Constructing a Standalone Navier-Stokes Solver

Application of Generalized Finite-Difference, Non-Standard Boundary Conditions, and Gradient-Ascent Mesh-Morphing

William van Noordt

MATHEMATICS

COLORADO STATE UNIVERSITY

Department of Mathematics
Colorado State University
May 11, 2018

Note:

The following document was completed in the Spring of 2018 in completion of my thesis for my undergraduate degree in applied mathematics at Colorado State University. As such, it contains errors, inefficiencies, and unusual conventions that some may find objectionable. I chose not to address these when I uploaded this document here because I felt the need to preserve the state of my development at the time this was written. Ultimately, we all must start somewhere, and I consider this to be where I started. Thank you for reading.

Will van Noordt

Contents

Constructing a Standalone Navier-Stokes Solver
Application of Generalized Finite-Difference, Non-Standard Boundary Conditions, and Gradient-Ascent Mesh-Morphing

Will van Noordt

May 11, 2018

Abstract

The purpose of the material presented hereafter is to explore a number of novel techniques as applied to numerically estimating the solution to the Navier-Stokes system of equations under some simplifying assumptions. There are three techniques that were explored: applying boundary conditions to the pressure distribution, using a generalized finite-difference method for non-orthogonal grids, and a novel gradient-ascent mesh-morphing algorithm. It was determined that, while applying boundary conditions to the pressure distribution resulted in marginally erroneous results, the finite-difference method and mesh-morphing algorithm that were used improved the quality of solutions.

1 Foreword

In addition to exploring the aforementioned novel techniques, this document has a secondary purpose: to serve as a loose guide for any individual interested in constructing a standalone solver. Although numerous solving packages and programs are available, the process of developing one serves as an excellent guide to gaining an intimate understanding of the mechanisms that solvers necessarily employ.

This document will explore three primary techniques, and each section will be loosely segregated into sections that each discuss one of these techniques.

2 Background

2.1 Background - CFD

Computational Fluid Dynamics (hereafter referred to as CFD) is a field of mechanical engineering that involves applying numerical PDE solving techniques to equations governing viscous fluid flow and heat transfer around and through physical boundaries. Typical applications of CFD include the following:

- Development, analysis, and evaluation of aerodynamic bodies (*i.e.* airfoils, drag-critical vehicles, aerodynamic thrust mechanisms, etc.)
- Simulation of combustion processes
- Analysis of free and forced convection heat transfer devices
- Simulation of intravenous blood flow and homeostatic processes

The numerical schemes in CFD typically require high-quality mesh generation for the purpose of producing a numerical solution. Typical features of high-quality meshes will include high node density in areas of interest (boundary layers, turbulent regions, pressure wavefronts) and low node density in regions of low interest (free streams, large isothermal surfaces). Generating high-quality meshes is critical to the accuracy and usability of the numerical solutions.

2.2 Background - Finite-Difference

When a numerical solution for a PDE is to be computed on some domain, a typical approach is to divide the domain into a grid, the express the derivatives present in the differential equations governing the solution in terms of approximations that depend on the value of the solution at a set of nodes. In many cases (such as typical practice in AMR applied to many CFD problems), these nodes lie on an orthogonal, Cartesian grid (see Section 5.2), and therefore yield reasonably high-fidelity approximations given below, for second-order (and lower order) derivatives:

$$\begin{bmatrix} u_x \\ u_y \\ u_{xx} \\ u_{yy} \\ u_{xy} \end{bmatrix} \approx \begin{bmatrix} \frac{u(x+\Delta x,y)-u(x-\Delta x,y)}{2\Delta x} \\ \frac{u(x,y+\Delta y)-u(x,y-\Delta y)}{2\Delta y} \\ \frac{u(x+\Delta x,y)+u(x-\Delta x,y)-2u(x,y)}{\Delta x^2} \\ \frac{u(x,y+\Delta y)+u(x,y-\Delta y)-2u(x,y)}{\Delta y^2} \\ \frac{(u(x+\Delta x,y+\Delta y)-u(x-\Delta x,y+\Delta y))-(u(x+\Delta x,y-\Delta y)-u(x-\Delta x,y-\Delta y))}{4\Delta x\Delta y} \end{bmatrix}.$$

These approximations are of acceptable accuracy for many applications, but it may not necessarily be the case that the quantities above can be easily calculated, especially if the underlying grid is not Cartesian. A typical example of this arises in grids generated for the large-eddy turbulence analysis over an airfoil, where unstructured grids are generated as in Figure 2.1. Note that near the boundary layer of the airfoil (the fluid body closest to the surface of the airfoil), there is a high node-density.

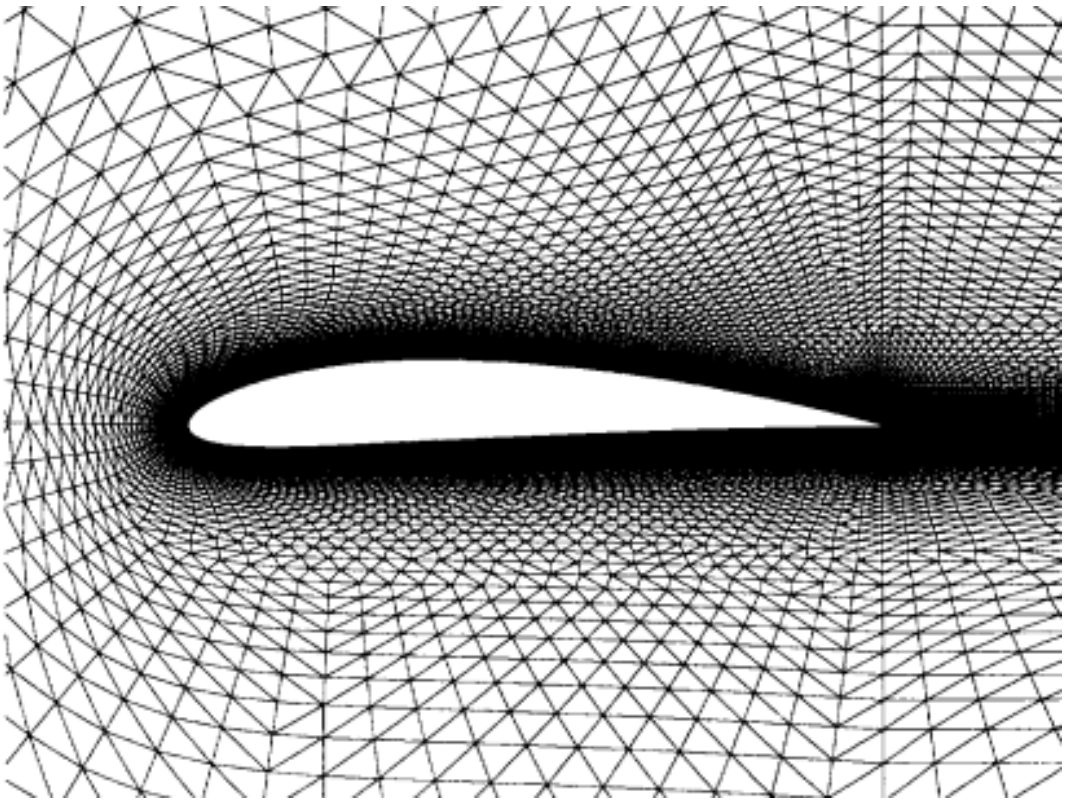


Figure 2.1: A typical example of an unstructured grid superimposed over an airfoil-flow domain. Image courtesy of K. Jansen [7].

Therefore, when solving problems on grids that are not Cartesian and orthogonal, it becomes necessary to improve upon the approximation given above to solve governing equations.

2.3 Background - Fluid Equations

2.3.1 Terms and Definitions

It will be helpful to understand the following vocabulary when considering the ideas presented in this material:

- *Viscous/Laminar flow*: a state of fluid flow where motion is governed primarily by viscous forces.
- *Inviscid/Turbulent flow*: a state of a fluid flow where motion is governed primarily by the inertia of fluid particles.

It will also be of use to understand the following terms in later equations:

- ρ: the density of a fluid (a positive constant).
- μ: the viscosity of a fluid (a positive constant).
- k: the heat conduction coefficient of a fluid (a positive constant).
- M_∞: The ratio between the free-stream velocity and the speed of sound in the conditions of the free stream (a positive constant).
- γ: The ratio of specific heat capacities of a fluid (a positive constant).
- E_t: the total energy per unit volume of a fluid.
- Pr: the Prandtl number of a fluid (a positive constant).
- Re_L: the Reynolds number of a fluid (a positive constant).

2.3.2 Euler Fluid Equaitons

Leonhard Euler was one of the first individuals to attempt to describe the motion of fluids by investigating fluid properties. Euler's description involved expressing the behavior of conserved quantities moving through infinitessimal control volumes through the domain of interest (see Figure 2.2). In doing so, and applying Newton's second law of motion Euler concluded that the velocity $\mathbf{v}$ of a fluid was governed by equation (2.1), which simply imposes a force balance on an infinitessimal control volume.

$$\rho \frac{D^{[\mathbf{v}]}\mathbf{v}}{Dt} = -\nabla p \tag{2.1}$$

Note that if $\mathbf{v}$ is expressed in terms of an orthonormal basis $\cup_{i=1}^{N}\{\mathbf{e}_i\}$, then

$$\frac{D^{[\mathbf{v}]}}{Dt} = \frac{\partial}{\partial t} + \sum_{i=1}^{N} v_i \frac{\partial}{\partial \mathbf{e}_i}.$$

For example, if $\mathbf{v} = \begin{bmatrix} u & v & w \end{bmatrix}^T$, then

$$\frac{D^{[\mathbf{v}]}\mathbf{v}}{Dt} = \frac{\partial}{\partial t}\begin{bmatrix} u \\ v \\ w \end{bmatrix} + u\frac{\partial}{\partial x}\begin{bmatrix} u \\ v \\ w \end{bmatrix} + v\frac{\partial}{\partial y}\begin{bmatrix} u \\ v \\ w \end{bmatrix} + w\frac{\partial}{\partial z}\begin{bmatrix} u \\ v \\ w \end{bmatrix}.$$

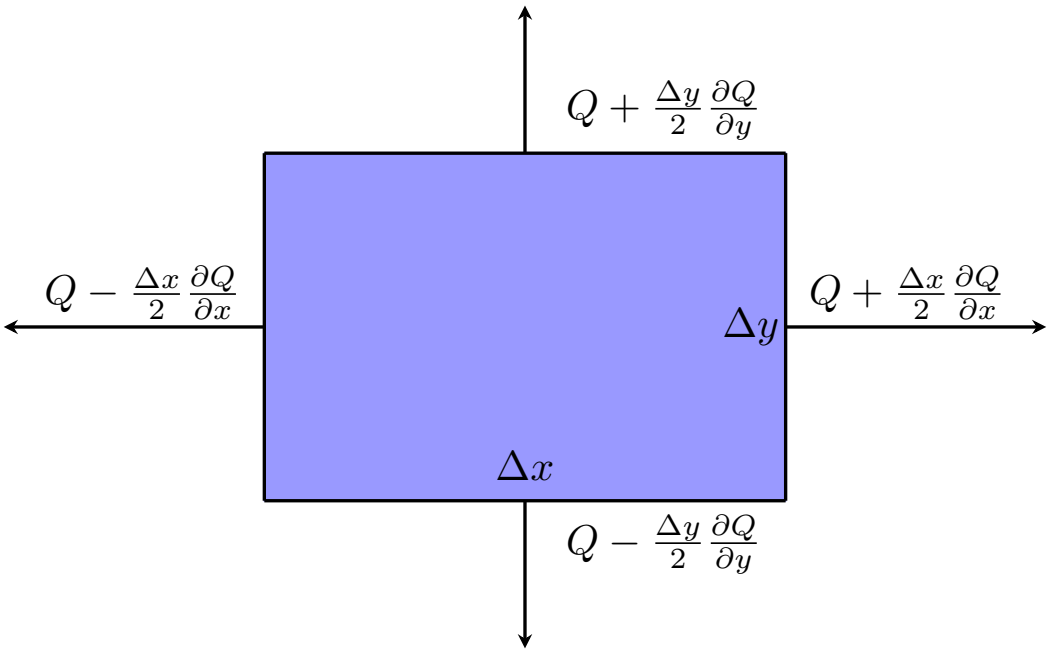


Figure 2.2: An example of a cartesian control volume considered when describing the behavior of a conserved quantity Q.

For the purpose of the material presented hereafter, it will be assumed that the operating vector $\mathbf{v}$ is a velocity vector in a fluid flow, and so the notation

$$\frac{D^{[\mathbf{v}]}}{Dt} = \frac{D}{Dt}$$

will be used.

The primary issue with this set of equations is that the force balance that Euler applied was missing some terms, namely those arising from viscous interactions between fluid elements and their surrounding aspects.

2.3.3 Navier-Stokes Fluid Equations

In order to more accurately model fluid flow, the viscous force on a fluid element must be considered. These considerations are not present in (2.1). The classical Navier-Stokes equation (given in equation 2.2) is thereby an important tool used for modeling fluid motion. For the remainder of this material, it will be assumed that $\mathbf{v} = \begin{bmatrix} u & v & w \end{bmatrix}^T$ is the velocity of the flow field in question.

$$\rho \frac{D\mathbf{v}}{Dt} = -\nabla p + \tau \tag{2.2}$$

$$\nabla \cdot (\rho \mathbf{v}) = 0 \tag{2.3}$$

$$k = \left[\frac{M_\infty \mu}{Re_L Pr(\gamma - 1)} \right] \tag{2.4}$$

$$\mu = C_1 \frac{T^{\frac{3}{2}}}{T + C_2} \tag{2.5}$$

$$p = (\gamma - 1)\left[E_t - \frac{\rho}{2}|\mathbf{v}|^2 \right] \tag{2.6}$$

The relation between the pressure of a fluid and its velocity is given in (2.6), and the continuity equation given in (2.3) is an expression of a reduced mass-conservation law, assuming constant density ρ. Note that as given in (2.2), from R. Cummings et. al [1], we have

$$\tau = \begin{bmatrix} \frac{\partial}{\partial x}\Big(2\mu\frac{\partial u}{\partial x} - \frac{2}{3}\mu\nabla\cdot\mathbf{v}\Big) + \frac{\partial}{\partial y}\Big(\mu\Big(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\Big)\Big) + \frac{\partial}{\partial z}\Big(\mu\Big(\frac{\partial w}{\partial x} + \frac{\partial u}{\partial z}\Big)\Big) \\ \frac{\partial}{\partial x}\Big(\mu\Big(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\Big)\Big) + \frac{\partial}{\partial y}\Big(2\mu\frac{\partial v}{\partial y} - \frac{2}{3}\mu\nabla\cdot\mathbf{v}\Big) + \frac{\partial}{\partial z}\Big(\mu\Big(\frac{\partial w}{\partial y} + \frac{\partial v}{\partial z}\Big)\Big) \\ \frac{\partial}{\partial x}\Big(\mu\Big(\frac{\partial w}{\partial x} + \frac{\partial u}{\partial z}\Big)\Big) + \frac{\partial}{\partial y}\Big(\mu\Big(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y}\Big)\Big) + \frac{\partial}{\partial z}\Big(2\mu\frac{\partial w}{\partial z} - \frac{2}{3}\mu\nabla\cdot\mathbf{v}\Big) \end{bmatrix}. \tag{2.7}$$

These equations are not in the forms typically used in computational fluid dynamics. These forms typically give rise to numerical instabilities, which can prove troublesome to handle.

R. Cummings et. al [1] suggests a transformed version of the equations (2.2) through (2.6) given by

$$\frac{\partial\mathbf{Q}}{\partial t} + \frac{\partial(\mathbf{F}-\mathbf{F}_v)}{\partial x} + \frac{\partial(\mathbf{G}-\mathbf{G}_v)}{\partial y} + \frac{\partial(\mathbf{H}-\mathbf{H}_v)}{\partial z}. \tag{2.8}$$

The vectors in the equation above are given by inviscid terms

$$\mathbf{Q} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ E_t \end{bmatrix} \quad \mathbf{F} = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uw \\ (E_t + p)u \end{bmatrix} \quad \mathbf{G} = \begin{bmatrix} \rho v \\ \rho vu \\ \rho v^2 + p \\ \rho vw \\ (E_t + p)v \end{bmatrix} \quad \mathbf{H} = \begin{bmatrix} \rho w \\ \rho wu \\ \rho wv \\ \rho w^2 + p \\ (E_t + p)w \end{bmatrix} \tag{2.9}$$

and viscous terms

$$\mathbf{F_v} = \begin{bmatrix} 0 \\ \tau_{xx} \\ \tau_{xy} \\ \tau_{xz} \\ u\tau_{xx} + v\tau_{xy} + w\tau_{xz} + q_x \end{bmatrix} \quad \mathbf{G_v} = \begin{bmatrix} 0 \\ \tau_{yx} \\ \tau_{yy} \\ \tau_{yz} \\ u\tau_{yx} + v\tau_{yy} + w\tau_{yz} + q_y \end{bmatrix} \quad \mathbf{H_v} = \begin{bmatrix} 0 \\ \tau_{zx} \\ \tau_{zy} \\ \tau_{zz} \\ u\tau_{zx} + v\tau_{zy} + w\tau_{zz} + q_z \end{bmatrix} \tag{2.10}$$

Note that if x_1 denotes the x-direction, x_2 denotes the y-direction, and x_3 denotes the z-direction, then

$$\tau_{x_i,x_j} = \frac{M_\infty}{Re_L}\left[\mu\Big(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\Big) - \frac{2}{3}\mu\nabla\cdot\mathbf{v}\delta_{i,j}\right]$$

where $\delta_{i,j}$ denotes the Kroenecker delta, and

$$q_{x_i} = -\left[\frac{M_\infty\mu}{Re_L Pr(\gamma-1)}\right]\frac{\partial T}{\partial x_i} = -k\frac{\partial T}{\partial x_i}. \quad \text{(see (6.6))}$$

Note that we have the relation that

$$\frac{M_\infty}{Re_L} = \frac{k\mu}{Pr(\gamma-1)}.$$

Observation of (2.9) and (2.10) might reflect that there are now five partial differential equations (since $\dim(\mathbf{Q}) = \dim(\mathbf{F}) = \dim(\mathbf{G}) = ... = \dim(\mathbf{H_v}) = 5$) and three expressions relating fluid properties, given by (2.4), (2.5), and (2.6). The unkowns being considered are three components of the velocity field u, v, and w, the pressure field p, the temperature field T, the density field ρ, and the viscosity and thermal conductivity μ and k, respectively.

3 Related Work

3.1 Related Work - Mesh-Morphing

Multiple studies of AMR techniques have been published, and a vast majority all of them explore algorithms for designating refinement locations across solutions or interfacing or superimposing existing refined grids.

AMR generally follows the following procedure:

1. Begin with pre-generated mesh, apply numerical scheme to solve for approximate solution.
2. Identify locations of interest (see Section 8.1.4).
3. Overlay refined,Cartesian subgrids on locations of interest.
4. Couple existing solution with newly generated grid.
5. Resolve conservation equations on newly generated grid.
6. Iterate as necessary.

Figure 3.1 shows a typical subgrid applied to a mesh after one iteration of AMR. Figure 3.2 shows how AMR

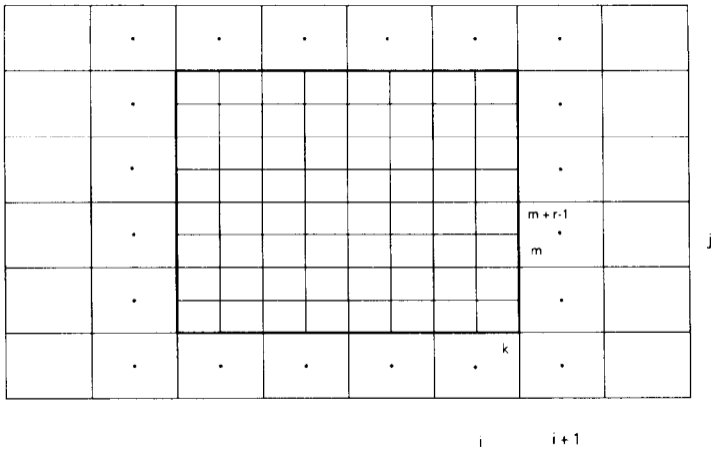


Figure 3.1: A typical Cartesian subgrid used in AMR. Image courtesy of M.J. Berger and P. Colella [4].

subgrids are typically arranged throughout a domain. These subgrids usually have high node density.

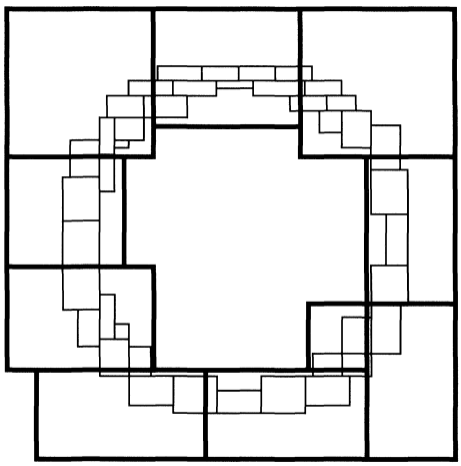


Figure 3.2: Distribution of AMR subgrids for a typical AMR method. Image courtesy of J. Bell, M. Berger, J. Saltzman, and M. Welcome [3].

3.2 Related Work - Finite-Difference

There are many methods and algorithms that have been developed for estimating derivatives on grids. There have been many methods proposed for cases such as in [12] (primarily for control volumes), where grids are *block-uniform*, or uniform for certain regions within the domain (see Figure 3.3). Such methods are particularly useful for post-processing of distribution data when considering problems in heat transfer, CFD, electromagnetics, and other problems where flux quantities are important.

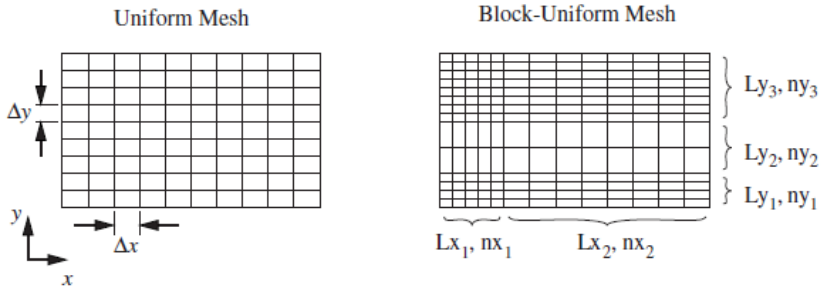


Figure 3.3: A comparison of a uniform grid and a block-uniform grid. Image courtesy of G. Recktenwald [12].

