A CUBIC SPLINE BASED LOCAL DEFECT CORRECTION APPROACH FOR THE SIMULATION OF INCOMPRESSIBLE VISCOUS FLOWS

A Project Report Submitted

for the Course

MA498 Project I

by

Aditya Gupta | Vishal Kumar

(130123051 | 130123043)

to the

DEPARTMENT OF MATHEMATICS

INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI

GUWAHATI - 781039, INDIA

November 2016

CERTIFICATE

This is to certify that the work contained in this project report entitled “**A cubic spline based local defect correction approach for the simulation of incompressible viscous flow**" submitted by **Aditya Gupta** (**Roll No.: 130123051**) and **Vishal Kumar** (**Roll No.: 130123043**) to Department of Mathematics, Indian Institute of Technology Guwahati towards the requirement of the course **MA498 Project I** has been carried out by him/her under my supervision.

Guwahati - 781 039	(Dr. Jiten C. Kalita)
November 2016	Project Supervisor

ABSTRACT

The project is focused on developing a numerical scheme by using local defect correction methods in conjunction with cubic splines for interpolating boundary conditions for the simulation of incompressible viscous flows.

The governing equations for many problems (e.g. the Navier-Stokes equations) in fluid dynamics involve ordinary or partial differential equations, defined continuously on a certain surface or volume. It is very difficult to solve such systems through analytical methods. The process of discretization provides a numerical approach to solve such models. The finite difference method is a very popular tool of discretization in computational fluid dynamics (CFD). It involves setting up a grid of points in the domain space. Then we replace the derivatives with finite differences.

However, for some problems, as we shall see, a single uniform grid is either difficult to work with or doesn't provide an accurate enough solution. This is where the local defect correction method (LDC) comes in. First we see a general approach to the LDC method. Afterwards we take a look at the cubic spline method. Then we shall apply both these paradigms, first to steady state heat conduction equations, and then to the famous Lid driven Cavity problem using Navier-Stokes equations.

Contents

List of Figures

Chapter 1

Introduction

The boundary value problems (BVP) we will explore in this paper show rapid variations in their solution in some parts of the domain. If we use a single uniform grid for the whole domain this we will require a very fine grid leading to a lot of unnecessary computations. Local Defect correction (LDC) offers a way to deal with such problems. It involves using different discretization schemes for the local and global problem, instead of using a single set of difference equations. In our case, we use a coarser grid as a global discretization scheme, which is combined with a finer grid (having smaller step size) used for the sub-domain.

1.1 The Local Defect Correction method

In this section we give a general introduction to the Local Defect Correction method.[12] We may have a boundary value problem with domain Ω, like

$$Lu = f \tag{1.1}$$

with boundary condition

$$Bu = g \tag{1.2}$$

on

$$\Gamma = \delta\Omega \tag{1.3}$$

where for example $L = -\Delta$. A discretization can be given by

$$L_h u_h = f_h \tag{1.4}$$

where h is grid parameter. The defect correction iteration for the given problem shall be

$$u_0^h = f_h L_h^1 \tag{1.5}$$

$$u_h^{i+1} = u * i_h - L_h^{-1}(L_h^{'} u_h^i - f_h^{'}) \tag{1.6}$$

for $i = 1, 2, ...$ The first step of this iteration can be written as

$$u_h^1 = u_h^0 - L_h^{-1}(L_h^{'} u_h^0 - f_h^{'}) \tag{1.7}$$

where d_h^0 is the defect $d_h^0 = L_h^{'} u_h^0 - f_h^{'}$ Let $\omega \Subset \Omega$ be a sub-region. Its characteristics function is

$$\chi_\omega(x) = \begin{pmatrix} 0 & x\epsilon\Omega\backslash\omega \\ 1 & x\epsilon\omega \end{pmatrix} \tag{1.8}$$

The local defect is

$$d_0^h = \chi_\omega(L_h^{'} u_h - f_h^{'}) \tag{1.9}$$

Together the equations (1.7) and (1.9) give us the local defect correction.

$$u_h^1 - u_h^* = L_h^{-1}\chi_\omega(L_h - L_h^{'})(u_h^0 - u_h^*) - L_h^{-1}\chi_\omega(L_h^{'} u_h^* - f'_h) + L_1^h(1 - \chi_\omega)L_h^{'}(u_h^0 - u_h^*) \tag{1.10}$$

The local discretization for our original BVP can be denoted by

$$L_h u_{h,\Omega} = f_{h,\Omega} \tag{1.11}$$

including the boundary conditions on Γ. The underlying grid is Ω_h . The local discretization problem in the subdomain $\omega \subset \Omega$ can be written as $Lu_\omega = f$ If $\Gamma_0 = \overline{w} \cap \Gamma$ is not empty, u_ω must satisfy in Γ_0

$$Bu_\omega = g \tag{1.12}$$

On the interior part $\Gamma_1 = \overline{\omega} \backslash \Gamma$ we have

$$u_\omega = u_\Omega \tag{1.13}$$

The global discretization is given by

$$L^{'}_{h'} u^{'}_{h',\omega} = f^{'}_{h',\omega} \tag{1.14}$$

$L^{'}$ and $u^{'}$ indicate the discretization may be different from L_h and u_h. $h^{'}$ subscript means a different step size from h. The discretization of (1.13) is given by:

$$u^{'}_{h',\omega} = \gamma u_{h,\Omega} \tag{1.15}$$

1.2 An Algorithm for LDC

An algorithm to combine the local and global discretization[1] is given by
Start:

$$f^0_{h,\Omega} = f_{h,\Omega} \tag{1.16}$$

obtained from the global discretization.
Given $f^i_{h,\Omega}$: Compute the solution to the coarse grid problem to initialize $u^i_{h,\omega}$ on Ω_h

$$L_h u^i_{h,\omega} = f^i_{h,\Omega} \tag{1.17}$$

Compute the boundary values on $\Gamma_{1,h'}$

$$g^i = \gamma u^i_{h,\Omega} \tag{1.18}$$

The preceding two equations give the solution to the coarser grid problem. Solve this finer grid problem in $\omega_{h'}$

$$L^{'}_{h'} u^{'}_{h',\omega} = f^{'}_{h',\omega} \tag{1.19}$$

and

$$g^i = \gamma u^i_{h,\omega} \tag{1.20}$$

We have sub-grids $\omega^{"}_h \Subset \omega^{'}_h \Subset \omega$
Interpolate $u^i_{h,\omega}$ on $\omega^{'}_h = \omega^{'} \epsilon \Omega_h$

$$\overline{u}_{h,\omega'} = \pi u_{h',\omega} \tag{1.21}$$

Compute the defect in w^"_h

$$d_{h,w^{"}} = L_h \overline{u}_{h,\omega'} - f_{h,\Omega} \tag{1.22}$$

Now for the next iteration, define in the coarser grid

$$f^{i+1}_{h,\Omega} = f_{h,\Omega} + \chi_{\omega^{"}} d_{h,w^{"}} \tag{1.23}$$

1.3 Background

A lot of problems have already been experimented with the inclusion of LDC methods, for example diffusion equations [3] and combustion equations [2], to name a few. Also, [5] provides a close correspondence between LDC and Fast Adaptive Composite Grid (FAC) methods [9]. [10] provide a local defect correction technique for time-dependent problems, suitable for solving partial differential equations characterized by high activity.

