section 2.1.1 is an example for Eulerian coordinates. There are also finite difference methods for Lagrangian coordinates, though these are outside the scope of this thesis. FPM uses Lagrangian coordinates.

Second, the family of partial differential equations (PDEs) is vast and consists of many different problems. Various ways of characterising them exist - e.g. elliptic, parabolic and hyperbolic PDEs - and how to solve them varies between these groupings. In this thesis we are focusing on the elliptic equation seen in Eq. 2.1. Here, Δ stands for the Laplace operator, thus (in two dimensions)

$$\Delta u = \frac{\delta^2 u}{\delta x^2} + \frac{\delta^2 u}{\delta y^2},$$

for $(x, y) \in \Omega \subset \mathbb{R}^2$, where Ω is bounded and connected with a piecewise smooth boundary and $f \equiv f(x, y)$ is a known function. Together with this we have boundary conditions. In this thesis, only Dirichlet boundary conditions, i.e.

$$u(x, y) = g_D(x, y), \ \ \forall (x, y) \in \delta\Omega,$$

and mixed boundary conditions

$$\begin{aligned} u(x, y) &= g_D(x, y), \ \ \forall (x, y) \in \delta\Omega_D, \\ \frac{\delta u}{\delta \vec{n}} &= g_N(x, y), \ \ \forall (x, y) \in \delta\Omega_N, \end{aligned}$$

are considered.

2.1.1 Finite Difference Method

The main idea of the FDM is to replace every derivative with a linear combination of discrete values from the grid. A good introduction to FDM is given in [6]. Here, we only give the most important results.

In one dimension, $\Omega = [x_0, x_L] \in \mathbb{R}$, second order derivatives are usually approximated by central differences, thus at an inner grid point j we have

$$\frac{\delta^2 u}{\delta x^2} \approx \frac{u_{j+1} - 2u_j + u_{j-1}}{dx^2} \quad \text{at} \quad x_j \in \Omega \setminus \delta\Omega,$$

where dx denotes the distance between two adjacent points. On a uniform grid, all distances between neighbouring points are the same. For N inner grid points, this gives the approximated system matrix A of size $N \times N$;

$$A = \frac{1}{dx^2} \begin{pmatrix} -2 & 1 & 0 & \dots & 0 & 0 & 0 \\ 1 & -2 & 1 & \dots & 0 & 0 & 0 \\ 0 & 1 & -2 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & -2 & 1 & 0 \\ 0 & 0 & 0 & \dots & 1 & -2 & 1 \\ 0 & 0 & 0 & \dots & 0 & 1 & -2 \end{pmatrix}. \tag{2.2}$$

Dirichlet boundary conditions can be incorporated on the right hand side of the equation, thus

$$b = \begin{pmatrix} f_1 - \frac{1}{dx^2} u_0 \\ f_2 \\ \vdots \\ f_{N-1} \\ f_N - \frac{1}{dx^2} u_N \end{pmatrix}, \tag{2.3}$$

where $u(x_0) = u_0$ and $u(x_L) = u_N$. This gives us a sparse, positive definite and symmetric system $A\tilde{u} = b$ to compute $\tilde{u} = [u_1\ u_2\ \dots\ u_N]^T$, which are the values of our function u at the grid points. If we instead have a Neumann boundary condition, it will be approximated either by a forward difference,

$$\frac{\delta u}{\delta x} \approx \frac{u_{j+1} - u_j}{dx}, \quad x_j \in \delta\Omega_N,$$

or by a backward difference

$$\frac{\delta u}{\delta x} \approx \frac{u_j - u_{j-1}}{dx}, \quad x_j \in \delta\Omega_N,$$

and incorporated into Eq. 2.2 and Eq. 2.3.

In two dimensions, the common five-point stencil to approximate the Laplace operator at point (i, j) is

$$\Delta u_{i,j} \approx \frac{u_{i-1,j} + u_{i+1,j} - 4u_{i,j} + u_{i,j-1} + u_{i,j+1}}{dx^2} \quad \text{at} \quad (x, y) \in \Omega \setminus \delta\Omega,$$

for a uniform grid, i.e. $dx = dy$. The boundary conditions are treated in a similar fashion to the one dimensional case.

2.1.2 Finite Pointset Method

FPM uses a point cloud on which the computations are performed. The points serve as the numerical "grid" of the method, are treated as carriers of information (pressure, velocity etc.) and can move with time over the flow geometry. The method has been described in many papers, see e.g. [4], [7], [18], [20]. Here only a short overview is given. The general idea of FPM is to model a differential equation by direct approximations of the operators, based on each of the particles and its neighbours. This gives us a strong formulation of our equations.

First, the domain is filled with N particles. Particle k has access to its position, $\vec{x}_k$, as well as, in the case of a non-stationary problem, initial values for all fluid information such as density, velocity and pressure. Additionally, a smoothing length h is defined. The neighbours of a particle $\vec{x}_k$ are defined as all particles within the ball $B(\vec{x}_k, h)$.

Second, consider the set of N particles. On each of the particles, the differential operator is constructed. For this, the particle itself and its neighbours are used. In order for the particles closer to the selected particle $\vec{x}_k$ to influence its properties more, a weight function is defined as

$$w_k = w(\vec{x}_k, h) = \begin{cases} \exp\left(-\eta \dfrac{\|\vec{x}_k - \vec{x}\|^2}{h^2}\right), \text{ if } \dfrac{\|\vec{x}_k - \vec{x}\|}{h} \leq 1, \\ 0, \text{ else,} \end{cases} \tag{2.4}$$

where h is the smoothing length as already mentioned and η a positive constant, with an optimal value between 2 and 6. Then, shape functions ϕ_i are constructed such that

$$\Delta u \approx \sum_{i=1}^{m} \phi_i u_i =: \tilde{\Delta} u, \tag{2.5}$$

where m is the number of neighbours for particle k. A Taylor series expansion of $u(\vec{x}_k)$ around an arbitrary neighbouring particle $\vec{x}$ then gives an approximation of the Laplace operator

$$\Delta u = \sum_{i=1}^{m} \phi_i u_i + O(\|d\vec{x}_i\|^3) = \tilde{\Delta} u + O(\|d\vec{x}_i\|^3),$$

with the constraints

$$\begin{aligned} &\sum_{i=1}^{m} \phi_i dx_i = 0, && \sum_{i=1}^{m} \phi_i dy_i = 0, && \sum_{i=1}^{m} \phi_i dz_i = 0, \\ &\sum_{i=1}^{m} \phi_i dx_i dy_i = 0, && \sum_{i=1}^{m} \phi_i dx_i dz_i = 0, && \sum_{i=1}^{m} \phi_i dy_i dz_i = 0, \\ &\text{as well as} \\ &\sum_{i=1}^{m} \phi_i = 0, && \sum_{i=1}^{m} \phi_i dx_i^2 = 2, \\ &\sum_{i=1}^{m} \phi_i dy_i^2 = 2 \quad \text{and} && \sum_{i=1}^{m} \phi_i dz_i^2 = 2. \end{aligned} \tag{2.6}$$

The constraints are denoted by G_1 to G_{10}. The distance between particle $\vec{x}_k$ and neighbouring particle $\vec{x}_i$ is defined as

$$[dx_i \; dy_i \; dz_i]^T = \vec{x}_k - \vec{x}_i.$$

To decide the Laplace stencil with ϕ_i for each particle $\vec{x}_k$ the functional

$$J = \sum_{i=1}^{m} \frac{\phi_i^2}{w_i} \tag{2.7}$$

is minimized with respect to the constraints in Eq. 2.6. This is equivalent to

$$\frac{\delta J}{\delta \phi_i} + \sum_{j=1}^{10} \lambda_j \frac{\delta G_j}{\delta \phi_i} = 0, \quad i = 1, \ldots, m,$$

where λ_j are the Lagrange multipliers. The weighted least squares minimization then gives

$$\bar{a} = (M^T W M)^{-1}(M^T W)\bar{b}, \tag{2.8}$$

where

$$\bar{a} = \begin{pmatrix}\lambda_1 & \lambda_2 & \dots & \lambda_{10}\end{pmatrix}^T, \qquad W = \operatorname{diag}[w_1, \dots, w_m],$$

$$M = \begin{pmatrix} 1 & dx_1 & dy_1 & dz_1 & dx_1^2 & dx_1dy_1 & dx_1dz_1 & dy_1^2 & dy_1dz_1 & dz_1^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & dx_m & dy_m & dz_m & dx_m^2 & dx_mdy_m & dx_mdz_m & dy_m^2 & dy_mdz_m & dz_m^2 \end{pmatrix},$$

and

$$b = -2\begin{pmatrix}\frac{\phi_1}{w_1} & \frac{\phi_2}{w_2} & \dots & \frac{\phi_m}{w_m}\end{pmatrix}^T.$$

Now, the stencil can be obtained by Eq. 2.9,

$$\begin{aligned}\phi_i = -\frac{w_i}{2}(\lambda_1 + dx_i\lambda_2 + dy_i\lambda_3 + dz_i\lambda_4 + dx_i^2\lambda_5 + dx_idy_i\lambda_6 + \\ + dx_idz_i\lambda_7 + dy_i^2\lambda_8 + dy_idz_i\lambda_9 + dz_i^2\lambda_{10}).\end{aligned} \tag{2.9}$$

Inserting all functions ϕ_i for all particles k in a matrix, Ψ, gives the discretised system

$$\Psi u = f. \tag{2.10}$$

In Eq. 2.10, Ψ is the matrix representing the discretised operator. It covers all particles in the domain, also those on the boundary, thus the boundary conditions are incorporated directly. This gives us a system that is not symmetric, nor necessarily positive definite.

This method works for uniformly as well as randomly distributed particles. The bandwidth of the matrix Ψ depends on the choice of the smoothing length h. For uniformly distributed particles, and $h < 2dx$, this system will be close to the system obtained for a five-point stencil in FDM.

2.1.3 Pseudoinverses

There are many matrices for which we cannot calculate inverses. These might be singular matrices, or simply matrices that are not square. One might still want to calculate a more generalised inverse, called a pseudoinverse. Here follows an overview of pseudoinverses. For more detailed information, see [2]. A generalised inverse of a matrix is defined as follows.

Definition 1. *Let m, $n \in \mathbb{N}$ and $A \in \mathbb{C}^{m\times n}$. A matrix $B \in \mathbb{C}^{n\times m}$ is said to be a generalised inverse, or a pseudoinverse, of A if it satisfies the conditions*

- $ABA = A$,
- $BAB = B$.

It is worth noting that if A is non-singular and square, then A^{-1} will satisfy the properties of Def. 1. This definition is quite wide and will cover many options for pseudoinverses. Therefore, a more specific pseudoinverse is introduced, the Moore-Penrose pseudoinverse.

Definition 2. *Let m, $n \in \mathbb{N}$ and $A \in \mathbb{C}^{m\times n}$. A matrix $A^\dagger \in \mathbb{C}^{n\times m}$ is said to be a Moore-Penrose pseudoinverse of A if it satisfies the following conditions:*

- $AA^\dagger A = A$,
- $A^\dagger AA^\dagger = A^\dagger$,
- $AA^\dagger \in \mathbb{C}^{m\times m}$ *and* $A^\dagger A \in \mathbb{C}^{n\times n}$ *are self-adjoint.*

Both uniqueness and existence of the Moore-Penrose pseudoinverse can be shown [2, p. 4-5]. The Moore-Penrose pseudoinverse is useful when trying to solve the linear equation

$$Ax = y,\ A \in \mathbb{C}^{m\times n},\ x \in \mathbb{C}^n. \tag{2.11}$$

If A is singular, or not square, the simple solution $x = A^{-1}y$ does not exist. In that case there might be no solution at all, or it is not unique. The Moore-Penrose pseudoinverse can be used to find the $x' \in \mathbb{C}^n$ that minimizes $\|Ax' - y\|_2$. These vectors x' make out what is called the minimizing set of the linear problem in Eq. 2.11. This set of vectors is provided by the Moore-Penrose pseudoinverse as can be seen in Thm. 1.

Theorem 1. *Let $A \in \mathbb{C}^{m\times n}$ and $y \in \mathbb{C}^m$ be given. Then, the set of all vectors in $\mathbb{C}^n$ for which the map $\mathbb{C}^n \ni x \mapsto \|Ax - y\|_2 \in [0, \inf)$ assumes a minimum coincides with the set*

$$A^\dagger y + \ker(A) = \{A^\dagger y + (\mathbb{1}_n - A^\dagger A)z, z \in \mathbb{C}^n\}.$$

A proof of Thm. 1 can be found in [2]. The Moore-Penrose pseudoinverse can be written as

$$A^\dagger = A^* \left(AA^*\right)^{-1},$$

if AA^* is invertible, or alternatively

$$A^\dagger = \left(A^*A\right)^{-1} A^*,$$

if A^*A is invertible, where A^* is the Hermitian transpose of A.

2.2 Method

Our deflation method consists of a number of steps. First, the differential operator as in Eq. 2.1 is discretised either by FDM or FPM. This gives us the linear system

$$Au = b, \tag{2.12}$$

where b represents the right hand side of our differential equation (i.e. the sources), u the desired quantity on each particle and A the stencil for the differential operator. We note that depending on how our discretisation is made, and which boundary conditions are imposed, our matrix A will have different properties. In general, we cannot assume either symmetry or positive definiteness.

Investigating the condition number of the system matrices shows that it can be vastly improved. Therefore, as a second step, we scale our system to try to decrease it. Different techniques for this are mentioned in Section 2.2.2.

Third, we decompose our domain into a given number, M, of subdomains or deflation cells. On each of these subdomains we project our solution given different ansatz functions. How this decomposition is done may vary. In one and two dimensions, the domain is simply divided into M equally large subdomains, independent of how the particles are spread out. For three dimensions, the idea would be to perform a recursive coordinate bisection, as described in [16]. In either dimension, no special thought is spent on the boundary particles.

The next step is to deflate our system to something much smaller, preferably on which the solution can be obtained by a direct method. This is done by restricting the particles on a given subdomain to a predetermined number of "new" particles. How these are placed - and how many there are - is decided by which sort of ansatz function we want to use. This is done by

$$\bar{u} = P^\dagger u \Leftrightarrow u = P\bar{u}.$$

We define the operator $P^\dagger$ as a the Moore-Penrose pseudoinverse of P.

Now, the construction of our deflated system is done by using the projection P on the deflated vector $\bar{u}$ and restricting our full system matrix and right-hand side by $P^\dagger$. Thus, we obtain the following system

$$P^\dagger A P \bar{u} = P^\dagger b \Leftrightarrow \bar{A}\bar{u} = \bar{b}. \tag{2.13}$$

The system in Eq. 2.13 has a smaller dimension than the original system. How big the change in size is will depend on which approximations are used for constructing P. From Eq. 2.13, we obtain the deflated solution $\bar{u}$ by using a direct method. Finally, the approximated solution $\ddot{u}$ is produced by $\ddot{u} = P\bar{u}$.

All in all, the algorithm can be summarised as in Alg. 1.

```
**Data**: Boundary conditions, domain information and RHS $f$
**Result**: Deflated solution $\bar{u}$ and approximated solution $\ddot{u}$
Decompose domain into $M$ subdomains;
Discretise operator either by FDM or FPM, to obtain system matrix $A$;
Scale $A$ to decrease condition number;
Construct $P$, either by constant, linear or quadratic approximations;
Construct $P^\dagger$ as the Moore-Penrose pseudoinverse of $P$;
Deflate system by $P^\dagger A P \bar{u} = P^\dagger b$;
Solve deflated system with respect to $\bar{u}$;
Approximate solution $\ddot{u} = P\bar{u}$;
```

Algorithm 1: Deflation Algorithm

2.2.1 Construction of P and $P^\dagger$

We construct P according to which ansatz functions we are interested in using. The idea is to take the information in each subdomain of the deflation and map that information to the particles in the original full domain. By construction, P will map our deflated values to an approximated solution,

$$\ddot{u} = P\bar{u}.$$

Then, $P^\dagger$ is constructed as the Moore-Penrose pseudoinverse of P.

Constant Approximations

In one as well as two dimensions, when approximating constant functions on the subdomains of the deflation each subdomain needs only one computational point. The value of $\bar{u}$ at this point is then mapped to all particles in said subdomain. For each deflation cell, we choose to place the computational point, c, in the middle of our subdomain.

An example in one dimension for $N = 10$ particles and $M = 3$ deflation cells would be

$$P = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}^T, \tag{2.14}$$

where the first and last subdomains contain three particles each, and the middle subdomain contains four. The corresponding pseudoinverse $P^\dagger$ is then

$$P^\dagger = \begin{pmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{pmatrix}. \tag{2.15}$$

Linear Approximations

For one dimension, the linear deflation will need two computational points per subdomain. The idea is to interpolate a linear function from these computational points, onto which the particles will be mapped. Then, P will have the size $N \times 2M$. The one dimensional formula for constructing P is as follows: For particle $\vec{x}_i$ in cell k we have

$$P_{i,2k-1} = \frac{-x_i + c_2}{c_2 - c_1} \qquad \text{and} \qquad P_{i,2k} = \frac{x_i - c_1}{c_2 - c_1}. \tag{2.16}$$

The computational points of each subdomain are denoted by c_1 and c_2. They are placed uniformly on the subdomains, as

$$c_1 = M_1 + \frac{1}{3}dM \qquad \text{and} \qquad c_2 = M_1 + \frac{2}{3}dM,$$

where dM is the size of the subdomain and M_1 its leftmost boundary.

In two dimensions, our linear approximation consists of approximating a plane on each subdomain. For this, three computational points per subdomain are needed. The projection P will have the size $N \times 3M$, where $N = N_xN_y$ is the total number of particles, and $M = M_xM_y$ the total number of deflation cells. Thus, for particle $\vec{x}_l = (x_i, y_j)$ in cell k we have

$$\begin{aligned}
P_{l,3k-2} &= \frac{(-1)(x_i - m_x) + (1 + c_{1,x} - m_x)}{(c_{3,x} - c_{3,y})(c_{1,y} - c_{2,y}) - (c_{1,y} - c_{2,x})} + \\
&\quad + \frac{(y_i - m_y)(c_{3,x} - c_{1,x}) - (c_{3,x} - c_{2,x})(c_{1,y} - m_y)}{(c_{1,y} - c_{2,y})(c_{3,x} - c_{2,1}) - (c_{3,y} - c_{2,y})(c_{1,x} - c_{2,x})}, \\
P_{l,3k-1} &= \frac{(x_i - m_x)(1 - c_{1,y} + c_{2,y}) + (1 - c_{1,y} + c_{2,y})(c_{1,x} - m_x)}{(c_{3,x} - c_{3,y})(c_{1,y} - c_{2,y}) - (c_{1,y} - c_{2,x})} + \\
&\quad + \frac{(y_i - m_y)(c_{1,x} - c_{3,x}) - (c_{3,x} - c_{2,x})(c_{1,y} - m_y)}{(c_{1,y} - c_{2,y})(c_{3,x} - c_{2,1}) - (c_{3,y} - c_{2,y})(c_{1,x} - c_{2,x})}, \\
P_{l,3k} &= \frac{(x_i - m_x)(c_{1,y} - c_{2,y}) + (c_{1,y} - c_{2,y})(c_{1,x} - m_x)}{(c_{3,x} - c_{3,y})(c_{1,y} - c_{2,y}) - (c_{1,y} - c_{2,x})} + \\
&\quad + \frac{(y_i - m_y)(c_{3,x}c_{1,x}) - (c_{3,x} - c_{1,x})(c_{1,y} - m_y)}{(c_{1,y} - c_{2,y})(c_{3,x} - c_{2,1}) - (c_{3,y} - c_{2,y})(c_{1,x} - c_{2,x})}.
\end{aligned} \tag{2.17}$$

Here, $m = (m_x, m_y)$ is defined as the mid point of the subdomain and c_1, c_2 and c_3 are three computational points placed according to a reference subdomain, as seen in Fig. 2.1. The computational points lie with 120 degrees spacing between each other on a circle with radius $r = \frac{1}{2}dM_x$. We choose to define our deflation subdomains such that $dM_x = dM_y$. As before, $P^\dagger$ is the Moore-Penrose pseudoinverse of P.

Quadratic Approximations

A quadratic approximation is constructed by interpolating a quadratic function on each subdomain from its computational points. In one dimension, a quadratic approximation requires three computational points per subdomain.

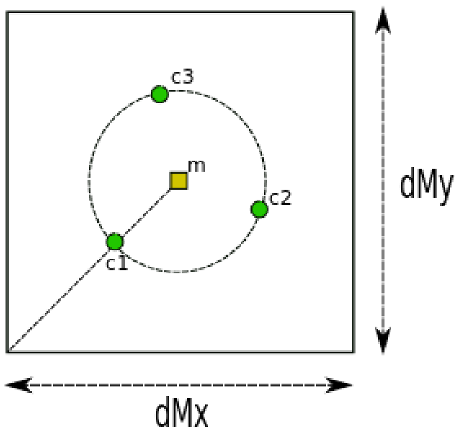

Figure 2.1: Example of deflation cell and placement of computational points in 2D for linear approximations.