Meanwhile, for unstructured grids, derivative estimation schemes are usually more complex. These methods are frequently based on regressions, nonlinear systems, or statistics, though some more explicit methods do exist. J. Benito [5] suggests a method that uses an arbitrary number of nodes for derivative estimation at any given node (see Figure 3.4), but resulting systems exhibit nonlinearity.

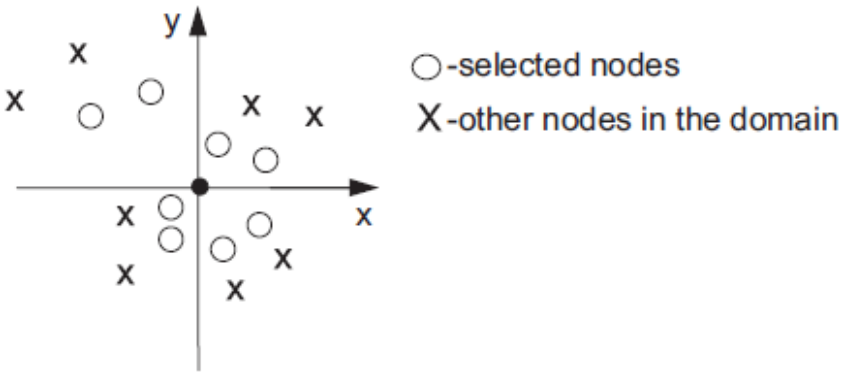


Figure 3.4: A stencil defined using quadrant criteria for an unstructured grid finite difference method. Image courtesy of J. Benito [5].

3.3 Related Work - Navier-Stokes

Although the form given in (2.2) represents the Navier-Stokes system in a simple expression, [11] suggests expressing (2.2) by defining the vorticity of a flow $\mathbf{v} = \begin{bmatrix} u & v \end{bmatrix}^T$ by

$$\zeta = |\zeta| = |\nabla \times \mathbf{v}| = \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y} \tag{3.1}$$

and defining the stream function ψ by

$$\frac{\partial \psi}{\partial y} = u \qquad \text{and} \qquad \frac{\partial \psi}{\partial x} = -v. \tag{3.2}$$

Doing so results in a transformation of 2.2 of the form given by

$$\frac{\partial \zeta}{\partial t} + u\frac{\partial \zeta}{\partial x} + v\frac{\partial \zeta}{\partial y} = \nabla^2 \zeta \tag{3.3}$$

$$\nabla^2 \psi = -\zeta,$$

which ultimately requires the use of a third-order solver. Both [9] and [11] present implementations of the orthogonal finite-difference approximation seen in section 2.2.

A problem-solving flowchart given in [11] can be found in Figure 3.5. More information on the notation used

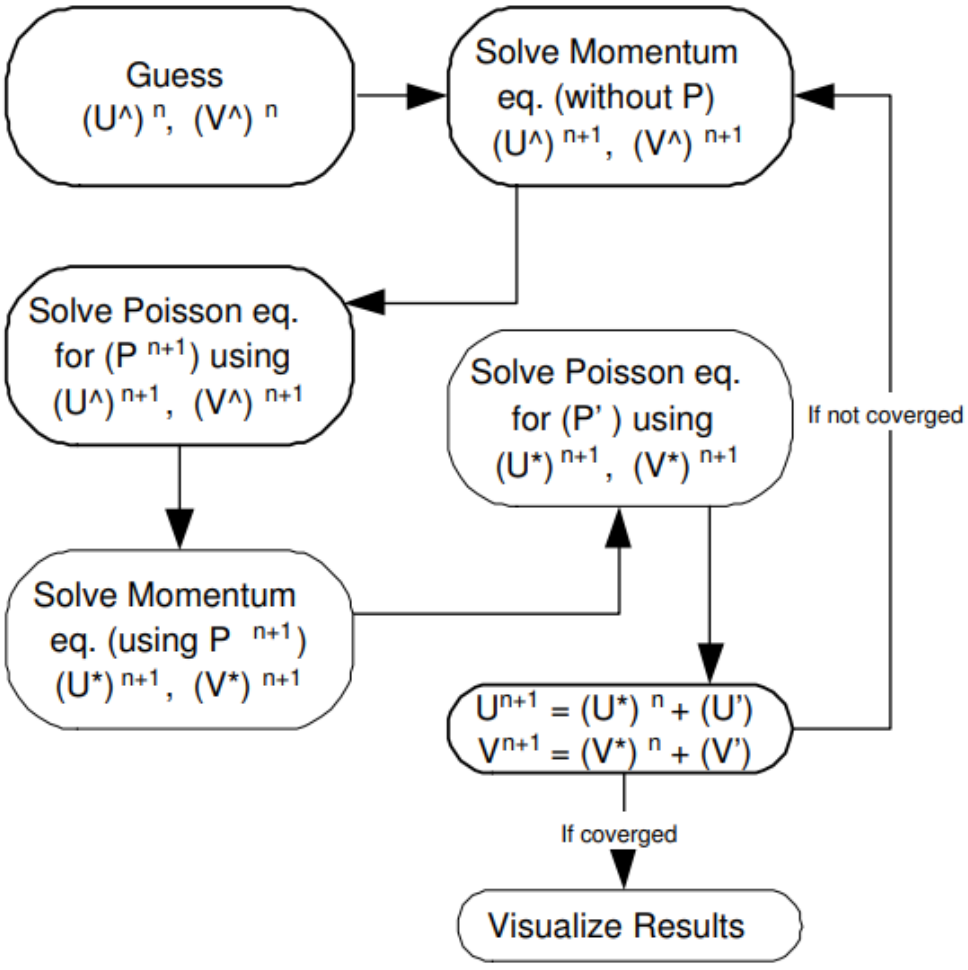


Figure 3.5: A flowchart for estimating solutions to the vorticity-stream function Navier-Stokes system. Image courtesy of M. Matyka [11].

in Figure 3.5 can be found in [9].

4 Problem Statements

4.1 Problem Formulation - Mesh Morphing

Let R be a Cartesian discretization of some rectangular domain $\Omega \in \mathbb{R}^2$, and let u be a solution to some system of partial differential equations considered on Ω with smooth boundary conditions on $\partial\Omega$, assuming that such a solution exists. Let u^* be the approximation of u over R.

The objectives are as follows:

- Find a transformation Z that can be performed on R such that the discretization of u^* over $Z(R)$ is optimal[1].
- Create a framework suitable for the future implementation of this scheme. See Section **??** for the full project source code.

The following notation will be used:

- $R = R^{[0]}$ will denote a Cartesian mesh with uniform discretization over Ω. This discretization will serve as an initial condition for an iterative application of the transformation Z.
- $R^{[1]} = Z(R^{[0]}) = Z(R)$ will denote the mesh transformed by 1 application of the transformation Z.
- $R^{[i]} = Z^i(R^{[0]}) = Z^i(R)$ will denote the resulting discretization after i applications of the transformation Z.
- $(x, y)_{h,k} = (x_{h,k}, y_{h,k})$ will denote a node in some mesh at the (h, k) position.

For transient problems, the solution u^* will contain necessary information to define Z.

The repeated application of Z to some mesh $R^{[i]}$ will involve the following for each node at the (h, k) position:

- Approximate various derivatives of u^* via interpolation.
- Use derivative information at node $(x, y)_{h,k}$ to define a function $z(x, y) : \mathbb{R}^2 \to \mathbb{R}^2$.
- Set the node at the (h, k) position of $R^{[i+1]}$ to be $z(x_{h,k}, y_{h,k})$.

For the purpose of illustration, Figure 4.1 depicts a special case of one iteration Z acting upon a single node.

4.2 Problem Formulation - Finite-Difference

We wish to estimate all second derivatives of some twice-differentiable solution u on some domain Ω at a node with index coordinates (i, j), and we assume that this node (call it u_0) does not lie on $\partial\Omega$ (lest adjacent nodes not lie on Ω).

Figure 4.2 illustrates the type of stencil upon which the derivatives of u are to be estimated.

We wish to estimate

$$\xi[u] = \begin{bmatrix} \frac{\partial u}{\partial x} \\ \frac{\partial u}{\partial y} \\ \frac{\partial^2 u}{\partial x^2} \\ \frac{\partial^2 u}{\partial y^2} \\ \frac{\partial^2 u}{\partial x \partial y} \end{bmatrix} = \begin{bmatrix} u_x \\ u_y \\ u_{xx} \\ u_{yy} \\ u_{xy} \end{bmatrix}.$$

Again, $\xi[u]$ is typically estimated via the expression in Section 2.2. Here we wish to find a matrix A such that if

$$\mathbf{u} = \begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix},$$

[1] See Section 8.1.4 for discussion on optimality of a such discretization.

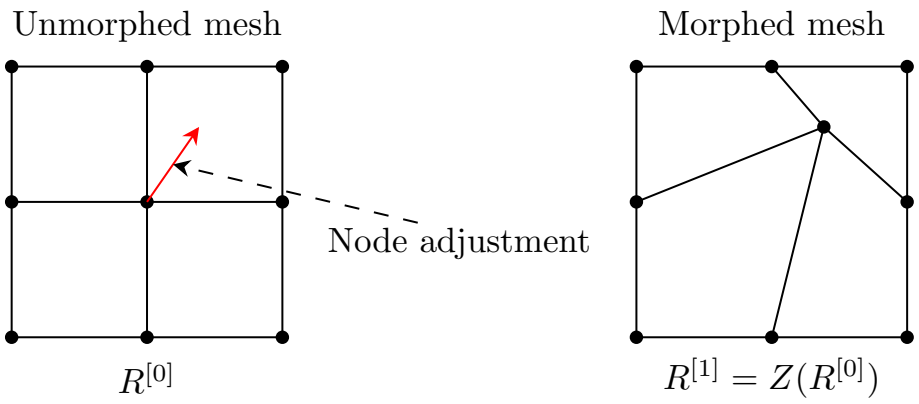


Figure 4.1: Illustration of single-node case of a single iteration of Z.

then we can approximate $\xi[u]$ with the system

$$\xi[u] \approx A \cdot \mathbf{u}.$$

The provision that $\xi[u]$ can be expressed in this linear form implies that for any second-order boundary value problem given by

$$\mathbf{v} \cdot \xi^*[u] = f \qquad \mathbf{x} \in \Omega,$$

and

$$u = g \qquad \mathbf{x} \in \partial\Omega$$

the solution to this problem, provided it exists, can be implicitly estimated using this scheme.

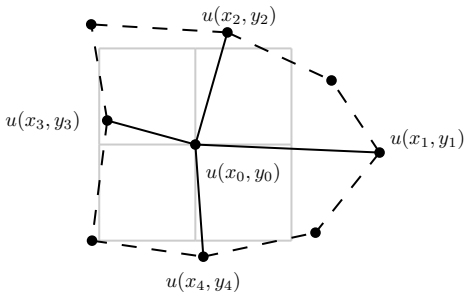


Figure 4.2: An illustration of an example stencil to have this method applied to it. The non-orthogonal stencil has been superimposed over an orthogonal stencil.

4.3 Problem Formulation - Navier-Stokes System

We wish to approximate a solution to the system given by

$$\rho\left(\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y}\right) = -\frac{\partial p}{\partial x} + \mu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right)$$

$$\rho\left(\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y}\right) = -\frac{\partial p}{\partial y} + \mu\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) \tag{4.1}$$
$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0$$

without necessitating the use of a third order finite-difference method. Namely, the finite-difference method outlined in Section 4.2 will be used to estimate solutions to (4.1).

5 Approaching the Problems

5.1 General Approach - Mesh Morphing

The general approach that this method will take will involve modifying a single existing subgrid using some iterative transformation. This process will, after generating a single subgrid, conserve the node density by translating nodes in the subgrid to regions of higher interest. The details of the transformation can be found in Section 4.1.

5.2 General Approach - Finite-Difference Method

The grids that this derivative scheme will apply to are of a particular form. In order to understand the exact type of grid that the scheme presented hereafter will apply to, it will be necessary to define the following terms:

- *Cartesian:* (of a grid) a property where for any node in a grid and not lying on the boundary of a domain, there are exactly 4 adjacent to it (though derivative approximations may extend beyond the adjacency), and each node can be uniquely described with a pair of integer index coordinates (i, j) and furthermore, the sum of the differences of the indices of any adjacent node to the node at (i, j) is exactly 1.
- *Structured:* (of a grid) a property where for any node in a grid and not on the boundary of a domain, there are a fixed number of nodes adjacent to that node.
- *Orthogonal:* (of a grid) a property where, for any given node, the angle between the vectors connecting that node and any two unique adjacent nodes is either 0, $\frac{\pi}{2}$, or π.

This scheme will apply specifically to non-orthogonal, Cartesian grids (see Figure 5.1).

Since the approximation expressed in Section 2.2 will not suffice for the type of grid shown in Figure 5.1, it will be necessary to develop a new one. However, it is expressly unnecessary to apply a scheme that may be used on an unstructured grid (see [5] and Figure 2.1), as these schemes are usually formed on the basis that the grid has no underlying structure, and are therefore more computationally intensive than they need be.

The scheme proposed by this material will take advantage of the grid's underlying structure by assuming that every derivative of some solution u at some node can be closely approximated by some linear combination of the values at the adjacent nodes.

5.3 General Approach - Navier-Stokes Estimation

The system in (4.1) will be discretized on an arbitrary, Cartesian grid. The discretization will give rise to a high-dimensional linear system which will then be solved using an iterative algorithm (lest a direct method prove prohibitively slow). Initial estimates for u and v (denoted u^* and v^* in Figure 6.3) will be applied, then updated with each iteration until convergence is reached.

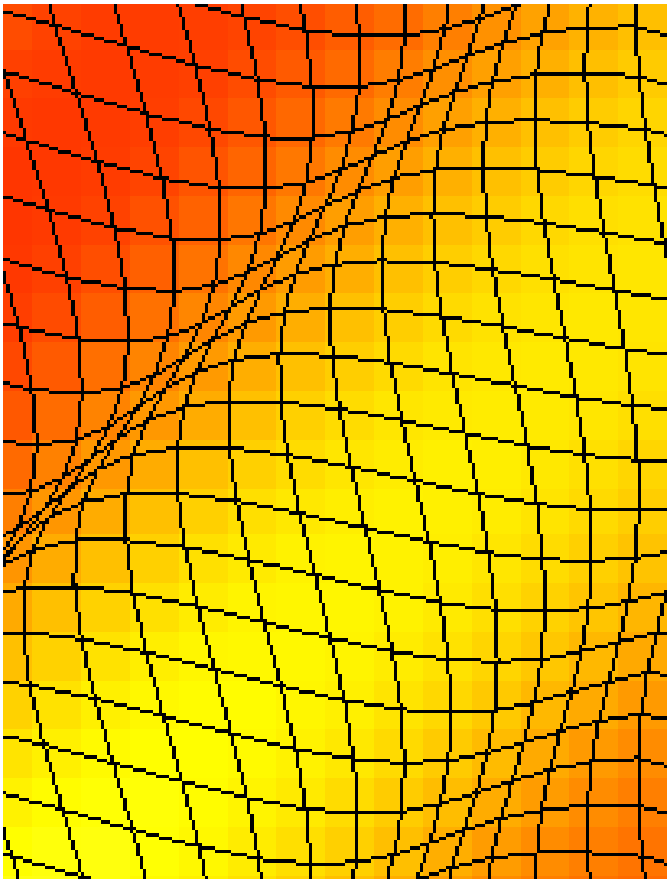


Figure 5.1: An example of a non-orthogonal, Cartesian grid.

6 Methods

6.1 Methods - Mesh-Morphing

6.1.1 A note on mesh quality

As was briefly stated in Section 2.1, it is typical of high-quality meshes to have high node density in regions where rapid changes in u^* are present. In more mathematical terms, it is of great interest that regions in which $|\nabla u|$ is large contain many nodes.

6.1.2 Gradient as an optimization quantity

As mentioned in Section 6.1.1, nodes should be placed where $|\nabla u|$ attains high values. This will motivate the following method for defining the transformation Z:

- At some node, calculate the following derivatives[2] for some fixed value of $\Delta x, \Delta y$:
  - $u_x = \frac{\partial u}{\partial x} \approx \frac{u(x+\Delta x,y)-u(x-\Delta x,y)}{2\Delta x}$
  - $u_y = \frac{\partial u}{\partial y} \approx \frac{u(x,y+\Delta y)-u(x,y-\Delta y)}{2\Delta y}$
  - $u_{xx} = \frac{\partial^2 u}{\partial x^2} \approx \frac{u(x+\Delta x,y)+u(x-\Delta x,y)-2u(x,y)}{\Delta x^2}$
  - $u_{yy} = \frac{\partial^2 u}{\partial y^2} \approx \frac{u(x,y+\Delta y)+u(x,y-\Delta y)-2u(x,y)}{\Delta y^2}$

[2]Recall that for the purpose of this project, derivatives are estimated using interpolation.

- $u_{xy} = u_{yx} = \frac{\partial}{\partial x}\frac{\partial u}{\partial y} \approx \frac{(u(x+\Delta x,y+\Delta y)-u(x-\Delta x,y+\Delta y))-(u(x+\Delta x,y-\Delta y)-u(x-\Delta x,y-\Delta y))}{4\Delta x\Delta y}$ (see section 8.1.2 for more information on this assumption.)

- Use the derivatives to calculate $\mathbf{m} = \nabla|\nabla u|^2$. Note that maximizing $\nabla|\nabla u|^2$ and maximizing $\nabla|\nabla u|$ are equivalent.
  - $\mathbf{m} = \nabla|\nabla u|^2 = 2\begin{bmatrix} u_{xx}u_x + u_{xy}u_y \\ u_{yy}u_y + u_x u_{yx} \end{bmatrix}$

- Define some step size s, then let $z(x,y)$ be given by $z(x,y) = \begin{bmatrix} x \\ y \end{bmatrix} + s\mathbf{m}$.

- If $\begin{bmatrix} x \\ y \end{bmatrix} + s\mathbf{m} \notin \Omega$, use adjusted step size $s^* = s \cdot 2^{-p}$, where $p = \min\{c \in \mathbb{Z} : \begin{bmatrix} x \\ y \end{bmatrix} + s\mathbf{m} \in \Omega\}$[3].

- Repeat for every node.

Applying this method at every time step of a known transient solution results in a discretization that transforms along with the known solution in time (see Section **??**).

Note that for each node, this method is identical to an optimization problem approached using the gradient-ascent method of optimization.

6.2 Methods - Finite-Difference

6.2.1 Derivative estimation

The objective of this scheme is to estimate the derivatives of a distribution u using a 5-node stencil. Let $\xi[\cdot]$ be a differential operator such that

$$\xi[u] = \begin{bmatrix} \frac{\partial u}{\partial x} \\ \frac{\partial u}{\partial y} \\ \frac{\partial^2 u}{\partial x^2} \\ \frac{\partial^2 u}{\partial y^2} \\ \frac{\partial^2 u}{\partial x \partial y} \end{bmatrix} = \begin{bmatrix} u_x \\ u_y \\ u_{xx} \\ u_{yy} \\ u_{xy} \end{bmatrix}.$$

Once again, for an orthogonal Cartesian grid, these terms are usually estimated as follows, for some fixed $(\Delta x, \Delta y)$:

$$\xi[u] \approx \begin{bmatrix} \frac{u(x+\Delta x,y)-u(x-\Delta x,y)}{2\Delta x} \\ \frac{u(x,y+\Delta y)-u(x,y-\Delta y)}{2\Delta y} \\ \frac{u(x+\Delta x,y)+u(x-\Delta x,y)-2u(x,y)}{\Delta x^2} \\ \frac{u(x,y+\Delta y)+u(x,y-\Delta y)-2u(x,y)}{\Delta y^2} \\ \frac{(u(x+\Delta x,y+\Delta y)-u(x-\Delta x,y+\Delta y))-(u(x+\Delta x,y-\Delta y)-u(x-\Delta x,y-\Delta y))}{4\Delta x\Delta y} \end{bmatrix}.$$

However, when using a morphed mesh, the grid is no longer orthogonal (see Figure 6.1) and requires other methods of estimation, since $(\Delta x, \Delta y)$ is not fixed.

Note that the surplus nodes (as depicted in Figure 4.2) are necessary in the approximation of $\xi[u]$. Note that it is not necessary to choose a surplus node in this particular way, as it could be chose using other methods. This choice was motivated by the fact that it gave rise to a simpler form of the matrix A, simplifying solution estimation.

In order to approximate $\xi[u]$, we use the multivariable variant of Taylor's theorem (see [6] for notation), which states that for any sufficiently differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ and any $\mathbf{a}, \mathbf{h} \in \mathbb{R}^n$, we have, for some $c \in (0,1)$:

$$f(\mathbf{a}+\mathbf{h}) = \sum_{|\alpha|\le k} \frac{\partial^\alpha f(\mathbf{a})}{\alpha!}\mathbf{h}^\alpha + \sum_{|\alpha|=k+1} \frac{\partial^\alpha f(\mathbf{a}+c\mathbf{h})}{\alpha!}\mathbf{h}^\alpha. \tag{6.1}$$

Evaluating this expression (truncating at the $k = 2$ term) we get the following quantities, provided that, in this case, $\dim(\alpha) = 2$:

[3]Note that the determination of the step size will change based on the derivative estimation scheme used.