Chapter 2

Interpolating boundary conditions

In our discretization scheme, we will use the cubic spline algorithm[4] to find the boundary values for the finer grid using values from the coarser grid. Piecewise polynomial approximation is used to approximate the value of a function on an interval using a set of given data points.

2.1 Cubic Splines

Definition 2.1.1. Given a function f defined on $[a, b]$ and set of n+1 points such that

$$a < x_0 < x_1 < x_2 ... < x_n < b$$

A cubic spline interpolant S for the function f satisfies the following conditions:

1. $S(x)$ is a cubic polynomial, denoted $S_j(x)$ for every jth interval $[x_j, x_{j+1}]$, $j = 0, 1, 2, ..., n-1$
2. $S_j(x_{j+1}) = S_j(x_{j+1})$ for all $j = 0, 1, 2, ..., n-2$
3. $S_j'(x_{j+1}) = S_j'(x_{j+1})$ for all $j = 0, 1, 2, ..., n-2$

4. $S_j^{''}(x_{j+1}) = S_j^{''}(x_{j+1})$ for all $j = 0, 1, 2, ..., n-2$

5. One of the following boundary conditions is satisfied:

   (a) $S^{''}(x_0) = S^{''}(x_n)$ (free boundary)

   (b) $S^{'}(x_0) = f^{'}(x_0)$ and $S^{'}(x_n) = f^{'}(x_n)S^{'}(x_n)$ (clamped boundary)

2.2 Construction of a cubic spline

We take the general form of the cubic spline interpolant for each interval as follows:

$$a_j + b_j(x - x_j) + c_j(x - x_j)^2 + d_j(x - x_j)^3 \tag{2.1}$$

Clearly, the we need to determine $4n$ constants to construct the interpolant based on n nodes. We apply the conditions mentioned in the definition of the cubic spline to each S_j to find the values of corresponding a_j, b_j, c_j, d_j for every $j = 0, 1, ..., n-2$. Since $S_j(x_j) = a_j = f(x_j)$, we can use condition 3 to get

$$a_{j+1} = S_{j+1}(x_{j+1}) = S_j(x_{j+1}) \tag{2.2}$$

for every $j = 1, ..., n-1$ we introduce the notation

$$h_j = x_{j+1} - x_j \tag{2.3}$$

We can define $a_n = f(x_n)$, giving us the equation

$$a_{j+1} = a_j + b_j h_j + c_j h_j^2 + d_j h_j^3 \tag{2.4}$$

which holds for all $j = 0, 1, ..., n-1$ Similarly we can define $b_n = S^{'}(x_n)$ (trivial). If we take the first derivative of $S_j(x)$ we see

$$S_j^{'}(x) = b_j + 2c_j(x - x_j) + 3d_j(x - x_j)^2 \tag{2.5}$$

Hence $S_j^{'}(x_j) = b_j$ for all $j = 0, 1, ..., n-1$. Using condition 4 we get for all $j = 0, 1, ..., n-1$

$$b_{j+1} = b_j + 2c_jh_j + 3d_jh_j^2 \tag{2.6}$$

Similarly we can define $c_n = S^{"}(x_n)/2$ and use condition 5 to get for all j = 0,1, ... ,n-1 the following equation

$$c_{j+1} = c_j + 3d_jh_j \tag{2.7}$$

From the last three equations, for all j = 0,1, ... ,n-1 we get

$$a_{j+1} = \frac{a_j + b_jh_j + h_j^2(2c_j + c_{j+1})}{3} \tag{2.8}$$

and

$$b_{j+1} = b_j + h_j(c_j + c_{j+1}) \tag{2.9}$$

By reducing index j by 1 we get

$$b_j = b_{j-1} + h_{j-1}(c_{j-1} + c_j) \tag{2.10}$$

Solving for b_j we get

$$b_j = \frac{(a_{j+1} - a_j)}{h_j} - \frac{(2c_{j+1} + c_j)h_j}{3} \tag{2.11}$$

We can get b_{j-1} by reducing the index j in the preceding equation

$$b_{j-1} = \frac{(a_j - a_{j-1})}{h_{j-1}} - \frac{(2c_j + c_{j-1})h_{j-1}}{3} \tag{2.12}$$

Substituting the values for b_j and b_{j-1} gives us for $all j = 1, ..., n-1$ this set of linear equations

$$h_{j-1} + c_{j-1} + 2(h_{j-1} + h_j)c_j + h_j c_{j+1} = 3\frac{(a_{j+1} - a_j)}{h_j} - 3\frac{(a_j - a_{j-1})}{h_{j-1}} \tag{2.13}$$

Notice that in this system we already know h_j and a_j for $j = 1, 2, ..., n-1$. Once c_j for $j = 1, 2, ..., n-1$ are calculated the remaining coefficients can be easily determined by the equations listed above. The following section demonstrates that these coefficients can indeed be uniquely calculated if one of the boundary conditions mentioned in the definition of the cubic spline is imposed.

2.3 Natural splines

The following theorem proves that for any given set of n points, there will always exist a natural spline interpolant.