For problems in one dimension P will have the size $N \times 3M$ and is constructed by

$$\begin{aligned}
P_{i,3k-2} = x_i^2 \cdot & \frac{c_1c_2 - c_1c_3 + c_2c_3 - c_3^2}{(c_1^2c_2 - c_1^2c_3 - c_1c_2^2 + c_1c_3^2 + c_2^2c_3 - c_2c_3^2)(c_1+c_3)} + \\
& + x_i \cdot \frac{-c_2^2 + c_3^2}{c_1^2c_2 - c_1^2c_3 - c_1c_2^2 + c_1c_3^2 + c_2^2c_3 - c_2c_3^2} + \\
& + \frac{c_1c_2^2c_3 - c_1c_2c_3^2 + c_2^2c_3^2 - c_2c_3^3}{(c_1^2c_2 - c_1^2c_3 - c_1c_2^2 + c_1c_3^2 + c_2^2c_3 - c_2c_3^2)(c_1+c_3)}, \\
P_{i,3k-1} = x_i^2 \cdot & \frac{-c_1^2 + c_3^2}{(c_1^2c_2 - c_1^2c_3 - c_1c_2^2 + c_1c_3^2 + c_2^2c_3 - c_2c_3^2)(c_1+c_3)} + \\
& + x_i \cdot \frac{c_1^2 - c_3^2}{c_1^2c_2 - c_1^2c_3 - c_1c_2^2 + c_1c_3^2 + c_2^2c_3 - c_2c_3^2} + \\
& + \frac{-c_1^3c_3 + c_1c_3^3}{(c_1^2c_2 - c_1^2c_3 - c_1c_2^2 + c_1c_3^2 + c_2^2c_3 - c_2c_3^2)(c_1+c_3)}, \\
P_{i,3k} = x_i^2 \cdot & \frac{-c_1^2 - c_1c_2 + c_1c_3 + 2c_2^2 - c_2c_3}{(c_1^2c_2 - c_1^2c_3 - c_1c_2^2 + c_1c_3^2 + c_2^2c_3 - c_2c_3^2)(c_1+c_3)} + \\
& + x_i \cdot \frac{-c_1^2 + c_2^2}{c_1^2c_2 - c_1^2c_3 - c_1c_2^2 + c_1c_3^2 + c_2^2c_3 - c_2c_3^2} + \\
& + \frac{2c_1^4 + c_1^3c_2 - 3c_1^3c_2 - 3c_1^2c_2^2 + c_1^2c_2c_3 - c_1c_2^2c_3}{(c_1^2c_2 - c_1^2c_3 - c_1c_2^2 + c_1c_3^2 + c_2^2c_3 - c_2c_3^2)(c_1+c_3)},
\end{aligned} \tag{2.18}$$

for particle $\vec{x}_i$ in subdomain k. The computational points of the deflated system are c_1, c_2 and c_3, placed by the formula

$$c_1 = M_1 + \frac{1}{4}dM, \qquad c_2 = M_1 + \frac{1}{2}dM \qquad \text{and} \qquad c_3 = M_1 + \frac{3}{4}dM,$$

where dM is the size of the deflation subdomain and M_1 its leftmost coordinate.

2.2.2 Varieties

Our method can be varied in a number of ways. First, when decomposing our domain, we can choose to do it either geometrically or by number of particles. Using different decompositions should not affect our approximations a great deal, since the particles, despite not necessarily being uniformly distributed, should still cover the domain somewhat evenly. Also, the domain decomposition could be done taking the boundary particles into particular consideration. By placing them in subdomains of their own, the boundary conditions can be explicitly enforced. In one, and maybe two, dimensions this is a possible approach, but as soon as we start looking at three dimensions it will not be practically possible.

Second, the computational points used in the approximations could be placed in a number of different ways. Here we choose to place them uniformly over our domain in 1D. In 2D they are placed as in Fig. 2.1. More random placings could be considered, as well as other structures.

Third, the matrix A from Eq. 2.12 will be scaled. Here, this is done by scaling the entire matrix and the vector b such that

$$A_{ii} = 1, \ \ \forall i \in [1, N].$$

Other approaches to scaling exist. One could, for example, weight the rows of A and b corresponding to the boundary particles - to ensure the boundary conditions are fulfilled.

2.2.3 Evaluation

We evaluate our results by investigating the approximated solution $\dddot{u}$. This is done both by comparing it to an already known solution, i.e.

$$e = u - \dddot{u}, \tag{2.19}$$

as well as computing the residual

$$r = b - A\dddot{u}, \tag{2.20}$$

in the two-norm, $\|\cdot\|_2$.

Additionally, the computational effort is taken into consideration. We investigate the improvement of convergence in the BiCGSTAB algorithm in MATLAB when $\dddot{u}$ is used as initial guess compared to using the zero-vector. For these tests we use the standard tolerance of the algorithm, tol $= 10^{-6}$, and no preconditioners.

Chapter 3

Results

In this chapter we present our deflation results for FPM. Results for the deflation of FDM on two test cases can be found in Appendix B. As mentioned in Section 2.2.3 we investigate the deviation from the correct solution as well as the residuals. Also, the convergence of BiCGSTAB is examined. All implementations are made in MATLAB. All tests are performed on uniform particle distributions.

3.1 Results in 1D

We test our deflation in the domain $\Omega = [0, 1] \in \mathbb{R}$. The boundary value problems to investigate are

$$-\Delta u = 1 \text{ in } \Omega, \qquad u = 0 \text{ on } \delta\Omega, \tag{3.1}$$

$$\Delta u = 0 \text{ in } \Omega, \qquad u(0) = 1, \;\; u(1) = 0, \tag{3.2}$$

$$\Delta u = 0 \text{ in } \Omega, \qquad \frac{\delta u}{\delta \vec{n}}(0) = -1, \;\; u(1) = 0, \text{ and} \tag{3.3}$$

$$-\Delta u = 1 \text{ in } \Omega, \qquad \frac{\delta u}{\delta \vec{n}}(0) = -1, \;\; u(1) = 0. \tag{3.4}$$

For all these test cases we test how the approximated solution changes with an increasing number of deflation cells, M. This is done for $N = 200$ particles. The error and residual changes are investigated on a particle distribution of 2050 particles, as are the BiCGSTAB convergence tests. The increase in N is to ensure that larger values of M can be tested. Constants used for FPM are the smoothing length $h = 2.1dx$ and $\eta = 4.5$. In Appendix B the solution and deflation of the FDM system for Eq. 3.3 can be found.

3.1.1 Constant Approximations

By using the constant approximations as seen in Section 2.2.1 we obtain the following results for the deflation of FPM. For each deflation subdomain one computational point is needed, here chosen to be the midpoint of each subdomain.

Dirichlet Boundary Conditions

The test cases with Dirichlet boundary conditions are Eq. 3.1 and Eq. 3.2. The results shown are obtained on the normalised system, where all elements on the diagonal of A are 1 as described in Section 2.2.2. For $N = 200$ the linear systems before and after deflation, $M = 3$, of Eq. 3.1 and Eq. 3.2 have the following condition numbers

$$\text{cond}(A_{\text{Eq.3.1, Eq.3.2}}) = 1.54 \cdot 10^4, \qquad \text{cond}(\bar{A}_{\text{Eq.3.1, Eq.3.2}}) = 3.99.$$

We see that the deflation vastly reduces the condition numbers of the system. As the number of deflation cells increases the condition number of $\bar{A}$ does too, while it stays approximately the same as N increases. This can be seen in Appendix A.

How the approximated solutions look like can be seen in Fig. 3.1 and Fig. 3.2. These figures portray how an increasing number of deflation cells M changes the approximated result. One can see that for Eq. 3.2 the approximated solution converges faster than for Eq. 3.1. For both equations, the Dirichlet boundary conditions are fulfilled. Increasing the number of deflation cells changes the error evaluated as in Section 2.2.3. This can be seen in Fig. 3.3 and Fig. 3.4 where we investigate the deflation for $M = 2^i$, $i = 1 \dots 11$, deflation cells with $N = 2050$ particles. Clearly, an increasing number of deflation cells improves the approximated solution in both cases. Again, we see a much faster convergence rate for Eq. 3.2 than Eq. 3.1. The residual, evaluated as in Section 2.2.3, will stay small for Eq. 3.1 although slightly fluctuating. For Eq. 3.2 it decreases rapidly with increasing M. Both residuals can be found in Appendix A, in Fig. A.25 and Fig. A.26 respectively.

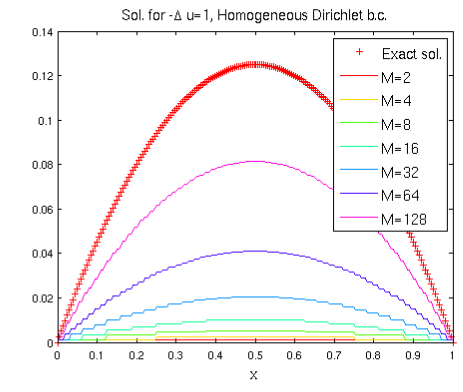

Figure 3.1: Exact and approximated solutions of Eq. 3.1 using constant ansatz functions, $N = 200$, with increasing number of deflation cells $M = 2^i$, $i = 1, \dots, 7$.

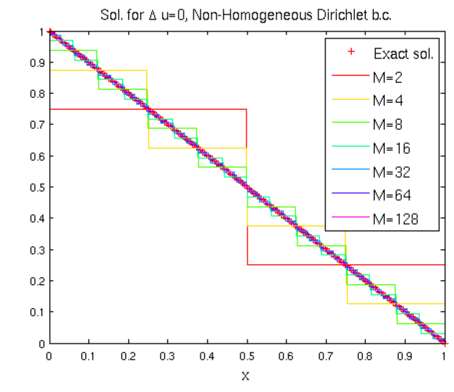

Figure 3.2: Exact and approximated solutions of Eq. 3.2 using constant ansatz functions, $N = 200$, with increasing number of deflation cells $M = 2^i$, $i = 1, \dots, 7$.

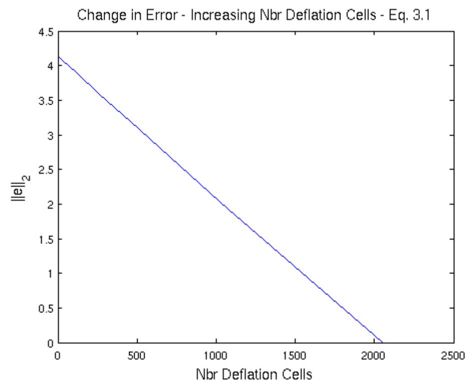

Figure 3.3: Change in error between exact and approximated solutions of Eq. 3.1 using constant ansatz functions, $N = 2050$, with increasing number of deflation cells $M = 2^i$, $i = 1, \dots, 11$.

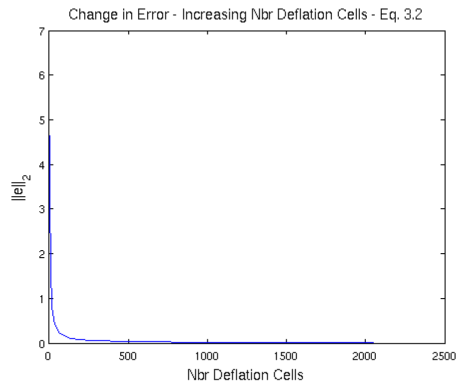

Figure 3.4: Change in error between exact and approximated solutions of Eq. 3.2 using constant ansatz functions, $N = 2050$, with increasing number of deflation cells $M = 2^i$, $i = 1, \dots, 11$.

Mixed Boundary Conditions

Both Eq. 3.3 and Eq. 3.4 have mixed boundary conditions, with a Neumann condition on the $x = 0$ side and a Dirichlet condition at $x = 1$. For $N = 200$, the condition numbers of the normalised linear systems for Eq. 3.3 and Eq. 3.4 given by FPM and the deflated systems for $M = 3$ are

$$\text{cond}(A_{\text{Eq.3.3, Eq.3.4}}) = 7.04 \cdot 10^4, \qquad \text{cond}(\bar{A}_{\text{Eq.3.3, Eq.3.4}}) = 13.93.$$

Also for these test cases a constant deflation reduces the condition number significantly. How the condition numbers change as M and N increase can be seen in Appendix A.

The constant approximations of these two equations can be seen in Fig. 3.5 and Fig. 3.6. The approximated solutions are obtained using $M = 2^i$, $i = 1, \ldots, 7$, deflation cells. One can see that an increasing number of deflation cells does not lead to a better approximation of u. Instead, the largest M seems to give the biggest deviation from the exact solution on the Neumann boundary. For both deflations, the Dirichlet boundary condition is fulfilled. That the method does not converge is also shown by the increasing error in Fig. 3.7 and Fig. 3.8, where we use $N = 2050$ particles and $M = 2^i$, $i = 1, \ldots, 11$, deflation cells. Although the error reaches a minimum in both cases at around $M \approx 1000$ deflation cells, it is still undesirably big at $\|e\|_2 > 10$ for Eq. 3.1 and $\|e\|_2 > 20$ for Eq. 3.2. For both equations the residuals stay small, which can be seen in Fig. A.27 and Fig. A.28.

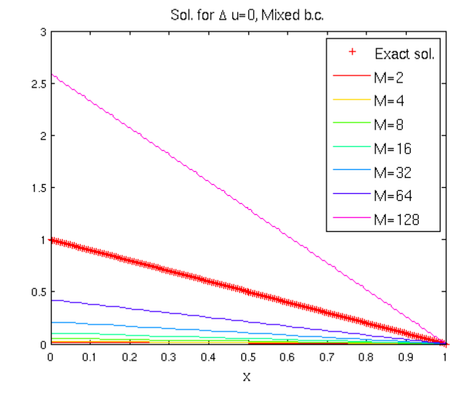

Figure 3.5: Exact and approximated solutions of Eq. 3.3 using constant ansatz functions, $N = 200$, with increasing number of deflation cells $M = 2^i$, $i = 1, \ldots, 7$.

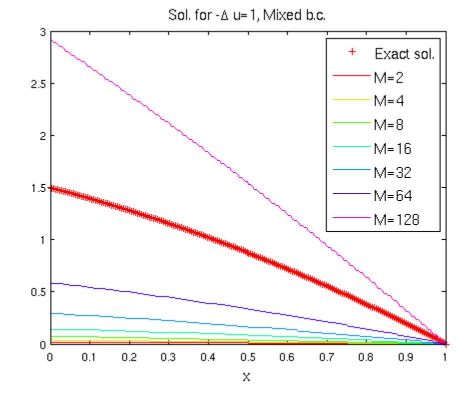

Figure 3.6: Exact and approximated solutions of Eq. 3.4 using constant ansatz functions, $N = 200$, with increasing number of deflation cells $M = 2^i$, $i = 1, \ldots, 7$.

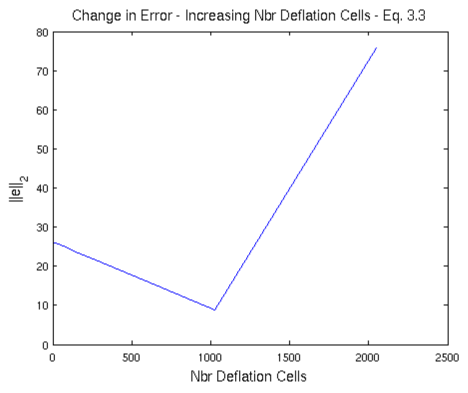

Figure 3.7: Change in error for Eq. 3.3 using constant ansatz functions, $N = 2050$, with increasing number of deflation cells $M = 2^i$, $i = 1, \ldots, 11$.

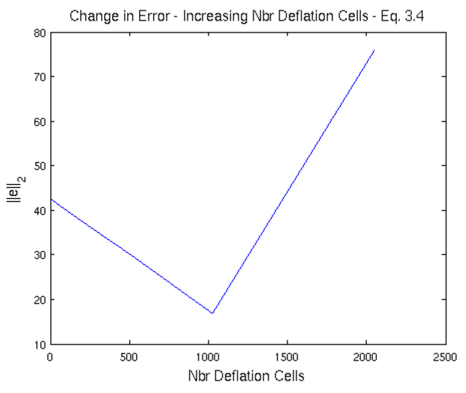

Figure 3.8: Change in error for Eq. 3.4 using constant ansatz functions, $N = 2050$, with increasing number of deflation cells $M = 2^i$, $i = 1, \ldots, 11$.

The convergence of the BiCGSTAB algorithm in MATLAB is investigated. We compare the number of iterations needed for convergence with the initial guess $u_{init} = 0$ to the number of iterations needed when using our approximated solution as initial guess, $u_{init} = \ddot{u}$. The approximation $\ddot{u}$ is obtained for $M = 2^i$, $i = 1, \ldots, 11$, deflation cells. How the convergence behaves for Eq. 3.1, Eq. 3.2, Eq. 3.3 and Eq. 3.4 can be seen in Fig. 3.9, Fig. 3.10, Fig. 3.11 and Fig. 3.12 respectively. For these tests, we use $N = 2050$ particles. We see that for all equations, BiCGSTAB performs better with a zero initial guess than our deflated approximation and that a larger number of deflation cells not necessarily improves the convergence rate. In some cases, Eq. 3.1 and Eq. 3.4, a larger M will even worsen the convergence rate. Our approximation performs better for Dirichlet problems than for problems with mixed boundary conditions.

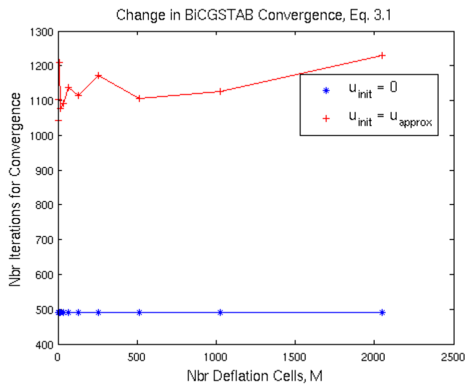

Figure 3.9: Change in convergence rate for Eq. 3.1 and BiCGSTAB with $u_{init} = u_{approx}$ when the number of deflation cells changes as $M = 2^i$, $i = 1, \ldots, 11$, $N = 2050$. Compared to convergence rate for $u_{init} = 0$.

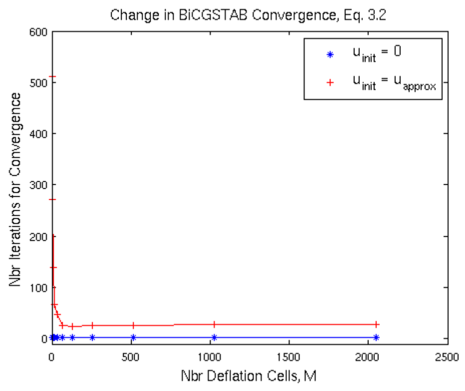

Figure 3.10: Change in convergence rate for Eq. 3.2 and BiCGSTAB with $u_{init} = u_{approx}$ when the number of deflation cells changes as $M = 2^i$, $i = 1, \ldots, 11$, $N = 2050$. Compared to convergence rate for $u_{init} = 0$.

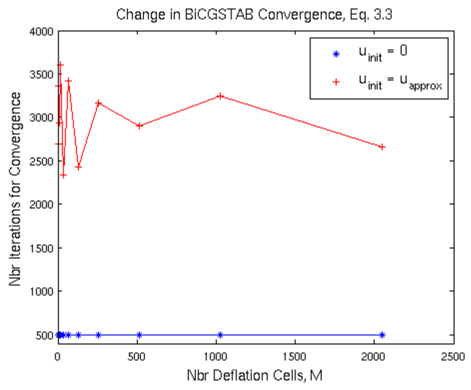

Figure 3.11: Change in convergence rate for Eq. 3.3 and BiCGSTAB with $u_{init} = u_{approx}$ when the number of deflation cells changes as $M = 2^i$, $i = 1, \ldots, 11$, $N = 2050$. Compared to convergence rate for $u_{init} = 0$.

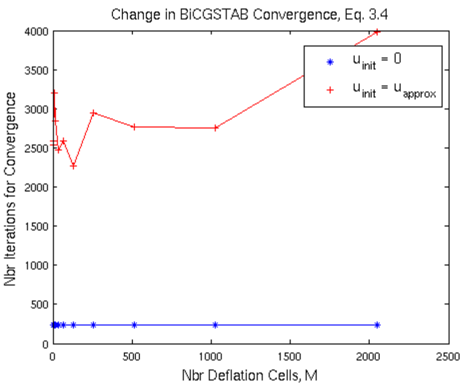

Figure 3.12: Change in convergence rate for Eq. 3.4 and BiCGSTAB with $u_{init} = u_{approx}$ when the number of deflation cells changes as $M = 2^i$, $i = 1, \ldots, 11$, $N = 2050$. Compared to convergence rate for $u_{init} = 0$.

As a final investigation of our constant deflation in one dimension we examine how the convergence rate of the BiCGSTAB algorithm changes as the number of particles, N, changes. Here, we test for

$$N = [200\ 400\ 800\ 1600\ 3200\ 4000]^T \tag{3.5}$$

particles. The number of deflation cells M is selected as the M with the smallest number of iterations in Fig. 3.9, Fig. 3.10, Fig. 3.11 and Fig. 3.12 respectively, thus

$$M_{\text{Eq.3.1}} = 16, \qquad M_{\text{Eq.3.2}} = 64, \qquad M_{\text{Eq.3.3}} = 32 \qquad \text{and} \qquad M_{\text{Eq.3.4}} = 128.$$

The results can be seen in Fig. 3.13, Fig. 3.14, Fig. 3.15 and Fig. 3.16. Worth noting is that for Eq. 3.2 the BiCGSTAB algorithm performs exceptionally well with $u_{init} = 0$ and the deflation will only give worse results, see Fig. 3.14. For all other test cases the deflation yields slightly faster convergence, especially when N becomes very large.

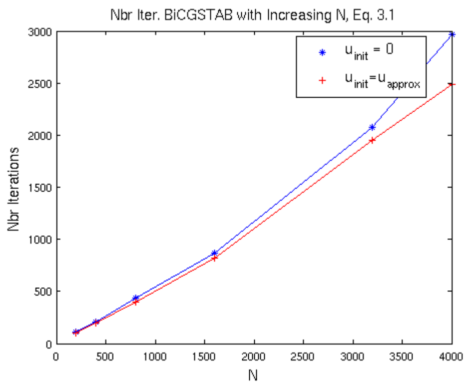

Figure 3.13: Change in convergence rate for Eq. 3.1 and BiCGSTAB with $u_{init} = u_{approx}$ for $M = 16$. The number of particles N changes as in Eq. 3.5. Compared to convergence rate for $u_{init} = 0$.