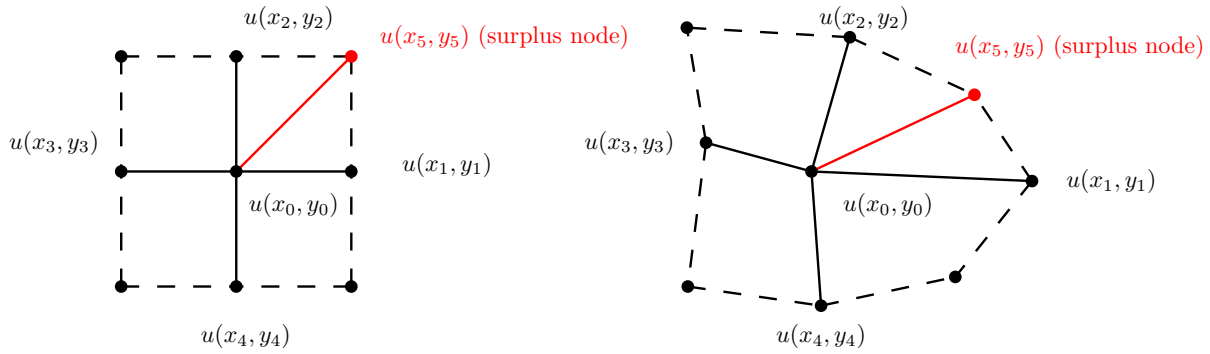


Figure 6.1: A depiction of an orthogonal stencil (left) and a non-orthogonal stencil (right).

Table 1: Taylor expansion terms given in equation 6.1, for $k = 2$ and $h = (h_1, h_2)$.

α	$\partial^\alpha f$	$\alpha!$	h^α
(0,0)	$f(\mathbf{a})$	$0!0! = 1$	$h_1^0 h_2^0 = 1$
(1,0)	$f_x(\mathbf{a})$	$1!0! = 1$	$h_1^1 h_2^0 = h_1$
(0,1)	$f_y(\mathbf{a})$	$0!1! = 1$	$h_1^0 h_2^1 = h_2$
(2,0)	$f_{xx}(\mathbf{a})$	$2!0! = 2$	$h_1^2 h_2^0 = h_1^2$
(0,2)	$f_{yy}(\mathbf{a})$	$0!2! = 2$	$h_1^0 h_2^2 = h_2^2$
(1,1)	$f_{xy}(\mathbf{a})$	$1!1! = 1$	$h_1^1 h_2^1 = h_1 h_2$

We note that for some given displacement vector $\mathbf{h}$, we have, for some $c \in (0, 1)$:

$$f(\mathbf{a}+\mathbf{h}) = f(\mathbf{a}) + h_1 f_x(\mathbf{a}) + h_2 f_y(\mathbf{a}) + \frac{1}{2}h_1^2 f_{xx}(\mathbf{a}) + \frac{1}{2}h_2^2 f_{yy}(\mathbf{a}) + h_1 h_2 f_{xy}(\mathbf{a}) + \sum_{|\alpha|=k+1} \frac{\partial^\alpha f(\mathbf{a}+c\mathbf{h})}{\alpha!}\mathbf{h}^\alpha. \quad (6.2)$$

Since we have 4 nodes and 1 surplus node on a given stencil (see Figure 6.1), we are given 5 equations. Neglecting the 3^{rd} order error terms, we have

$$f(x_1, y_1) = f(x_0, y_0) + (x_1 - x_0) f_x(x_0, y_0) + (y_1 - y_0) f_y(x_0, y_0) + \frac{1}{2}(x_1 - x_0)^2 f_{xx}(x_0, y_0) + \frac{1}{2}(y_1 - y_0)^2 f_{yy}(x_0, y_0)$$
$$+(x_1 - x_0)(y_1 - y_0) f_{xy}(x_0, y_0)$$

$$\dots$$

$$f(x_5, y_5) = f(x_0, y_0) + (x_5 - x_0) f_x(x_0, y_0) + (y_5 - y_0) f_y(x_0, y_0) + \frac{1}{2}(x_5 - x_0)^2 f_{xx}(x_0, y_0) + \frac{1}{2}(y_5 - y_0)^2 f_{yy}(x_0, y_0)$$
$$+(x_5 - x_0)(y_5 - y_0) f_{xy}(x_0, y_0)$$

Note that all of the above equations have similar forms for each node. Now, letting

$$\Delta\mathbf{f} = \begin{bmatrix} f(x_1, y_1) - f(x_0, y_0) \\ f(x_2, y_2) - f(x_0, y_0) \\ f(x_3, y_3) - f(x_0, y_0) \\ f(x_4, y_4) - f(x_0, y_0) \\ f(x_5, y_5) - f(x_0, y_0) \end{bmatrix} \quad \text{and} \quad A = \begin{bmatrix} (x_1 - x_0) & (y_1 - y_0) & \frac{1}{2}(x_1 - x_0)^2 & \frac{1}{2}(y_1 - y_0)^2 & (x_1 - x_0)(y_1 - y_0) \\ (x_2 - x_0) & (y_2 - y_0) & \frac{1}{2}(x_2 - x_0)^2 & \frac{1}{2}(y_2 - y_0)^2 & (x_2 - x_0)(y_2 - y_0) \\ (x_3 - x_0) & (y_3 - y_0) & \frac{1}{2}(x_3 - x_0)^2 & \frac{1}{2}(y_3 - y_0)^2 & (x_3 - x_0)(y_3 - y_0) \\ (x_4 - x_0) & (y_4 - y_0) & \frac{1}{2}(x_4 - x_0)^2 & \frac{1}{2}(y_4 - y_0)^2 & (x_4 - x_0)(y_4 - y_0) \\ (x_5 - x_0) & (y_5 - y_0) & \frac{1}{2}(x_5 - x_0)^2 & \frac{1}{2}(y_5 - y_0)^2 & (x_5 - x_0)(y_5 - y_0) \end{bmatrix},$$

we can see that, letting $f = u$ that

$$\xi[u] = A\Delta\mathbf{u}.$$

6.2.2 Conditioning the system matrix

Note that for fine meshes, the condition number $\text{cond}(A)$ could be large, leading to errors in the derivative estimates. For this purpose, we introduce a conditioning matrix K such that $\text{cond}(KA) < \text{cond}(A)$. Then, as long as $(KA)^{-1}$ exists, then approximating $\xi[u]$ by computing $\xi[u] = K(KA)^{-1} = KA^{-1}K^{-1}$.

6.2.3 Building a linear system

Let $A^* = KA$. We now estimate $\xi[u]$ by noting that

$$\xi[u] \approx K(A^*)^{-1}\Delta\mathbf{u}$$

for some conditioning matrix

$$K = \begin{bmatrix} k & 0 & 0 & 0 & 0 \\ 0 & k & 0 & 0 & 0 \\ 0 & 0 & k^2 & 0 & 0 \\ 0 & 0 & 0 & k^2 & 0 \\ 0 & 0 & 0 & 0 & k^2 \end{bmatrix}.$$

For the purpose of this project, arbitrary values of k were chosen based on empirical results[4] from numerical experiments. We let $B = K(A^*)^{-1}$, where

$$B = \begin{bmatrix} b_{1,1} & b_{1,2} & ... & b_{1,5} \\ b_{2,1} & b_{2,2} & ... & b_{2,5} \\ ... & ... & ... & ... \\ b_{5,1} & b_{5,2} & ... & b_{5,5} \end{bmatrix}$$

Previously, at any given node u_0, we defined nodes on that stencil as $u_1, ..., u_5$. In order to build an implicit matrix, we must use a different notation.

We recognize that at any node (i, j), we can estimate all second order derivatives by using the system $(\xi[u])_{i,j} \approx B_{[i,j]}\Delta\mathbf{u}_{i,j}$. Note that $B_{[i,j]}$ denotes a matrix unique to node (i, j) rather than the entry of B in the i^{th} row and j^{th} column. The latter case shall be expressed, if necessary, as $B_{i,j}$. We expand to find that

$$\begin{bmatrix} u_x \\ u_y \\ u_{xx} \\ u_{yy} \\ u_{xy} \end{bmatrix} \approx \begin{bmatrix} b_{1,1} & b_{1,2} & ... & b_{1,5} \\ b_{2,1} & b_{2,2} & ... & b_{2,5} \\ ... & ... & ... & ... \\ b_{5,1} & b_{5,2} & ... & b_{5,5} \end{bmatrix}_{[i,j]} \begin{bmatrix} u_{i+1,j} - u_{i,j} \\ u_{i,j+1} - u_{i,j} \\ u_{i-1,j} - u_{i,j} \\ u_{i,j-1} - u_{i,j} \\ u_{i+r_1,j+r_2} - u_{i,j} \end{bmatrix},$$

where $r_1, r_2 \in \{-1, 1\}$ are surplus node relative indices. Noting that, for some general linear, second-order PDE to be solved on a domain Ω, we have that

$$\langle \mathbf{v}, \xi[u] \rangle + ku = f \qquad \text{for} \qquad \mathbf{x} \in \Omega \tag{6.3}$$

and the Dirichlet boundary condition

$$u = g \qquad (\mathbf{x}) \in \partial\Omega$$

for constants $v_1, ..., v_5$, and that

$$\xi[u]_{i,j} \approx \begin{bmatrix} b_{1,1}(u_{i+1,j} - u_{i,j}) + b_{1,2}(u_{i,j+1} - u_{i,j}) + b_{1,3}(u_{i-1,j} - u_{i,j}) + b_{1,4}(u_{i,j-1} - u_{i,j}) + b_{1,5}(u_{i+r_1,j+r_2} - u_{i,j}) \\ b_{2,1}(u_{i+1,j} - u_{i,j}) + b_{2,2}(u_{i,j+1} - u_{i,j}) + b_{2,3}(u_{i-1,j} - u_{i,j}) + b_{2,4}(u_{i,j-1} - u_{i,j}) + b_{2,5}(u_{i+r_1,j+r_2} - u_{i,j}) \\ b_{3,1}(u_{i+1,j} - u_{i,j}) + b_{3,2}(u_{i,j+1} - u_{i,j}) + b_{3,3}(u_{i-1,j} - u_{i,j}) + b_{3,4}(u_{i,j-1} - u_{i,j}) + b_{3,5}(u_{i+r_1,j+r_2} - u_{i,j}) \\ b_{4,1}(u_{i+1,j} - u_{i,j}) + b_{4,2}(u_{i,j+1} - u_{i,j}) + b_{4,3}(u_{i-1,j} - u_{i,j}) + b_{4,4}(u_{i,j-1} - u_{i,j}) + b_{4,5}(u_{i+r_1,j+r_2} - u_{i,j}) \\ b_{5,1}(u_{i+1,j} - u_{i,j}) + b_{5,2}(u_{i,j+1} - u_{i,j}) + b_{5,3}(u_{i-1,j} - u_{i,j}) + b_{5,4}(u_{i,j-1} - u_{i,j}) + b_{5,5}(u_{i+r_1,j+r_2} - u_{i,j}) \end{bmatrix}.$$

We make a substitution into (6.3) at node (i, j) to find that

[4] A more analytical avenue may yield insightful results.

$$f_{i,j} = v_1\Bigg(b_{1,1}(u_{i+1,j}-u_{i,j}) + b_{1,2}(u_{i,j+1}-u_{i,j}) + b_{1,3}(u_{i-1,j}-u_{i,j}) + b_{1,4}(u_{i,j-1}-u_{i,j}) + b_{1,5}(u_{i+r_1,j+r_2}-u_{i,j})\Bigg)$$

$$+v_2\Bigg(b_{2,1}(u_{i+1,j}-u_{i,j}) + b_{2,2}(u_{i,j+1}-u_{i,j}) + b_{2,3}(u_{i-1,j}-u_{i,j}) + b_{2,4}(u_{i,j-1}-u_{i,j}) + b_{2,5}(u_{i+r_1,j+r_2}-u_{i,j})\Bigg)$$

$$+v_3\Bigg(b_{3,1}(u_{i+1,j}-u_{i,j}) + b_{3,2}(u_{i,j+1}-u_{i,j}) + b_{3,3}(u_{i-1,j}-u_{i,j}) + b_{3,4}(u_{i,j-1}-u_{i,j}) + b_{3,5}(u_{i+r_1,j+r_2}-u_{i,j})\Bigg)$$

$$+v_4\Bigg(b_{4,1}(u_{i+1,j}-u_{i,j}) + b_{4,2}(u_{i,j+1}-u_{i,j}) + b_{4,3}(u_{i-1,j}-u_{i,j}) + b_{4,4}(u_{i,j-1}-u_{i,j}) + b_{4,5}(u_{i+r_1,j+r_2}-u_{i,j})\Bigg)$$

$$+v_5\Bigg(b_{5,1}(u_{i+1,j}-u_{i,j}) + b_{5,2}(u_{i,j+1}-u_{i,j}) + b_{5,3}(u_{i-1,j}-u_{i,j}) + b_{5,4}(u_{i,j-1}-u_{i,j}) + b_{5,5}(u_{i+r_1,j+r_2}-u_{i,j})\Bigg),$$

or

$$f_{i,j} = -\Bigg(\sum_{h=1}^{5} v_h\Bigg(\sum_{k=1}^{5} b_{h,k}\Bigg)\Bigg)u_{i,j} + \Bigg(\sum_{h=1}^{5} v_h b_{h,1}\Bigg)u_{i+1,j} + \Bigg(\sum_{h=1}^{5} v_h b_{h,2}\Bigg)u_{i,j+1} + \Bigg(\sum_{h=1}^{5} v_h b_{h,3}\Bigg)u_{i-1,j} + \Bigg(\sum_{h=1}^{5} v_h b_{h,4}\Bigg)u_{i,j-1}$$
$$+\Bigg(\sum_{h=1}^{5} v_h b_{h,5}\Bigg)u_{i+r_1,j+r_2}. \tag{6.4}$$

We wish to form a complete implicit system, where

$$W\mathbf{u} = \mathbf{k}.$$

From (6.4), it becomes clear that it will be of great convenience to choose (r_1, r_2) such that, at any node (i, j), we have that $(x, y)_{i+r_1,j+r_2} \notin \partial\Omega$. Note also that if one of the nodes considered in (6.4) lies on $\partial\Omega$, it will be necessary to move one of the terms on the right side to the left side. Assuming that Ω has been discretized initially by an $m \times n$ grid, it becomes clear that $u_{i,j}$ is known if $i \in \{0, m\}$ or $j \in \{0, n\}$.

We now know that we have $(m-2)(n-2)$ unknowns that must be solved for. These are

$$\{u_{i,j} \quad 1 \le i \le m-1,\ 1 \le j \le n-1\}.$$

The system is thereby fully defined.

6.2.4 Error evaluation

We estimate the error by applying the norm

$$||u - u^*||^2 = \int_\Omega \Big(u(\mathbf{x}) - u^*(\mathbf{x})\Big)^2\, d\mathbf{x}.$$

In order to evaluate this norm, we assume that in a region $P_{i,j}$ surrounding a node (i, j) of u^*, we have that

$$u^*(\mathbf{x}) = u_{i,j} \quad \text{when} \quad \mathbf{x} \in P_{i,j}.$$

To define the region $P_{i,j}$ more specifically, we consider a parallelogram formed by the centers of mass of the adjacent nodes (see Figure 6.2).

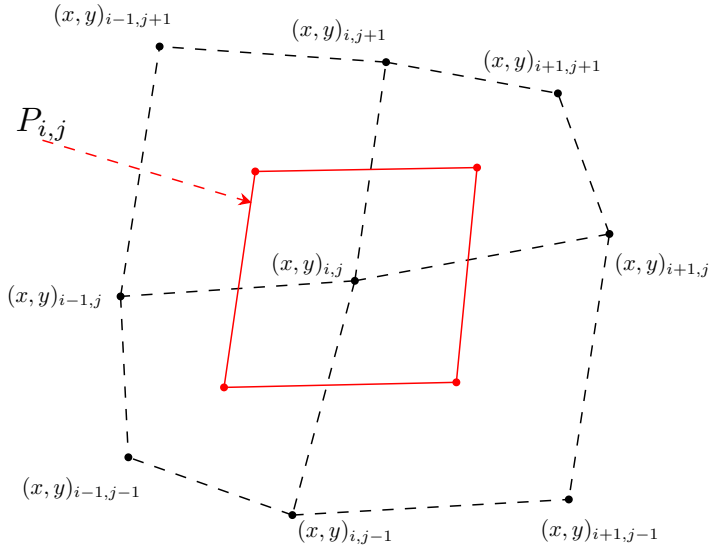


Figure 6.2: Example of the region $P_{i,j}$.

using this assumption, it becomes apparent that

$$||u - u^*||^2 \approx \sum_{(x,y)_{i,j} \in \Omega} \int_{P_{i,j}} \Big(u(\mathbf{x}) - u_{i,j}\Big)^2 \, d\mathbf{x}.$$

Assuming that u is can be considered constant on every $P_{i,j}$, we attain the error as the sum given by

$$\epsilon^2 = ||u - u^*||^2 \approx \sum_{(x,y)_{i,j} \in \Omega} \Big(u((x,y)_{i,j}) - u_{i,j}\Big)^2 A(P_{i,j}),$$

or

$$\epsilon \approx \sqrt{\sum_{(x,y)_{i,j} \in \Omega} \Big(u((x,y)_{i,j}) - u_{i,j}\Big)^2 A(P_{i,j})}$$

where $A(P_{i,j})$ is the area of a region $P_{i,j} \subset \mathbb{R}^2$.

6.3 Methods - Navier-Stokes

6.3.1 Minor Simplifying Assumptions

The following simplifying assumptions will be used, and discussion of these assumptions can be found in Section 8.3.2.

- Isothermal flow (implying constant viscosity flow):

$$\frac{\partial T}{\partial x_i} = \frac{\partial T}{\partial t} = 0 \qquad \rightarrow \qquad \frac{\partial \mu}{\partial T} = 0 \quad \text{(constant viscosity)} \tag{6.5}$$

- Constant thermal conductivity:

$$k = \left[\frac{M_\infty \mu}{Re_L Pr(\gamma - 1)}\right] \quad \text{(constant thermal conductivity)} \tag{6.6}$$

- Adiabatic flow:

$$q_{\mathbf{n}} = 0 \tag{6.7}$$

- Incompressible flow (constant density):

$$\frac{\partial \rho}{\partial t} = 0 \qquad \text{and} \qquad \nabla \rho = \mathbf{0} \tag{6.8}$$

  which implies that (2.3) is reduced to

$$\nabla \cdot \mathbf{v} = 0 \tag{6.9}$$

- 2-dimensional flow (this assumption will be discussed in Section 6.3.2)

Note that (6.9) implies that (2.7) reduces to

$$\tau = \begin{bmatrix} \frac{\partial}{\partial x}\left(2\mu\frac{\partial u}{\partial x}\right) + \frac{\partial}{\partial y}\left(\mu\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right)\right) + \frac{\partial}{\partial z}\left(\mu\left(\frac{\partial w}{\partial x} + \frac{\partial u}{\partial z}\right)\right) \\ \frac{\partial}{\partial x}\left(\mu\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right)\right) + \frac{\partial}{\partial y}\left(2\mu\frac{\partial v}{\partial y}\right) + \frac{\partial}{\partial z}\left(\mu\left(\frac{\partial w}{\partial y} + \frac{\partial v}{\partial z}\right)\right) \\ \frac{\partial}{\partial x}\left(\mu\left(\frac{\partial w}{\partial x} + \frac{\partial u}{\partial z}\right)\right) + \frac{\partial}{\partial y}\left(\mu\left(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y}\right)\right) + \frac{\partial}{\partial z}\left(2\mu\frac{\partial w}{\partial z}\right) \end{bmatrix} = \mu \begin{bmatrix} 2\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 v}{\partial x \partial y} + \frac{\partial^2 w}{\partial x \partial z} + \frac{\partial^2 u}{\partial z^2} \\ \frac{\partial^2 u}{\partial x \partial y} + \frac{\partial^2 v}{\partial x^2} + 2\frac{\partial^2 v}{\partial y^2} + \frac{\partial^2 w}{\partial z \partial y} + \frac{\partial^2 v}{\partial z^2} \\ \frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 u}{\partial x \partial z} + \frac{\partial^2 v}{\partial z \partial y} + \frac{\partial^2 w}{\partial y^2} + 2\frac{\partial^2 w}{\partial z^2} \end{bmatrix}. \tag{6.10}$$

6.3.2 Planar (2D) Flow Assumption

So far, all equations have been stated or developed as though the flow field in question is 3-dimensional, as not to lose any critical information. However, the computations carried out for the purpose of this material assumed a 2-dimensional flow:

$$\frac{\partial \mathbf{v}}{\partial z} = 0 \qquad \text{and} \qquad w = 0. \tag{6.11}$$

Note that the assumptions in (6.11) impact the development of the equations given in Section 6.3.1, most critically, (6.10), and (2.2). Specifically:

$$\frac{D\mathbf{v}}{Dt} = \frac{\partial}{\partial t}\begin{bmatrix} u \\ v \end{bmatrix} + u\frac{\partial}{\partial x}\begin{bmatrix} u \\ v \end{bmatrix} + v\frac{\partial}{\partial y}\begin{bmatrix} u \\ v \end{bmatrix} \tag{6.12}$$

and

$$\tau = \mu \begin{bmatrix} 2\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 v}{\partial x \partial y} \\ \frac{\partial^2 u}{\partial x \partial y} + \frac{\partial^2 v}{\partial x^2} + 2\frac{\partial^2 v}{\partial y^2} \end{bmatrix}. \tag{6.13}$$

Note that by (6.9),

$$\tau = \mu \begin{bmatrix} \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial}{\partial x}\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) \\ \frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} + \frac{\partial}{\partial y}\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) \end{bmatrix} = \mu \begin{bmatrix} \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \\ \frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \end{bmatrix} = \mu \nabla^2 \mathbf{v}.$$

6.3.3 Node Equations

As this derivation is lengthy, the full derivation has been placed in Appendix D. After some length derivation, we can approximate the system on some domain Ω given by

$$\begin{aligned} \frac{1}{\rho}\frac{\partial p}{\partial x} - \nu\frac{\partial^2 u}{\partial x^2} - \nu\frac{\partial^2 u}{\partial y^2} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} &= -\frac{\partial u}{\partial t} \\ \frac{1}{\rho}\frac{\partial p}{\partial y} - \nu\frac{\partial^2 v}{\partial x^2} - \nu\frac{\partial^2 v}{\partial y^2} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} &= -\frac{\partial v}{\partial t} \end{aligned} \tag{6.14}$$