Theorem 2.3.1. *If a function f is defined at nodes $x_0, x_1, ..., x_n$ such that $a < x_0 < x_1 ... < x_n < b$, then there exists a unique natural spline interpolant S for the given function passing through the points $(x_0, f(x_0), (x_1, f(x_1)...(x_n, f(x_n)$ satisfying the natural boundary condition $S^{''}(a) = 0$ and $S^{''}(b) = 0$.*

Proof. Imposing the boundary condition gives us

$$c_n = S^{''}(x_n) = 0 \tag{2.14}$$

$$S''(x_0) = 0 = 2c_0 + 6d_0(x_0 - x_0) \tag{2.15}$$

Hence $c_0 = x_0$. We get a system of linear equations of the form $AX = B$ where

$$A = \begin{bmatrix} 1 & 0 & 0 & . & . & . & . & . & 0 \\ h_0 & 2(h_0+h_1) & h1 & 0 & . & . & . & . & 0 \\ 0 & h_1 & 2(h_1+h_2) & h_2 & 0 & . & . & . & 0 \\ 0 & 0 & . & . & . & . & . & . & 0 \\ . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & h_{n-2} & 2(h_{n-2}+h_{n-1}) & h_{n-1} \\ 0 & . & . & . & . & . & 0 & 0 & 1 \end{bmatrix}$$

Notice that A has three non-zero diagonals.

$$B = \begin{bmatrix} 0 \\ 3(a_2 - a_1)/h_1 \\ . \\ . \\ 3(a_n - a_{n-1})/h_{n-1} - 3(a_{n-1} - a_{n-2})/h_{n-2} \\ 0 \end{bmatrix}$$

and

$$X = \begin{bmatrix} c_0 \\ c_1 \\ . \\ . \\ c_n \end{bmatrix}$$

We can see that matrix A is clearly diagonally dominant, i.e. in every row the diagonal entry is greater than the sum of all other elements in the same row. Hence, by Levy–Desplanques theorem[7] we can conclude that the matrix A is singular and the system $AX = B$ has a unique solution. Hence, the theorem is proved. The natural spline has been used in our implementation of the discretization processes. The algorithm for formulating cubic splines is discussed in further sections of this paper. $\square$

Chapter 3

Steady State Heat Conduction

To begin experimenting with our scheme, we started with steady state heat conduction problem with Dirichlet boundary conditions.

3.1 Problem Description and Governing Equations

The following PDE governs the steady-state temperature T(x,y):

$$T_{xx} + T_{yy} = 0 \tag{3.1}$$

subject to the following constraints:

$$0 < x < 4$$
$$0 < y < 1$$

and the following boundary conditions:

$$T(x,0) = 200x(x-4)$$
$$T(x,1) = T(0,y) = T(4,y) = 0$$

3.2 Analytical Solution

The analytical solution satisfying the above PDE with the given boundary conditions is :

$$T(x,y) = \sum_{n=1}^{\infty} cosech(\frac{-n\pi}{4})\frac{12800}{n^3\pi^3}[(-1)^n - 1]sin(\frac{n\pi}{4}x)sinh[\frac{n\pi}{4}(y-1)] \tag{3.2}$$

3.3 Discretization

To obtain the FDE we replace T_{xx} and T_{yy} by their respective second-order centered-difference approximation at grid point (i,j) :

$$\overline{T}_{xx}(i,j) = \frac{\overline{T}_{i+1,j} - 2\overline{T}_{i,j} + \overline{T}_{i-1,j}}{\Delta x^2} + O(\Delta x^2) \tag{3.3}$$
$$\overline{T}_{yy}(i,j) = \frac{\overline{T}_{i,j+1} - 2\overline{T}_{i,j} + \overline{T}_{i,j-1}}{\Delta y^2} + O(\Delta y^2) \tag{3.4}$$

Now putting (3.3) and (3.4) in (3.1) we get:

$$\frac{\overline{T}_{i+1,j} - 2\overline{T}_{i,j} + \overline{T}_{i-1,j}}{\Delta x^2} + \frac{\overline{T}_{i,j+1} - 2\overline{T}_{i,j} + \overline{T}_{i,j-1}}{\Delta y^2} + E_{i,j} = 0 \tag{3.5}$$

with the following truncation error:

$$E_{i,j} = O(\Delta x^2) + O(\Delta y^2)$$

Now, taking β as $\frac{\Delta x}{\Delta y}$, we finally get:

$$T_{i,j} = \frac{T_{i+1,j} + \beta^2 T_{i,j+1} + T_{i-1,j} + \beta^2 T_{i,j-1}}{2(1+\beta^2)} \tag{3.6}$$

Now, for solving the steady state FDE, we used the Gauss-Siedel method as our iterative solver and experimented with β as 1.
The sweep was done in the following manner :

- The outer sweep was done in Y- direction from 1 to 0.
- The inner sweep was done in X- direction from 0 to 4.

The tolerence was set as 10^{-5} for each cases and the error was taken as absolute and the max norm ($||.||_\infty$)was used.
To crosscheck and to comment on the correctness and accuracy of the iterative solver for solving the above FDE, we also evaluated the analytical values whereby setting the tolerance limit to 10^{-5} for computing the infinite series and since the absolute value of the terms of the series were monotonically decreasing, the stopping criterion was practical to get a suitable answer. The code was written in C++.

3.4 Using Local Defect Correction with Cubic Splines

The global domain (0<x<4 and 0<y<1) was discretized using a coarser of dimension 41x11 points(step size of .1 in both directions) and a finer grid was imposed in the lower right corner (2<x<4 and 0<y<0.5) of dimension 41x11 points(step size of .05 in both directions). The grid structure looked like this:

As is evident, some point are common between the finer and the coarser grids and these points were used to update the information from coarser to finer and

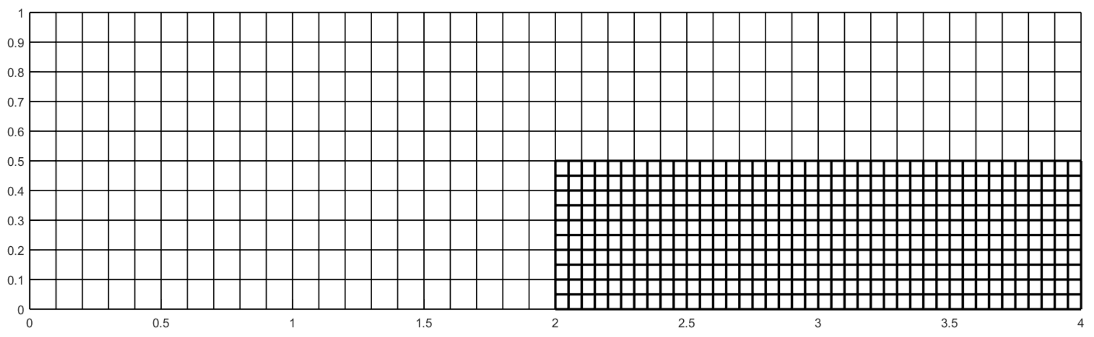


Figure 3.1: Grid structure

vice versa. Thus, the solution for each grid was refined using an 'back and forth' approach:

1. The coarser grid is solved first(till convergence or for some fixed number of iterations) by initializing it with proper boundary conditions and zero values for the interior points.