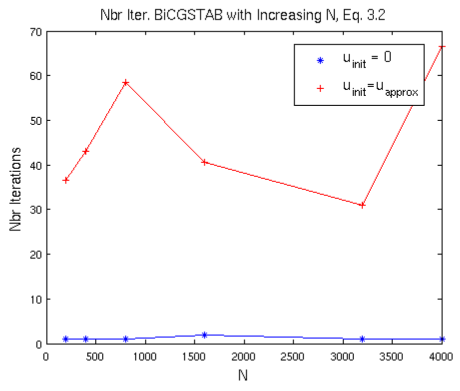

Figure 3.14: Change in convergence rate for Eq. 3.2 and BiCGSTAB with $u_{init} = u_{approx}$ for $M = 64$. The number of particles N changes as in Eq. 3.5. Compared to convergence rate for $u_{init} = 0$.

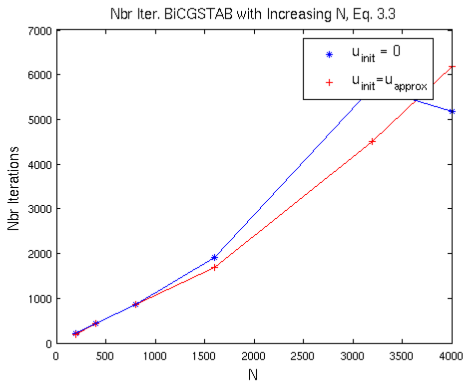

Figure 3.15: Change in convergence rate for Eq. 3.3 and BiCGSTAB with $u_{init} = u_{approx}$ for $M = 32$. The number of particles N changes as in Eq. 3.5. Compared to convergence rate for $u_{init} = 0$.

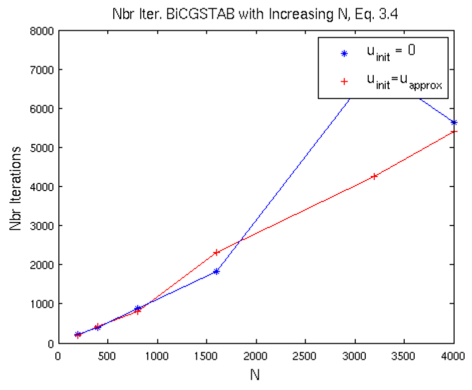

Figure 3.16: Change in convergence rate for Eq. 3.4 and BiCGSTAB with $u_{init} = u_{approx}$ for $M = 128$. The number of particles N changes as in Eq. 3.5. Compared to convergence rate for $u_{init} = 0$.

3.1.2 Linear Approximations

In the same way as for constant approximations, we investigate the results generated by linear approximations in our deflation. How the linear approximation is constructed is described in Section 2.2.1.

Dirichlet Boundary Conditions

The results of the linear deflation for Eq. 3.1 and Eq. 3.2 can be seen in Fig. 3.17 and Fig. 3.18 for $M = 2^i$, $i = 1, \dots, 6$, deflation cells and $N = 200$ particles. For this N and $M = 3$ the deflated systems of Eq. 3.1 and Eq. 3.2 have the condition number

$$\mathrm{cond}(\bar{A}_{\mathrm{Eq.3.1,\,Eq.3.2}}) = 638.48.$$

More results regarding the condition numbers of the deflated systems can be found in Appendix A.

For Eq. 3.2 the deflation gives a solution very close to the exact one and the error is very small, see Fig. 3.20. The deflated solution for Eq. 3.1 converges rapidly, which also can be seen for the error in Fig. 3.19. After a certain number of deflation cells, increasing M will no longer improve the deflated solution notably. For Eq. 3.1 this is the case after approximately $M = 16$ deflation cells. In comparison, $M = 2$ deflation cells is enough for Eq. 3.2. The change in residuals can be seen in Fig. A.29 and Fig. A.30. For both equations these are very small and decreasing with an increasing M.

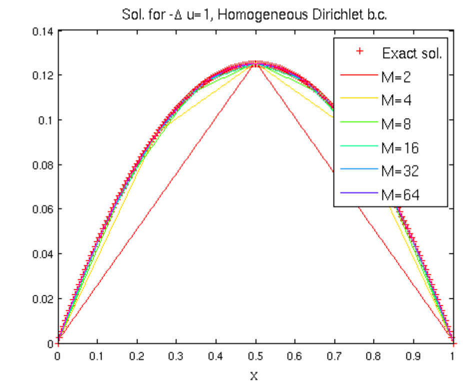

Figure 3.17: Exact and approximated solutions of Eq. 3.1 using linear ansatz functions, $N = 200$, with increasing number of deflation cells $M = 2^i$, $i = 1, \dots, 6$.

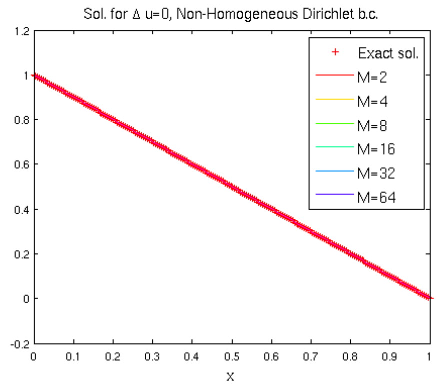

Figure 3.18: Exact and approximated solutions of Eq. 3.2 using linear ansatz functions, $N = 200$, with increasing number of deflation cells $M = 2^i$, $i = 1, \dots, 6$.

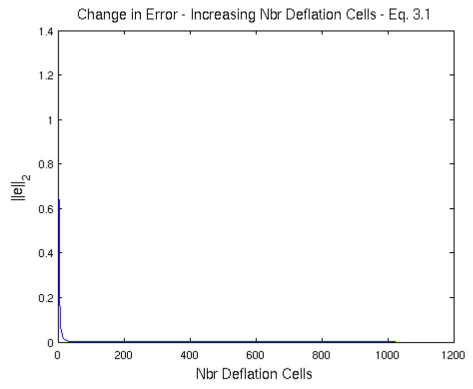

Figure 3.19: Change in error for Eq. 3.1 using linear ansatz functions, $N = 2050$, with increasing number of deflation cells $M = 2^i$, $i = 1, \dots, 10$.

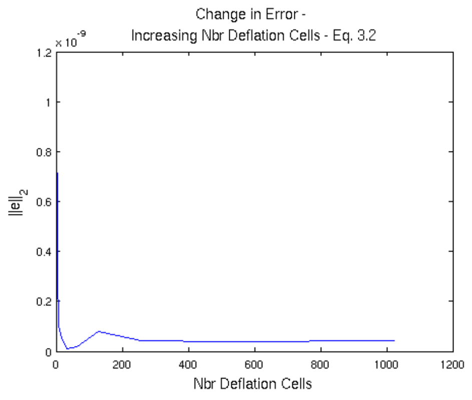

Figure 3.20: Change in error for Eq. 3.2 using linear ansatz functions, $N = 2050$, with increasing number of deflation cells $M = 2^i$, $i = 1, \dots, 10$.

Mixed Boundary Conditions

The results for the test cases with mixed boundary conditions, Eq. 3.3 and Eq. 3.4, can be seen in Fig. 3.21 and Fig. 3.22. For $N = 200$ and $M = 3$ the condition number of the deflated systems is

$$\mathrm{cond}(\bar{A}_{\mathrm{Eq.3.3,\,Eq.3.4}}) = 2.42 \cdot 10^3.$$

More results regarding the condition numbers can be found in Appendix A.

In Fig. 3.21 we see that the approximation is very close to the exact solution for all M. As seen in Fig. 3.22, for Eq. 3.4 the linear approximation converges quickly to the exact solution. Fig. 3.23 and Fig. 3.24 show how the error changes with an increased number of deflation cells, $M = 2^i$, $i = 1, \ldots, 10$. For both test cases the error is relatively small for all M, although much smaller for Eq. 3.3. Also the residuals are very small, especially for Eq. 3.3, which is shown in Fig. A.31 and Fig. A.32.

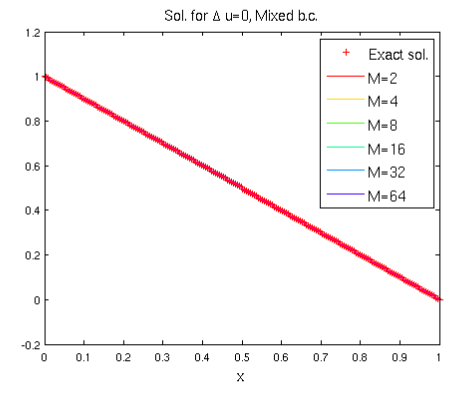

Figure 3.21: Exact and approximated solutions of Eq. 3.3 using linear ansatz functions, $N = 200$, with increasing number of deflation cells $M = 2^i$, $i = 1, \ldots, 6$.

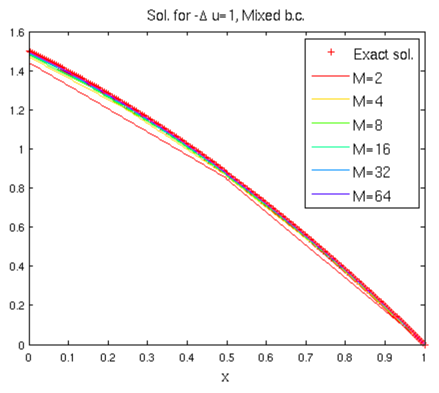

Figure 3.22: Exact and approximated solutions of Eq. 3.4 using linear ansatz functions, $N = 200$, with increasing number of deflation cells $M = 2^i$, $i = 1, \ldots, 6$.

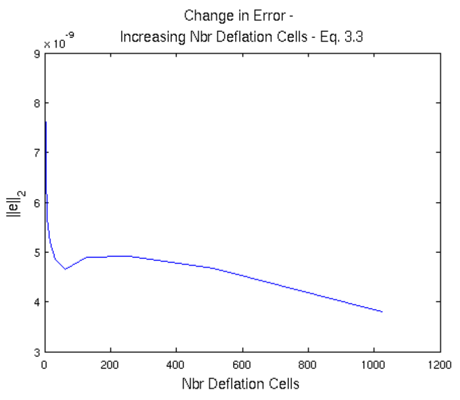

Figure 3.23: Change in error for Eq. 3.3 using linear ansatz functions, $N = 2050$, with increasing number of deflation cells $M = 2^i$, $i = 1, \ldots, 10$.

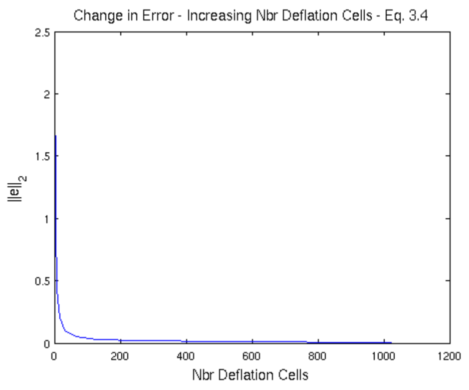

Figure 3.24: Change in error for Eq. 3.4 using linear ansatz functions, $N = 2050$, with increasing number of deflation cells $M = 2^i$, $i = 1, \ldots, 10$.

The change in convergence rate for BiCGSTAB can be seen in Fig. 3.25, Fig. 3.26, Fig. 3.27 and Fig. 3.28. As can be expected, for the two test cases with linear exact solutions, Eq. 3.2 and Eq. 3.3, the deflated solution in BiCGSTAB gives instant convergence whilst $u_{init} = 0$ needs approximately 2 and 500 iterations, respectively, for convergence. For Eq. 3.1 it is initially slower to use the deflated solution, but for $M \approx 500$ deflation cells an improvement compared to $u_{init} = 0$ can be observed. Regarding Eq. 3.4, using $u_{init} = \ddot{u}$ will always give slower convergence than $u_{init} = 0$. In that case, the convergence rate stays more or less constant for changing M.

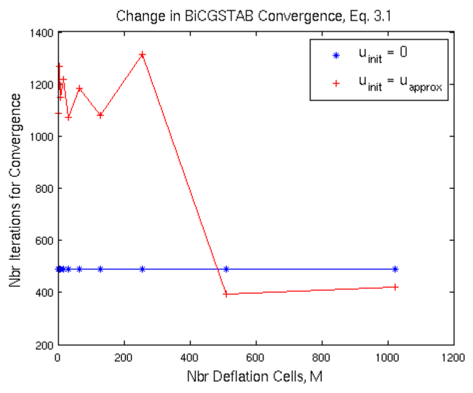

Figure 3.25: Change in convergence rate for Eq. 3.1 and BiCGSTAB with $u_{init} = u_{approx}$ when the number of deflation cells changes as $M = 2^i$, $i = 1, \ldots, 10$, $N = 2050$. Compared to convergence rate for $u_{init} = 0$.

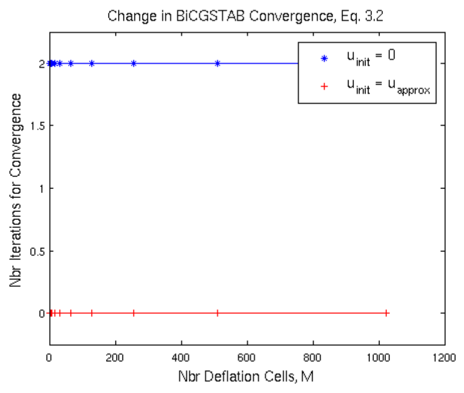

Figure 3.26: Change in convergence rate for Eq. 3.2 and BiCGSTAB with $u_{init} = u_{approx}$ when the number of deflation cells changes as $M = 2^i$, $i = 1, \ldots, 10$, $N = 2050$. Compared to convergence rate for $u_{init} = 0$.

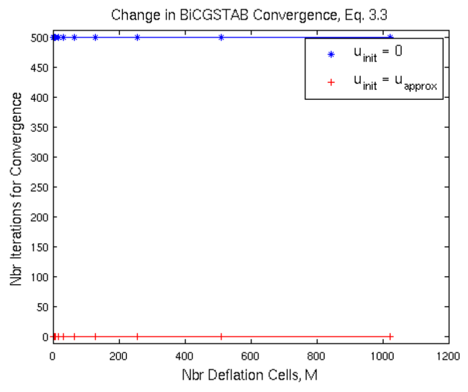

Figure 3.27: Change in convergence rate for Eq. 3.3 and BiCGSTAB with $u_{init} = u_{approx}$ when the number of deflation cells changes as $M = 2^i$, $i = 1, \ldots, 10$, $N = 2050$. Compared to convergence rate for $u_{init} = 0$.

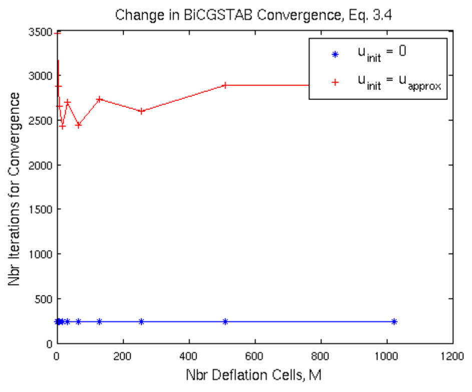

Figure 3.28: Change in convergence rate for Eq. 3.4 and BiCGSTAB with $u_{init} = u_{approx}$ when the number of deflation cells changes as $M = 2^i$, $i = 1, \ldots, 10$, $N = 2050$. Compared to convergence rate for $u_{init} = 0$.

Investigating how the BiCGSTAB convergence rate changes with an increasing number of particles N, the results in Fig. 3.29, Fig. 3.30, Fig. 3.31 and Fig. 3.32 are obtained. The tests are performed on N chosen as in Eq. 3.5. For Eq. 3.1 and Eq. 3.4, M is the number of deflation cells that yields the best convergence as seen in Fig. 3.25 and Fig. 3.28, but smaller than 100 so as to have $2M < N$ for the smallest N. This gives us

$$M_{\text{Eq.3.1}} = 32, \qquad \text{and} \qquad M_{\text{Eq.3.4}} = 16.$$

Since Eq. 3.2 and Eq. 3.3 converges directly with the deflated solution, for those two test cases we set $M = 3$ in our tests. As before, the convergence for $u_{init} = \ddot{u}$ is instant in Fig. 3.30 and

Fig. 3.31. For Eq. 3.2, also convergence for $u_{init} = 0$ is fast. In contrast, the number of iterations needed for convergence for Eq. 3.3 grows by a factor of two for $u_{init} = 0$. For Eq. 3.1 in Fig. 3.29 our deflated solution proves slightly worse, and for Eq. 3.4 in Fig. 3.32 slightly better, than the zero-vector.

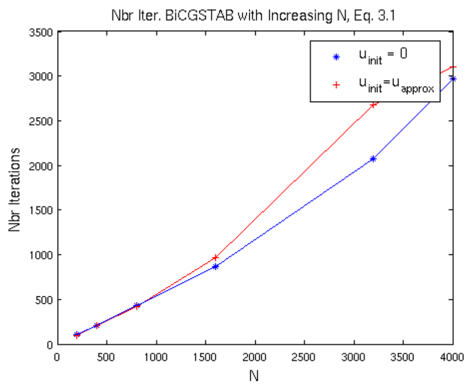

Figure 3.29: Change in convergence rate for Eq. 3.1 and BiCGSTAB with $u_{init} = u_{approx}$ for $M = 32$. The number of particles N changes as in Eq. 3.5. Compared to convergence rate for $u_{init} = 0$.

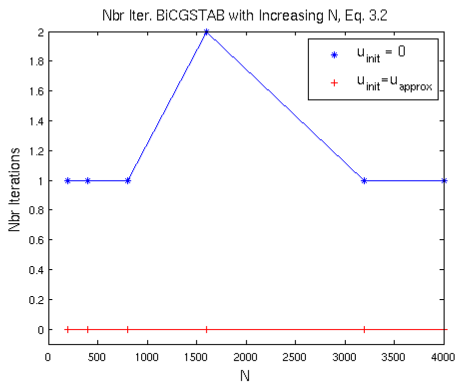

Figure 3.30: Change in convergence rate for Eq. 3.2 and BiCGSTAB with $u_{init} = u_{approx}$ for $M = 3$. The number of particles N changes as in Eq. 3.5. Compared to convergence rate for $u_{init} = 0$.

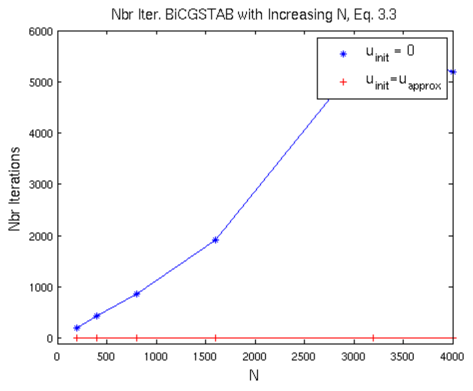

Figure 3.31: Change in convergence rate for Eq. 3.3 and BiCGSTAB with $u_{init} = u_{approx}$ for $M = 3$. The number of particles N changes as in Eq. 3.5. Compared to convergence rate for $u_{init} = 0$.

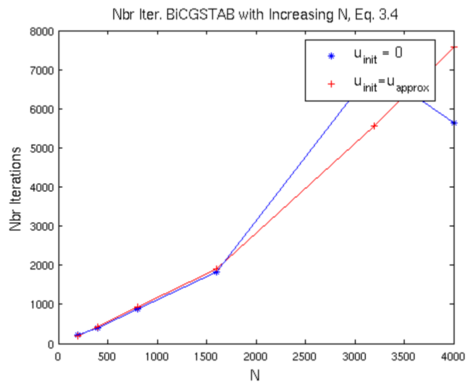

Figure 3.32: Change in convergence rate for Eq. 3.4 and BiCGSTAB with $u_{init} = u_{approx}$ for $M = 16$. The number of particles N changes as in Eq. 3.5. Compared to convergence rate for $u_{init} = 0$.

3.1.3 Quadratic Approximations

The quadratic deflation requires three computational points per cell, which means that the maximum number of deflation cells M one can use with $N = 200$ particles is 66. The approximation is described in Section 2.2.1. We show the deflation first for $N = 200$ particles with an increasing number of deflation cells, $M = 2^i$, $i = 2, \dots, 6$. Then the error, the residuals and the convergence rate of BiCGSTAB are investigated using $N = 2050$ particles.

Dirichlet Boundary Conditions

For Eq. 3.1 and Eq. 3.2 the approximated solutions are as seen in Fig. 3.33 and Fig. 3.34. Increasing the number of deflation cells hardly changes the approximated solution, at least not visibly in our results. For $N = 200$, $M = 3$, the condition number of the deflated system will be

$$\mathrm{cond}(\bar{A}_{\mathrm{Eq.3.1,\,Eq.3.2}}) = 1.30 \cdot 10^7.$$

More information can be found in Appendix A.

As is shown by the plots of the error in Fig. 3.35 and Fig. 3.36 the error is always small, but increases slightly for larger M. Also the residuals are very small, as can be seen in Fig. A.33 and Fig. A.34. For M larger than $M = 2^5$, $N = 2050$, the deflated solutions yield big errors. This is because our system $\bar{A}\bar{u} = \bar{b}$ then becomes ill-conditioned. For example, for Eq. 3.1 with $N = 2050$ and $M = 2^6$ the condition number of $\bar{A}$ is $\mathrm{cond}(\bar{A}) = 2.1 \cdot 10^{17}$. Thus, we need to limit M such that $\bar{A}\bar{u} = \bar{b}$ remains easily solvable.