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

Using the finite-difference method in Section 6.2.3 to get, as long as $\mathbf{x}_{i,j} \notin \partial\Omega$,

$$P_0 p_{i,j} + P_1 p_{i+1,j} + P_2 p_{i,j+1} + P_3 p_{i-1,j} + P_4 p_{i,j-1} + P_5 p_{i+r_1,j+r_2} \tag{6.15}$$
$$+Q_0 u_{i,j} + Q_1 u_{i+1,j} + Q_2 u_{i,j+1} + Q_3 u_{i-1,j} + Q_4 u_{i,j-1} + Q_5 u_{i+r_1,j+r_2} = -\frac{1}{\delta t} u_p$$

$$R_0 p_{i,j} + R_1 p_{i+1,j} + R_2 p_{i,j+1} + R_3 p_{i-1,j} + R_4 p_{i,j-1} + R_5 p_{i+r_1,j+r_2} \tag{6.16}$$
$$+S_0 v_{i,j} + S_1 v_{i+1,j} + S_2 v_{i,j+1} + S_3 v_{i-1,j} + S_4 v_{i,j-1} + S_5 v_{i+r_1,j+r_2} = -\frac{1}{\delta t} v_p$$

$$T_0 u_{i,j} + T_1 u_{i+1,j} + T_2 u_{i,j+1} + T_3 u_{i-1,j} + T_4 u_{i,j-1} + T_5 u_{i+r_1,j+r_2}$$
$$+W_0 v_{i,j} + W_1 v_{i+1,j} + W_2 v_{i,j+1} + W_3 v_{i-1,j} + W_4 v_{i,j-1} + W_5 v_{i+r_1,j+r_2} = 0 \tag{6.17}$$

where all coefficients are given by the following:

$$P_0 = \left(-\frac{1}{\rho} \sum_k b_{1,k} \right) \quad P_1 = \frac{b_{1,1}}{\rho} \quad P_2 = \frac{b_{1,2}}{\rho} \quad P_3 = \frac{b_{1,3}}{\rho} \quad P_4 = \frac{b_{1,4}}{\rho} \quad P_5 = \frac{b_{1,5}}{\rho}$$

$$Q_0 = \left(-u^* \sum_k b_{1,k} - v^* \sum_k b_{2,k} + \nu \sum_k b_{3,k} + \nu \sum_k b_{4,k} - \frac{1}{\delta t} \right) \quad Q_1 = u^* b_{1,1} + v^* b_{2,1} - \nu b_{3,1} - \nu b_{4,1}$$
$$Q_2 = u^* b_{1,2} + v^* b_{2,2} - \nu b_{3,2} - \nu b_{4,2} \quad Q_3 = u^* b_{1,3} + v^* b_{2,3} - \nu b_{3,3} - \nu b_{4,3} \quad Q_4 = u^* b_{1,4} + v^* b_{2,4} - \nu b_{3,4} - \nu b_{4,4}$$
$$Q_5 = u^* b_{1,5} + v^* b_{2,5} - \nu b_{3,5} - \nu b_{4,5}$$

$$R_0 = \left(-\frac{1}{\rho} \sum_k b_{2,k} \right) \quad R_1 = \frac{b_{2,1}}{\rho} \quad R_2 = \frac{b_{2,2}}{\rho} \quad R_3 = \frac{b_{2,3}}{\rho} \quad R_4 = \frac{b_{2,4}}{\rho} \quad R_5 = \frac{b_{2,5}}{\rho}$$

$$S_0 = \left(-u^* \sum_k b_{1,k} - v^* \sum_k b_{2,k} + \nu \sum_k b_{3,k} + \nu \sum_k b_{4,k} - \frac{1}{\delta t} \right) \quad S_1 = u^* b_{1,1} + v^* b_{2,1} - \nu b_{3,1} - \nu b_{4,1}$$
$$S_2 = u^* b_{1,2} + v^* b_{2,2} - \nu b_{3,2} - \nu b_{4,2} \quad S_3 = u^* b_{1,3} + v^* b_{2,3} - \nu b_{3,3} - \nu b_{4,3} \quad S_4 = u^* b_{1,4} + v^* b_{2,4} - \nu b_{3,4} - \nu b_{4,4}$$
$$S_5 = u^* b_{1,5} + v^* b_{2,5} - \nu b_{3,5} - \nu b_{4,5}$$

$$T_0 = \sum_k b_{1,k} \quad T_1 = b_{1,1} \quad T_2 = b_{1,2} \quad T_3 = b_{1,3} \quad T_4 = b_{1,4} \quad T_5 = b_{1,5}$$

$$W_0 = \sum_k b_{2,k} \quad W_1 = b_{2,1} \quad W_2 = b_{2,2} \quad W_3 = b_{2,3} \quad W_4 = b_{2,4} \quad W_5 = b_{2,5}.$$

Note that u^* and v^* represent estimates for u and v, respectively, and that u_p and v_p represent the respective estimates for u and v in the previous time frame.

6.3.4 Solution Procedure

It is important to apply an appropriate solution procedure. As seen in Figure 3.5, it is typical to formulate a flowchart-style procedure. The procedure applied in this solver is depicted in Figure 6.3.

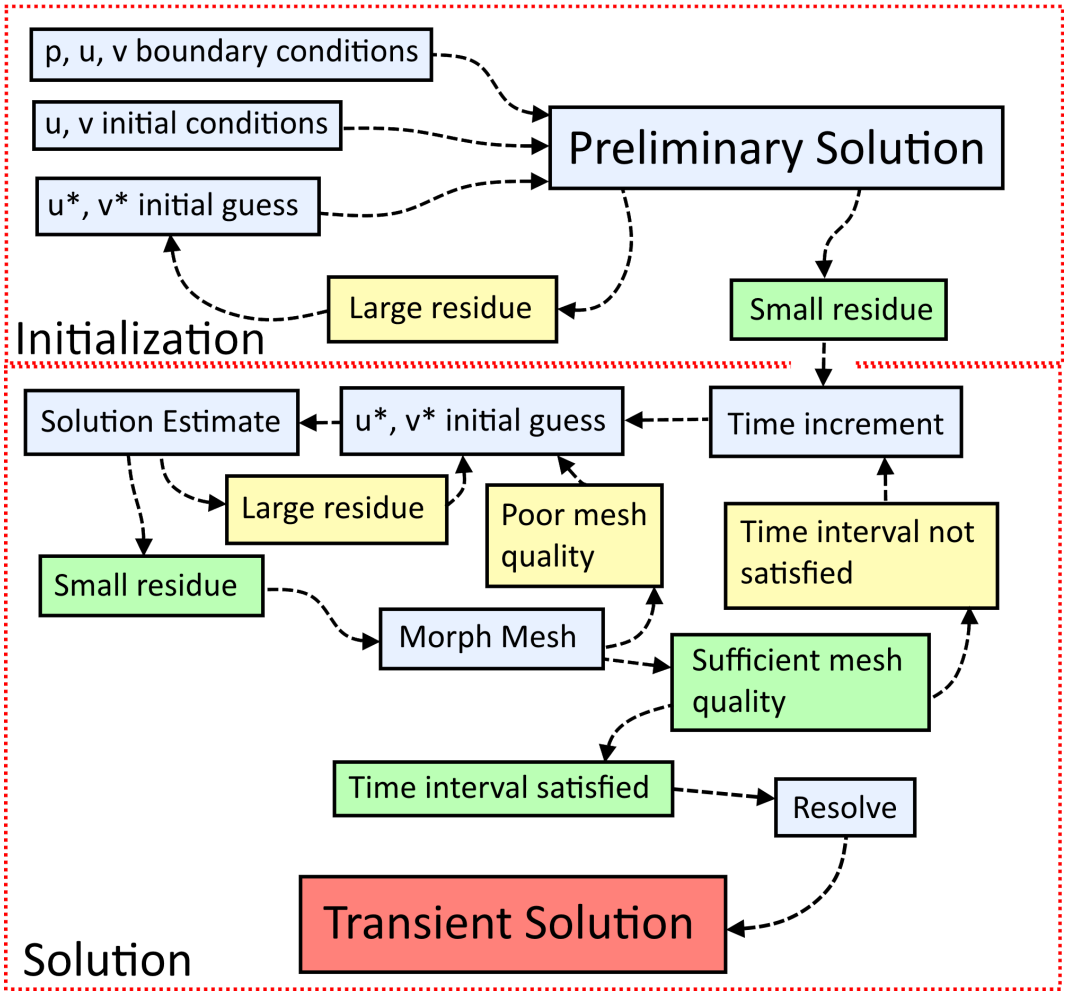


Figure 6.3: Solution procedure.

6.3.5 Kaczmarz's Algorithm

Applying equations (6.15), (6.16), and (6.17) to every node of an $M \times N$ grid results in a typical linear system given, in general form, by

$$A\mathbf{x} = \mathbf{b},$$

where

$$\mathbf{x} = \begin{bmatrix} u_{1,1} & v_{1,1} & p_{1,1} & u_{2,1} & v_{2,1} & p_{2,1} & u_{3,1} & v_{3,1} & p_{3,1} & \dots \end{bmatrix}^T.$$

An issue arises when we notice that $\dim(\mathbf{x}) = 3(M-2)(N-2)$. Most of the test cases were run with $N = M = 20$ or $M = N = 30$, so the system dimension ranged between 972×972 and 2352×2352. These systems proved far too large to solve via direct methods.

As such, an iterative scheme was applied, namely, a slightly modified version of Kaczmarz's algorithm (see [8]). The algorithm for an $N \times N$ system is as follows.

Let

$$A = \begin{bmatrix} \mathbf{a}_1 & \mathbf{a}_2 & \dots & \mathbf{a}_N \end{bmatrix}^T$$

and let $\mathbf{x}_0$ be an initial guess given by

$$\mathbf{x_0} = \begin{bmatrix} 1 & 1 & \dots & 1 \end{bmatrix}^T \quad \text{if on first iteration}$$

or

$$\mathbf{x_0} = \mathbf{x}_p \quad \text{otherwise}$$

where $\mathbf{x}_p$ is the previous solution estimate. Using some initial estimate $\mathbf{x}_k$, we obtain the next estimate as follows:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \frac{P_{r,i}(\mathbf{x}_k, \mathbf{a}_{i+1}) - \mathbf{b}_{i+1}}{|\mathbf{a}_{i+1}|^2}\mathbf{a}_{i+1} \quad \text{for } i \equiv k+1 \mod N. \tag{6.18}$$

Here, $P_{r,k}$ denotes a modified euclidean inner product given by

$$P_{r,k}(\mathbf{x}, \mathbf{y}) = \sum_{i=m}^{M} \mathbf{x}_i \mathbf{y}_i$$

where

$$m = \max(1, k-r) \qquad \text{and} \qquad M = \min(N, k+r).$$

The convergence for this algorithm can be proven to be exponential, but that will not be explored here. The convergence of this algorithm is visualized in Figures C.1 and C.2.

7 Results

7.1 Results - Mesh-Morphing

This scheme was allowed to run using two iterations of this method with a step size $s = \frac{0.06}{2}$. The known solution u for all test runs solved the wave equation:

$$\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial x^2}.$$

Figures A.1 and A.2 in Section A show an example grid under the transformation Z. This grid was well formed and had desirable qualities of a high-quality mesh.

Figure A.3 shows another instance of this method applied to a different solution (also to the wave equation). The transformed grid, in this case, exhibited both desirable and undesirable features of mesh generation. These features are shown in Figures A.5 and A.4.

7.2 Results - Generalized Finite-Difference

Note: the results presented in this section primarily involve variants of Laplace's equation, primarily due to to the fact that fast solution methods could be applied, and due to the fact that analytic solutions are readily available.

7.2.1 Solver results on orthogonal grid

For the purpose of preliminary solution validation, the solver was configured to solve Laplace's equation (7.1) using a variety of boundary conditions. The first test case run involved a variation of temperature distribution problem (see Figure 7.1) using a sinusoidal boundary condition, with $\Delta T = 5$, and $H = L = 10$.

$$\nabla^2 u = 0 \tag{7.1}$$

The analytic solution to this problem is given by

$$T(\mathbf{x}) = \frac{\sinh(\frac{\pi y}{L})}{\sinh(\frac{\pi H}{L})} \sin(\frac{\pi x}{L}).$$

A plot of the analytic solution can be found in Figure B.1, and the approximation given by the method can be found in Figure B.2.

The error for this approximation was also plotted, and can be found in Figure B.3. Note that the error is greatest near the center of the domain.

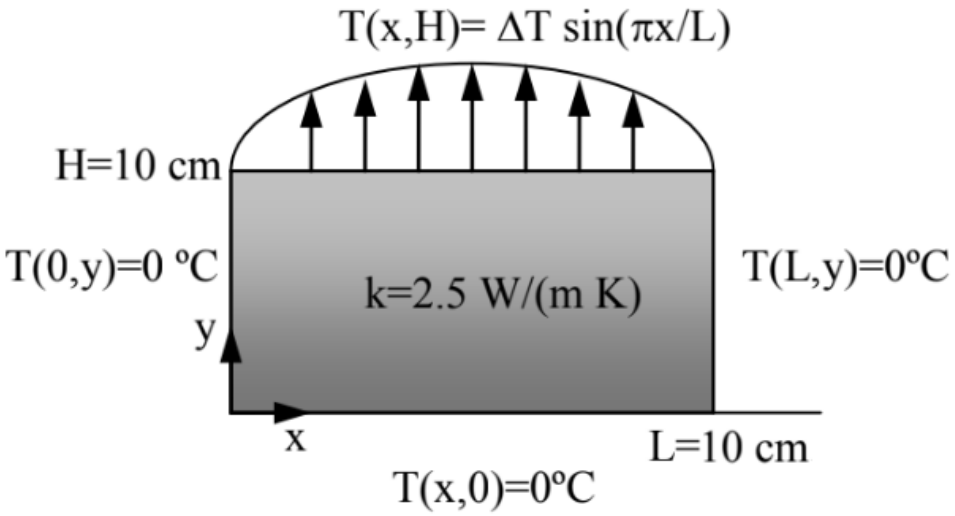


Figure 7.1: Boundary conditions for test case.

7.2.2 Solver results on non-orthogonal grid

In order to validate the quality of solutions on non-orthogonal grids, some test cases were run for solutions with different test cases. One is presented here. This test case was a solution to Laplace's equation with logistic boundary conditions. The fact that all approximate solutions were consistent with each other indicates that the mesh-convergence values of each node are invariant to the mesh.

Figure B.4 depicts four similar solutions on four different grids. Note the consistency of the solutions.

The grids in Figure B.4 were randomly generated, each with different allowable cell sizes, with (b) having the smallest sizes, and (d) having the largest.

7.2.3 Validation of gradient-ascent mesh-morphing

Initially, gradient-ascent mesh-morphing was applied iteratively to the solution to the problem posed in Figure 7.1. The quality of the solution was monitored by evaluating the norm of the difference (see section 6.2.4) between the numerical and analytic solutions. The iterative process was run 10 times. The error plot can be found in Figure B.5.

A second test case was run for boundary conditions that were highly periodic. The problem for this case was given as follows, with $\omega = 4$ an $H = L = 10$:

$$\frac{1}{H^2}\frac{\partial^2 u}{\partial x^2} + \left(\frac{\pi\omega}{L}\right)^2 \frac{\partial^2 u}{\partial y^2} = 0 \qquad \text{when} \qquad \mathbf{x} \in \Omega$$

and

$$u(0,y) = u(L,y) = 0 \qquad \text{and} \qquad u(x,0) = \sin(\frac{\pi\omega x}{L}) \qquad \text{and} \qquad u(x,H) = e \cdot \sin(\frac{\pi\omega x}{L}).$$

The analytic solution to this problem is given by

$$u(\mathbf{x}) = \sin(\frac{\pi\omega x}{L})e^{\frac{y}{H}}.$$

After a few iterations, the results of the solver output were as depicted in Figure B.6.

The tracked error for this iterative scheme can be found in Figure B.7.

7.3 Results - Navier-Stokes

Two distinct types of test cases were investigated. The two types were classified as

- open solutions, where
$$w = \oint_{\partial\Omega} |\mathbf{v} \cdot d\mathbf{n}| > 0 \quad \text{(net inflow), and}$$
- closed solutions, where
$$w = \oint_{\partial\Omega} |\mathbf{v} \cdot d\mathbf{n}| = 0 \quad \text{(no net inflow).}$$

Note that w is a parameter that can be calculated a priori, as no knowledge of the solution need be attained.

Two test cases are presented here: the first can be found in Figure 7.2, and the second in Figure 7.4

7.3.1 Results - $w > 0$

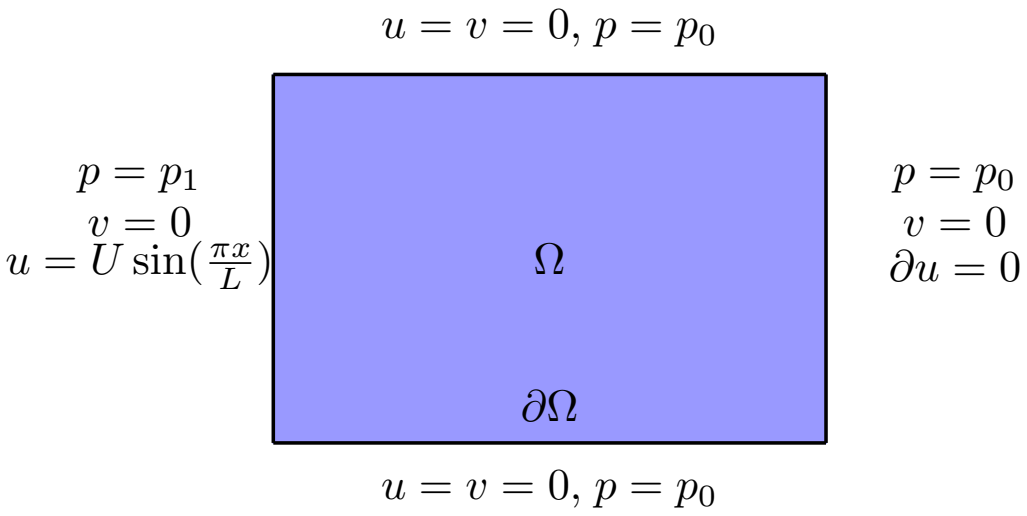


Figure 7.2: A test case with $w > 0$.

The test case found in Figure 7.2 was run, and it was discovered that the use of pressure boundary conditions did not produce a coherent transient solution. An example output of the transient solution under these boundary conditions can be found in Figure 7.3. Unfortunately, no known results were available to check the solution against. However, it was determined that this solution was inaccurate, as the transient solution exhibited instability.

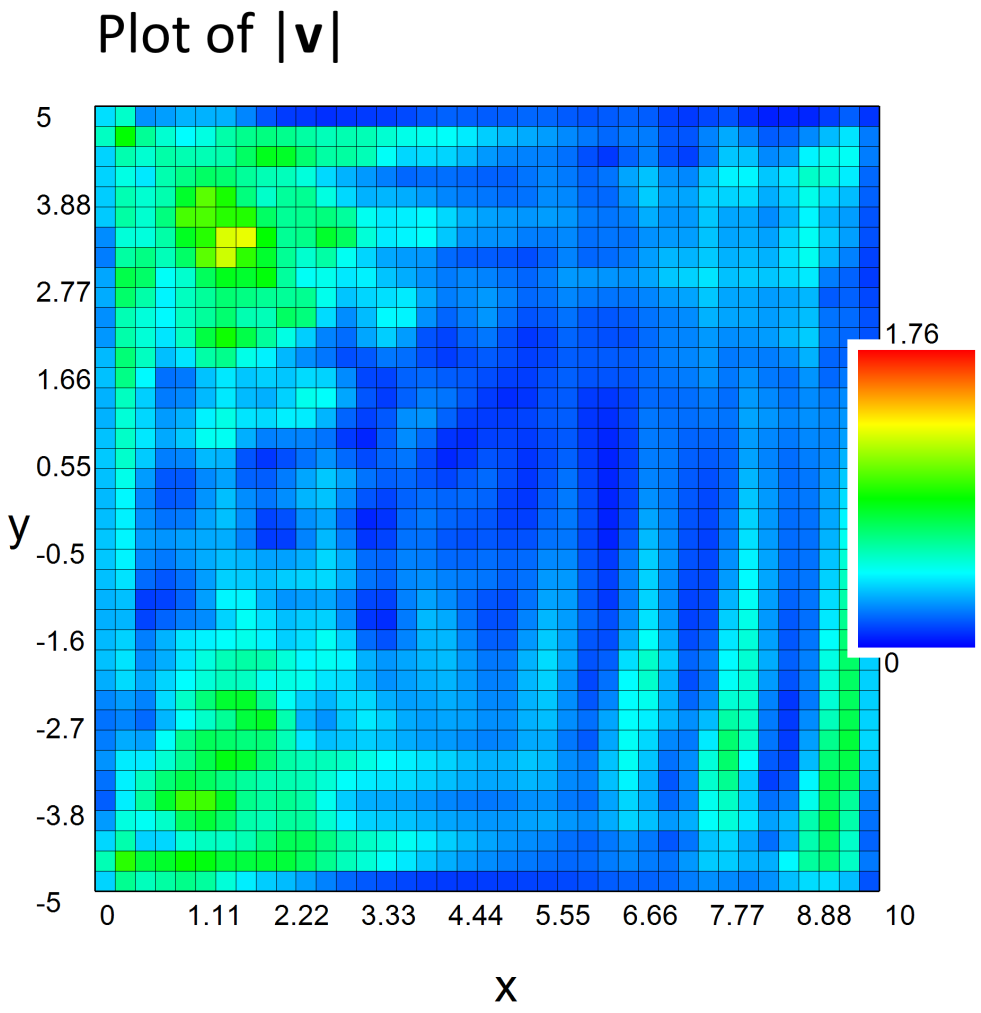


Figure 7.3: An example output from the boundary conditions in Figure 7.2. $|\mathbf{v}|$ is plotted.

7.3.2 Results - $w = 0$

The test case found in Figure 7.4 was investigated, and compared with available results.