2. The common points between the finer and coarser grid are used to then update the finer grid for boundary conditions as well as interior points(a better initial approximation).

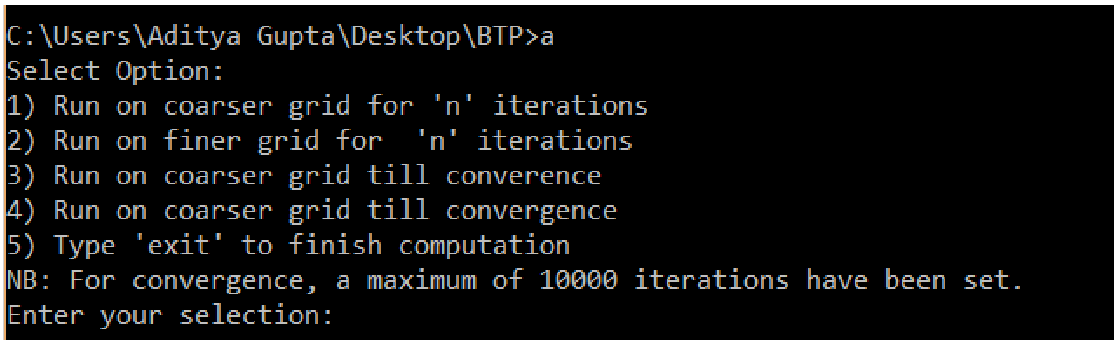
3. The remaining boundary points are then interpolated using cubic spline for the two virtual boundaries in our case(top and left boundary in this case).

4. The finer grid is then solved using a similar scheme as used in the coarser grid in step 1.

5. The common points are now used to update the coarser grid and we move back to step 1. The process is repeated multiple time.

3.5 Numerical Results and Observations

For experimentation, at each stage 4 options were available:



```
C:\Users\Aditya Gupta\Desktop\BTP>a
Select Option:
1) Run on coarser grid for 'n' iterations
2) Run on finer grid for  'n' iterations
3) Run on coarser grid till converence
4) Run on coarser grid till convergence
5) Type 'exit' to finish computation
NB: For convergence, a maximum of 10000 iterations have been set.
Enter your selection:
```

Figure 3.2: Runtime options

We experimented with a lot of options:

1. **Running for specific (10-100) number of iterations on both the coarser and finer grid.** It was observed as the finer grid had more points in comparison to the coarser grid it required more iteration to reach the refinement level of the coarser grid. For example: The following is the contour plot when 10 iteration are done on both the grids:

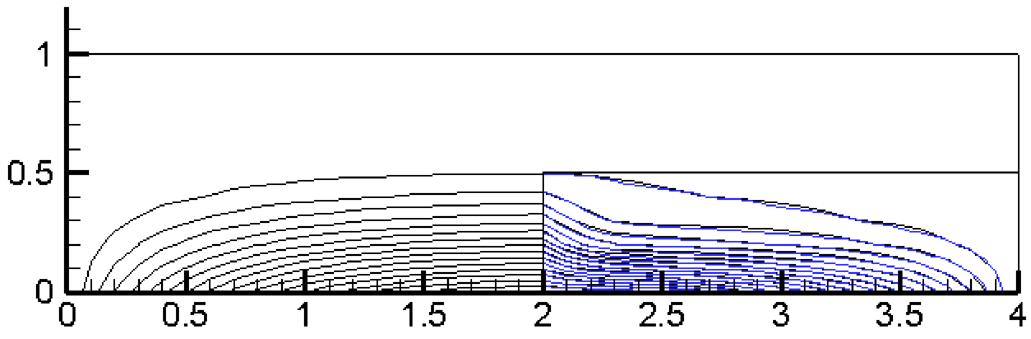


Figure 3.3: Running for 10 iterations on coarser and then the finer grid.

2. **Going back and forth performing a fixed (10-100) number of iterations on both the coarser and finer grid.**

3. **Running till convergence for the coarser grid and for the finer grid.**
4. **Going back and forth achieving convergence for the coarser grid and for the finer grid.**
5. **Using a combination of the above 4.**

For options 2-5, good results were obtained in the finer grid which were in accordance with the coarser grid.

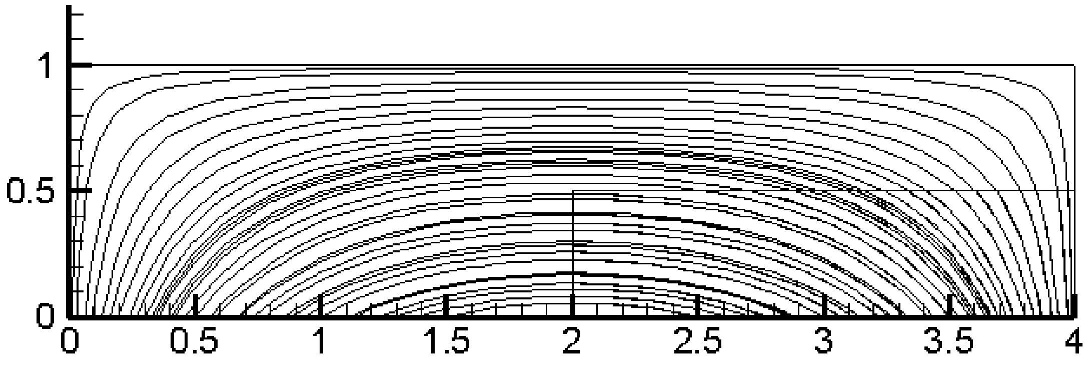


Figure 3.4: Contour plot of the whole domain consisting of the coarser and finer grids.