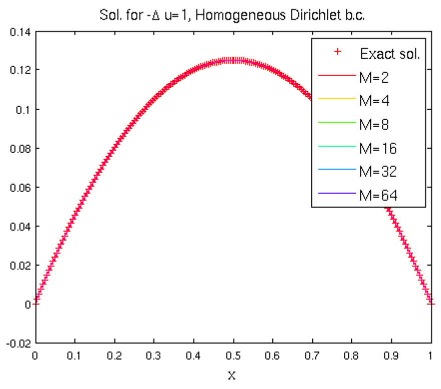

Figure 3.33: Exact and approximated solution of Eq. 3.1 using quadratic ansatz functions, $N = 200$, $M = 2^i$, $i = 2, \ldots, 6$.

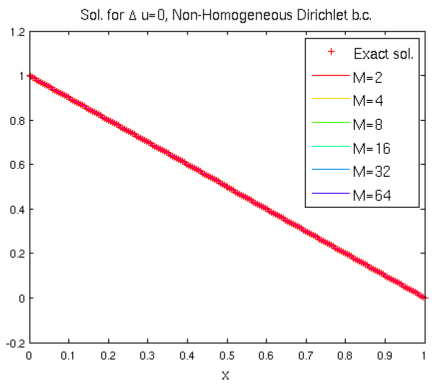

Figure 3.34: Exact and approximated solution of Eq. 3.2 using quadratic ansatz functions, $N = 200$, $M = 2^i$, $i = 2, \ldots, 6$.

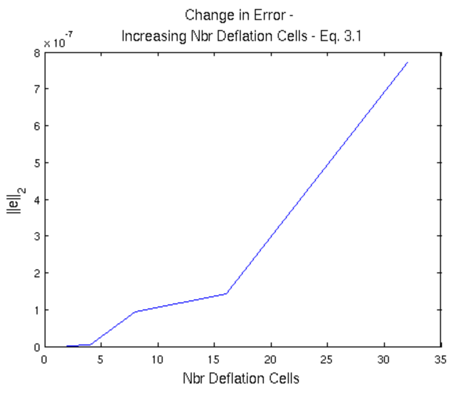

Figure 3.35: Change in error for Eq. 3.1 using quadratic ansatz functions, $N = 2050$, with increasing number of deflation cells $M = 2^i$, $i = 1, \ldots, 5$.

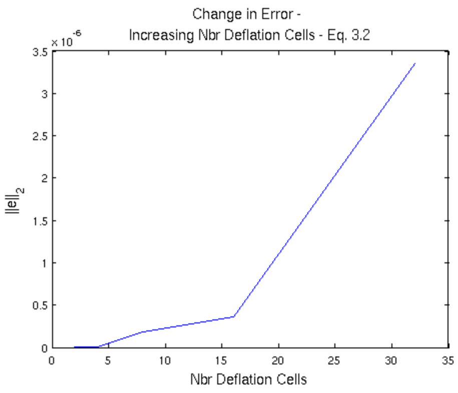

Figure 3.36: Change in error for Eq. 3.2 using quadratic ansatz functions, $N = 2050$, with increasing number of deflation cells $M = 2^i$, $i = 1, \ldots, 5$.

Mixed Boundary Conditions

For Eq. 3.3 and Eq. 3.4 with $M = 2^i$, $i = 2, \ldots, 5$, deflation cells the approximated solutions are as seen in Fig. 3.37 and Fig. 3.38 respectively. The condition number grows large quickly, for $N = 200$, $M = 3$ we have

$$\text{cond}(\bar{A}_{\text{Eq.3.3, Eq.3.4}}) = 2.10 \cdot 10^7,$$

and it increases with changing M and N as seen in Appendix A.

As for the Dirichlet problems, the quadratic deflation for mixed boundary value problems gives results very close to the exact solution. Increasing the number of deflation cells does not affect the approximated solution much, although the error will increase slightly, as is shown in Fig. 3.39 and Fig. 3.40, albeit it will still be small. The residuals in Fig. A.35 and Fig. A.36 stay small but grow a little with an increase in M.

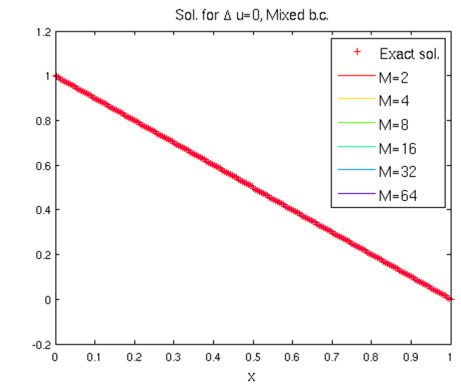

Figure 3.37: Exact and approximated solution of Eq. 3.3 using quadratic ansatz functions, $N = 200$, $M = 3$.

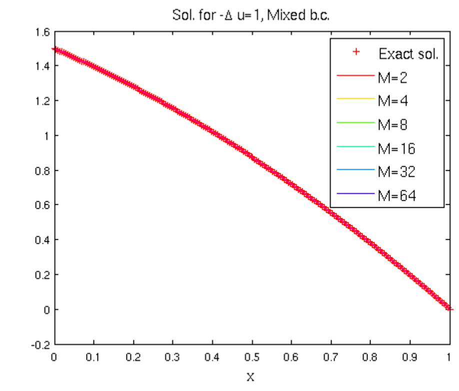

Figure 3.38: Exact and approximated solution of Eq. 3.4 using quadratic ansatz functions, $N = 200$, $M = 3$.

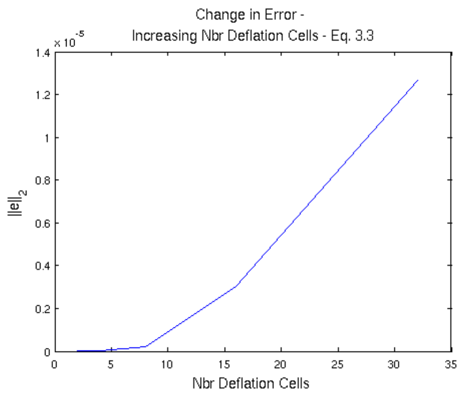

Figure 3.39: Change in error for Eq. 3.3 using quadratic ansatz functions, $N = 2050$, with increasing number of deflation cells $M = 2^i$, $i = 1, \ldots, 9$.

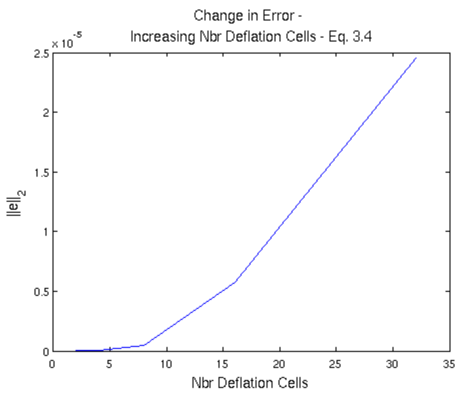

Figure 3.40: Change in error for Eq. 3.4 using quadratic ansatz functions, $N = 2050$, with increasing number of deflation cells $M = 2^i$, $i = 1, \ldots, 9$.

We investigate the convergence rate of BiCGSTAB with $N = 2050$ particles, and an increasing number of deflation cells $M = 2^i$, $i = 1, \ldots, 5$. In Fig. 3.41, Fig. 3.42, Fig. 3.43 and Fig. 3.44 we see that $u_{init} = \ddot{u}$ gives a much faster convergence for all test cases when M is small but performs much worse for large M. Investigating convergence for $M = 3$ and N as in Eq. 3.5, we see that the number of iterations needed for convergence for $u_{init} = 0$ grows steadily, whilst for $u_{init} = \ddot{u}$ convergence is instant. This is shown in Fig. 3.45, Fig. 3.46, Fig. 3.47 and Fig. 3.48.

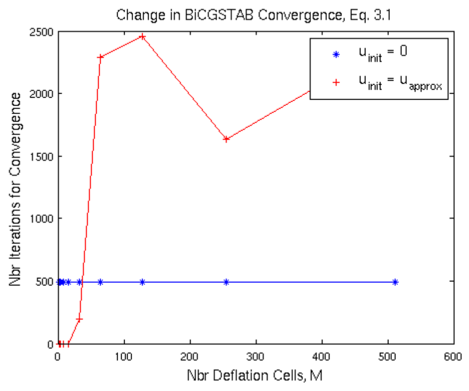

Figure 3.41: Change in convergence rate for Eq. 3.1 and BiCGSTAB with $u_{init} = u_{approx}$ when the number of deflation cells changes as $M = 2^i$, $i = 1, \ldots, 9$, $N = 2050$. Compared to convergence rate for $u_{init} = 0$.

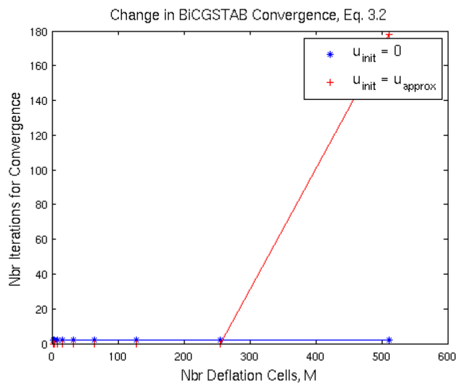

Figure 3.42: Change in convergence rate for Eq. 3.2 and BiCGSTAB with $u_{init} = u_{approx}$ when the number of deflation cells changes as $M = 2^i$, $i = 1, \ldots, 9$, $N = 2050$. Compared to convergence rate for $u_{init} = 0$.

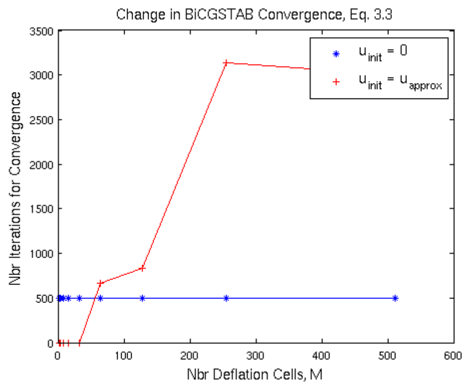

Figure 3.43: Change in convergence rate for Eq. 3.3 and BiCGSTAB with $u_{init} = u_{approx}$ when the number of deflation cells changes as $M = 2^i$, $i = 1, \ldots, 9$, $N = 2050$. Compared to convergence rate for $u_{init} = 0$.

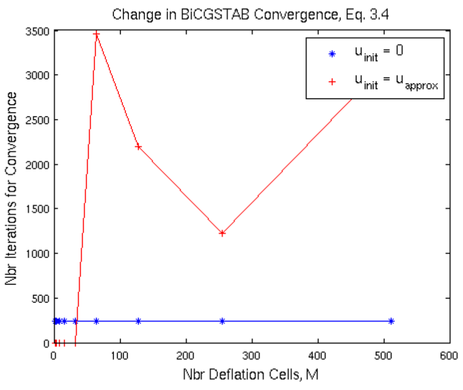

Figure 3.44: Change in convergence rate for Eq. 3.4 and BiCGSTAB with $u_{init} = u_{approx}$ when the number of deflation cells changes as $M = 2^i$, $i = 1, \ldots, 9$, $N = 2050$. Compared to convergence rate for $u_{init} = 0$.

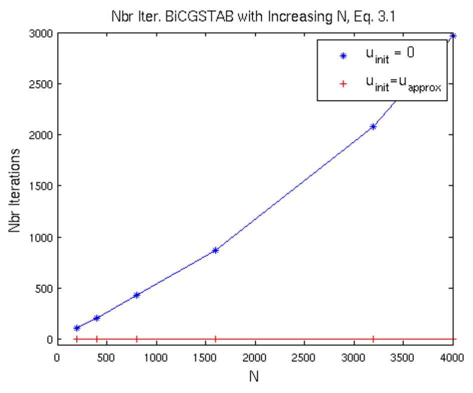

Figure 3.45: Change in convergence rate for Eq. 3.1 and BiCGSTAB with $u_{init} = u_{approx}$ for $M = 3$ deflation cells as N changes as in Eq. 3.5. Compared to convergence rate for $u_{init} = 0$.

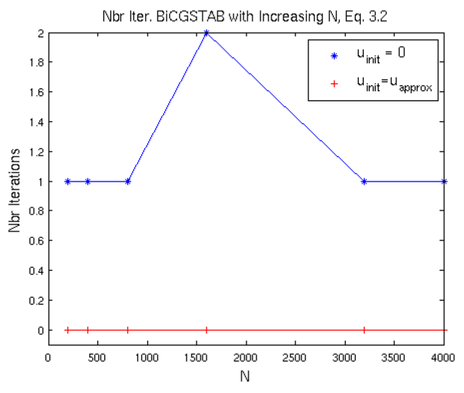

Figure 3.46: Change in convergence rate for Eq. 3.2 and BiCGSTAB with $u_{init} = u_{approx}$ for $M = 3$ deflation cells as N changes as in Eq. 3.5. Compared to convergence rate for $u_{init} = 0$.

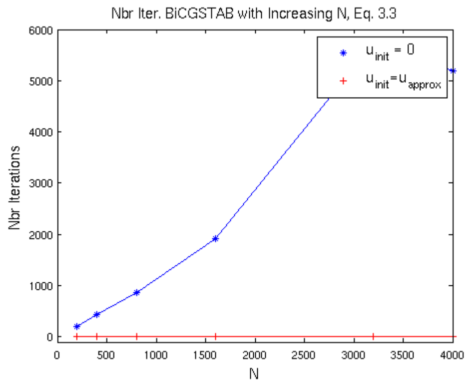

Figure 3.47: Change in convergence rate for Eq. 3.3 and BiCGSTAB with $u_{init} = u_{approx}$ for $M = 3$ deflation cells as N changes as in Eq. 3.5. Compared to convergence rate for $u_{init} = 0$.

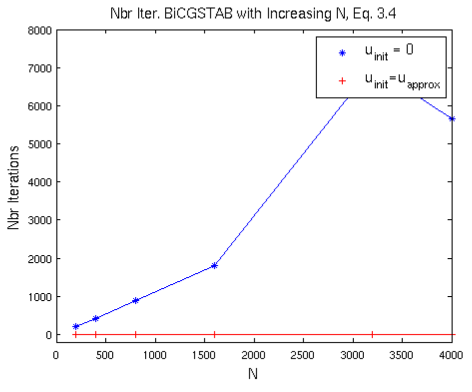

Figure 3.48: Change in convergence rate for Eq. 3.4 and BiCGSTAB with $u_{init} = u_{approx}$ for $M = 3$ deflation cells as N changes as in Eq. 3.5. Compared to convergence rate for $u_{init} = 0$.

3.2 Results in 2D

To test our method in two dimensions, we investigate three different settings;

$$-\Delta u = 2 \text{ in } \Omega, \quad u = 0 \text{ on } \delta\Omega, \qquad \text{and} \tag{3.6}$$

$$-\Delta u = 2 \text{ in } \Omega, \quad u = 0 \text{ on } \delta\Omega_D \text{ and } \frac{\delta u}{\delta \vec{n}} = 0 \text{ on } \delta\Omega_N, \tag{3.7}$$

where we define $\Omega = [0,1] \times [0,1]$, the boundary $\delta\Omega_D$ as the boundary where $x = 0$ and $y = 0$, and $\delta\Omega_N$ as the boundary where $x = 1$ and $y = 1$. The third setting to investigate is

$$\Delta u = 0 \text{ in } \Omega, \tag{3.8}$$

with boundary condition $\frac{\delta u}{\delta \vec{n}} = -1$ on the leftmost boundary, $\frac{\delta u}{\delta \vec{n}} = 0$ on the upper and lower boundaries and $u = 0$ on the rightmost boundary. We investigate our deflation method for Eq. 3.8 in a long and narrow tube-like domain, especially interested in how BiCGSTAB converges as the tube grows longer. Additionally, we investigate how the deflation behaves for Eq. 3.8 in a domain where we have inserted rectangular obstacles. These obstacles are defined as holes in the domain, here of size $22dx \times 6dy$ with a homogeneous Neumann boundary.

As before, the constants used in FPM are $\eta = 4.5$ and $h = 2.1dx$. We use a uniform particle distribution of $N_x = 40$ and $N_y = 40$ particles for visualising the deflation and for convergence tests.

3.2.1 Constant Approximations

As for the results in one dimension, the constant approximation used is the one described in Section 2.2.1. As before, only one computational point per subdomain is needed. For Eq. 3.6, the solution given by FPM without deflation is seen in Fig. 3.49. Deflating the system with constant

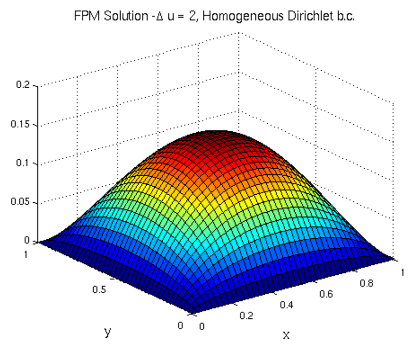

Figure 3.49: Exact solution of Eq. 3.6, $N_x =$ 40, $N_y = 40$.

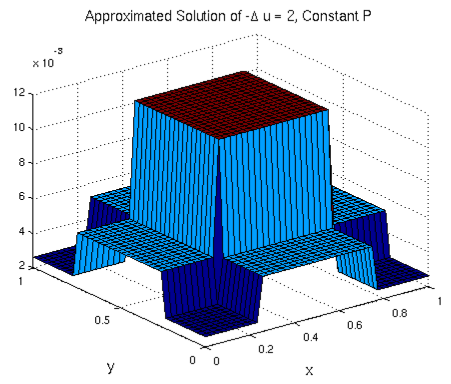

Figure 3.50: Approximated solution of Eq. 3.6, $N_x = 40$, $N_y = 40$, with constant ansatz functions and $M_x = 4$, $M_y = 4$.

ansatz functions gives us the approximated solution that can be seen in Fig. 3.50 using $M_x = 4$ and $M_y = 4$ deflation cells. Clearly, the deflation does not fulfill all Dirichlet boundary conditions and the approximated solution has very low values. When increasing the number of deflation cells, the change in error can be seen in Fig. 3.53. In general, the bigger M we use the better the approximation for Eq. 3.6 becomes. How the condition numbers for A and $\bar{A}$ change as M and N increase can be seen in Appendix A. In general, the constant deflation provides a system with much lower condition number than the original system. However, as the number of deflation cells increases the condition number also grows. The residuals of the two test cases behave in the same way as in the one dimensional case.

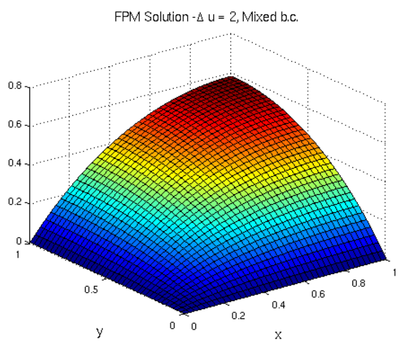

Figure 3.51: Exact solution of Eq. 3.7, $N_x =$ 40, $N_y = 40$.

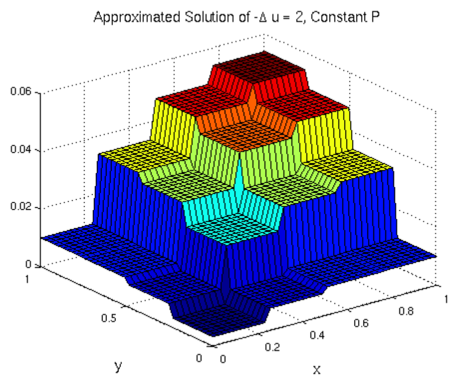

Figure 3.52: Approximated solution of Eq. 3.7, $N_x = 40$, $N_y = 40$, with constant ansatz functions and $M_x = 4$, $M_y = 4$.

The solution for Eq. 3.7 given by FPM can be seen in Fig. 3.51. Using $M_x = M_y = 4$ deflation cells, the approximated solution is as in Fig. 3.52. Worth noting is that the constant deflation seems to handle the homogeneous Neumann boundary conditions well. Also the Dirichlet boundary conditions are close to fulfilled. The change in error with an increasing number of deflation

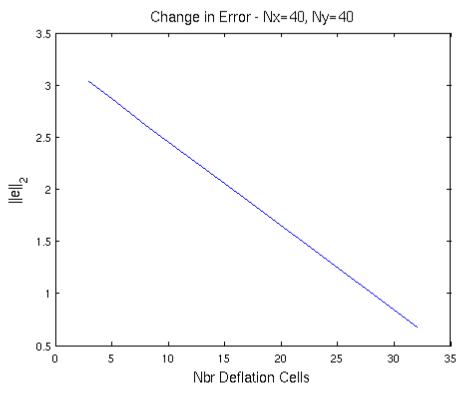

Figure 3.53: Change in error between exact and approximated solution of Eq. 3.6, $N_x =$ 40, $N_y = 40$, with constant ansatz functions and $M_x = M_y = 3, 4, 5, 8, 16, 32$.

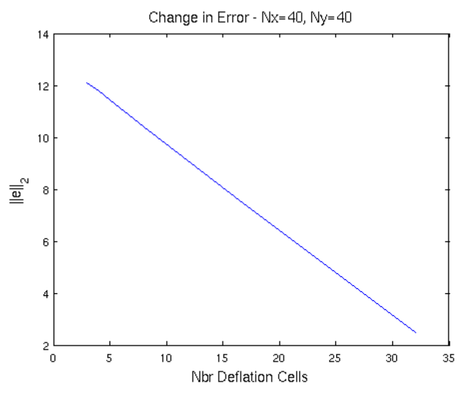

Figure 3.54: Change in error between exact and approximated solution of Eq. 3.7, $N_x =$ 40, $N_y = 40$, with constant ansatz functions and $M_x = M_y = 3, 4, 5, 8, 16, 32$.

cells can be seen in Fig. 3.53. For Eq. 3.7 the error is significantly bigger than for Eq. 3.6, but it decreases faster for Eq. 3.7.