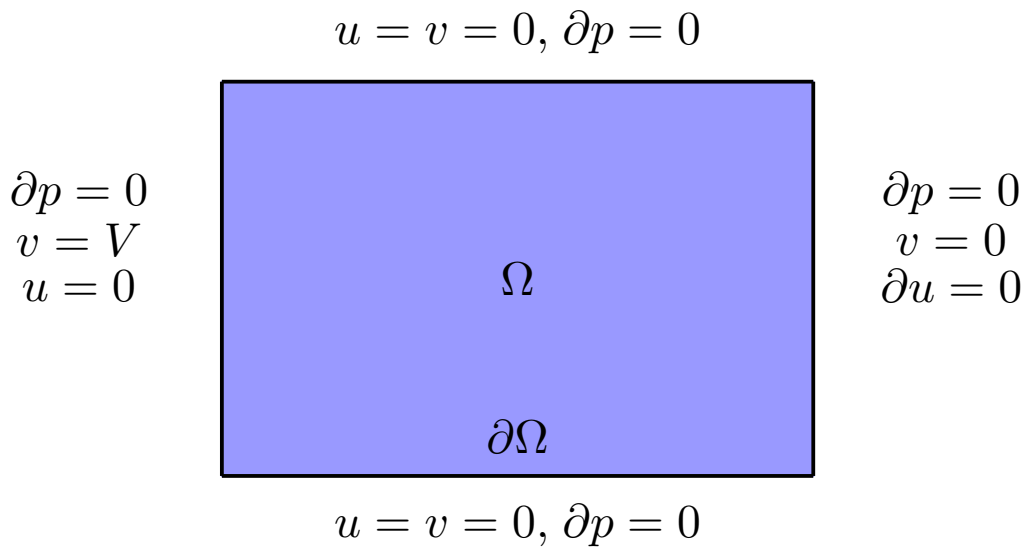


Figure 7.4: A test case with $w = 0$.

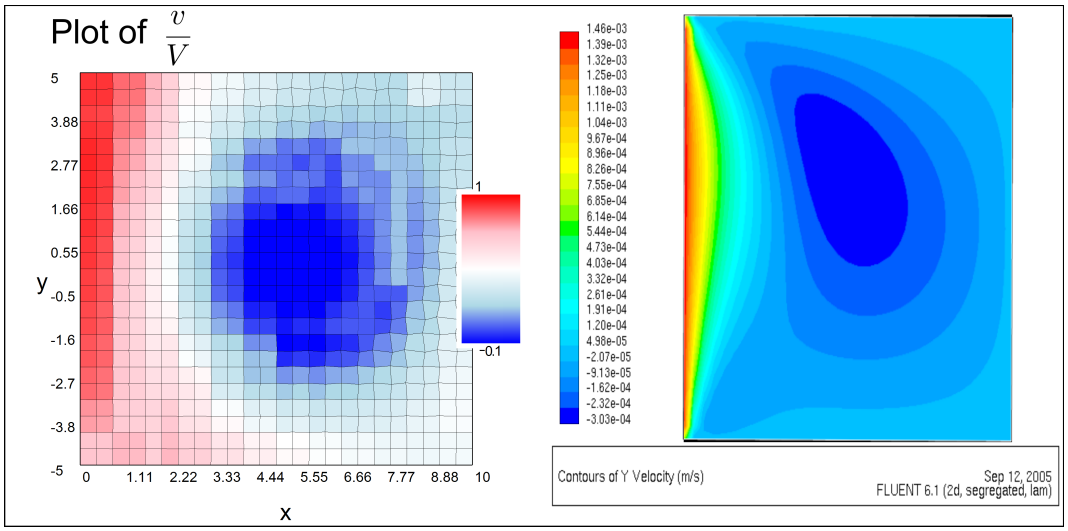


Figure 7.5: Comparison of results for a lid-driven flow problem: the solver created for this project (left) and a similar problem solved using ANSYS Fluent (right). Solution estimated at $Re_L \approx 175 < 12000$, an approximate laminar-turbulent threshold suggested by [10]. Image courtesy of CFD-online.

8 Discussion

8.1 Discussion - Mesh-Morphing

8.1.1 Discussion of results

As seen in Section A, this method, as it was implemented on this framework, is not perfect. Observation of Figure A.1 will show that the mesh transformation did not have a profoundly positive effect on the mesh, though an improvement was achieved. However, the intent of this method is to modify the analysis of solutions that have large gradients throughout Ω, and this is not a property of the solution seen in Figure A.1. Therefore, this may be considered a case where the mesh-morphing algorithm is superfluous.

However, undesirable mesh qualities notwithstanding, Figure A.3 shows that for solutions with a larger number of features in a given domain, this method has the potential to generate much higher-quality meshes than simple Cartesian meshes. Furthermore, the nature of the poor quality mesh features tend to suggest that the modified step size s^* from Section 6.1.2 might yield better results after having some more optimization or limiting codes applied to it.

Overall, the results of this first implementation of this method show that with some small modifications, high-quality meshes can be generated automatically using this scheme.

8.1.2 Limitations of this method

This method assumes that for any PDE that must be solved on these meshes, there must exist a smooth solution given smooth enough boundary conditions. This may not necessarily be the case. Specifically, consider the case of the Navier-Stokes equation. Unfortunately, it is still a matter of conjecture whether or not this system of equations has a smooth solution for sufficiently smooth boundary conditions[5].

The existence and smoothness of solutions, however, is not an assumption that is unique to this method. Countless other methods for solving various PDE systems have been built on the foundation of this assumption.

8.1.3 A note on degenerate meshes

The degenerate mesh feature seen in Figure A.4 has been identified as such because there are cells (systems of four adjacent nodes) that overlap regions of the domain. This is a problem if one is interested in calculating quantities that involve performing integration over the cells. Degenerate features like this can be fixed by reassigning the coordinates of nearby nodes to prevent cell overlap from happening.

8.1.4 A note on empiricism in defining mesh quality

The original intent of this project was to quantify the quality of a mesh by defining a quality function. The purpose of this function would be to serve as a datum by which transformed meshes could be compared to the untransformed meshes from which they are calculated. However, the problem with this approach is twofold:

1. This method builds itself from optimizing $|\nabla u|$ for each node. If a quality function were to be defined that favored nodes being in locations where $|\nabla u|$ is large, then the optimal value for this function would occur when all nodes are at the point $(x, y) \in \Omega$ where $|\nabla u|$ is maximized. This would not yield a desirable mesh.

2. Since this method is based on gradient-ascent, there is presumably no finite number of times one can apply the transformation Z to a mesh to attain a limiting value of the quality function. That is, the quality function would (assuming convergence of the node transformation) continuously increase, or, at the very least, change. This would imply that it would be necessary to implement a limit on the number of times one could apply Z, thereby making the quality function obsolete.

[5]This problem is so popular that it is actually one of the Clay Mathematics Institute's Millennium Prize Problems. The monetary prize for proving that the Navier-Stokes equations have a smooth solution for sufficiently smooth boundary conditions is currently 1,000,000 USD.

The problems that arise when attempting to properly define a quality function like this one tend to suggest that a degree of empiricism is necessary to evaluate the quality of meshes, and that limiting codes will be more worthwhile to develop than rigorously formulating a function with the aforementioned qualities.

8.2 Discussion - Generalized Finite Difference

8.2.1 General discussion

It is clear from section 7.2.3 that the gradient-ascent mesh-morphing algorithm may not necessarily optimize solution accuracy for every problem. It is suspected that since the problem considered in this section has highly periodic boundary conditions, this particular algorithm is detrimental to solution accuracy (see Section 8.2.2 for further discussion). By contrast, the results presented for the first problem posed in section 7.2.3 suggest that for problems that have solutions with single, large-gradient features (resembling a wave-front), the algorithm improves solution accuracy.

The results presented in Sections 7.2.1 and 7.2.2 suggest that this method of estimating the second-order derivatives is suitable for solving BVPs on a non-orthogonal domain. These results also suggest that this estimation might be suitable for application to problems with unstructured grids, though this implementation would require a significant amount of adaptation.

8.2.2 A note on evaluation of solution quality

Although it is natural to precisely quantify error as in Section 6.2.4, caution should be employed when using this definition as a metric by which to judge the quality of a solution.

As an example, suppose we wish to model a combustion wavefront. We might apply the gradient-ascent mesh-morphing algorithm to a known, analytical solution to find that the error increases upon each iteration. We may observe the nature of the estimated solution to find that the inaccuracies lie primarily in large regions of low variation, and choose to keep this approximation. This may be an example of a solution that is not *accurate*, but is *resolute*.

By contrast, it might be the case that, on a certain grid, the approximation of the solution is highly accurate, but the majority of nodes lie in regions of low interest. This might be an example of a solution that is *accurate*, but not *resolute*.

Both of these qualities are important in worthwhile approximations, so evaluating the quality of solutions must be approached carefully.

8.3 Discussion - Navier-Stokes System

8.3.1 Discussion of Results

The issues exhibited in Figure 7.3 tend to suggest that the use of the non-standard boundary conditions on the pressure distribution p causes problems. As suggested in [9], physically interpreting boundary conditions on p is complicated, and should be avoided. The results in Figure 7.3 confirm this, at least for $w > 0$. Recall that

$$w = \oint_{\partial\Omega} |\mathbf{v} \cdot d\mathbf{n}|.$$

By contrast, the results seen in Figure 7.5 tend to suggest that as long as $w = 0$, the use of boundary conditions on p produce closer results that the $w > 0$ case. Results could possibly be improved by placing a tighter restriction on the convergence of (6.18). Results may also be significantly improved by implementing the vorticity-stream function approach given in [11], (3.2), and (3.3).

8.3.2 Discussion of Simplifying Assumptions

It is important to physically interpret the following assumptions to check their validity:

- **Isothermal Flow:** Isothermal flows are of a uniform temperature. As long as flows are not driven by density gradients (which are, in turn, driven by temperature differences and gravitational forces), then it can be assumed that the flow is isothermal.
- **Constant Thermal Conductivity:** A typical fluid's thermal conductivity will change by a margin of less than 9% over a temperature difference of a few hundred degrees. This change was neglected, and this assumption is consistent with the isothermal flow assumption.
- **Adiabatic Flow:** Heat transfer in a fluid is driven by 3 major phenomena: conduction, transport, and viscous dissipation. Both conduction and transport can be neglected in an isothermal flow, since temperature difference drives conduction, and transport is flow-driven, but the flow exists at one temperature. Viscous dissipation terms are considered to be negligible at low Reynolds numbers (at relatively low velocities).
- **Incompressible Flow:** It is common practice to apply this assumption when the velocities in the flow are slower than 30% of the speed of sound in that fluid.

9 Conclusion

Although some novel techniques were applied in the construction of this numerical solver, it should be clear that not all choices made were ideal. The mesh-morphing scheme presented here would benefit from some further investigation, perhaps by making a better choice of the step size at each node. Although solution quality did benefit from this algorithm in some cases, it proved detrimental in other cases. Investigation into what class of problems benefit from the algorithm might yield interesting results. The application of the generalized finite difference method proved to be a successful way of solving general BVPs. The problem solving scheme in Figure 6.3 also proved moderately successful, as seen in Figure 7.5. Although the solutions obtained using this process were somewhat noisy, these solutions were obtained using parallel computing on four CPU cores. Higher-quality solutions may be obtained using more robust hardware.

10 Future Work

Future modifications of this solver include the following:

1. **Application to Unstructured Grids:** The finite-difference method explored in this document need nod be constrained to Cartesian grids. Unstructured solvers are far more useful in applications than structured solvers, so this utility may motivate the application of these concepts to a new solver in the future.
2. **Vorticity-Stream Function Approach:** The solver would benefit from the implementation of third-order derivative estimates. This would allow for an efficient linearization of the transformations given in (3.2) and (3.3). It may also be possible to estimate solutions under the vorticity-stream function approach using only second-order approximations.
3. **Expediting GPU Hardware:** Although this solver was constructed to do all computations in parallel on 4 CPU cores, it would speed computations up by many orders of magnitude if computations were carried out on available GPU hardware. Also, further refinements may be made to the parallelism used in this solver.

11 Acknowledgments

Special thanks is extended to the following individuals:

- **Dr. Patrick Shipman:** Dr. Shipman's advice on this project has been essential to its completion.
- **Dr. Iuliana Oprea:** Dr. Oprea's advice on boundary conditions and flow function transformations was critical in understanding the quality of the solver results.

- **Sean Walters:** Mr. Walters and his contribution on assumptions made in the Navier-Stokes Model were critical in making this problem more manageable.

References

[1] R. Cummings, W. Mason, S. Morton, and D. McDaniel. *Applied Computational Aerodynamics.* Cambridge University Press (2015).

[2] A. Chorin. *Numerical Solution of the Navier-Stokes Equations.* Courant Institute of Mathematical Sciences, NewYork University (1968).

[3] J. Bell, M. Berger, J. Saltzman, and M. Welcome. *Three-Dimensional Adaptive Mesh Refinement for Hyperbolic Conservation Laws.* SIAM Journal of Scientific Computing, Vol. 15, 127-138 (1994).

[4] M.J. Berger and P. Colella. *Local Adaptive Mesh Refinement for Shock Hydrodynamics.* Journal of Computational Physics 82, 64-84 (1989).

[5] J.J. Benito, F. Urena, and L. Gavete. *Solving Parabolic and Hyperbolic Equations by the Generalized Finite Difference Method.* Journal of Computational and Applied Mathematics 209, 208-233 (2007).

[6] G.B.Folland. *Higher-Order Derivatives and Taylor's Formula in Several Variables.* University of Washington Department of Mathematics, Accessed 2018.

[7] K. Jansen. *Unstructured-grid large-eddy simulation of flow over an airfoil.* Center for Turbulence Research (1994).

[8] J. Liu, S. Wright, and S. Sridhar. *An Asynchronous Parallel Randomized Kaczmarz Algorithm.* University of Wisconsin-Madison, Department of Computer Science (2014).

[9] B. Seibold. *A Compact and Fast Matlab Code Solving the Incompressible Navier-Stokes Equations on Rectangular Domains.* Massachusetts Institute of Technology (2008).

[10] R. Bouffanis et. al. *Large-Eddy Simulation of the Flow in a Lid-Driven Cavity.* Laboratoire de Mathematiques de lUniversite de Saint-Etienne (2008).

[11] M. Matkya. *Solution to Two-Dimensional Incompressible Navier-Stokes Equations with SIMPLE, SIMPLER and Vorticity-Stream Function Approaches. Driven-Lid Cavity Problem: Solution and Visualization.* University of Wroclaw (2004).

[12] G. Recktenwald. *The Control-Volume Finite Difference Approximation to the Diffusion Equation.* (2014). Retrieved from

`http://web.cecs.pdx.edu/~gerry/class/ME448/notes_2012/pdf/CVFDdiffusion2D.pdf`

A Appendix - Mesh-Morphing Figures

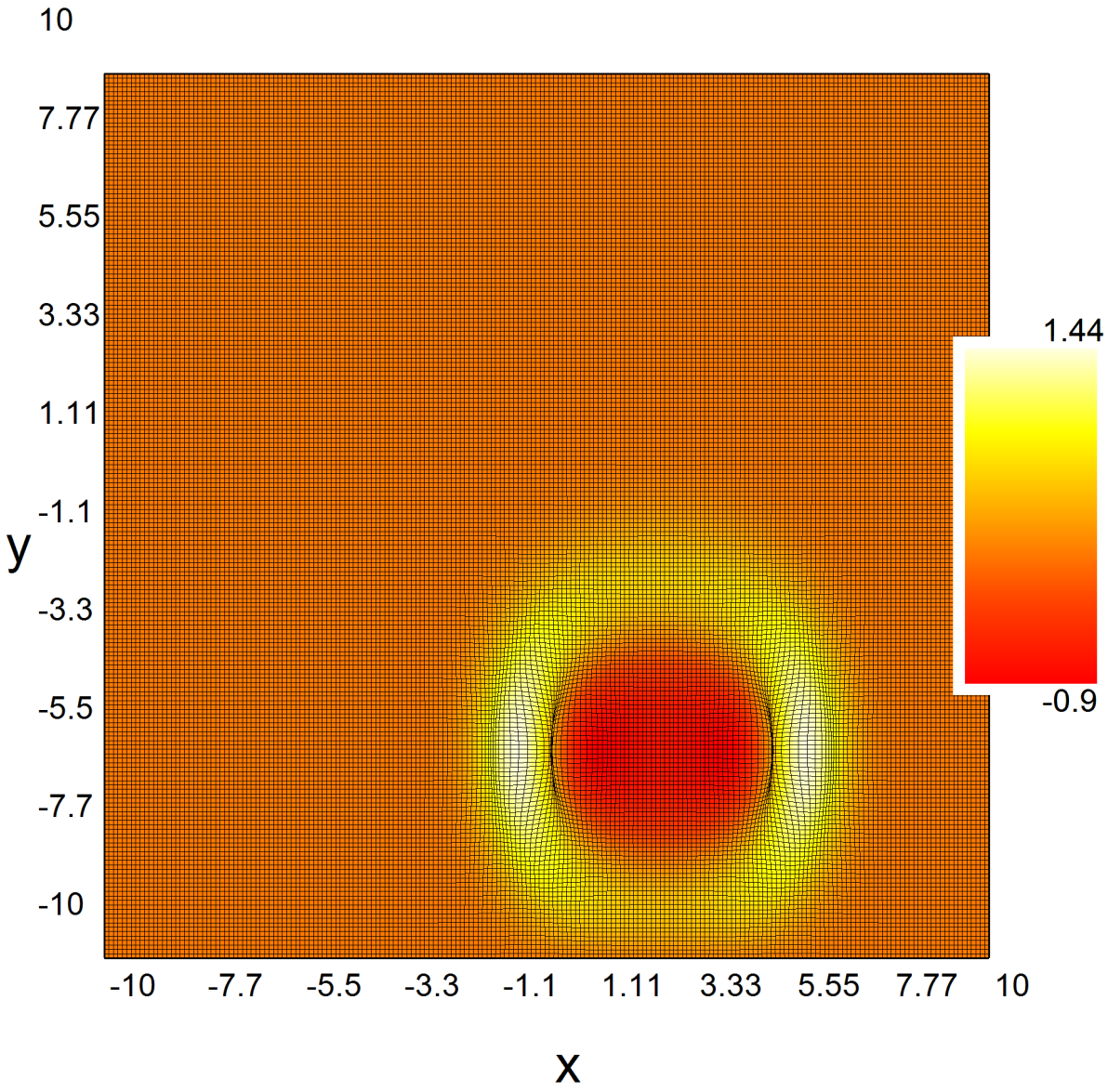


Figure A.1: A solution u with transformed grid superimposed.

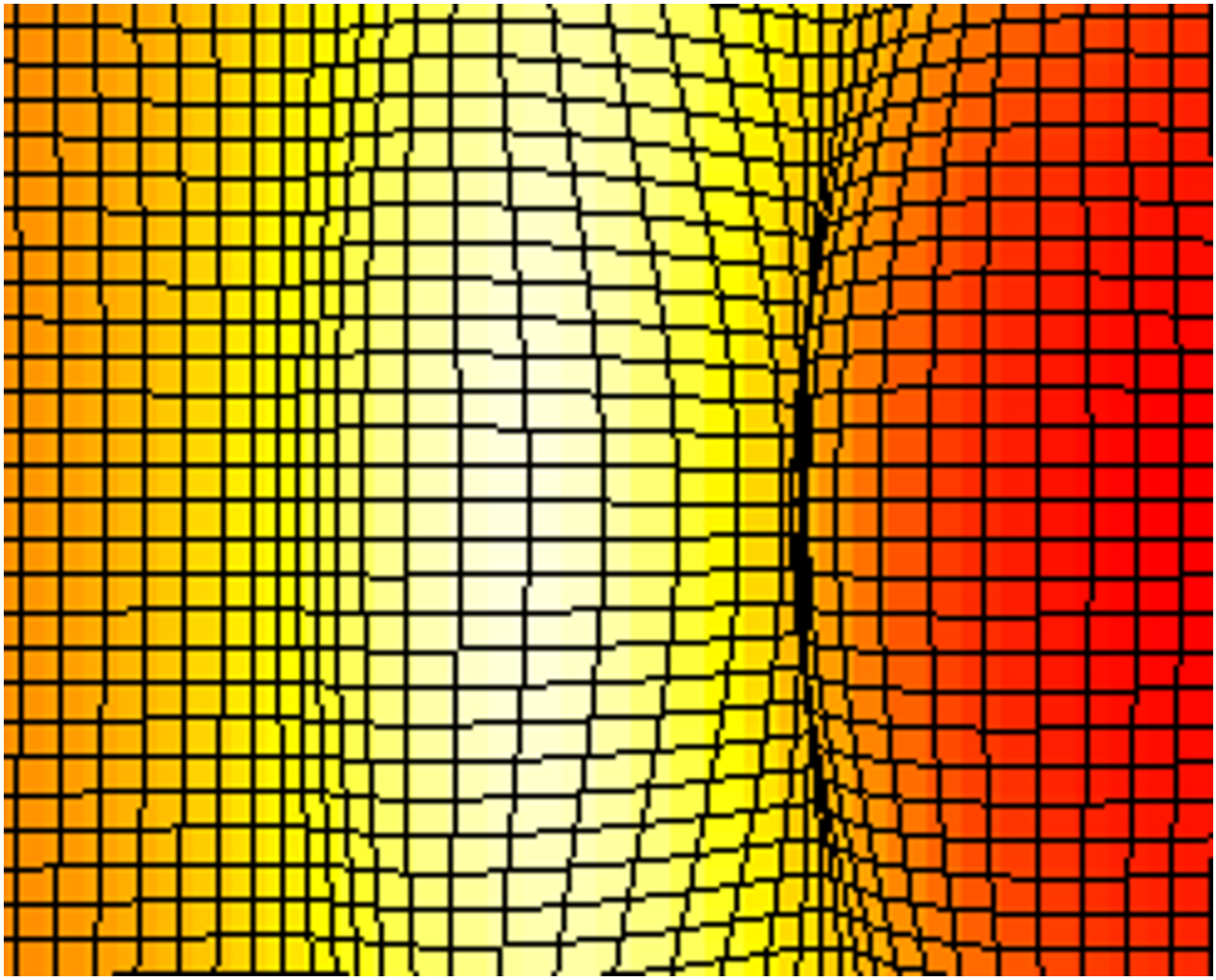


Figure A.2: A close view of the transformed discretization after 2 iterations of the transformation Z at $(x, y) \approx (-1.15, -6.5)$.