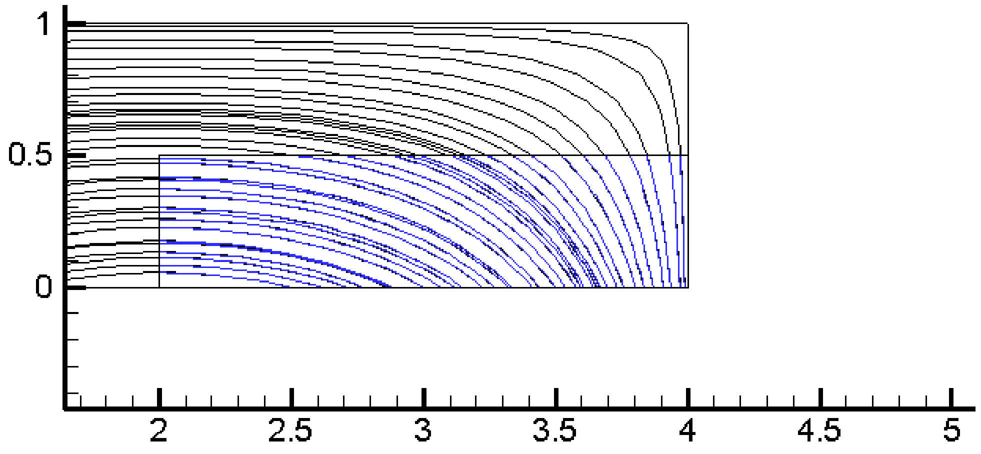


Figure 3.5: Contour plot of the lower right region

This gave us the fair confidence to use cubic splines for boundary condition interpolation for our further problems.

NB: We did not compare our solution with any analytical solution as our primary aim was to first match the quality of the solution with the global scheme followed in the coarser grid. Improving the overall solution will be dealt later by incorporating HOC schemes and better improved solvers.

Chapter 4

Lid Driven Cavity(LDC) for incompressible fluids

Having gained confidence in our implementation of local defect correction in conjunction with the steady state heat conduction equations, we moved forward to the famous LDC problem

4.1 Problem Description

The problem essentially consists of numerically solving the 2D NS equations in a square cavity. The left, right and bottom walls of the cavity are stationary while the top wall moves from the left to right (in the positive x-direction) with a velocity U. Because of the no-slip condition for a viscous flow, the x- and y-velocities of the fluid are zero at the left, right and the bottom walls. The no-slip condition at the top wall requires that y-velocity of the fluid is zero but the x-velocity equals the wall velocity U. As a consequence, as the top wall moves from the left to right, so does the fluid in contact with that wall. Because of viscosity, this momentum in the x-direction diffuses in the transverse direction to the interior of the cavity and after some time the fluid in the whole cavity is set into motion (note that if the fluid had no viscosity, the wall motion would not have affected the fluid in the cavity at all and it would have remained stationary).

In this simple geometric setting, highly complex flow patterns arise as the Reynolds number (Re) increases. At relatively low Reynolds numbers, a large primary vortex (or eddy) is seen to form. As Reynolds number increases smaller vortices, called secondary, tertiary, quaternary etc. start developing at various corners of the cavity.

Theoretically, at high Reynolds numbers, an infinite series of vortices, ranging from the primary eddy (having the dimension of the cavity) to the very very small ones, are generated. Multiplicity of scales is characteristic of high-Re flows.

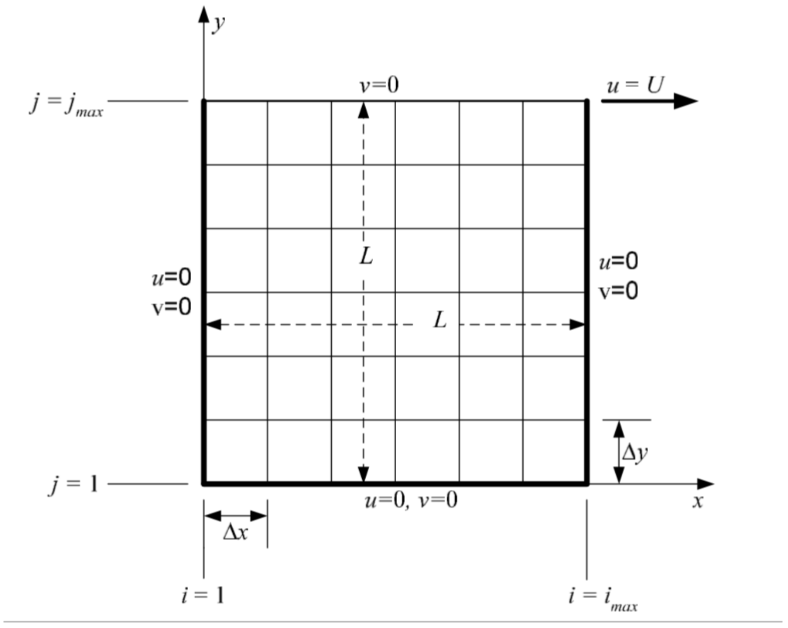


Figure 4.1: Lid Driven Square Cavity with an uniform grid for the whole domain, $Re = \frac{LU}{v}$.

4.2 Governing Equations

The governing equations for the present flow configuration are the two-dimensional (2D) incompressible Navier-Stokes (NS) equations. The 2D NS equations in the **non-dimensional** primitive-variables form are given by:

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \tag{4.1}$$

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = -\frac{\partial p}{\partial x} + \frac{1}{Re}\nabla^2 u = -\frac{\partial p}{\partial x} + \frac{1}{Re}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) \tag{4.2}$$

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} = -\frac{\partial p}{\partial x} + \frac{1}{Re}\nabla^2 v = -\frac{\partial p}{\partial x} + \frac{1}{Re}\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) \tag{4.3}$$

For a steady 2D flow, it is possible to have an equivalent non-dimensional system of two equations known as the streamfunction-vorticity (or $\psi - \omega$) form of the NS equations:

$$\nabla^2\psi = \frac{\partial^2 \psi}{\partial x^2} + \frac{\partial^2 \psi}{\partial y^2} = -\omega \tag{4.4}$$

$$u\frac{\partial \omega}{\partial x} + v\frac{\partial \omega}{\partial y} = \frac{1}{Re}\nabla^2\omega = \frac{1}{Re}\left(\frac{\partial^2 \omega}{\partial x^2} + \frac{\partial^2 \omega}{\partial y^2}\right) \tag{4.5}$$

4.3 Discretization

We now discretize our equations involving ψ and ω.