The convergence of the BiCGSTAB algorithm in MATLAB is investigated for $N_x = N_y = 40$ particles. In Fig. 3.55 and Fig. 3.56 the number of iterations needed for convergence for $u_{init} = \ddot{u}$ where

$$M_x = M_y = 3,\ 4,\ 5,\ 8,\ 16,\ 32,$$

is shown and compared to the number of iterations needed for $u_{init} = 0$. For Eq. 3.7 the deflation performs better than the zero-vector for very small and large M. In comparison, the deflation for Eq. 3.6 needs larger M to outperfom $u_{init} = 0$. Furthermore, we investigate how the convergence changes when the number of particles N increases. The results can be seen in Fig. 3.57 and Fig. 3.58. For this we compare the solution for the deflation $u_{init} = \ddot{u}$ for $M_x = M_y = 4$ deflation cells with $u_{init} = 0$ on the following particle distributions

$$N_x = N_y = 20,\ 40,\ 60\ ,80\ ,100. \tag{3.9}$$

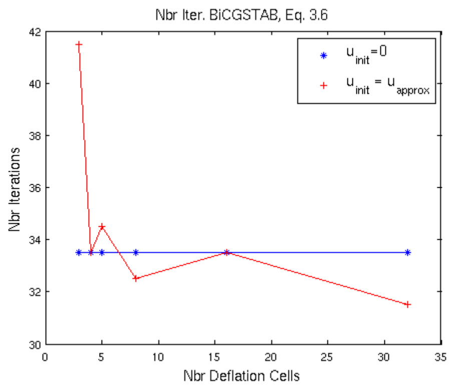

Figure 3.55: Number of iterations needed for convergence in BiCGSTAB for Eq. 3.6, $N_x = 40$, $N_y = 40$, with constant ansatz functions, $u_{init} = \ddot{u}$ and $M_x = M_y = 3, 4, 5, 8, 16, 32$. Compared to convergence for $u_{init} = 0$.

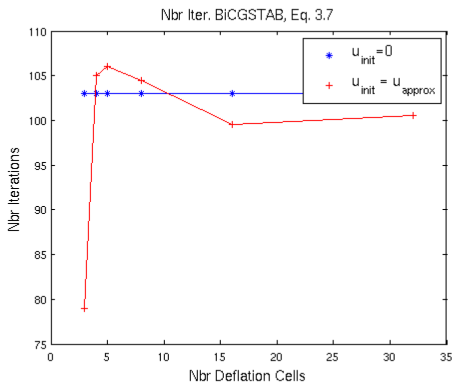

Figure 3.56: Number of iterations needed for convergence in BiCGSTAB for Eq. 3.7, $N_x = 40$, $N_y = 40$, with constant ansatz functions, $u_{init} = \ddot{u}$ and $M_x = M_y = 3, 4, 5, 8, 16, 32$. Compared to convergence for $u_{init} = 0$.

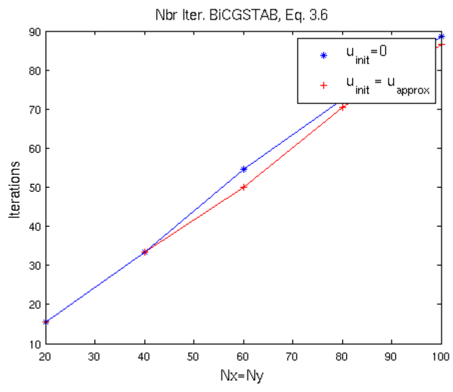

Figure 3.57: Number of iterations needed for convergence in BiCGSTAB for Eq. 3.6, $u_{init} = \ddot{u}$, with constant ansatz functions, $M_x = M_y = 4$ and N_x, N_y as in Eq. 3.9. Compared to convergence for $u_{init} = 0$.

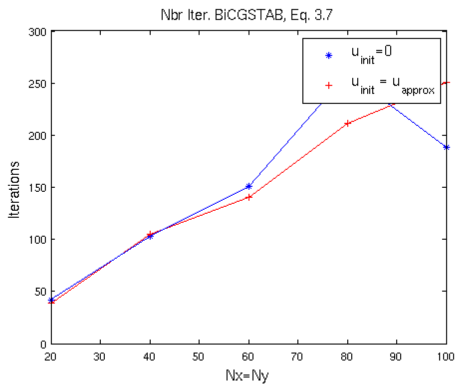

Figure 3.58: Number of iterations needed for convergence in BiCGSTAB for Eq. 3.7, $u_{init} = \ddot{u}$, with constant ansatz functions, $M_x = M_y = 4$ and N_x, N_y as in Eq. 3.9. Compared to convergence for $u_{init} = 0$.

Moreover, we investigate how our deflation behaves in more tube-like domains. Here, we solve Eq. 3.8 in the domain

$$\Omega = [x_0, x_L] \times [y_0, y_L] = [0, 5] \times [0, 0.1], \tag{3.10}$$

with $N_y = 6$ and $dy = dx$. This gives us $N_x = 51$ and $N = N_x N_y = 306$. The exact solution can be found in Fig. 3.59 and the constant approximation is found in Fig. 3.60. How the error and BiCGSTAB convergence change with an increase in M_x can be seen in Fig. 3.61 and Fig. 3.62. The change in convergence and error when M_y changes is similar, which can be seen in Appendix A, in Fig. A.45 and Fig. A.46.

The BiCGSTAB convergence rate is investigated when increasing the length of the domain, as

$$x_L = i, \; i = 1, \ldots, 15, \tag{3.11}$$

while $dy = dx$ is kept constant. How the deflated solution $u_{init} = \ddot{u}$ behaves in comparison to $u_{init} = 0$ can be seen in Fig. 3.63. Here, we have chosen $M_x = 4$ and $M_y = 1$ as these settings

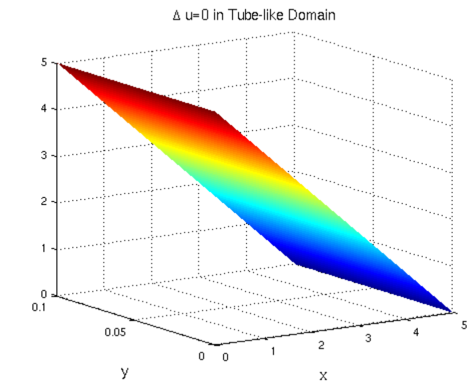

Figure 3.59: Exact solution for Eq. 3.8 in the tube-like domain Ω as in Eq. 3.10.

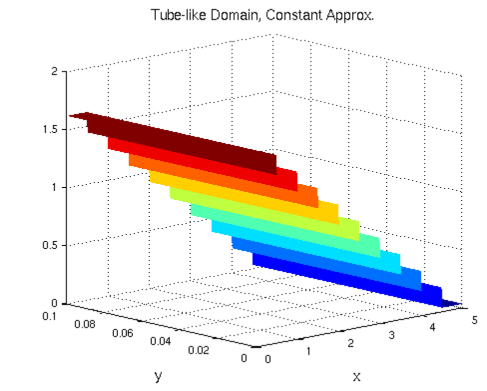

Figure 3.60: Deflated solution with constant approximations for Eq. 3.8 in the tube-like domain Ω as in Eq. 3.10, $M_x = 10$ and $M_y = 1$.

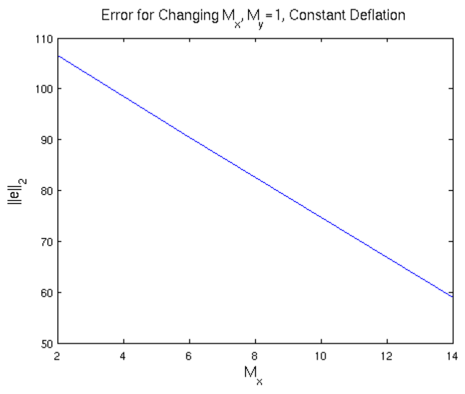

Figure 3.61: Change in error for Eq. 3.8 in the tube-like domain Ω as in Eq. 3.10 when $M_x = 2, 4, 6, 8, 10, 12, 14$ and $M_y = 1$.

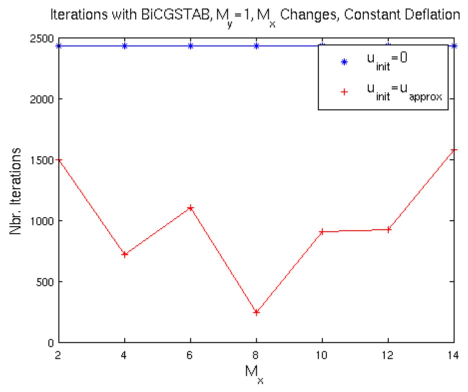

Figure 3.62: Change in convergence with constant approximations for Eq. 3.8 in the tube-like domain Ω as in Eq. 3.10, $M_x = 2, 4, 6, 8, 10, 12, 14$, and $M_y = 1$.

provided fast convergence for Eq. 3.8 in the domain Ω as in Eq. 3.10. It shows that the constant deflation provides an initial guess better than the zero vector for almost all N. At the same time, convergence is still slow.

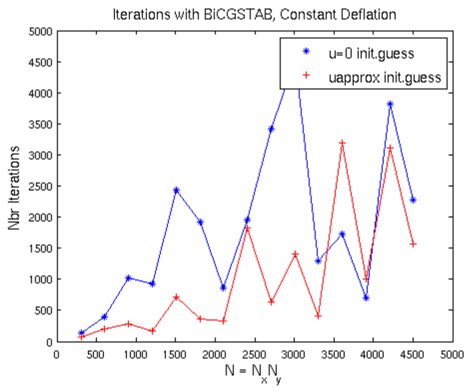

Figure 3.63: Convergence of BiCGSTAB for Eq. 3.8 in the tube-like domain Ω as in Eq. 3.10 with constant approximations, $M_x = 4$, $M_y = 1$.

As a final step, we investigate the results of Eq. 3.8 on the domain in Eq. 3.10 with added obstacles. There are three added obstacles, or holes in the domain, with boundary condition

$$\frac{\delta u}{\delta \vec{n}} = 0.$$

How the solution to this test case will look like can be seen in Fig. 3.64. The constant approximation, using $M_y = 1$ deflation cells in the y-direction and $M_x = 5$ in the x-direction, can be found in Fig. 3.65. There, we see that although the deflation performs well on the Dirichlet boundary, it struggles with the Neumann boundary. Additionally, it seems to have no problems handling the obstacles in the domain. The BiCGSTAB convergence for $u_{init} = 0$ compared to $u_{init} = \ddot{u}$ can be seen in Fig. 3.66. We see that the deflated solution performs better for small N but this does not hold true as N grows.

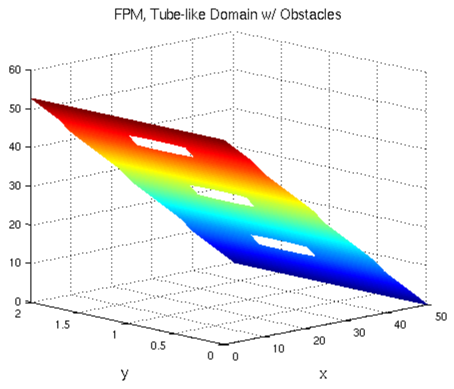

Figure 3.64: Solution to Eq. 3.8 in the tube-like domain Ω in Eq. 3.10 with added obstacles.

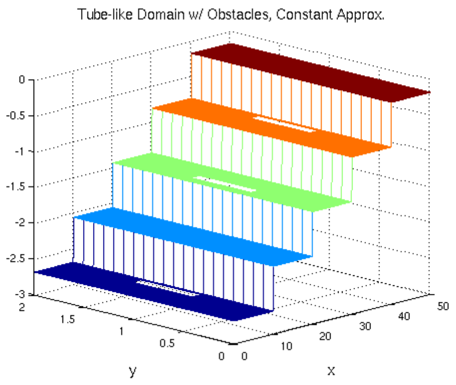

Figure 3.65: Approxoimated solution to Eq. 3.8 in the tube-like domain Ω in Eq. 3.10 with added obstacles, with constant approximations $M_y = 1$, $M_x = 5$.

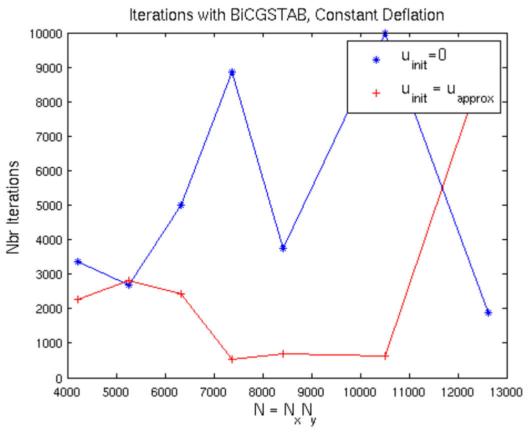

Figure 3.66: Change in convergence of Eq. 3.8 in the tube-like domain Ω in Eq. 3.10 with added obstacles, $M_x = 5$, $M_y = 1$ as N changes.

3.2.2 Linear Approximations

The linear approximation is described in Section 2.2.1. For Eq. 3.6 the approximated solution obtained with linear ansatz functions can be seen in Fig. 3.67 with $M_x = M_y = 4$. Already with few deflation cells it is very close to the exact solution in Fig. 3.49. As expected, the error decreases quickly and is small even for low values of M. How the error changes with an increasing number

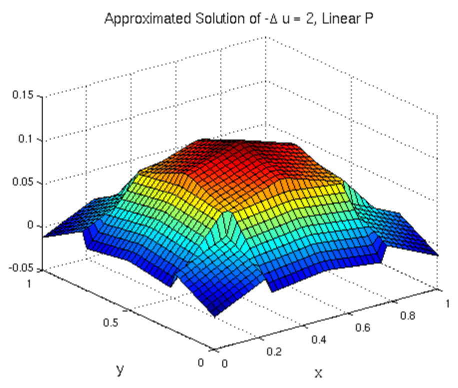

Figure 3.67: Approximated solution of Eq. 3.6, $N_x = 40$, $N_y = 40$, with linear ansatz functions and $M_x = 4$, $M_y = 4$.

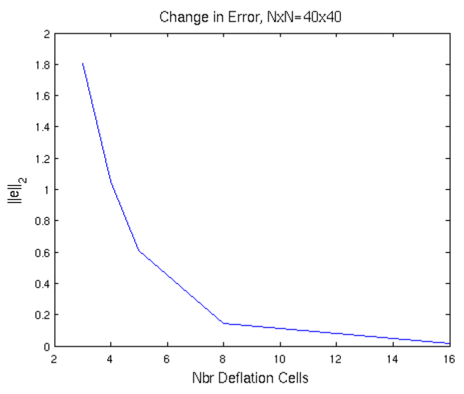

Figure 3.68: Change in error between exact and approximated solution of Eq. 3.6, $N_x = 40$, $N_y = 40$, with linear ansatz functions and $M_x = M_y = 3, 4, 5, 8, 16$.

of deflation cells can be seen in Fig. 3.68. The residuals perform as in the one dimensional case. The condition number of $\bar{A}$ grows rapidly and is in many cases bigger than the condition number for A, as can be seen in Appendix A.

For Eq. 3.7, the approximated solution for $M_x = M_y = 4$ deflation cells with linear ansatz functions is shown in Fig. 3.69. Again, the linear deflation performs well even for small M. When increasing the number of deflation cells the error changes as seen in Fig. 3.70. It decreases as M increases, but is slightly larger than for Eq. 3.6.

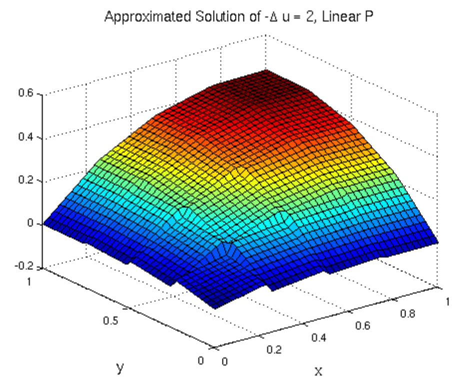

Figure 3.69: Approximated solution of Eq. 3.7, $N_x = 40$, $N_y = 40$, with linear ansatz functions and $M_x = 4$, $M_y = 4$.

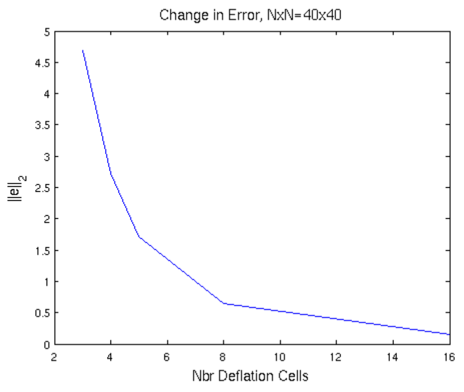

Figure 3.70: Change in error between exact and approximated solution of Eq. 3.7, $N_x = 40$, $N_y = 40$, with linear ansatz functions and $M_x = M_y = 3, 4, 5, 8, 16$.

The convergence of the BiCGSTAB algorithm in MATLAB is investigated for $N_x = N_y = 40$ particles. In Fig. 3.71 and Fig. 3.72 the number of iterations needed for convergence for $u_{init} = \ddot{u}$ where

$$M_x = M_y = 3\ ,4\ ,5\ ,8\ ,\ 16,\ 32,$$

is shown and compared to the number of iterations needed for $u_{init} = 0$. For Eq. 3.6 $u_{init} = u_{approx}$ performs slightly worse than $u_{init} = 0$ for small M. For Eq. 3.7 the deflation always performs better. Furthermore, we investigate how the convergence changes when the number of particles used increases. For this we compare the deflation $u_{init} = \ddot{u}$ for $M_x = M_y = 4$ deflation cells with

$u_{init} = 0$ on the particle distributions as in Eq. 3.9. The results can be seen in Fig. 3.73 and Fig. 3.74. Generally, the deflation is an improvement to $u_{init} = 0$.

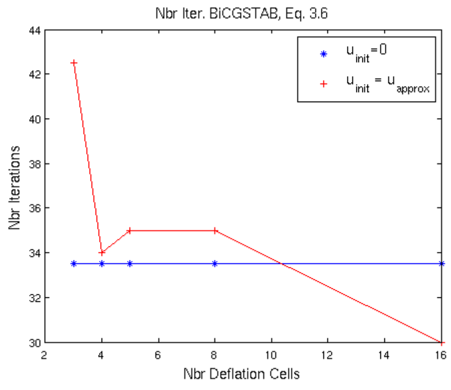

Figure 3.71: Number of iterations needed for convergence in BiCGSTAB for Eq. 3.6, $N_x = 40$, $N_y = 40$, with linear ansatz functions, $u_{init} = \ddot{u}$ and $M_x = M_y = 3, 4, 5, 8, 16$. Compared to convergence for $u_{init} = 0$.

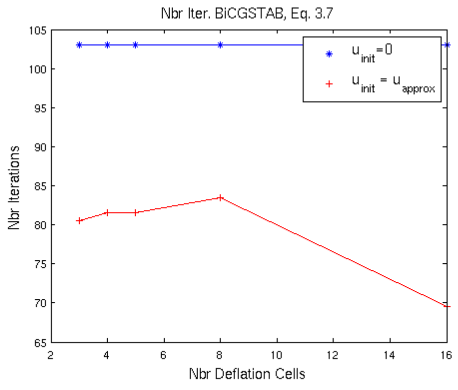

Figure 3.72: Number of iterations needed for convergence in BiCGSTAB for Eq. 3.7, $N_x = 40$, $N_y = 40$, with linear ansatz functions, $u_{init} = \ddot{u}$ and $M_x = M_y = 3, 4, 5, 8, 16$. Compared to convergence for $u_{init} = 0$.

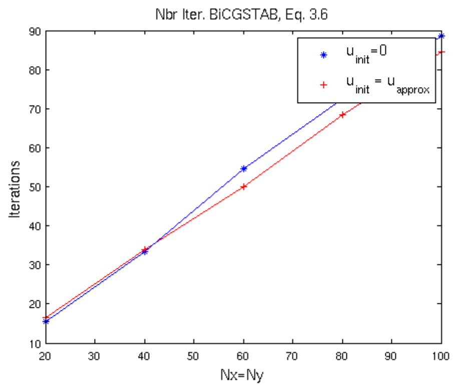

Figure 3.73: Number of iterations needed for convergence in BiCGSTAB for Eq. 3.6, $u_{init} = \ddot{u}$, with linear ansatz functions, $M_x = M_y = 4$ and N_x, N_y as in Eq. 3.9. Compared to convergence for $u_{init} = 0$.

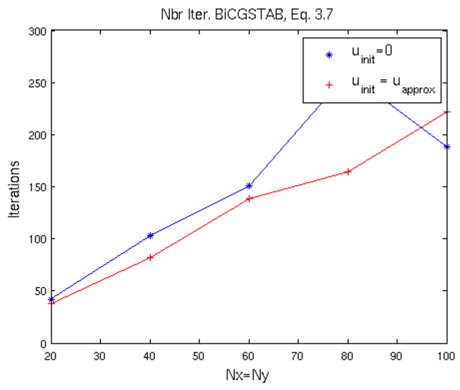

Figure 3.74: Number of iterations needed for convergence in BiCGSTAB for Eq. 3.7, $u_{init} = \ddot{u}$, with linear ansatz functions, $M_x = M_y = 4$ and N_x, N_y as in Eq. 3.9. Compared to convergence for $u_{init} = 0$.