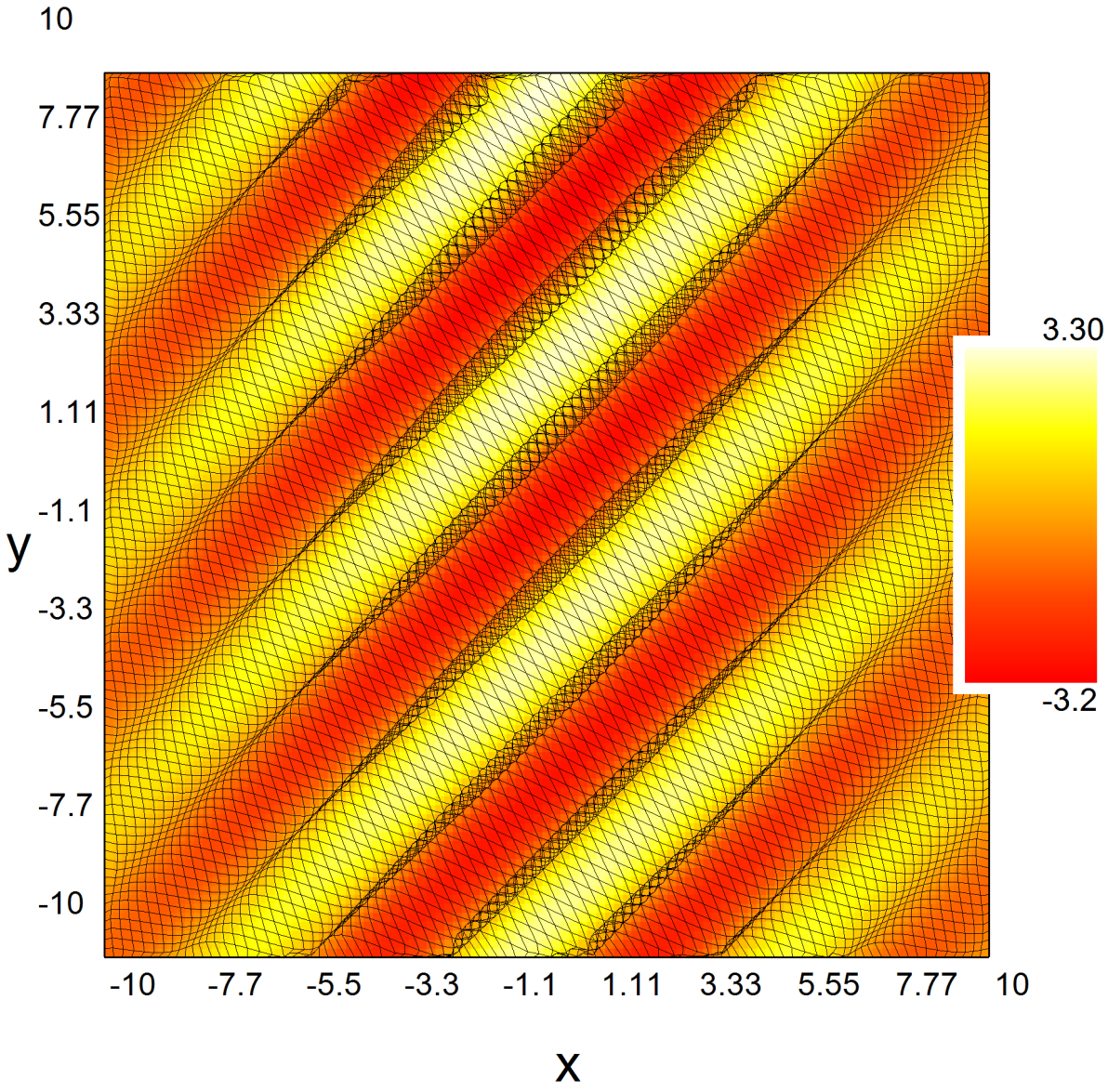


Figure A.3: A solution u with transformed grid superimposed.

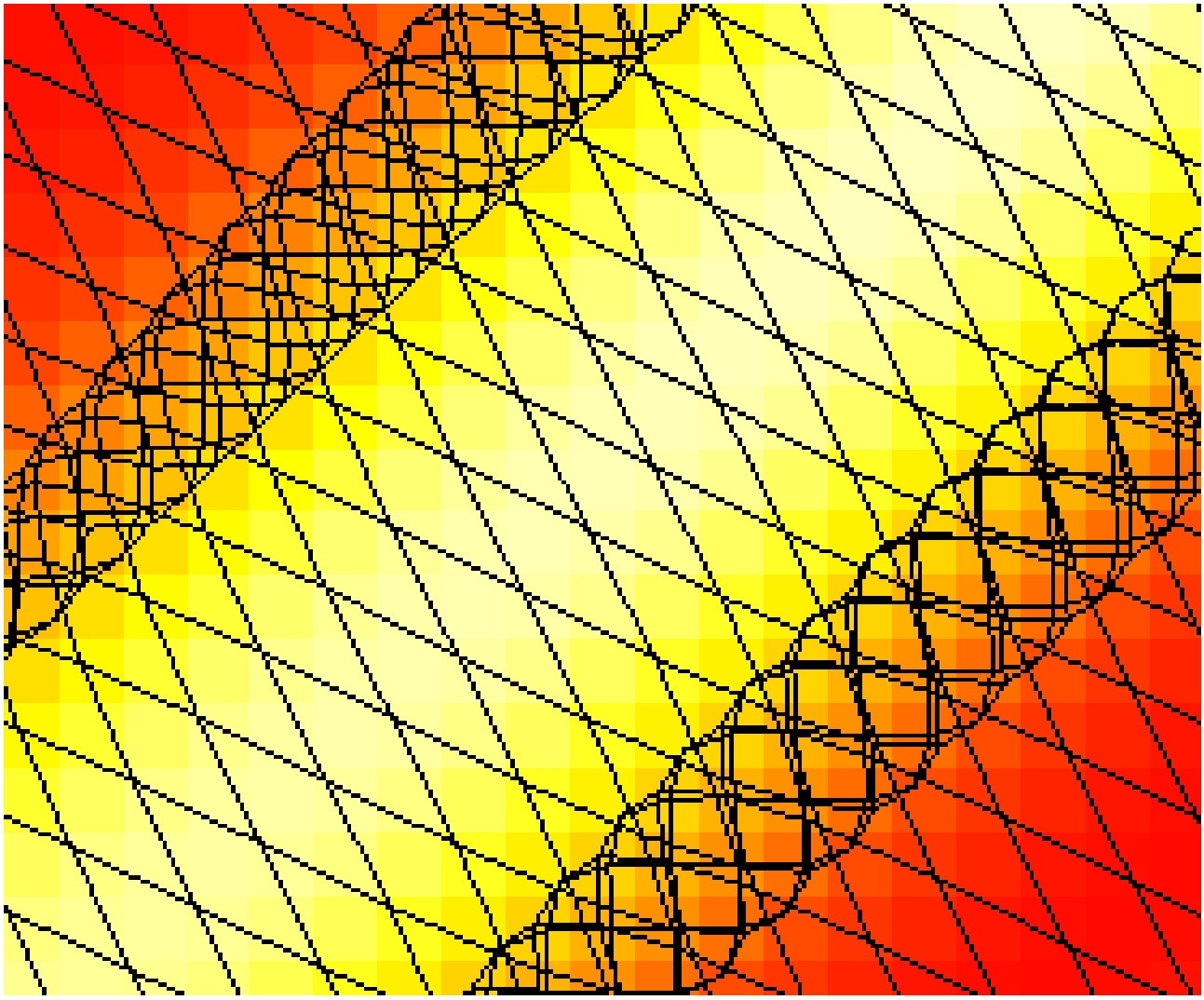


Figure A.4: A close view of an undesirable mesh quality, $(x, y) \approx (-1.15, 1.05)$.

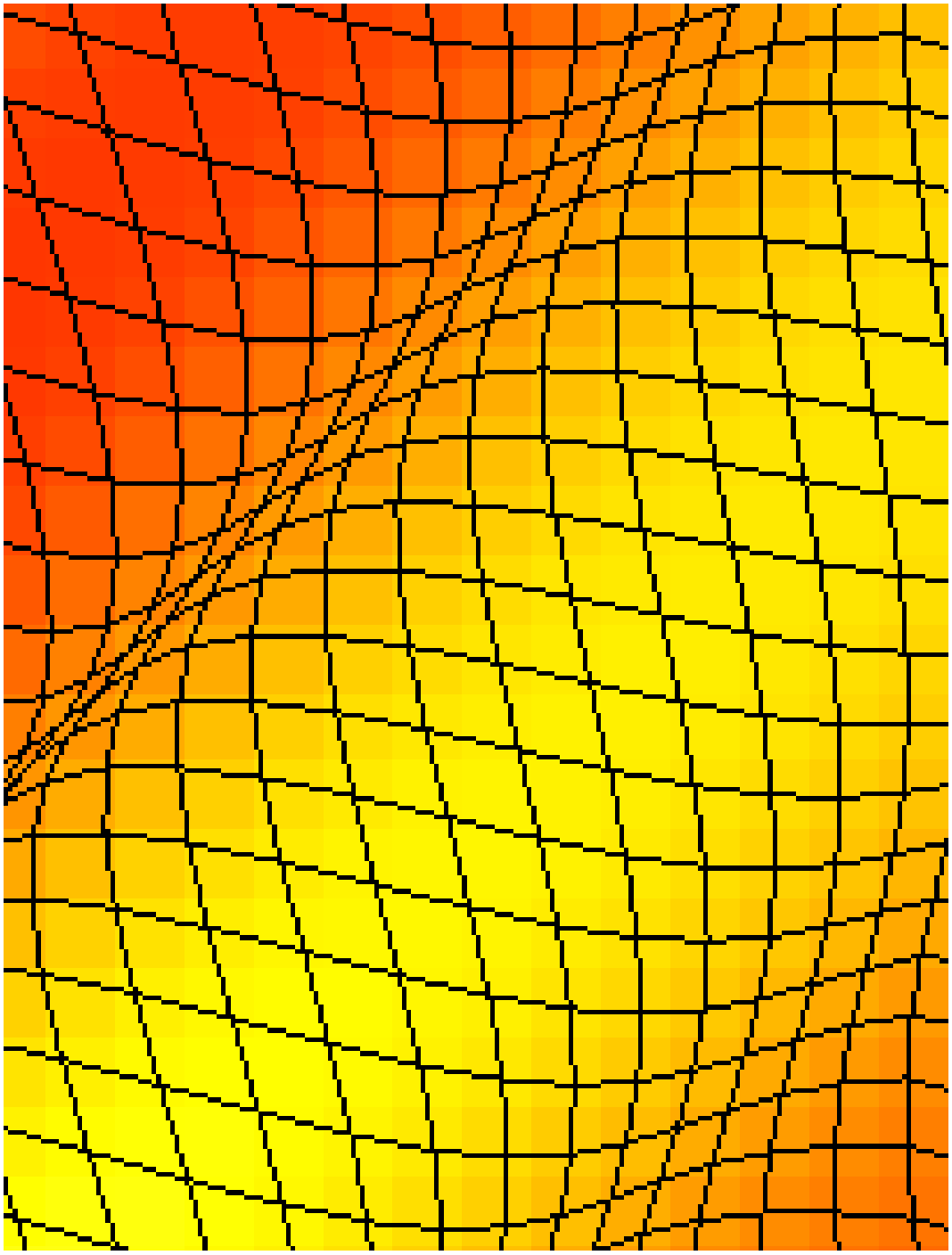


Figure A.5: A close view of an highly desirable mesh quality, $(x, y) \approx (7.56, 4.6)$.

B Appendix - Finite-Difference Figures

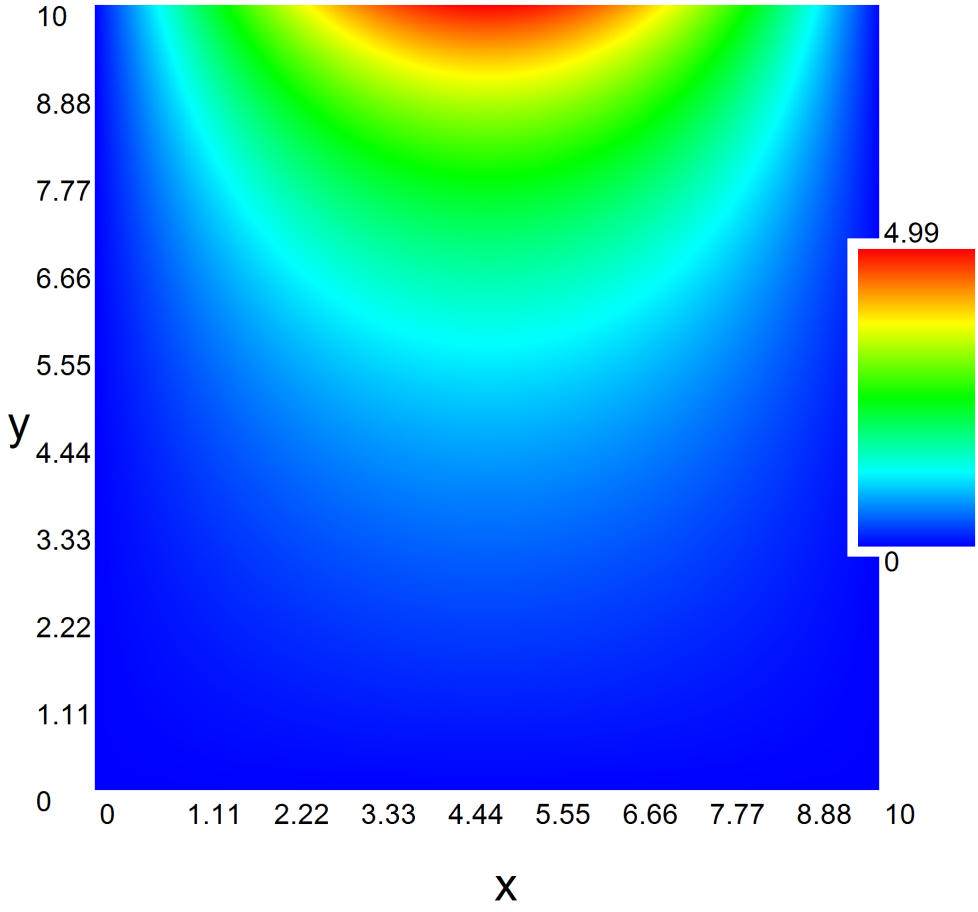


Figure B.1: Analytic solution for Laplace's equation with boundary conditions given in Figure 7.1.

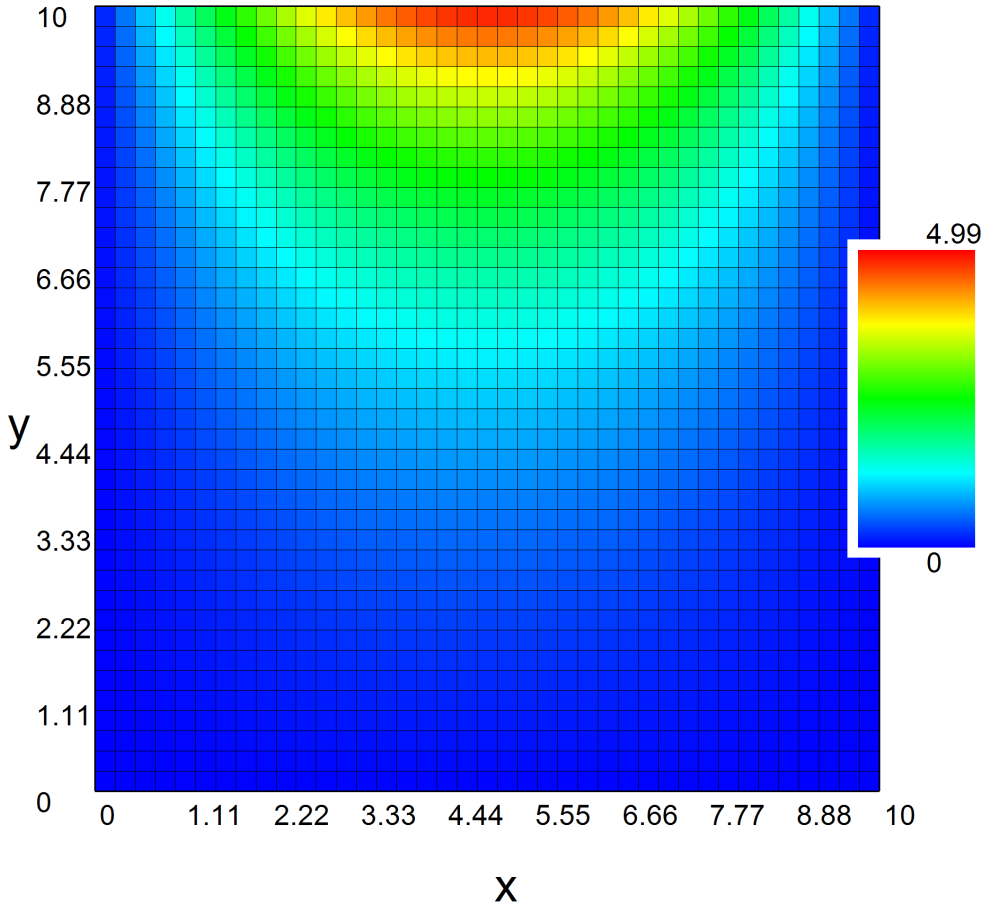


Figure B.2: Approximate solution for Laplace's equation with boundary conditions given in Figure 7.1.

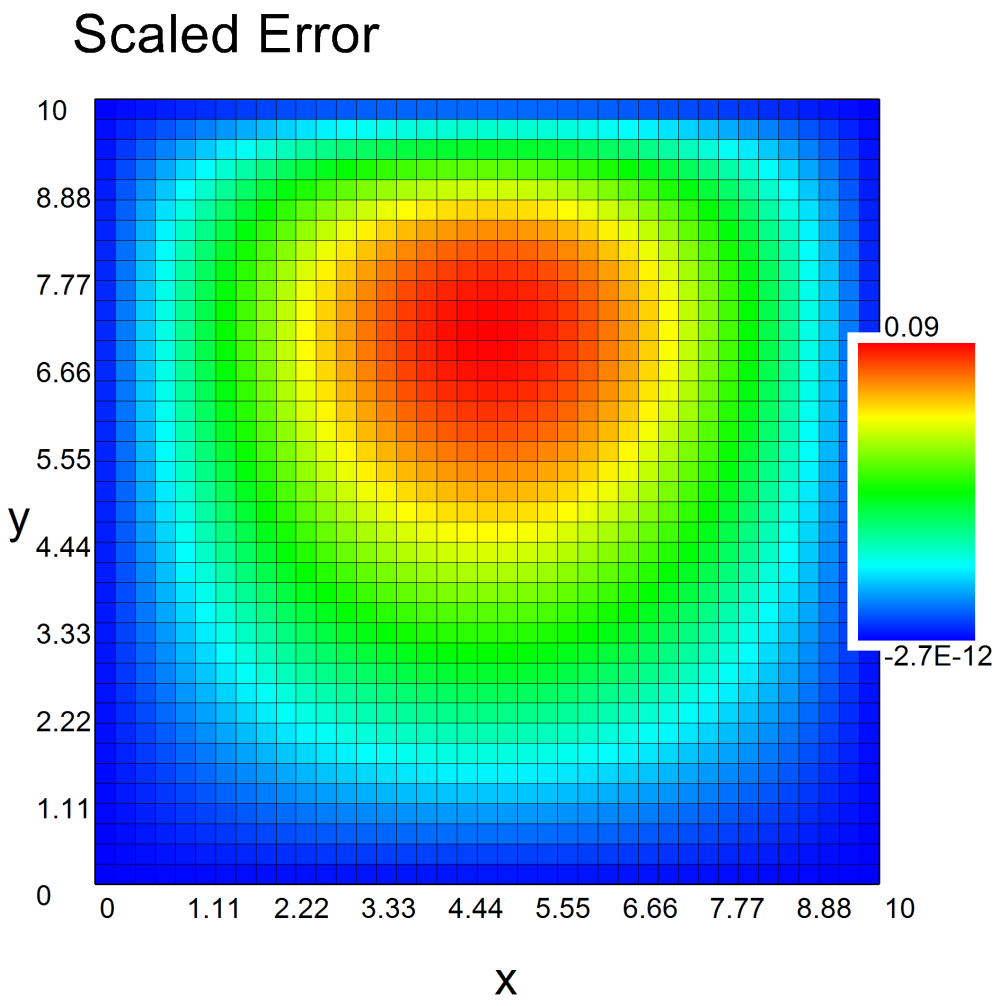
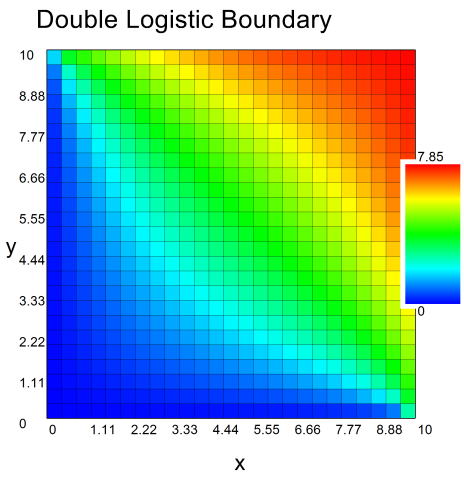
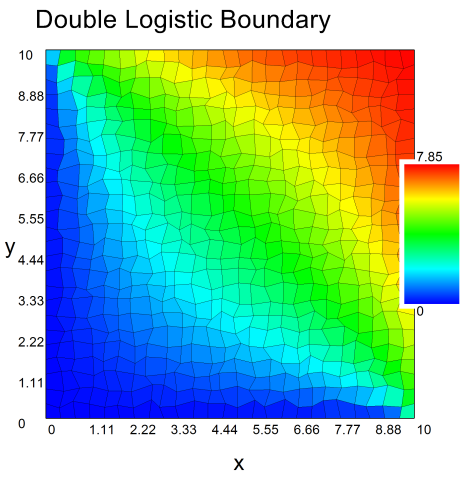


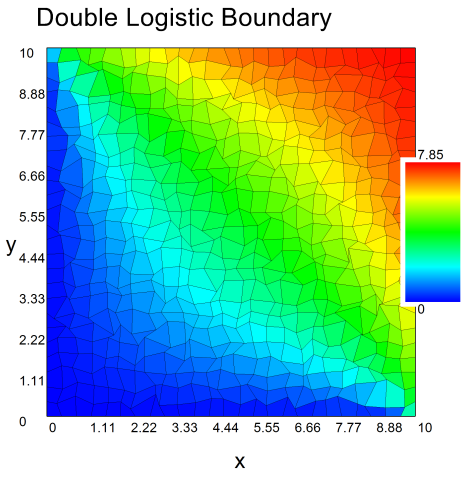
Figure B.3: Plot of the error between analytic and numerical solutions.