4.3.1 Interior points

Using 5 point formula, we get the following discretization for the interior points:

For ψ

We use second-order accurate central differencing for all the space derivatives.The discretized ψ-equation at point (i, j) is written as:

$$\frac{\psi_{i+1,j} - 2\psi_{i,j} + \psi_{i-1,j}}{(\Delta x)^2} + \frac{\psi_{i,j+1} - 2\psi_{i,j} + \psi_{i,j-1}}{(\Delta y)^2} = -\omega_{i,j} \tag{4.6}$$

We introduce the notation $\beta = \frac{\Delta x}{\Delta y}$. Upon rearrangement of equation (4.6) we shall get

$$\psi_{i,j} = \frac{\psi_{i+1,j} + \psi_{i-1,j} + \beta^2(\psi_{i,j+1} + \psi_{i,j-1}) + (\Delta x)^2\omega_{i.j}}{2(1+\beta^2)} \tag{4.7}$$

For ω

The discretized form of the ω equation at (i, j) point is

$$u_{i,j}\frac{\omega_{i+1,j} - \omega_{i-1,j}}{2\Delta x} + v_{i,j}\frac{\omega_{i,j+1} - \omega_{i,j-1}}{2\Delta y} = \frac{1}{Re}\left(\frac{\omega_{i+1,j} - 2\omega_{i,j} + \omega_{i-1,j}}{(\Delta x)^2} + \frac{\omega_{i,j+1} - 2\omega_{i,j} + \omega_{i,j-1}}{(\Delta y)^2}\right) \tag{4.8}$$

Upon rearrangement after multiplying on both sides by $(\Delta x)^2$ we get

$$\omega_{i,j} = \frac{\omega_{i+1,j} + \omega_{i-1,j} + \beta^2(\omega_{i,j+1} + \omega_{i,j-1}) - \frac{1}{2}\Delta x Re(\omega_{i+1,j} - \omega_{i-1,j})u_{i,j} - \frac{1}{2}\Delta x\beta Re(\omega_{i,j+1} - \omega_{i,j-1})v_{i,j}}{2(1+\beta^2)} \tag{4.9}$$

4.3.2 Boundary conditions

For ψ

Take $\psi = 0$ on all the walls (any real number can be used; but 0 is better for the sake of comparison as most papers use this value)

For ω

$$\text{Left wall: } \omega_{1,j} = -\frac{2\psi_{2,j}}{(\Delta x)^2} \qquad 2 \le j \le j_{max-1} \tag{4.10}$$

$$\text{Right wall: } \omega_{imax,j} = -\frac{2\psi_{imax-1,j}}{(\Delta x)^2} \qquad 2 \le j \le j_{max-1} \tag{4.11}$$

$$\text{Bottom wall} \omega_{i,1} = -\frac{2\psi_{i,2}}{(\Delta y)^2} \qquad 2 \le i \le i_{max-1} \tag{4.12}$$

$$\text{Top wall} \omega_{i,jmax} = -\frac{2\psi_{i,jmax-1} + 2\Delta y}{(\Delta y)^2} \qquad 2 \le i \le i_{max-1} \tag{4.13}$$

These conditions were obtained by coupling with (4.7) and (4.9) the conditions of zero wall-tangential velocity at the side and bottom walls and a tangential velocity of U at the top wall.

4.4 Local Defect Correction

The following grid structure was used to apply local defect correction to the lower left corner of the global domain.

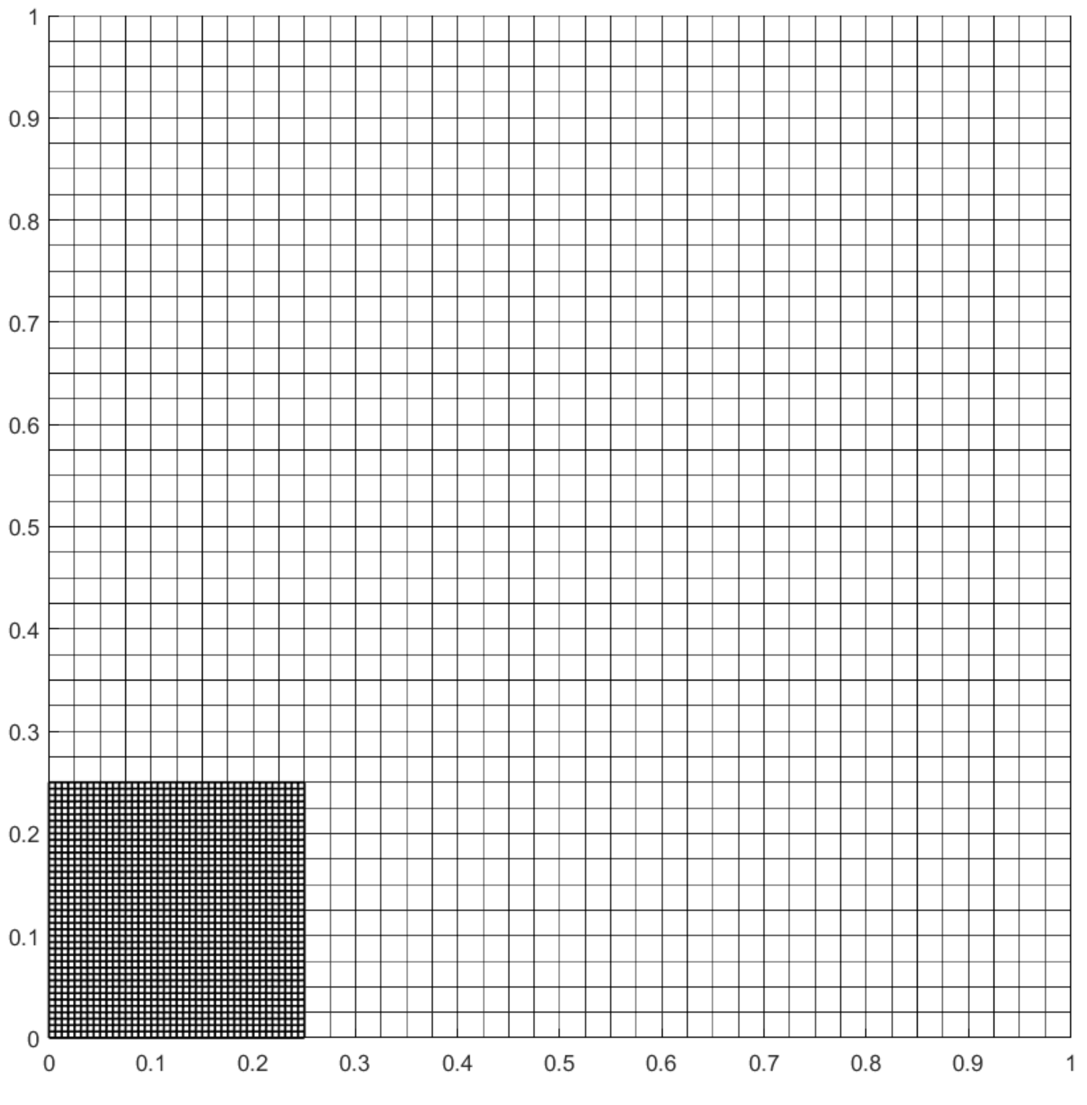


Figure 4.2: Grid structure for lid driven cavity

The process used was similar to what was done in the previous experiment where in a 'Back and Forth' approach was used.