Also, we investigate how our deflation behaves in more tube-like domains. Here, we solve Eq. 3.8 in the domain given by Eq. 3.10 with $N_y = 6$ and $dy = dx$. The exact solution can be found in Fig. 3.59 and the linear approximation is found in Fig. 3.75. When M changes as $M_x = 2$, 4, 8, 10, and $M_y = 1$ the error for the linear approximation stays constant at $\|e\|_2 = 13.4112$. The convergence changes as in Fig. 3.77. When changing M_y and keeping M_x constant, the error and convergence rate change as in Fig. A.47 and Fig. A.48 in Appendix A.

The BiCGSTAB convergence rate is then investigated when increasing the length of the domain, as

$$x_L = i, \; i = 1, \ldots, 15, \tag{3.12}$$

while $dy = dx$ is kept constant. How the deflated solution $u_{init} = \ddot{u}$ behaves in comparison to $u_{init} = 0$ can be seen in Fig. 3.78. It shows that the constant deflation provides an initial guess

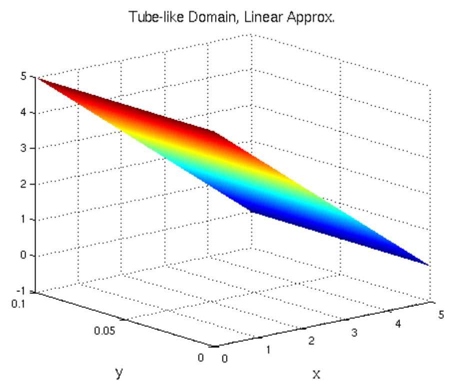

Figure 3.75: Deflated solution with linear approximations for Eq. 3.8 in the tube-like domain as in Eq. 3.10, $M_x = 10$ and $M_y = 1$.

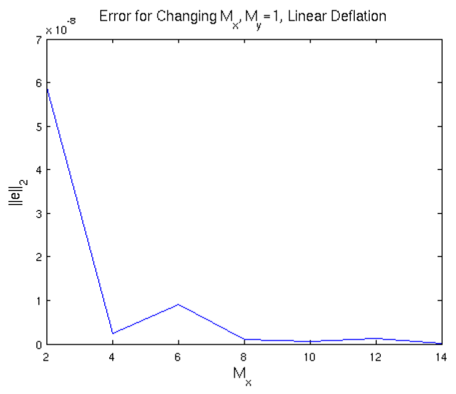

Figure 3.76: How the error between the exact and the linear deflated solutions changes for Eq. 3.7 in the tube-like domain Ω as in Eq. 3.10, $M_x = 2$, 4, 8, 10, $M_y = 1$ and N_x, N_y as in Eq. 3.10.

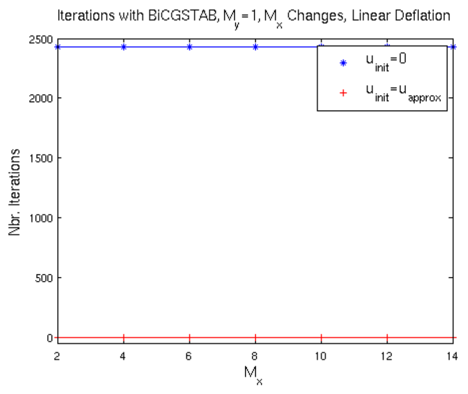

Figure 3.77: Number of iterations needed for convergence in BiCGSTAB for Eq. 3.7 in the tube-like domain Ω as in Eq. 3.10, $u_{init} = \ddot{u}$, with linear ansatz functions, $M_x = 2$, 4, 8, 10, $M_y = 1$ and N_x, N_y as in Eq. 3.9. Compared to convergence for $u_{init} = 0$.

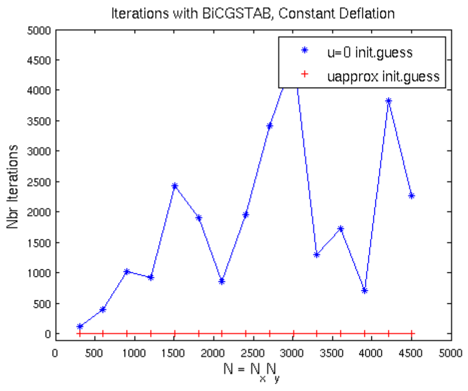

Figure 3.78: Convergence of BiCGSTAB for Eq. 3.8 in the tube-like domain Ω as in Eq. 3.10 with linear approximations for changing N, $M_x = 2$, $M_y = 1$.

better than the zero vector for all N. The deflation provides a vast improvement in convergence rate, as it gives instant convergence in BiCGSTAB whereas $u_{init} = 0$ needs a rapidly growing number of iterations.

Finally, we investigate how the linear approximations behave in the tube-like domain Ω with added obstacles. The deflated solution can be found in Fig. 3.79 for $M_x = 5$, $M_y = 1$, deflation cells. We see that it looks very close to the exact solution. The improvement in convergence can be seen in Fig. 3.80.

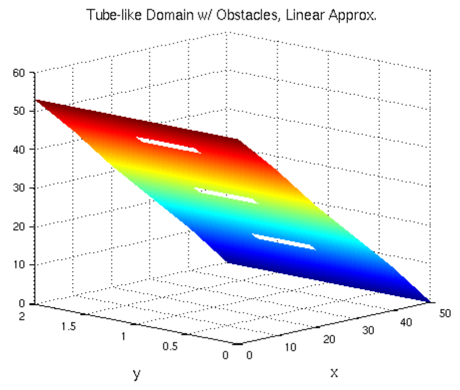

Figure 3.79: Deflated solution with linear approximations for Eq. 3.8 in the tube-like domain as in Eq. 3.10 with added obstacles, $M_x = 5$ and $M_y = 1$.

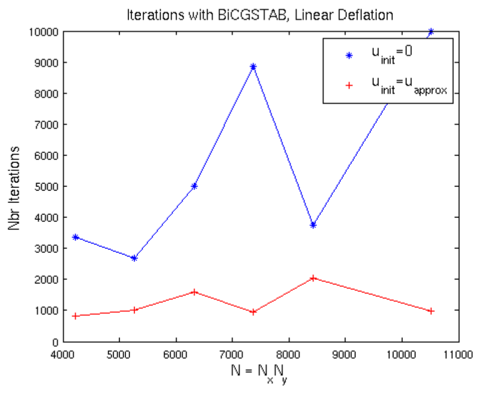

Figure 3.80: Number of iterations needed for convergence in BiCGSTAB for Eq. 3.7 in the tube-like domain Ω as in Eq. 3.10 with added obstacles, $u_{init} = \ddot{u}$, with linear ansatz functions, $M_x = 2, 4, 8, 10$, $M_y = 1$ and N_x, N_y as in Eq. 3.9. Compared to convergence for $u_{init} = 0$.

Chapter 4

Discussion and Conclusions

In this chapter we discuss the results shown in Chapter 3, as well as give suggestions for further work. Then we conclude with some final remarks regarding the deflation of FPM.

4.1 Discussion

We divide the discussion of our results into three main parts for the different ansatz functions used before comparing the three.

4.1.1 Constant Approximations

The deflation with constant ansatz functions works well in some cases, and not so well in others. Mainly, deflating problems with Dirichlet boundary conditions works better than deflating those with mixed boundary conditions since the deflation struggles on the Neumann boundaries. How to approximate Neumann boundaries with constant approximations is a problem both in one and two dimensions. As can be seen in Fig. 3.5 and Fig. 3.6, increasing the number of deflation cells M will not necessarily take the approximated solution nearer to the exact solution. Given the nature of the Neumann boundary conditions, this is not surprising. By performing a constant approximation on such a boundary, all information about the slope will be lost. A constant approximation will act as if the Neumann condition is homogeneous, thus with slope zero. That is why, in two dimensions, the constant approximation performs better for the Neumann boundaries for Eq. 3.7 than for either Eq. 3.3 or Eq. 3.4, see Fig. 3.2 compared to Fig. 3.5 and Fig. 3.6.

When deflating the system for problems such as Eq. 3.1 and Eq. 3.6 with homogeneous Dirichlet boundary conditions, the only problem with the approximated solution is the scale. The change in value between the cells is often not very big, and by normalising our system we have put extra emphasis on the boundary particles and their conditions. Contrary to the case for Neumann boundary conditions, for Dirichlet boundary conditions an increase in the number of deflation cells leads to a better approximated solution.

The convergence in BiCGSTAB is not affected significantly by the use of deflation in these cases, but rather worsened than improved. This leads us to the conclusion that constant approximations will not be worthwhile their computational effort for improving the FPM algorithm.

When deflating Eq. 3.8 in a long, narrow domain with constant approximations, again we struggle with the non-homogeneous Neumann boundary. Still, for small N our deflation gives an improvement on the BiCGSTAB convergence. As N grows, however, this improvement decreases.

The addition of obstacles to the tube-like domain does not change the constant approximations drastically. The improvement in convergence stays more or less the same, thus not always significantly better. The deflation has no issues dealing with the obstacles in the domain.

4.1.2 Linear Approximation

Obviously, a linear approximation of a linear function, such as the solution to Eq. 3.2, is equal to the exact solution independently of the number of deflation cells used, see e.g. Fig. 3.18. This result holds for both Dirichlet and Neumann boundary conditions. Additionally, using the exact solution as initial guess will of course lead to immediate convergence of the BiCGSTAB algorithm, see Fig. 3.26 and Fig. 3.27.

Although not as good, the linear ansatz functions also provide a good deflated solution to systems where the solution is not linear, e.g. to Eq. 3.1 and Eq. 3.4. Also, the more deflation cells used, the better the solution becomes. For these two test cases, the deflated solution does not provide a better initial guess to the BiCGSTAB algorithm than the zero-vector. We can therefore conclude that for linear deflation, the boundary conditions play a much lesser role in deciding the quality of the approximation than in the constant case. Instead, the source term will affect the behaviour of our deflation to a much greater extent.

In two dimensions, the linear approximation provides good results for both homogeneous Dirichlet and mixed boundary conditions. Even for only $M_x \times M_y = 4 \times 4$ deflation cells the deflated solution comes close to the true solution. As the number of deflation cells increases the error decreases rapidly. For both Eq. 3.6 and Eq. 3.7 our deflation gives a slight improvement in convergence rate, although not very significant.

The solution in the narrow domain in Eq. 3.8 is approximated well by our linear deflation. With only a few deflation cells it gives very accurate results. As initial guess to BiCGSTAB it gives a great improvement on the convergence rate, with instant convergence in comparison to the thousands of iterations needed for $u_{init} = 0$. The computational effort to obtain the deflation is therefore more than made up for in improved convergence rate.

When adding obstacles to the domain the linear deflation still gives an approximated solution very close to the exact solution. It also improves the convergence rate greatly. While the convergence rate with the deflated solution as intial guess stays more or less the same at approximately 1000 iterations, the convergence rate with $u_{init} = 0$ grows as N increases.

4.1.3 Quadratic Approximation

The quadratic approximation provides good deflated solutions for all our test cases as long as M is chosen small enough. The convergence of the BiCGSTAB algorithm is instantaneous for all cases.

Worth noting is that although the convergence of our iterative method improves drastically with the use of quadratic approximations, the computational effort needed to compute these approximations also increases. In one dimension, the cost is hardly noticable, but in higher dimensions this might not be the case.

4.1.4 Comparisons

Comparing the three different approximations, one can easily conclude that simply regarding the convergence rate of BiCGSTAB, the approximation of choice should be the quadratic ansatz functions. At the same time, the linear approximation performs well for all test cases in one and two dimensions for both kinds of boundary conditions. For linear solutions the convergence is even instant. Worst does the constant approximation perform, but it might still be useful for Dirichlet problems. Weighting computational cost against accuracy, the linear approximation seems to be the better choice.

In general, for all approximations the residuals are always relatively small even when the error is not, which shows they are not a preferable way of evaluating our system. The condition number for the deflated system is the smallest for constant approximations. Better approximations give a higher condition number that also increases faster for larger numbers of deflation cells. This is also something that should be taken into consideration when choosing a deflation.

Comparing the results for FPM with those for FDM in two of our test cases, we see that they are very similar. Thus, the deflation is not affected much by which system we are deflating. This is hardly surprising, since we have used uniform particle distributions in both cases.

4.1.5 Future Work

This deflation could be investigated further, especially by considering more complex geometries. Also geometries in three dimensions should be looked at. Additionally, to improve the deflation one could consider other approximations than the three considered here. Different ways of obtaining the deflated solution could also be investigated.

4.2 Conclusions

In this master's thesis different ways of deflating FPM have been constructed and investigated. A method has been constructed where the domain is divided into a given number of deflation cells on which different ansatz functions are defined. Constant, linear and quadratic test functions have been used. The particles of FPM in each deflation cell are restricted to a number of computational points per cell, resulting in a much smaller linear system to solve. The result from this system is then mapped to the particles within the deflation cells. The deflated solution should be used as an initial guess to the BiCGSTAB algorithm.

Using constant approximations works well for Dirichlet problems without source terms, but struggles otherwise. Linear approximations handle all test cases well and show convergence as the number of deflation cells increases. A linear deflated solution as initial guess to BiCGSTAB improves the convergence rates for a suitable number of deflation cells. Also, to obtain the deflated solution is computationally cheap. The quadratic approximations prove very exact for all test cases and a few deflation cells. As the number of deflation cells increases, however, the deflated system becomes problematic to solve. For both constant and linear approximations we see an advantage in using deflation in the convergence of BiCGSTAB for problems in long and narrow domains. Especially for linear approximations this is true, since they yield instant convergence.

Bibliography

[1] A. Abdel-Rehim, R. B. Morgan, and W. Wilcox. Deflated bicgstab for linear equations in qcd problems. *PoS*, LATTICE 2007(arXiv:0710.1988. BU-HEPP-07-13):026. 7 p, Oct 2007.

[2] J. C. A. Barata and M. S. Hussein. The moore–penrose pseudoinverse: A tutorial review of the theory. *Brazilian Journal of Physics*, 42(1-2):146–165, 2012.

[3] O. Coulaud, L. Giraud, P. Ramet, and X. Vasseur. Deflation and augmentation techniques in Krylov linear solvers. Rapport de recherche RR-8265, INRIA, February 2013. Preliminary version of the book chapter entitled 'Deflation and augmentation techniques in Krylov linear solvers" published in 'Developments in Parallel, Distributed, Grid and Cloud Computing for Engineering", ed. Topping, B.H.V and Ivanyi, P., Saxe-Coburg Publications, Kippen, Stirlingshire, United Kingdom, ISBN 978-1-874672-62-3, p. 249-275, 2013.

[4] C. Drumm, S. Tiwari, J. Kuhnert, and H.-J. Bart. Finite pointset method for simulation of the liquid–liquid flow field in an extractor. *Computers and Chemical Engineering*, 32(12):2946 – 2957, 2008.

[5] R. A. Gingold and J. J. Monaghan. Smoothed particle hydrodynamics - theory and application to non-spherical stars. *Monthly Notices of the Royal Astronomical Society*, 181:375–389, 1977.

[6] A. Iserles. *A First Course in the Numerical Analysis of Differential Equations*. Cambridge University Press, New York, NY, USA, 2nd edition, 2008.

[7] J. Kuhnert. *General smoothed particle hydrodynamics*. Dissertation, Departments of Mathematics, University of Kaiserslautern, 1999.

[8] L. B. Lucy. A numerical approach to the testing of the fission hypothesis. *Astronomical Journal*, 82:1013–1024, 1977.

[9] L. Mansfield. On the use of deflation to improve the convergence of conjugate gradient iteration. *Communications in Applied Numerical Methods*, 4(2):151–156, 1988.

[10] J. J. Monaghan. Smoothed particle hydrodynamics. *Reports on Progress in Physics*, 68(8):1703, 2005.

[11] R. Nabben and C. Vuik. A comparison of deflation and the balancing preconditioner. *SIAM Journal on Scientific Computing*, 27(5):1742 – 1759, 2006.

[12] R.A. Nicolaides. Deflation of conjugate gradients with applications to boundary value problems. *SIAM Journal on Numerical Analysis*, 24(2):355–365, 1987.

[13] J. F. Price. Lagrangian and eulerian representations of fluid flow: Kinematics and the equations of motion. Woods Hole Oceanographic Institution, Woods Hole MA, June 2006.

[14] Y. Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, 2 edition, 2003.

[15] B. Seibold. *Multigrid and M-matrices in the Finite Pointset Method for incompressible flows.*, volume 57 of *Lecture Notes in Computational Science and Engineering*. Technische Universität Kaiserslautern, Fachbereich Mathematik, 2007.

[16] A.B. Subramanyam, S. Sundar, A. Saxena, J. Kuhnert, S. Tiwari, and A. Joshi. On parallelization and load balancing aspects of the finite-pointset method. *International Journal of Computer Mathematics*, 88(2):360–374, 2011.

[17] S. Tiwari. A LSQ-SPH approach for solving compressible viscous flows. *International series of numerical mathematics*, 141, 2000.

[18] S. Tiwari and J. Kuhnert. Finite pointset method based on the projection method for simulations of the incompressible navier-stokes equations. In *Lecture Notes in Computational Science and Engineering: Meshfree Methods for Partial Differential Equations*. Springer, 2003.

[19] S. Tiwari and J. Kuhnert. Grid free method for solving poisson equation. *Wavelet Analysis and Applications*, pages 151–166, 2004.

[20] S. Tiwari and J. Kuhnert. Modeling of two-phase flows with surface tension by finite pointset method (FPM). *Journal of Computational and Applied Mathematics*, 203(2):376 – 386, 2007. Special Issue: The first Indo-German Conference on PDE, Scientific Computing and Optimization in Applications.

Appendix A

Further Results for FPM

Results 1D

Here, further results for the four test cases Eq. 3.1, Eq. 3.2, Eq. 3.3 and Eq. 3.4 are presented. First we investigate the condition numbers of the original FPM and the deflated systems further. Second, the residuals of the different deflations are presented.

Condition Numbers

We investigate how the condition numbers of the linear systems generated by FPM, A, and that of the deflated system, $\bar{A}$, change as the number of particles N changes. Moreover, we examine how the condition number of the deflated system changes as M grows, where $N = 2050$.

Constant Approximations

The constant approximations are as described in Section 2.2.1.

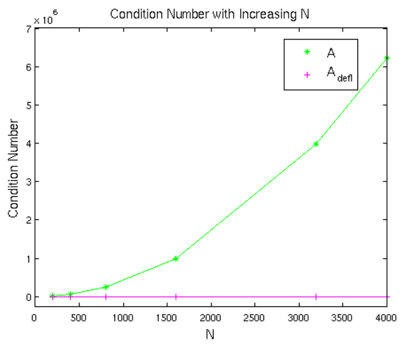

Figure A.1: Change in condition number of the original FPM system for Eq. 3.1 and for the constant deflated solution as N grows, $M = 3$.

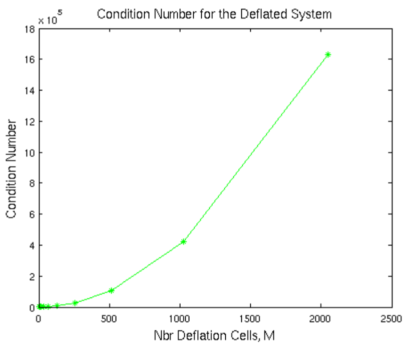

Figure A.2: Change in condition number for $\bar{A}$, cond($\bar{A}$), of Eq. 3.1 using constant ansatz functions, $N = 2050$, with increasing number of deflation cells $M = 2^i$, $i = 1, \ldots, 11$.

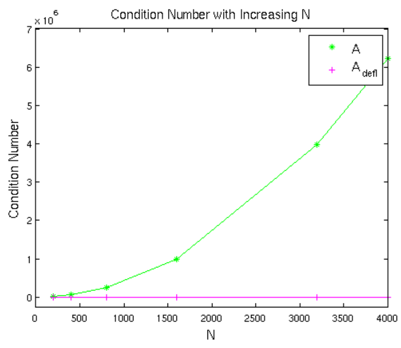

Figure A.3: Change in condition number of the original FPM system for Eq. 3.2 and for the constant deflated solution as N grows, $M = 3$.

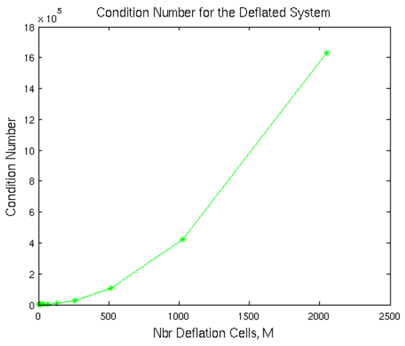

Figure A.4: Change in condition number for $\bar{A}$, cond($\bar{A}$), of Eq. 3.2 using constant ansatz functions, $N = 2050$, with increasing number of deflation cells $M = 2^i$, $i = 1, \ldots, 11$.

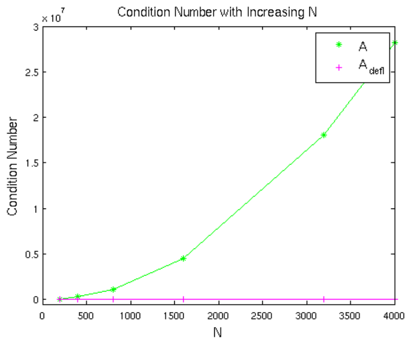

Figure A.5: Change in condition number of the original FPM system for Eq. 3.3 and for the constant deflated solution as N grows, $M = 3$.