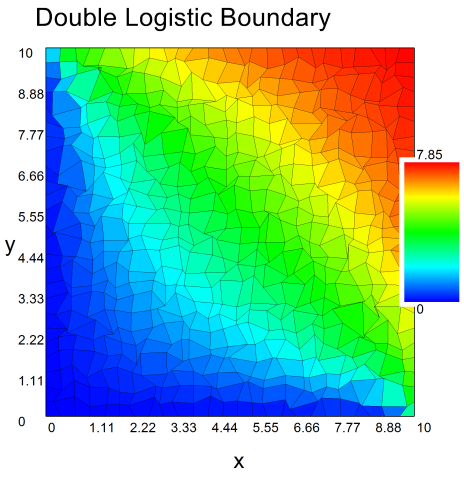
(a) Solution on orthogonal grid



(b) Solution on a non-orthogonal grid



(c) Solution on a non-orthogonal grid



(d) Solution on a non-orthogonal grid

Figure B.4: Solution on a collection of grids.

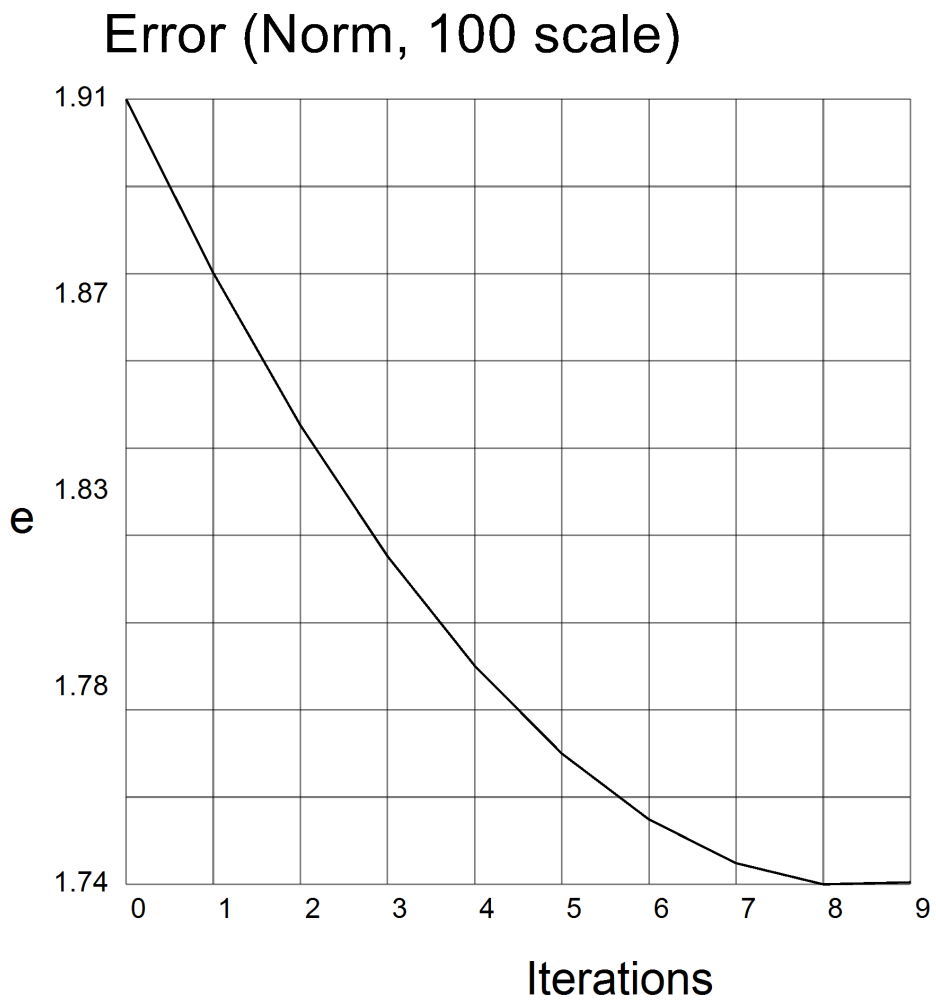


Figure B.5: Residual plot for gradient-ascent mesh-morphing for solution to problem posed in Figure 7.1. Note that the error has been scaled by a value of 100.

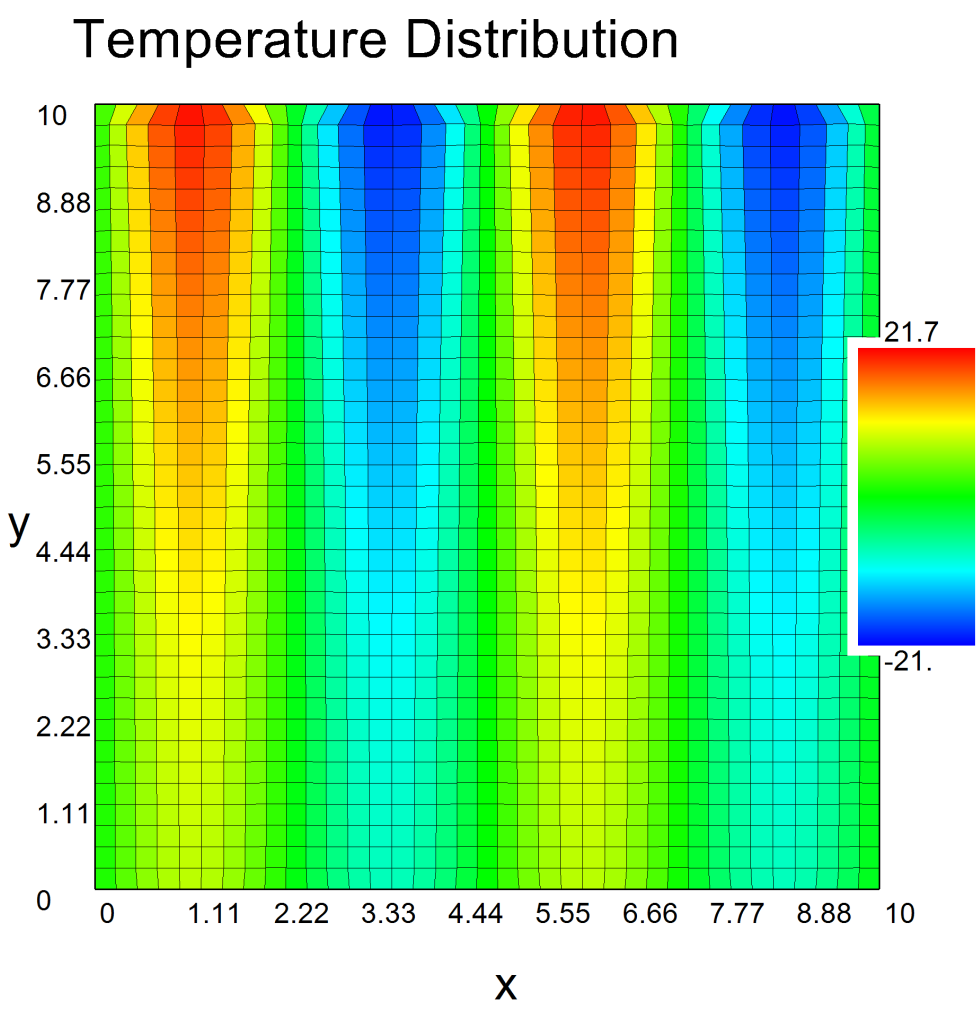


Figure B.6: Solution after iteration of gradient-ascent mesh-morphing.

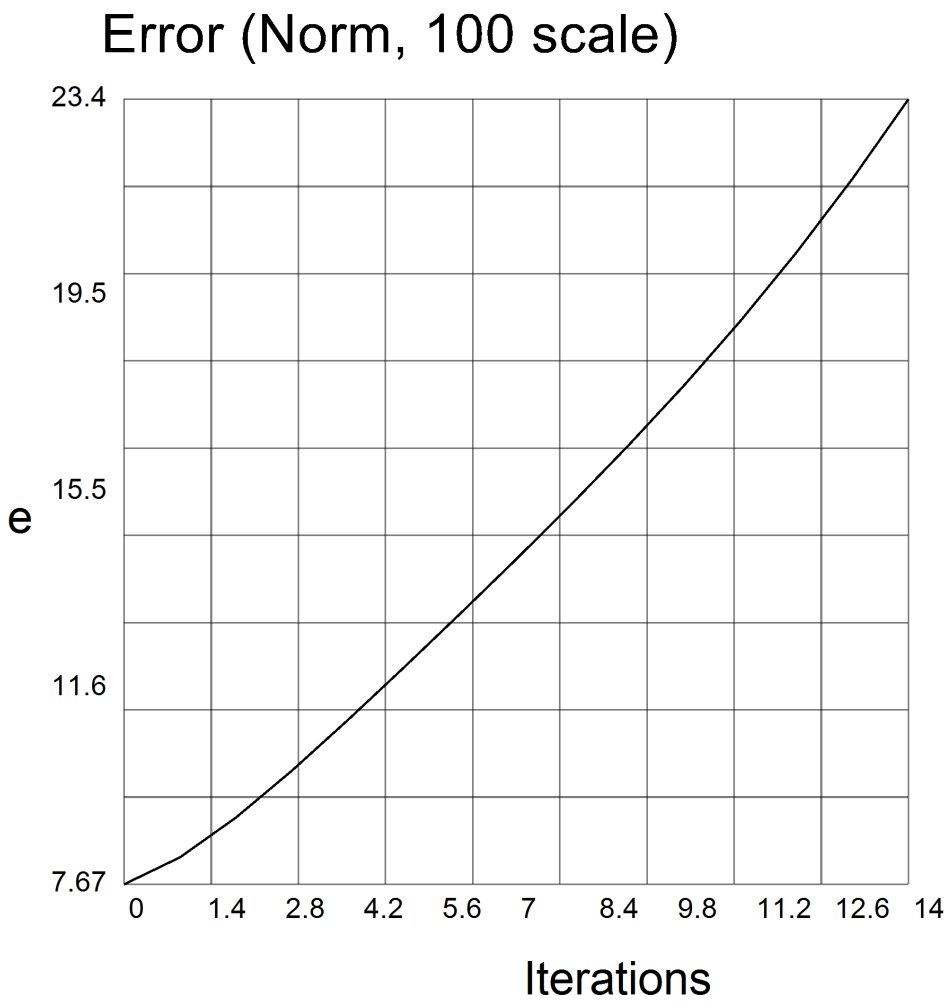


Figure B.7: Tracked errors, scaled by 100.

C Appendix - Modified Kaczmarz Convergence

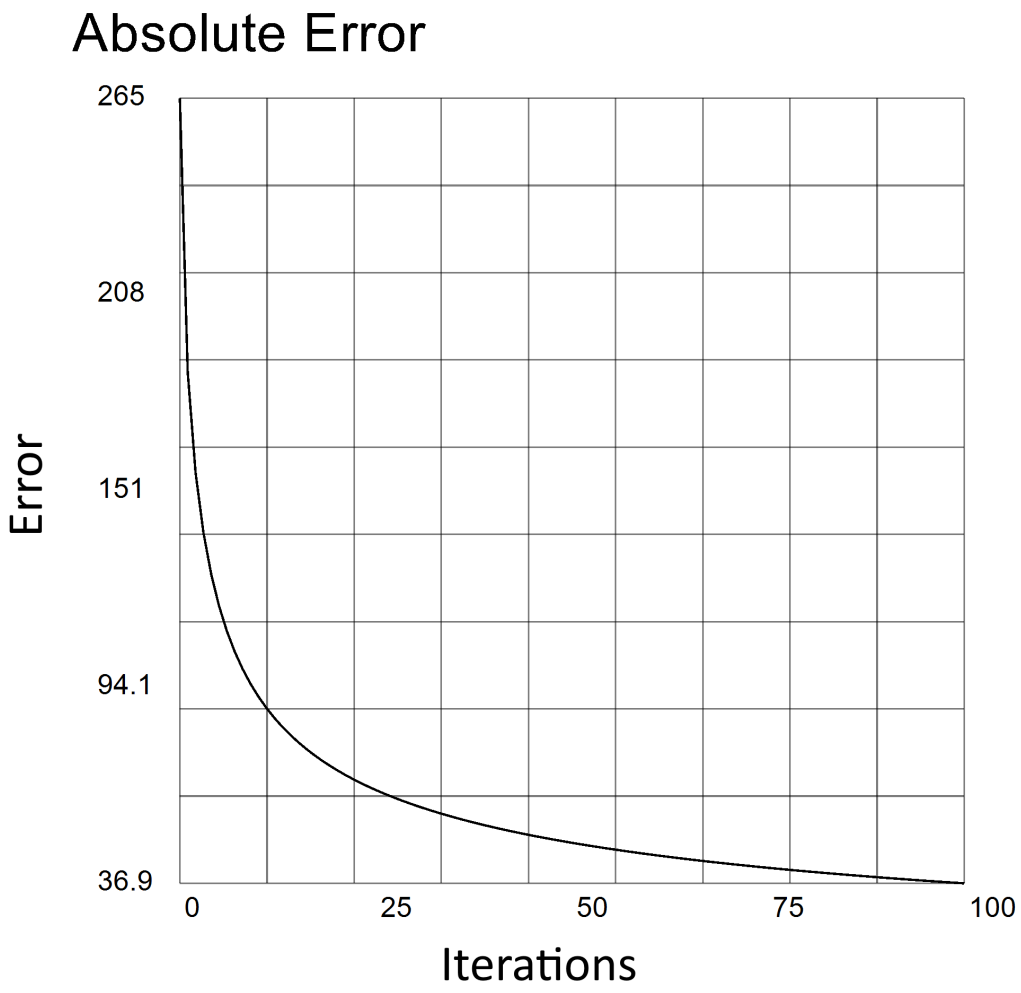


Figure C.1: Absolute convergence for the modified Kaczmarz algorithm on a 1444×1444 system.

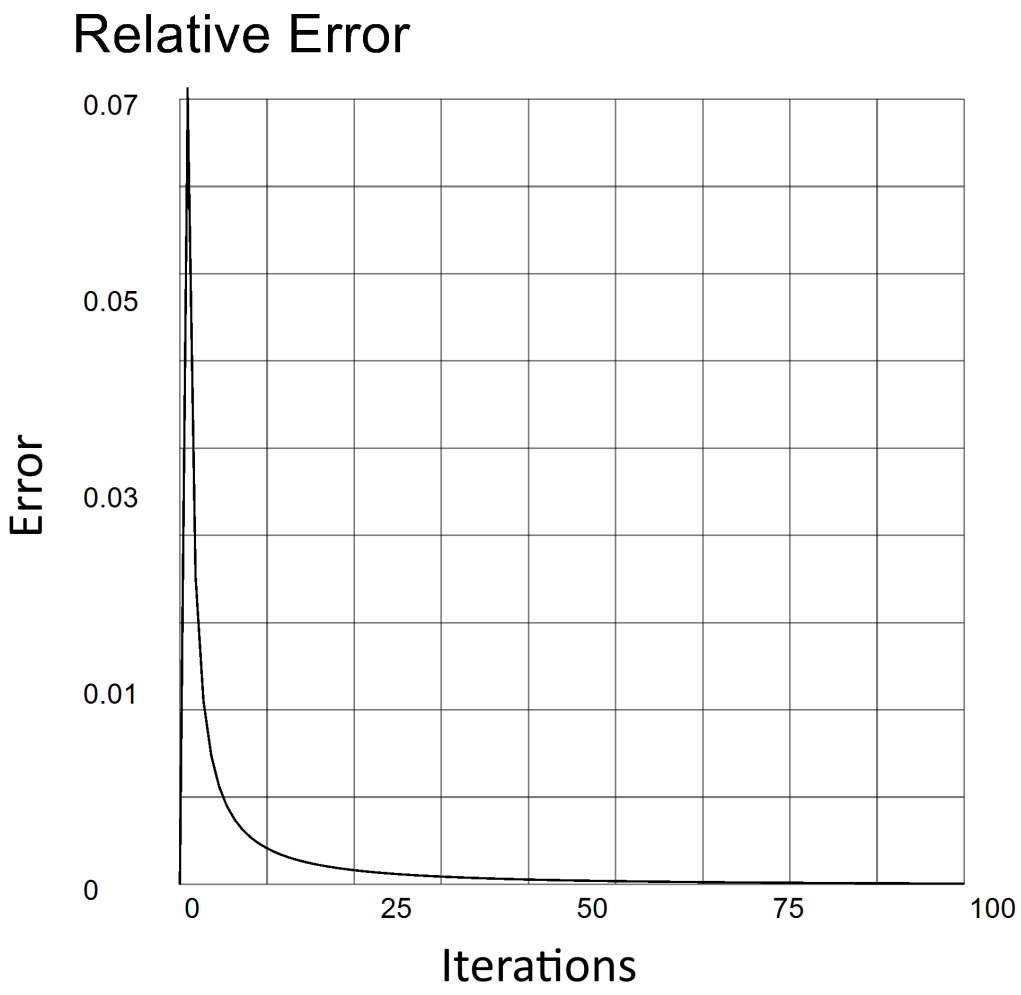


Figure C.2: Relative convergence for the modified Kaczmarz algorithm on a 1444×1444 system.

D Appendix: Derivation of Node Equations for Navier-Stokes System

We have the system given by

$$\rho\frac{D\mathbf{v}}{Dt} = -\nabla p + \mu\nabla^2\mathbf{v}$$
$$\nabla\cdot\mathbf{v} = 0.$$

We manipulate to get

$$\begin{aligned}
\frac{1}{\rho}\frac{\partial p}{\partial x} - \nu\frac{\partial^2 u}{\partial x^2} - \nu\frac{\partial^2 u}{\partial y^2} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} &= -\frac{\partial u}{\partial t}\\
\frac{1}{\rho}\frac{\partial p}{\partial y} - \nu\frac{\partial^2 v}{\partial x^2} - \nu\frac{\partial^2 v}{\partial y^2} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} &= -\frac{\partial v}{\partial t}\\
\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} &= 0.
\end{aligned} \tag{D.1}$$

We apply a generalized discretization from Sections 6.2.1 and 6.2.2, with

$$B = \begin{bmatrix} b_{1,1} & b_{1,2} & ... & b_{1,5}\\ b_{2,1} & b_{2,2} & ... & b_{2,5}\\ ... & ... & ... & ...\\ b_{5,1} & b_{5,2} & ... & b_{5,5}\end{bmatrix} = \begin{bmatrix}\mathbf{b_1}\\ \mathbf{b_2}\\ \mathbf{b_3}\\ \mathbf{b_4}\\ \mathbf{b_5}\end{bmatrix}$$

and, for a node variable Q,

$$\delta\mathbf{Q} = \mathbf{Q_o} - \mathbf{Q_n} = \begin{bmatrix} Q_{i+1,j}\\ Q_{i,j+1}\\ Q_{i-1,j}\\ Q_{i,j-1}\\ Q_{i+r_1,j+r_2}\end{bmatrix} - Q_{i,j}\begin{bmatrix}1\\1\\1\\1\\1\end{bmatrix}$$

to get, at a node (i,j):

$$\begin{aligned}
\frac{1}{\rho}\langle\delta\mathbf{p},\mathbf{b_1}\rangle - \nu\langle\delta\mathbf{u},\mathbf{b_3}\rangle - \nu\langle\delta\mathbf{u},\mathbf{b_4}\rangle + u^*\langle\delta\mathbf{u},\mathbf{b_1}\rangle + v^*\langle\delta\mathbf{u},\mathbf{b_2}\rangle &= \frac{u_{i,j}^{[n]} - u_{i,j}^{[n-1]}}{\delta t} = \frac{1}{\delta t}u_{i,j}^{[n]} - \frac{1}{\delta t}u_{i,j}^{[n-1]}\\
\frac{1}{\rho}\langle\delta\mathbf{p},\mathbf{b_2}\rangle - \nu\langle\delta\mathbf{v},\mathbf{b_3}\rangle - \nu\langle\delta\mathbf{v},\mathbf{b_4}\rangle + u^*\langle\delta\mathbf{v},\mathbf{b_1}\rangle + v^*\langle\delta\mathbf{v},\mathbf{b_2}\rangle &= \frac{v_{i,j}^{[n]} - v_{i,j}^{[n-1]}}{\delta t} = \frac{1}{\delta t}v_{i,j}^{[n]} - \frac{1}{\delta t}v_{i,j}^{[n-1]}\\
\langle\delta\mathbf{u},\mathbf{b_1}\rangle + \langle\delta\mathbf{v},\mathbf{b_2}\rangle &= 0
\end{aligned} \tag{D.2}$$

where $\langle\cdot,\cdot\rangle$ denotes the Euclidean inner product.