4.5 Experimental Results and Observation

The following plots were the obtained from standard discretization of the stream-function and vorticity equations by using just the coarser grid on the global domain.

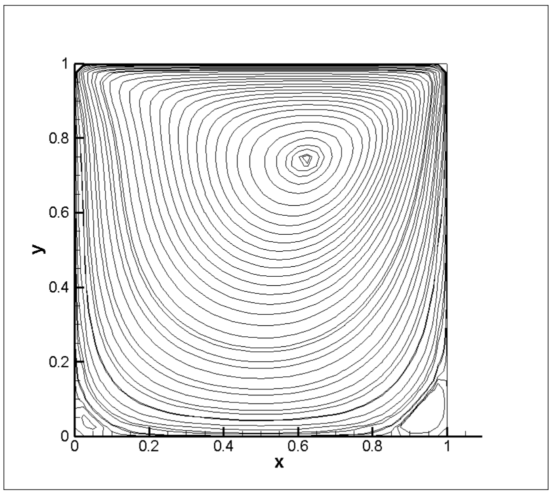


Figure 4.3: Contour plot of the global domain.

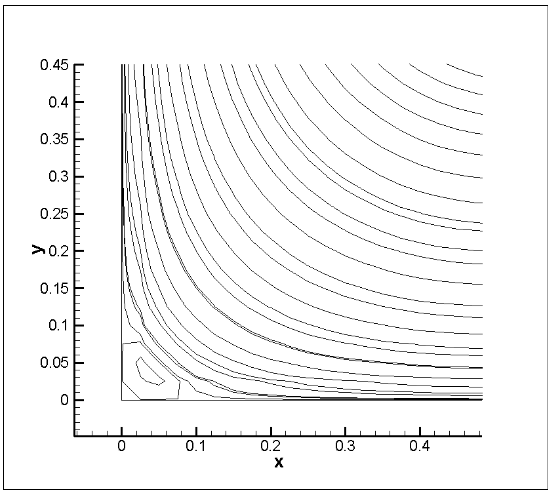


Figure 4.4: Contour plot of the lower left region

Observation: Using a uniform coarser grid global domain does not give us good results in high gradient regions. And making the grid finer over the whole region increases the computation cost. Also since our main purpose is to just improve the results in the high gradient regions, application of an uniform finer grid is an overkill.

For the coarser grid ψ has a Dirichlet BC and ω has Neumann BC. For the finer grid we experiment with different setting of the BCs.:

4.5.1 Using physical boundary conditions on all the virtual boundaries

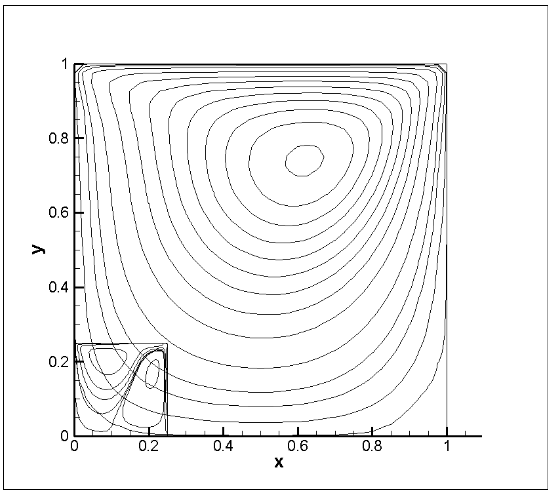


Figure 4.5: Contour plot of the whole domain.

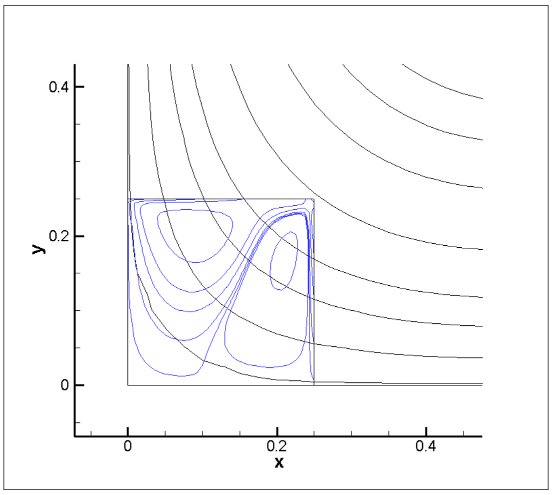


Figure 4.6: Contour plot of the lower right region.

As expected, we obtain results which would have been obtained if the finer domain were the whole global domain and the virtual boundaries were the actual physical boundaries.

Consequently, this gives us a solution where the finer grid now corresponds to a new global domain with actual physical boundaries. Since in the estimating ω on the actual physical boundaries we assumed the exterior point to be an image and the same generalization is not true for the interior point(s) which now act as points on the virtual boundary, this method is erroneous.

4.5.2 Keeping both ψ and ω fixed

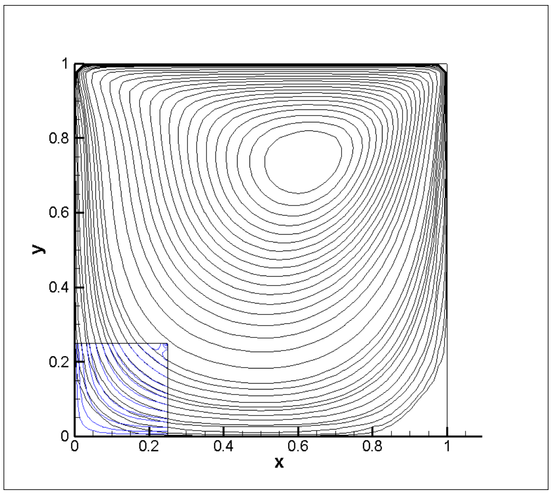


Figure 4.7: Contour plot of the whole domain.