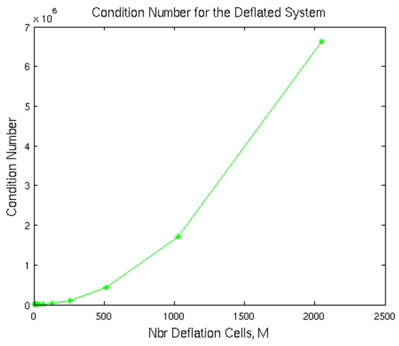

Figure A.6: Change in condition number for $\bar{A}$, cond($\bar{A}$), of Eq. 3.3 using constant ansatz functions, $N = 2050$, with increasing number of deflation cells $M = 2^i$, $i = 1, \ldots, 11$.

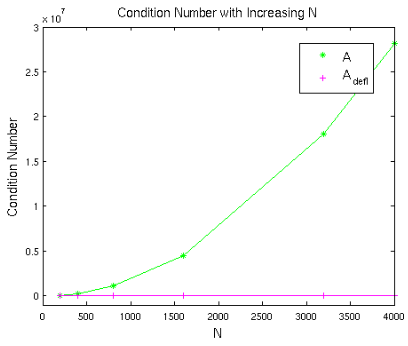

Figure A.7: Change in condition number of the original FPM system for Eq. 3.4 and for the constant deflated solution as N grows, $M = 3$.

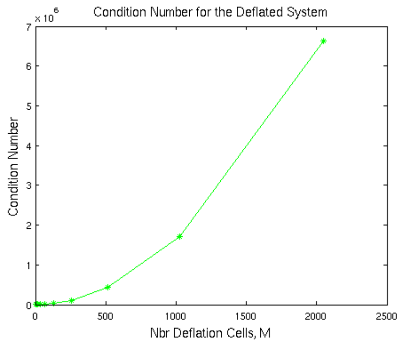

Figure A.8: Change in condition number for $\bar{A}$, cond($\bar{A}$), of Eq. 3.4 using constant ansatz functions, $N = 2050$, with increasing number of deflation cells $M = 2^i$, $i = 1, \ldots, 11$.

Linear Approximations

The linear approximations are constructed as in Section 2.2.1.

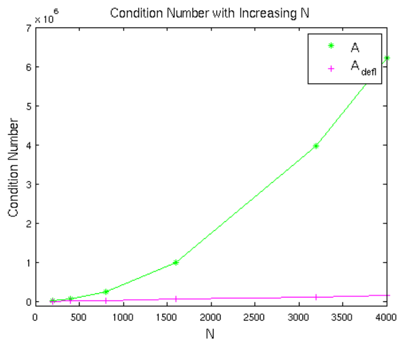

Figure A.9: Change in condition number of the original FPM system for Eq. 3.1 and for the linear deflated solution as N grows, $M = 3$.

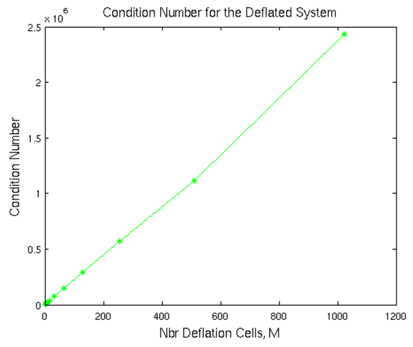

Figure A.10: Change in condition number for $\bar{A}$, cond($\bar{A}$), of Eq. 3.1 using linear ansatz functions, $N = 2050$, with increasing number of deflation cells $M = 2^i$, $i = 1, \ldots, 10$.

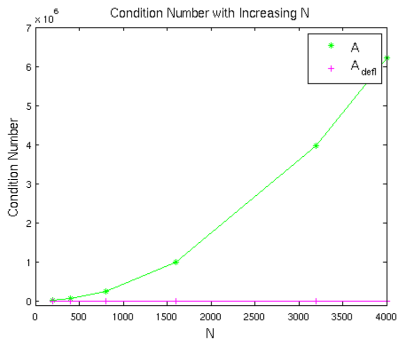

Figure A.11: Change in condition number of the original FPM system for Eq. 3.2 and for the linear deflated solution as N grows, $M = 3$.

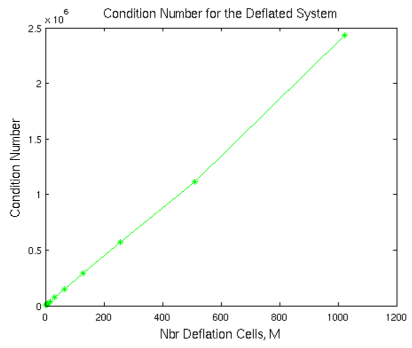

Figure A.12: Change in condition number for $\bar{A}$, cond($\bar{A}$), of Eq. 3.2 using linear ansatz functions, $N = 2050$, with increasing number of deflation cells $M = 2^i$, $i = 1, \ldots, 10$.

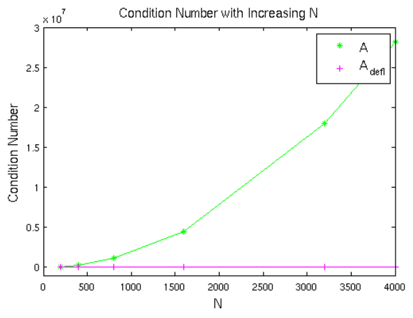

Figure A.13: Change in condition number of the original FPM system for Eq. 3.3 and for the linear deflated solution as N grows, $M = 3$.

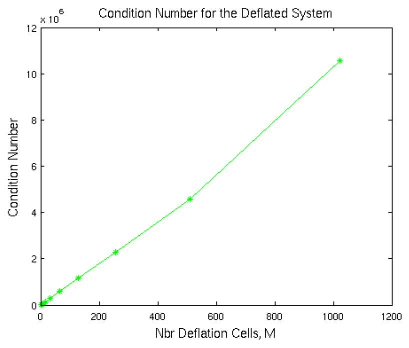

Figure A.14: Change in condition number for $\bar{A}$, cond($\bar{A}$), of Eq. 3.3 using linear ansatz functions, $N = 2050$, with increasing number of deflation cells $M = 2^i$, $i = 1, \ldots, 10$.

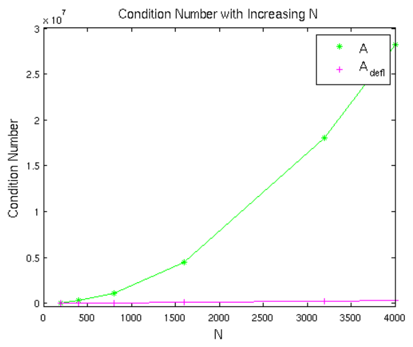

Figure A.15: Change in condition number of the original FPM system for Eq. 3.4 and for the linear deflated solution as N grows, $M = 3$.

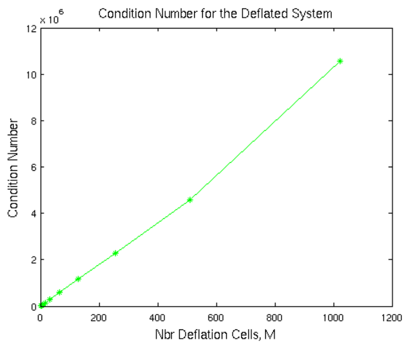

Figure A.16: Change in condition number for $\bar{A}$, cond($\bar{A}$), of Eq. 3.4 using linear ansatz functions, $N = 2050$, with increasing number of deflation cells $M = 2^i$, $i = 1, \ldots, 10$.

Quadratic Approximations

The quadratic approximations are made as in Section 2.2.1.

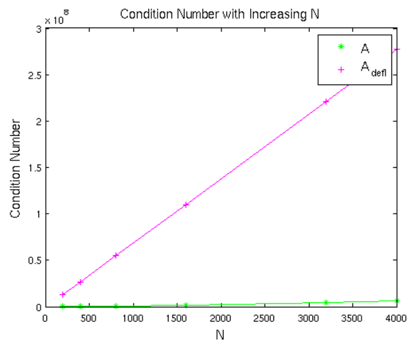

Figure A.17: Change in condition number of the original FPM system for Eq. 3.1 and for the quadratic deflated solution as N grows, $M = 3$.

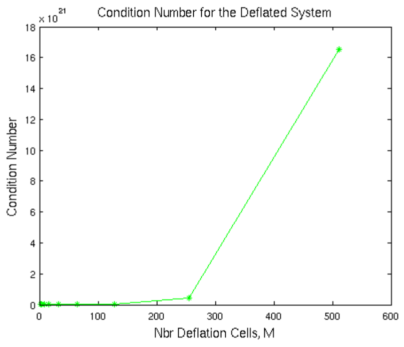

Figure A.18: Change in condition number for $\bar{A}$, cond$(\bar{A})$, of Eq. 3.1 using quadratic ansatz functions, $N = 2050$, with increasing number of deflation cells $M = 2^i$, $i = 1, \ldots, 9$.

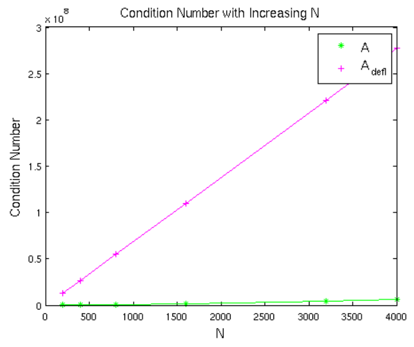

Figure A.19: Change in condition number of the original FPM system for Eq. 3.2 and for the quadratic deflated solution as N grows, $M = 3$.

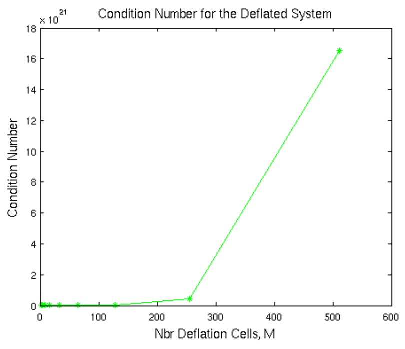

Figure A.20: Change in condition number for $\bar{A}$, cond$(\bar{A})$, of Eq. 3.2 using quadratic ansatz functions, $N = 2050$, with increasing number of deflation cells $M = 2^i$, $i = 1, \ldots, 9$.

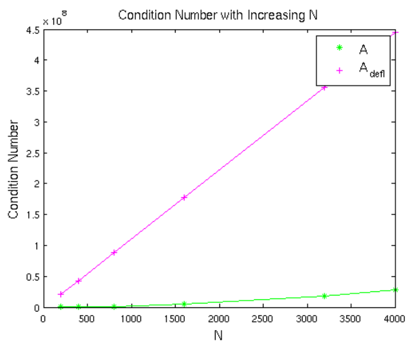

Figure A.21: Change in condition number of the original FPM system for Eq. 3.3 and for the quadratic deflated solution as N grows, $M = 3$.

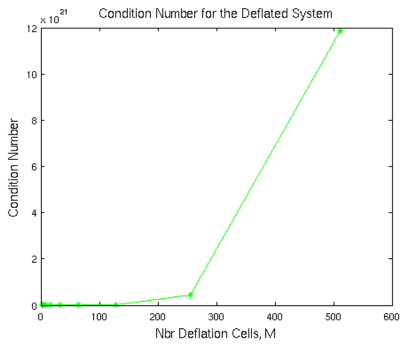

Figure A.22: Change in condition number for $\bar{A}$, cond($\bar{A}$), of Eq. 3.3 using quadratic ansatz functions, $N = 2050$, with increasing number of deflation cells $M = 2^i$, $i = 1, \ldots, 9$.

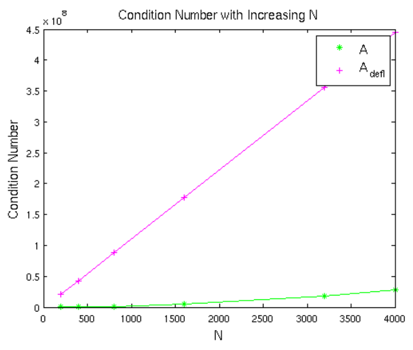

Figure A.23: Change in condition number of the original FPM system for Eq. 3.4 and for the quadratic deflated solution as N grows, $M = 3$.

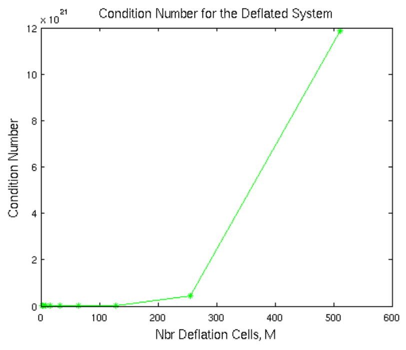

Figure A.24: Change in condition number for $\bar{A}$, cond($\bar{A}$), of Eq. 3.4 using quadratic ansatz functions, $N = 2050$, with increasing number of deflation cells $M = 2^i$, $i = 1, \ldots, 9$.

Residual Plots

We investigate the residuals as mentioned in Section 2.2.3.

Constant Approximations

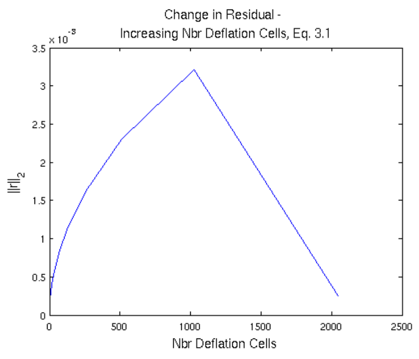

Figure A.25: Change in residual between exact and approximated solutions of Eq. 3.1 using constant ansatz functions, $N = 2050$, with increasing number of deflation cells $M = 2^i$, $i = 1, \ldots, 11$.

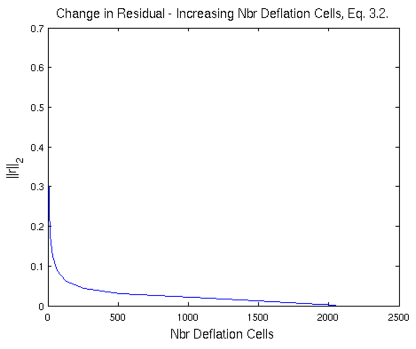

Figure A.26: Change in residual between exact and approximated solutions of Eq. 3.2 using constant ansatz functions, $N = 2050$, with increasing number of deflation cells $M = 2^i$, $i = 1, \ldots, 11$.

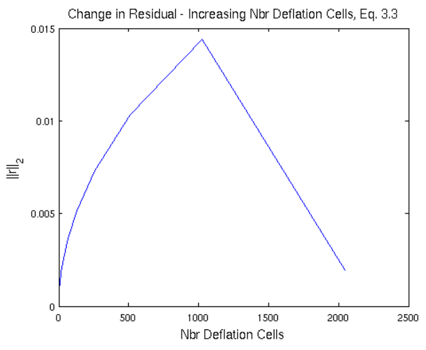

Figure A.27: Change in residual for Eq. 3.3 using constant ansatz functions, $N = 2050$, with increasing number of deflation cells $M = 2^i$, $i = 1, \ldots, 11$.

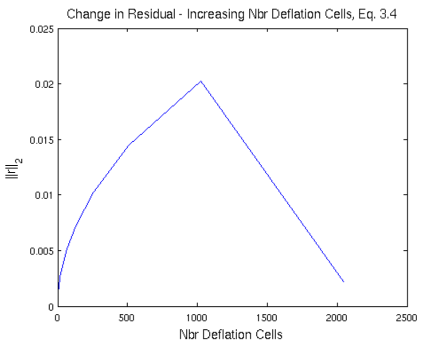

Figure A.28: Change in residual for Eq. 3.4 using constant ansatz functions, $N = 2050$, with increasing number of deflation cells $M = 2^i$, $i = 1, \ldots, 11$.

Linear Approximations

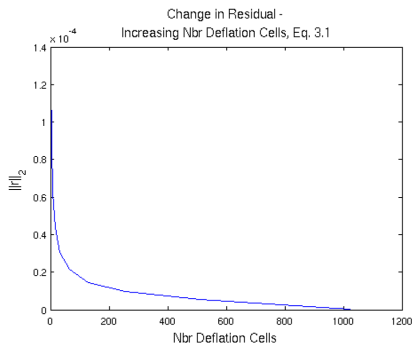

Figure A.29: Change in residuals for Eq. 3.1 using linear ansatz functions, $N = 2050$, with increasing number of deflation cells $M = 2^i$, $i = 1, \ldots, 10$.

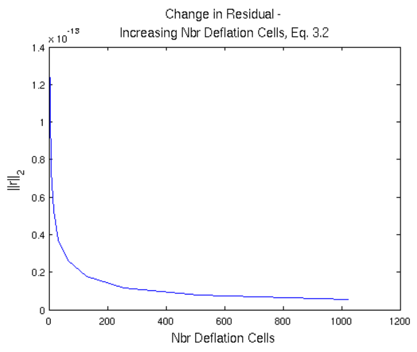

Figure A.30: Change in residuals for Eq. 3.2 using linear ansatz functions, $N = 2050$, with increasing number of deflation cells $M = 2^i$, $i = 1, \ldots, 10$.

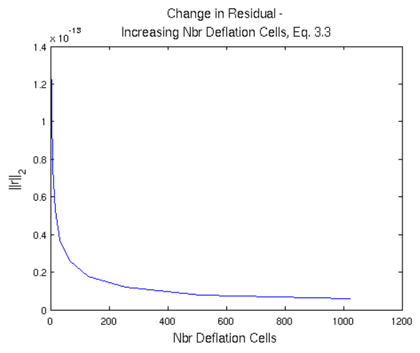

Figure A.31: Change in residuals for Eq. 3.3 using linear ansatz functions, $N = 2050$, with increasing number of deflation cells $M = 2^i$, $i = 1, \ldots, 10$.

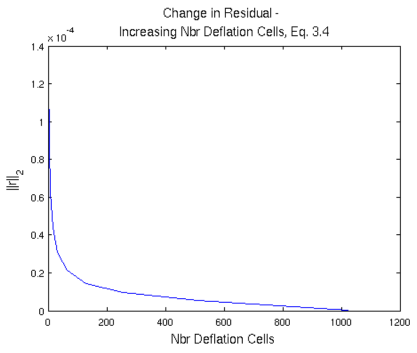

Figure A.32: Change in residuals for Eq. 3.4 using linear ansatz functions, $N = 2050$, with increasing number of deflation cells $M = 2^i$, $i = 1, \ldots, 10$.

Quadratic Approximations

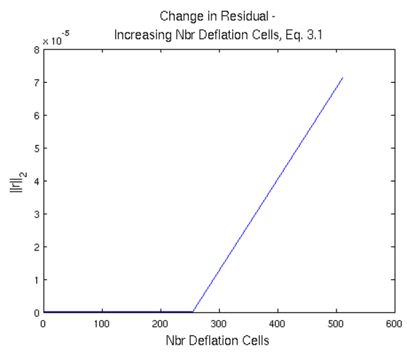

Figure A.33: Change in residuals for Eq. 3.1 using quadratic ansatz functions, $N = 2050$, with increasing number of deflation cells $M = 2^i$, $i = 1, \ldots, 5$.

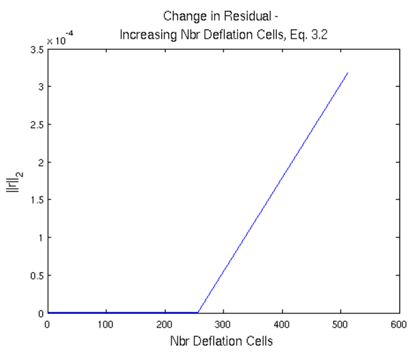

Figure A.34: Change in residuals for Eq. 3.2 using quadratic ansatz functions, $N = 2050$, with increasing number of deflation cells $M = 2^i$, $i = 1, \ldots, 5$.

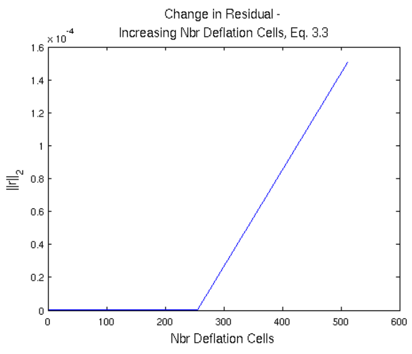

Figure A.35: Change in residuals for Eq. 3.3 using quadratic ansatz functions, $N = 2050$, with increasing number of deflation cells $M = 2^i$, $i = 1, \ldots, 9$.

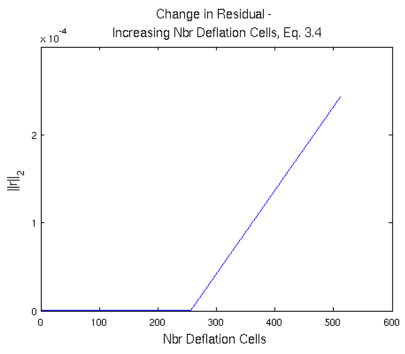

Figure A.36: Change in residuals for Eq. 3.4 using quadratic ansatz functions, $N = 2050$, with increasing number of deflation cells $M = 2^i$, $i = 1, \ldots, 9$.

Results 2D

Condition Numbers

We investigate the condition numbers of A and $\bar{A}$, both as N and M increases.

Constant Approximations

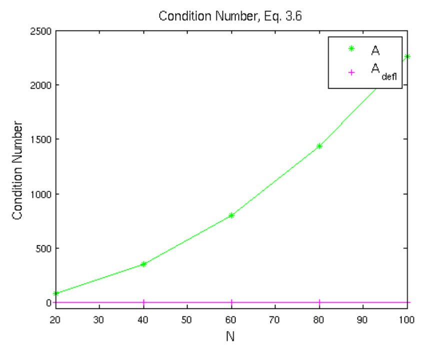

Figure A.37: Change in condition number of the original FPM system for Eq. 3.6 and for the constant deflated solution as N grows, $M_x = M_y = 4$.

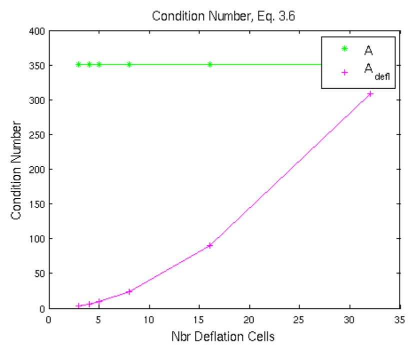

Figure A.38: Change in condition number of the original FPM system for Eq. 3.6 and for the constant deflated solution as M grows, $N_x = 40$, $N_y = 40$.

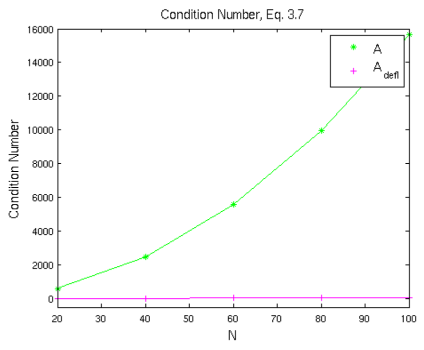

Figure A.39: Change in condition number of the original FPM system for Eq. 3.7 and for the constant deflated solution as N grows, $M_x = M_y = 4$.

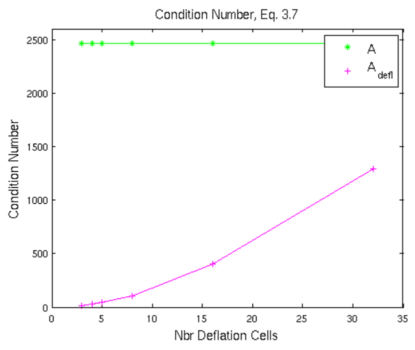

Figure A.40: Change in condition number of the original FPM system for Eq. 3.7 and for the constant deflated solution as M grows, $N_x = 40$, $N_y = 40$.

Linear Approximations

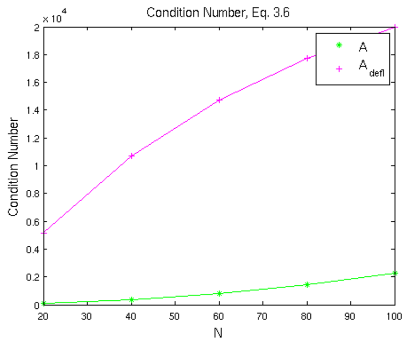

Figure A.41: Change in condition number of the original FPM system for Eq. 3.6 and for the linear deflated solution as N grows, $M_x = M_y = 4$.

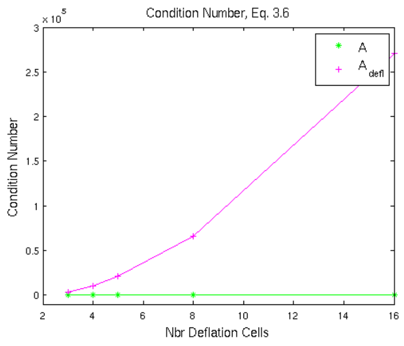

Figure A.42: Change in condition number of the original FPM system for Eq. 3.6 and for the linear deflated solution as M grows, $N_x = 40$, $N_y = 40$.

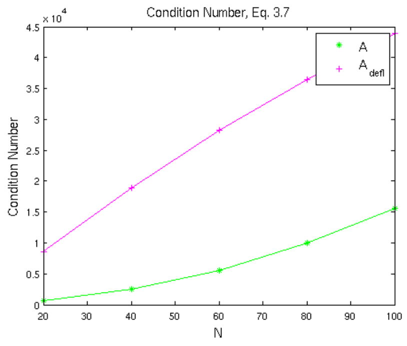

Figure A.43: Change in condition number of the original FPM system for Eq. 3.7 and for the linear deflated solution as N grows, $M_x = M_y = 4$.

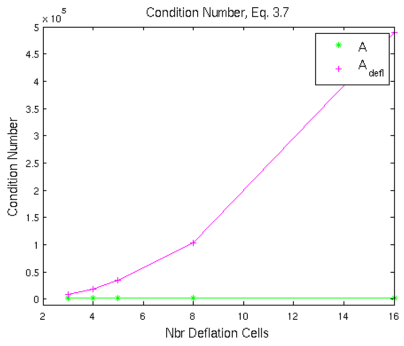

Figure A.44: Change in condition number of the original FPM system for Eq. 3.7 and for the linear deflated solution as M grows, $N_x = 40$, $N_y = 40$.

Convergence Rates and Errors

We investigate the error as evaluated in Section 2.2.3 and the rate of convergence in BiCGSTAB as M_y changes.

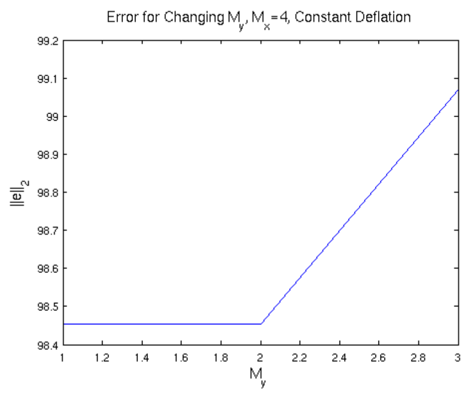

Figure A.45: Change in error for Eq. 3.8 in the tube-like domain Ω as in Eq. 3.10 when $M_y = 1, 2, 3$ and $M_x = 4$, for constant approximations.

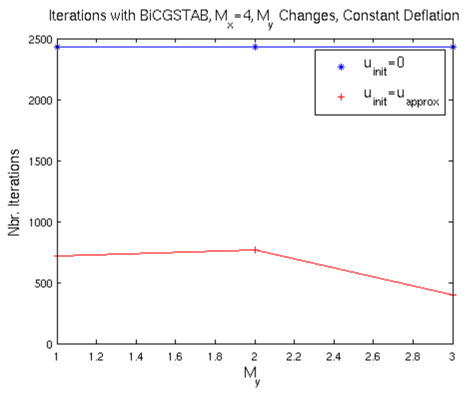

Figure A.46: Change in convergence with constant approximations for Eq. 3.8 in the tube-like domain Ω as in Eq. 3.10, $M_y = 1, 2, 3$ and $M_x = 4$.

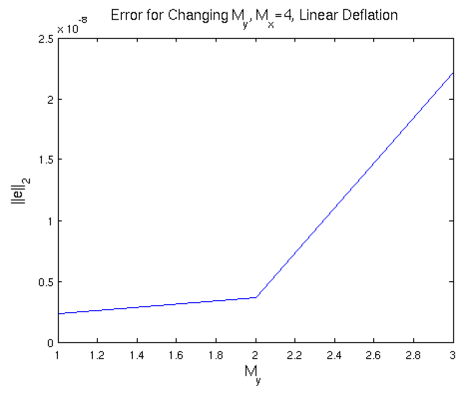

Figure A.47: Change in error for Eq. 3.8 in the tube-like domain Ω as in Eq. 3.10 when $M_y = 1, 2, 3$ and $M_x = 4$, for linear approximations.

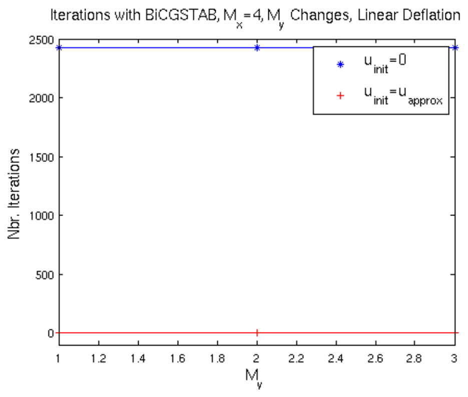

Figure A.48: Change in convergence with linear approximations for Eq. 3.8 in the tube-like domain Ω as in Eq. 3.10, $M_y = 1, 2, 3$ and $M_x = 4$.

Appendix B

Results for FDM

The FDM systems are constructed as in Section 2.1.1. We test constant and linear approximations for the test case Eq. 3.3 in one dimension and Eq. 3.7 in two dimensions.

Results 1D

The test case in one dimension is Eq. 3.3. It has a Neumann boundary at $x = 0$ and a Dirichlet boundary at $x = 1$.

Constant Approximations

The exact solution and the deflated solution for an increasing number of deflation cells

$$M = 2^i,\ i = 1, \dots, 11,$$

where $N = 2050$ shows that the deflation does not converge to the exact solution as M increases. This can also be seen for the error, which does not decrease with increasing M. The residual stays small throughout. Using the deflated solution as initial guess to BiCGSTAB will increase the number of iterations needed for convergence compared to $u_{init} = 0$.

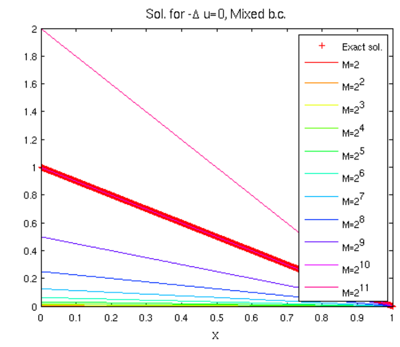

Figure B.1: Exact and deflated solution with constant approximations of Eq. 3.3 with FDM, $N = 2050$, $M = 2^i$, $i = 1, \dots, 11$.

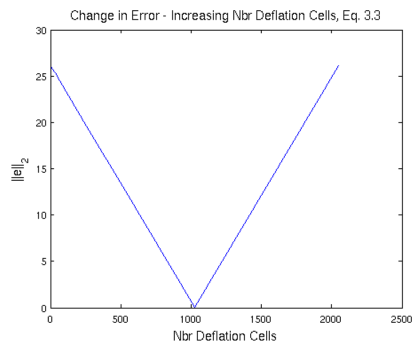

Figure B.2: Change in error for constant approximations of Eq. 3.3 with FDM.

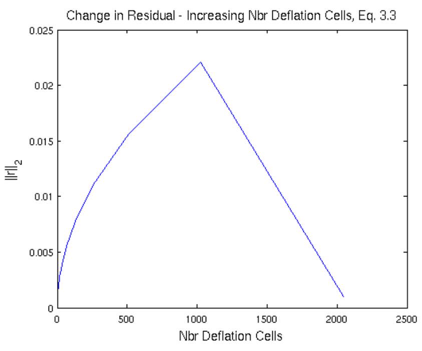

Figure B.3: Change in residual for constant approximations of Eq. 3.3 with FDM.

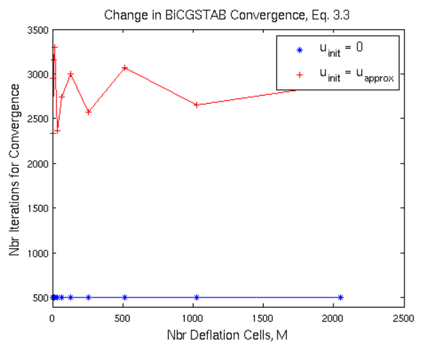

Figure B.4: Change in BiCGSTAB convergence rate for constant approximations of Eq. 3.3 with FDM as M changes as $M = 2^i$, $i = 1, \ldots, 11$, $N = 2050$, compared to $u_{init} = 0$.

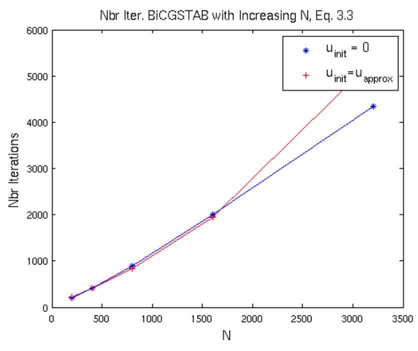

Figure B.5: Change in BiCGSTAB convergence rate for constant approximations of Eq. 3.3 with FDM as N changes as in Eq. 3.5, $M = 3$, compared to $u_{init} = 0$.

Linear Approximations

The linear deflation is close to the exact solution, independent of the choice of M. Also the error and residual are both very small. The deflated solution gives instant convergence in BiCGSTAB.

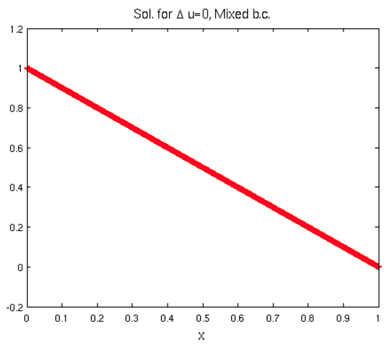

Figure B.6: Exact and deflated solution with linear approximations of Eq. 3.3 with FDM, $N = 2050$, $M = 2^i$, $i = 1, \ldots, 10$.

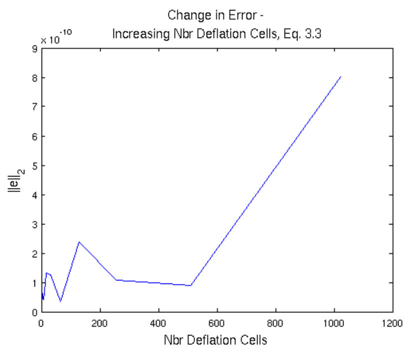

Figure B.7: Change in error for linear approximations of Eq. 3.3 with FDM.

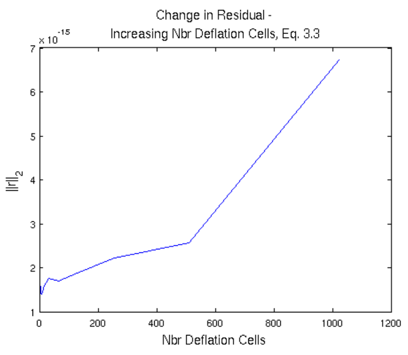

Figure B.8: Change in residual for linear approximations of Eq. 3.3 with FDM.

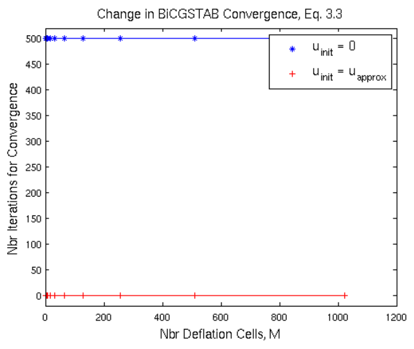

Figure B.9: Change in BiCGSTAB convergence rate for linear approximations of Eq. 3.3 with FDM as M changes as $M = 2^i$, $i = 1, \ldots, 10$, $N = 2050$, compared to $u_{init} = 0$.

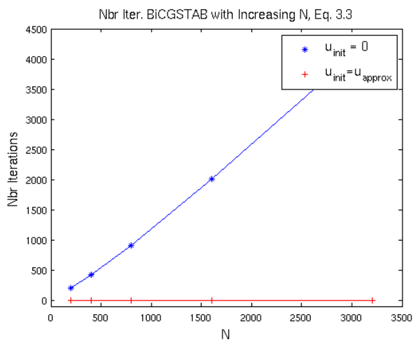

Figure B.10: Change in BiCGSTAB convergence rate for constant approximations of Eq. 3.3 with FDM as N changes as in Eq. 3.5, $M = 3$, compared to $u_{init} = 0$.

Results 2D

Constant Approximations

For the test case Eq. 3.7, the constant approximations give a solution with the same shape but smaller values than the exact solution. As the number of deflation cells increases the error will decrease. The deflation gives a slightly faster convergence rate than $u_{init} = 0$.

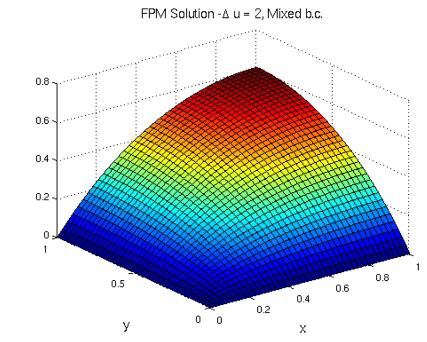

Figure B.11: Exact solution for Eq. 3.7 with FDM for $N_x = N_y = 40$.

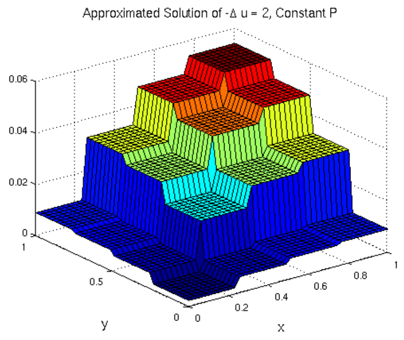

Figure B.12: Approximated solution with constant ansatz functions for Eq. 3.7, $N_x = N_y = 40$ and $M_x = M_y = 4$.

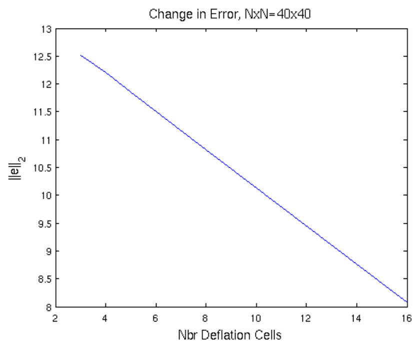

Figure B.13: Error of deflated solution with constant approximations for Eq. 3.7 as $M_x = M_y = 3, 4, 5, 8, 16$, $N_x = N_y = 40$.

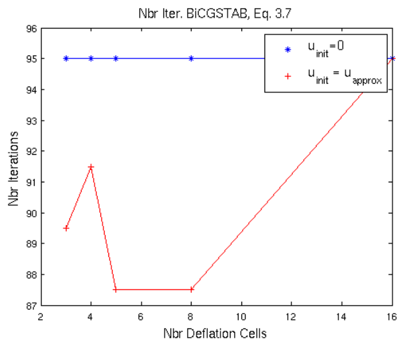

Figure B.14: How the convergence of BiCGSTAB changes with constant ansatz functions for Eq. 3.7, $N_x = N_y = 40$ and $M_x = M_y = 3, 4, 5, 8, 16$.

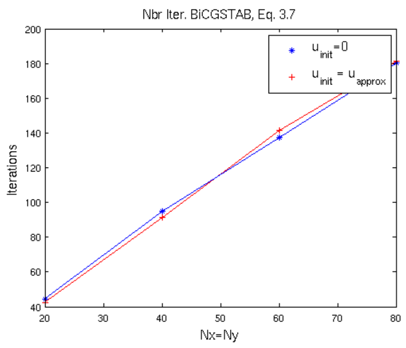

Figure B.15: How the convergence of BiCGSTAB changes with constant ansatz functions for Eq. 3.7, $M_x = M_y = 4$ and $N_x = N_y = 20, 40, 60, 80$.

Linear Approximations

The linear approximation gives a solution close to the exact solution. The error decreases rapidly as M increases. For most N the deflation gives a slightly faster convergence rate.

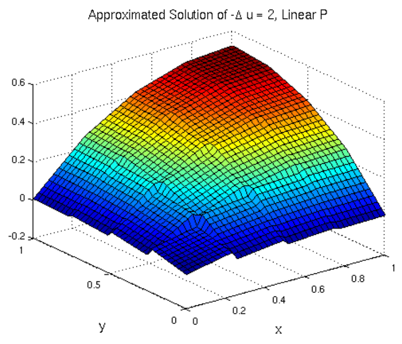

Figure B.16: Approximated solution with linear ansatz functions for Eq. 3.7, $N_x = N_y = 40$ and $M_x = M_y = 4$.

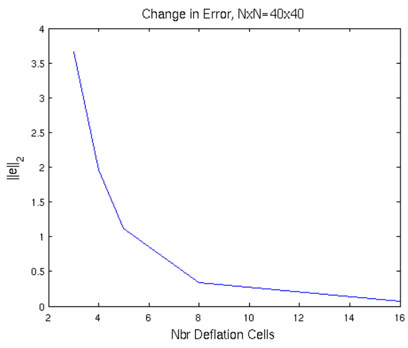

Figure B.17: Error of deflated solution with linear approximations for Eq. 3.7 as $M_x = M_y = 3, 4, 5, 8, 16$, $N_x = N_y = 40$.

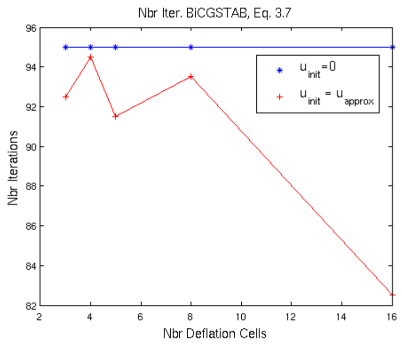

Figure B.18: How the convergence of BiCGSTAB changes with linear ansatz functions for Eq. 3.7, $N_x = N_y = 40$ and $M_x = M_y = 3, 4, 5, 8, 16$.

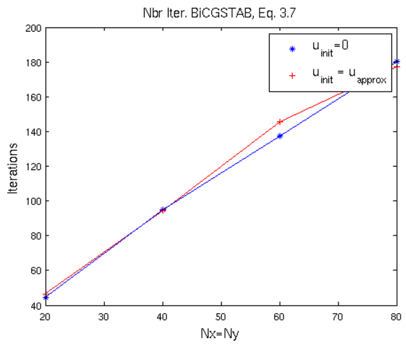

Figure B.19: How the convergence of BiCGSTAB changes with linear ansatz functions for Eq. 3.7, $M_x = M_y = 4$ and $N_x = N_y = 20, 40, 60, 80$.

Master’s Theses in Mathematical Sciences 2014:E37
ISSN 1404-6342

LUTFNA-3029-2014

Numerical Analysis
Centre for Mathematical Sciences
Lund University
Box 118, SE-221 00 Lund, Sweden

http://www.maths.lth.se/