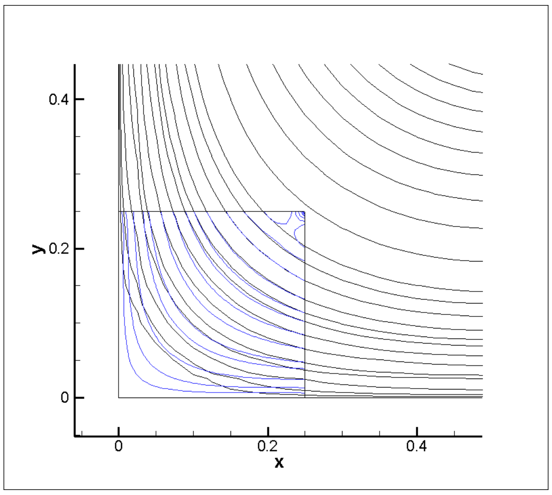


Figure 4.8: Contour plot of the lower right region.

Here, both the variables were held constant (initial interpolation by cubic ssplines) on the boundary. Since there is no refinement in the boundary conditions of the both ψ and ω, the error which arises due to the initial interpolation from cubic spline on the boundary prevails throughout the course of solution. Thus, the solution resembles the overall solution but the error is significantly high.

4.5.3 Keeping ψ fixed and varying ω

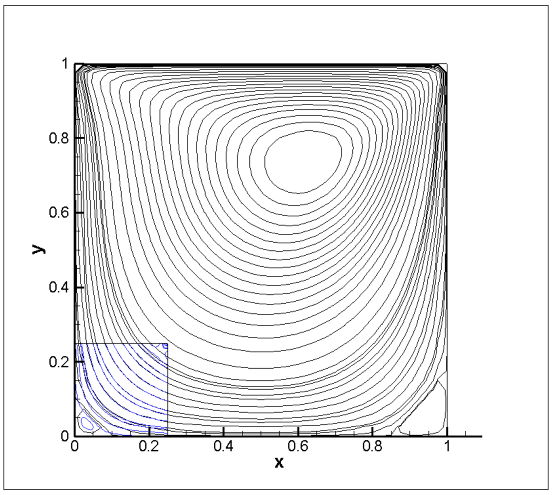


Figure 4.9: Contour plot of the whole domain.

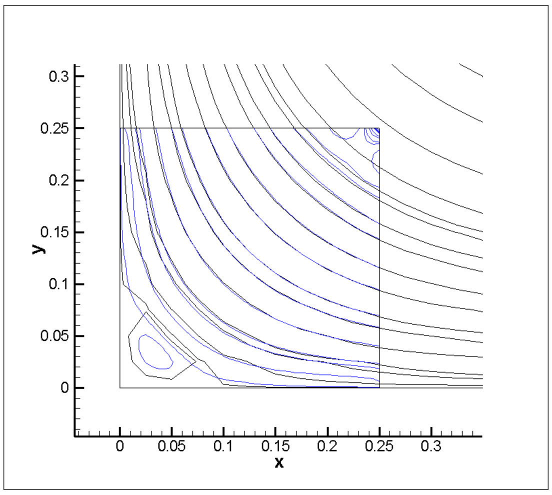


Figure 4.10: Contour plot of the lower right region.

Using the discretization for ω, we update ω every time by using the new values from the interior of the finer grid and old(fixed) values from the coarser grid to properly satisfy the neumann boundary conditions of ω. As can be observed from the contour plot this method proves superior from the previous two, as was expected. But still there are some 'unwanted vortices' in the upper right corner. The reason of which could be ascribed to error which exists due to cubic spline interpolation since the top right corner's result are affected values from both the top and the right boundaries (both are virtual in our case). And the rest three corners have at least one true boundary.

Moreover, the error prevails due to the interpolation in the values of ψ which are not refined over the course of the solution.

Chapter 5

Future Plans

5.1 Improving boundary condition interpolation

Better interpolation scheme for the virtual boundaries.

5.2 Higher Order Compact schemes

Deriving and experimenting with Higher Order Compact schemes.[8]

5.3 Efficient solvers

Using efficient solvers which are specific to the problem[11]. Currently, basic iterative solver, namely Gauss-Sidel, was used in all the experiments.

5.4 Parallel and Distributed computing

Parallel computing for faster back and forth computation. Also, computation on two disjoint finer grids.[6]

Bibliography

[1] Martijn J.H. Anthonissen. *Local Defect Correction Techniques: Analysis and Application to Combustion.* First Edition. Eindhoven University of Technology,, 2001.

[2] Martijn Johannes Hermanus Anthonissen. *Local defect correction techniques: analysis and application to combustion.* Eindhoven University of Technology, 2001.

[3] MJH Anthonissen, RMM Mattheij, and JHM ten Thije Boonkkamp. Convergence analysis of the local defect correction method for diffusion equations. *Numerische Mathematik*, 95(3):401–425, 2003.

[4] Richard L. Burden and J. Douglas Faires. *Numerical analysis.* Ninth Edition. Brooks/Cohle, Cengage Learning, Boston, 2011.

[5] Peter JJ Ferket and Arnold A Reusken. Further analysis of the local defect correction method. *Computing*, 56(2):117–139, 1996.

[6] Wolfgang Hackbusch. Local defect correction method and domain decomposition techniques. In *Defect correction methods*, pages 89–113. Springer, 1984.

[7] Roger A. Horn and Charles R. Johnson. *Matrix analysis.* Second Edition. Cambridge University Press, 2013.

[8] Rajendra K. Ray H.V.R. Mittal, Jiten C. Kalita. A class of finite difference schemes for interface problems with an hoc approach. *International journal for numerical methods in fluids*, 86, 2016.

[9] Steve McCormick. Fast adaptive composite grid (fac) methods: Theory for the variational case. In *Defect correction methods*, pages 115–121. Springer, 1984.

[10] R Minero, MJH Anthonissen, and RMM Mattheij. A local defect correction technique for time-dependent problems. *Numerical Methods for Partial Differential Equations*, 22(1):128–144, 2006.

[11] D.C. Dalal Swapan K. Pandit, Jiten C. Kalita. A fourth-order accurate compact scheme for the solution of steady navier–stokes equations on non-uniform grids. *Computers and fluids*, 37, 2007.

[12] Kiel W.Hackbusch. Local defect correction method and domain decomposition techniques. In K.Bohmer H.J.Stetter, editor, *Defect correction methods theory and applications*, chapter 6, pages 90–95. Springer-Verlag Wien, New York.