Using the fact that

$$\langle\delta\mathbf{Q},\mathbf{x}\rangle = \langle\mathbf{Q_o} - \mathbf{Q_n},\mathbf{x}\rangle = \langle\mathbf{Q_o},\mathbf{x}\rangle - \langle\mathbf{Q_n},\mathbf{x}\rangle,$$

we get the following from (D.2):

$$\begin{aligned}
\frac{1}{\rho}\langle\mathbf{p_o},\mathbf{b_1}\rangle - \frac{1}{\rho}\langle\mathbf{p_n},\mathbf{b_1}\rangle - \nu\langle\mathbf{u_o},\mathbf{b_3}\rangle + \nu\langle\mathbf{u_n},\mathbf{b_3}\rangle - \nu\langle\mathbf{u_o},\mathbf{b_4}\rangle + \nu\langle\mathbf{u_n},\mathbf{b_4}\rangle + u^*\langle\mathbf{u_o},\mathbf{b_1}\rangle&\\
-u^*\langle\mathbf{u_n},\mathbf{b_1}\rangle + v^*\langle\mathbf{u_o},\mathbf{b_2}\rangle - v^*\langle\mathbf{u_n},\mathbf{b_2}\rangle &= \frac{1}{\delta t}u_{i,j}^{[n]} - \frac{1}{\delta t}u_{i,j}^{[n-1]}\\
\frac{1}{\rho}\langle\mathbf{p_o},\mathbf{b_2}\rangle - \frac{1}{\rho}\langle\mathbf{p_n},\mathbf{b_2}\rangle - \nu\langle\mathbf{v_o},\mathbf{b_3}\rangle + \nu\langle\mathbf{v_n},\mathbf{b_3}\rangle - \nu\langle\mathbf{v_o},\mathbf{b_4}\rangle + \nu\langle\mathbf{v_n},\mathbf{b_4}\rangle + u^*\langle\mathbf{v_o},\mathbf{b_1}\rangle&\\
-u^*\langle\mathbf{v_n},\mathbf{b_1}\rangle + v^*\langle\mathbf{v_o},\mathbf{b_2}\rangle - v^*\langle\mathbf{v_n},\mathbf{b_2}\rangle &= \frac{1}{\delta t}v_{i,j}^{[n]} - \frac{1}{\delta t}v_{i,j}^{[n-1]}\\
\langle\mathbf{u_o},\mathbf{b_1}\rangle + \langle\mathbf{u_n},\mathbf{b_1}\rangle + \langle\mathbf{v_o},\mathbf{b_2}\rangle + \langle\mathbf{v_n},\mathbf{b_2}\rangle &= 0.
\end{aligned} \tag{D.3}$$

Noting that

$$\begin{bmatrix} u_{i,j}^{[n]} \\ v_{i,j}^{[n]} \end{bmatrix} = \begin{bmatrix} u_{i,j} \\ v_{i,j} \end{bmatrix} \qquad \text{and} \qquad \begin{bmatrix} u_{i,j}^{[n-1]} \\ v_{i,j}^{[n-1]} \end{bmatrix} = \begin{bmatrix} u_p \\ v_p \end{bmatrix}$$

and noting also that

$$\langle \mathbf{Q_n}, \mathbf{x} \rangle = Q_{i,j} \sum_k \mathbf{x}_k$$

we get that

$$\frac{1}{\rho}\langle \mathbf{p_o}, \mathbf{b_1} \rangle - p_{i,j}\frac{1}{\rho}\sum_k b_{1,k} - \nu\langle \mathbf{u_o}, \mathbf{b_3} \rangle + u_{i,j}\nu\sum_k b_{3,k} - \nu\langle \mathbf{u_o}, \mathbf{b_4} \rangle + u_{i,j}\nu\sum_k b_{4,k} + u^*\langle \mathbf{u_o}, \mathbf{b_1} \rangle - u_{i,j}u^*\sum_k b_{1,k}$$

$$+v^*\langle \mathbf{u_o}, \mathbf{b_2} \rangle - u_{i,j}v^*\sum_k b_{2,k} = \frac{1}{\delta t}u_{i,j} - \frac{1}{\delta t}u_p$$

$$\frac{1}{\rho}\langle \mathbf{p_o}, \mathbf{b_2} \rangle - p_{i,j}\frac{1}{\rho}\sum_k b_{2,k} - \nu\langle \mathbf{v_o}, \mathbf{b_3} \rangle + v_{i,j}\nu\sum_k b_{3,k} - \nu\langle \mathbf{v_o}, \mathbf{b_4} \rangle + v_{i,j}\nu\sum_k b_{4,k} + u^*\langle \mathbf{v_o}, \mathbf{b_1} \rangle - v_{i,j}u^*\sum_k b_{1,k} \quad \text{(D.4)}$$

$$+v^*\langle \mathbf{v_o}, \mathbf{b_2} \rangle - v_{i,j}v^*\sum_k b_{2,k} = \frac{1}{\delta t}v_{i,j} - \frac{1}{\delta t}v_p$$

$$\langle \mathbf{u_o}, \mathbf{b_1} \rangle + u_{i,j}\sum_k b_{1,k} + \langle \mathbf{v_o}, \mathbf{b_2} \rangle + v_{i,j}\sum_k b_{2,k} = 0.$$

We manipulate each equation in (D.4) separately:

$$\left(-\frac{1}{\rho}\sum_k b_{1,k}\right)p_{i,j} + \left(\frac{1}{\rho}b_{1,1}\right)p_{i+1,j} + \left(\frac{1}{\rho}b_{1,2}\right)p_{i,j+1} + \left(\frac{1}{\rho}b_{1,3}\right)p_{i-1,j} + \left(\frac{1}{\rho}b_{1,4}\right)p_{i,j-1} + \left(\frac{1}{\rho}b_{1,5}\right)p_{i+r_1,j+r_2}$$

$$+\left(-u^*\sum_k b_{1,k} - v^*\sum_k b_{2,k} + \nu\sum_k b_{3,k} + \nu\sum_k b_{4,k} - \frac{1}{\delta t}\right)u_{i,j} + (u^*b_{1,1} + v^*b_{2,1} - \nu b_{3,1} - \nu b_{4,1})u_{i+1,j} \quad \text{(D.5)}$$

$$+(u^*b_{1,2} + v^*b_{2,2} - \nu b_{3,2} - \nu b_{4,2})u_{i,j+1} + (u^*b_{1,3} + v^*b_{2,3} - \nu b_{3,3} - \nu b_{4,3})u_{i-1,j}$$

$$(u^*b_{1,4} + v^*b_{2,4} - \nu b_{3,4} - \nu b_{4,4})u_{i,j-1} + (u^*b_{1,5} + v^*b_{2,5} - \nu b_{3,5} - \nu b_{4,5})u_{i+r_1,j+r_2} = \frac{-1}{\delta t}u_p.$$

Upon observing (D.5), it may prove useful to define the following terms:

$$P_0 = \left(-\frac{1}{\rho}\sum_k b_{1,k}\right) \quad P_1 = \frac{b_{1,1}}{\rho} \quad P_2 = \frac{b_{1,2}}{\rho} \quad P_3 = \frac{b_{1,3}}{\rho} \quad P_4 = \frac{b_{1,4}}{\rho} \quad P_5 = \frac{b_{1,5}}{\rho}$$

$$Q_0 = \left(-u^*\sum_k b_{1,k} - v^*\sum_k b_{2,k} + \nu\sum_k b_{3,k} + \nu\sum_k b_{4,k} - \frac{1}{\delta t}\right) \quad Q_1 = u^*b_{1,1} + v^*b_{2,1} - \nu b_{3,1} - \nu b_{4,1}$$

$$Q_2 = u^*b_{1,2} + v^*b_{2,2} - \nu b_{3,2} - \nu b_{4,2} \quad Q_3 = u^*b_{1,3} + v^*b_{2,3} - \nu b_{3,3} - \nu b_{4,3} \quad Q_4 = u^*b_{1,4} + v^*b_{2,4} - \nu b_{3,4} - \nu b_{4,4}$$

$$Q_5 = u^*b_{1,5} + v^*b_{2,5} - \nu b_{3,5} - \nu b_{4,5}.$$

This allows (D.5) to be put into the following form:

$$P_0p_{i,j} + P_1p_{i+1,j} + P_2p_{i,j+1} + P_3p_{i-1,j} + P_4p_{i,j-1} + P_5p_{i+r_1,j+r_2} \quad \text{(D.6)}$$

$$+Q_0u_{i,j} + Q_1u_{i+1,j} + Q_2u_{i,j+1} + Q_3u_{i-1,j} + Q_4u_{i,j-1} + Q_5u_{i+r_1,j+r_2} = -\frac{1}{\delta t}u_p.$$

It is a trivial matter to show that, by a similar process, the second equation in (D.4) can be reduced to

$$R_0p_{i,j} + R_1p_{i+1,j} + R_2p_{i,j+1} + R_3p_{i-1,j} + R_4p_{i,j-1} + R_5p_{i+r_1,j+r_2} \quad \text{(D.7)}$$

$$+S_0 v_{i,j} + S_1 v_{i+1,j} + S_2 v_{i,j+1} + S_3 v_{i-1,j} + S_4 v_{i,j-1} + S_5 v_{i+r_1,j+r_2} = -\frac{1}{\delta t} v_p,$$

where

$$R_0 = \left(-\frac{1}{\rho} \sum_k b_{2,k} \right) \quad R_1 = \frac{b_{2,1}}{\rho} \quad R_2 = \frac{b_{2,2}}{\rho} \quad R_3 = \frac{b_{2,3}}{\rho} \quad R_4 = \frac{b_{2,4}}{\rho} \quad R_5 = \frac{b_{2,5}}{\rho}$$

$$S_0 = \left(-u^* \sum_k b_{1,k} - v^* \sum_k b_{2,k} + \nu \sum_k b_{3,k} + \nu \sum_k b_{4,k} - \frac{1}{\delta t} \right) \quad S_1 = u^* b_{1,1} + v^* b_{2,1} - \nu b_{3,1} - \nu b_{4,1}$$

$$S_2 = u^* b_{1,2} + v^* b_{2,2} - \nu b_{3,2} - \nu b_{4,2} \quad S_3 = u^* b_{1,3} + v^* b_{2,3} - \nu b_{3,3} - \nu b_{4,3} \quad S_4 = u^* b_{1,4} + v^* b_{2,4} - \nu b_{3,4} - \nu b_{4,4}$$

$$S_5 = u^* b_{1,5} + v^* b_{2,5} - \nu b_{3,5} - \nu b_{4,5}.$$

Node equations corresponding to the first two equations in (D.1) are now given by (D.6) and (D.7). The third term in (D.1) can be manipulated into a linear from also, yielding

$$T_0 u_{i,j} + T_1 u_{i+1,j} + T_2 u_{i,j+1} + T_3 u_{i-1,j} + T_4 u_{i,j-1} + T_5 u_{i+r_1,j+r_2}$$
$$+W_0 v_{i,j} + W_1 v_{i+1,j} + W_2 v_{i,j+1} + W_3 v_{i-1,j} + W_4 v_{i,j-1} + W_5 v_{i+r_1,j+r_2} = 0 \tag{D.8}$$

where

$$T_0 = \sum_k b_{1,k} \quad T_1 = b_{1,1} \quad T_2 = b_{1,2} \quad T_3 = b_{1,3} \quad T_4 = b_{1,4} \quad T_5 = b_{1,5}$$

$$W_0 = \sum_k b_{2,k} \quad W_1 = b_{2,1} \quad W_2 = b_{2,2} \quad W_3 = b_{2,3} \quad W_4 = b_{2,4} \quad W_5 = b_{2,5}.$$

E Appendix: Source Code

```
/* Copyright (c) 2018 William van Noordt */using System;using System.Collections.Generic;using
System.Linq;using System.Text;using System.IO;using System.Threading.Tasks;namespace m_435_NCAM
R{class BoundaryConditions{private string bc_repo = @"C:\Users\Will\Desktop\Folders\MATH435\rep
o\boundary-conditions";public enum Direction{Positive_X,Positive_Y,Negative_X,Negative_Y}public
 enum BoundaryConditionType{Dirichlet,Neumann}private RBounds2D bounds;private BoundaryConditio
nType positive_x;private BoundaryConditionType positive_y;private BoundaryConditionType negativ
e_x;private BoundaryConditionType negative_y;public RBounds2D Bounds{get { return bounds; }}pri
vate int xcount, ycount;public int Xcount{get { return xcount; }}public int Ycount{get { return
 ycount; }}private double[] positive_x_vals;private double[] positive_y_vals;private double[] n
egative_x_vals;private double[] negative_y_vals;public double this[int i, Direction indicator]{
get{switch (indicator){case Direction.Positive_X:{return positive_x_vals[i];}case Direction.Neg
ative_X:{return negative_x_vals[i];}case Direction.Positive_Y:{return positive_y_vals[i];}case
Direction.Negative_Y:{return negative_y_vals[i];}default:{throw new Exception("Invalid directio
n.");}}}set{switch (indicator){case Direction.Positive_X:{positive_x_vals[i] = value; break;}ca
se Direction.Negative_X:{negative_x_vals[i] = value; break;}case Direction.Positive_Y:{positive
_y_vals[i] = value; break;}case Direction.Negative_Y:{negative_y_vals[i] = value; break;}defaul
t:{throw new Exception("Invalid direction.");}}}}public BoundaryConditions(RBounds2D _bounds, i
nt _xcount, int _ycount, BoundaryConditionType _type){xcount = _xcount;ycount = _ycount;positiv
e_x_vals = new double[xcount];positive_y_vals = new double[xcount];negative_x_vals = new double
[ycount];negative_y_vals = new double[ycount];bounds = _bounds;SetAllBoundaryConditionTypes(_ty
pe);}public void SetAllBoundaryConditionTypes(BoundaryConditionType type){positive_x = type;pos
itive_y = type;negative_x = type;negative_y = type;}public BoundaryConditions(NCGrid_Distributi
on boundary_of, BoundaryConditionType _type){bounds = boundary_of.Bounds;SetAllBoundaryConditio
nTypes(_type);xcount = boundary_of.Xcount;ycount = boundary_of.Ycount;positive_x_vals = new dou
ble[xcount];positive_y_vals = new double[xcount];negative_x_vals = new double[ycount];negative_
y_vals = new double[ycount];for (int i = 0; i < xcount; i++){positive_x_vals[i] = boundary_of[i
, ycount - 1].Value;negative_x_vals[i] = boundary_of[i, 0].Value;}for (int j = 0; j < ycount; j
++){positive_y_vals[j] = boundary_of[xcount-1,j].Value;negative_y_vals[j] = boundary_of[0,j].Va
lue;}}public void SetBoundaryConditionType(Direction indicator, BoundaryConditionType type){swi
```

```
tch (indicator){case Direction.Positive_X:{positive_x = type;break;}case Direction.Negative_X:{
negative_x = type;break;}case Direction.Positive_Y:{positive_y = type;break;}case Direction.Neg
ative_Y:{negative_y = type;break;}}}public BoundaryConditionType GetBoundaryConditionType(Direc
tion indicator){switch (indicator){case Direction.Positive_X:{return positive_x;}case Direction
.Negative_X:{return negative_x;}case Direction.Positive_Y:{return positive_y;}case Direction.Ne
gative_Y:{return negative_y;}default:{throw new Exception("Error: Invalid direction.");}}}publi
c void WriteFile(string title, bool using_full_path, bool allow_overwrite){string path = bc_rep
o + "\\" + title + ".bc";if (using_full_path) { path = title; }if (File.Exists(path) && !allow_
overwrite){throw new Exception("Error: Overwriting permissions not granted.");}List<string> stu
ff = new List<string>();stuff.Add(bounds.ToString());stuff.Add(string.Format("Nx:{0}:Ny:{1}", x
count, ycount));string px = "positive_x:" + ConditionTypeToString(positive_x);stuff.Add(px);int
 px_count = positive_x_vals.Length;for (int i = 0; i < px_count; i++){stuff.Add(positive_x_vals
[i].ToString());}string nx = "negative_x:" + ConditionTypeToString(negative_x);stuff.Add(nx);in
t nx_count = negative_x_vals.Length;for (int i = 0; i < nx_count; i++){stuff.Add(negative_x_val
s[i].ToString());}string py = "positive_y:" + ConditionTypeToString(positive_y);stuff.Add(py);i
nt py_count = positive_y_vals.Length;for (int i = 0; i < py_count; i++){stuff.Add(positive_y_va
ls[i].ToString());}string ny = "negative_y:" + ConditionTypeToString(negative_y);stuff.Add(ny);
int ny_count = negative_y_vals.Length;for (int i = 0; i < ny_count; i++){stuff.Add(negative_y_v
als[i].ToString());}File.WriteAllLines(path, stuff.ToArray());}public BoundaryConditions(string
 title, bool using_full_path){string path = bc_repo + "//" + title + ".bc";if (using_full_path)
 { path = title; }string[] contents = File.ReadAllLines(path);bounds = RBounds2D.fromstring(con
tents[0]);if (!(int.TryParse(contents[1].Split(':')[1], out xcount) && int.TryParse(contents[1]
.Split(':')[3], out ycount))) { throw new Exception("Error: Improper file format."); }int px_of
fset = 3;int nx_offset = px_offset+1+xcount;int py_offset = nx_offset+1+xcount;int ny_offset =
py_offset+1+ycount;int terminal = ny_offset+1+ycount;positive_x_vals = new double[xcount];posit
ive_y_vals = new double[xcount];negative_x_vals = new double[ycount];negative_y_vals = new doub
le[ycount];positive_x = ConditionTypeFromString(contents[px_offset - 1].Split(':')[1]);negative
_x = ConditionTypeFromString(contents[nx_offset - 1].Split(':')[1]);positive_y = ConditionTypeF
romString(contents[py_offset - 1].Split(':')[1]);negative_y = ConditionTypeFromString(contents[
ny_offset - 1].Split(':')[1]);for (int i = px_offset; i < nx_offset-1; i++){double z;if (!doubl
e.TryParse(contents[i], out z)) { throw new Exception("Error: Improper file format"); }positive
_x_vals[i - px_offset] = z;}for (int i = nx_offset; i < py_offset-1; i++){double z;if (!double.
TryParse(contents[i], out z)) { throw new Exception("Error: Improper file format"); }negative_x
_vals[i - nx_offset] = z;}for (int i = py_offset; i < ny_offset-1; i++){double z;if (!double.Tr
yParse(contents[i], out z)) { throw new Exception("Error: Improper file format"); }positive_y_v
als[i - py_offset] = z;}for (int i = ny_offset; i < terminal-1; i++){double z;if (!double.TryPa
rse(contents[i], out z)) { throw new Exception("Error: Improper file format"); }negative_y_vals
[i - ny_offset] = z;}}public void SetConstant(double val, Direction indicator){switch (indicato
r){case Direction.Positive_X:{for (int i = 0; i < xcount; i++){positive_x_vals[i] = val;}break;
}case Direction.Negative_X:{for (int i = 0; i < xcount; i++){negative_x_vals[i] = val;}break;}c
ase Direction.Positive_Y:{for (int i = 0; i < ycount; i++){positive_y_vals[i] = val;}break;}cas
e Direction.Negative_Y:{for (int i = 0; i < ycount; i++){negative_y_vals[i] = val;}break;}}}pri
vate string ConditionTypeToString(BoundaryConditionType type){switch (type){case BoundaryCondit
ionType.Dirichlet:{return "dirichlet";}case BoundaryConditionType.Neumann:{return "neumann";}de
fault:{throw new Exception("Error: Invalid type.");}}}private BoundaryConditionType ConditionTy
peFromString(string source){switch (source){case "dirichlet":{return BoundaryConditionType.Diri
chlet;}case "neumann":{return BoundaryConditionType.Neumann;}default:{throw new Exception("Erro
r: Invalid type.");}}}}}/* Copyright (c) 2018 William van Noordt */using System;using System.Co
llections.Generic;using System.Linq;using System.Text;using System.Threading.Tasks;namespace m_
435_NCAMR{class BVPLinear2D{private bool console_output = false;private BoundaryConditions boun
dary_conditions;public BoundaryConditions Boundary_Conditions{get { return boundary_conditions;
 }set { boundary_conditions = value; }}LinearOperatorOrder2 lin_operator;public LinearOperatorO
rder2 LinearOperator{get { return lin_operator; }set { lin_operator = value; }}NCGrid_Distribut
ion discretization;public NCGrid_Distribution Discretization{get { return discretization; }set
{ discretization = value; }}public BVPLinear2D(BoundaryConditions _conditions, LinearOperatorOr
der2 _operator, NCGrid_Distribution _discretization){lin_operator = _operator;discretization =
```

```
_discretization;boundary_conditions = _conditions;bool[] all_necessary_compatibility_conditions
 ={boundary_conditions.Bounds.Xmax == discretization.Bounds.Xmax,boundary_conditions.Bounds.Xmi
n == discretization.Bounds.Xmin,boundary_conditions.Bounds.Ymax == discretization.Bounds.Ymax,b
oundary_conditions.Bounds.Ymin == discretization.Bounds.Ymin,boundary_conditions.Xcount == disc
retization.Xcount,boundary_conditions.Ycount == discretization.Ycount};bool compatible = true;f
oreach (bool i in all_necessary_compatibility_conditions){compatible = compatible && i;}if (!co
mpatible) { throw new Exception("Error: Boundary conditions and initial discretization are inco
mpatible."); }}private enum BoundaryCase{PositiveXBoundary,NegativeXBoundary,PositiveYBoundary,
NegativeYBoundary,LLCorner,LRCorner,ULCorner,URCorner,Body}public void EnableConsoleOutput(){co
nsole_output = true;}public void DisableConsoleOutput(){console_output = false;}public NCGrid_D
istribution Solve(){int m = discretization.Xcount;int n = discretization.Ycount;int total_nodes
 = (m - 2) * (n - 2);NCGrid_Distribution dist = discretization.Clone();dist.ApplyBoundary(bound
ary_conditions);Matrix system_matrix = new Matrix(total_nodes, total_nodes);Matrix RHS = new Ma
trix(total_nodes, 1);int neg_y = 0;int pos_y = 0;int neg_x = 0;int pos_x = 0;int body = 0;int l
l_corner = 0;int lr_corner = 0;int ur_corner = 0;int ul_corner = 0;if (console_output) { Consol
e.WriteLine("Populating linear system..."); }for (int i = 1; i < m - 1; i++){if (console_output
 && i % (m - 1) / 13 == 0){Console.WriteLine((100 * i / (m - 1)).ToString() + "%");}for (int j
= 1; j < n - 1; j++){double rhs_here = 0;//for now, assume zero forcing function.BoundaryCase _
case;bool interior = !isCloseToBoundary(i, j, m, n, out _case);int surplusi = 1;int surplusj =
1;Matrix b = dist.GetTaylorSystemCoeffs(i, j, surplusi, surplusj);double uijterm = 0;double ui1
jterm = 0;double uij1term = 0;double ui_1jterm = 0;double uij_1term = 0;double usurplusterm = 0
;int row = (m - 2) * (i - 1) + j - 1;for (int h = 0; h < 5; h++){ui1jterm += b[h, 0] * lin_oper
ator[h];uij1term += b[h, 1] * lin_operator[h];ui_1jterm += b[h, 2] * lin_operator[h];uij_1term
+= b[h, 3] * lin_operator[h];usurplusterm += b[h, 4] * lin_operator[h];double temp_ij = 0;for (
int k = 0; k < 5; k++){temp_ij += b[h, k];}uijterm += lin_operator[h] * temp_ij;}switch (_case)
{case BoundaryCase.NegativeYBoundary:{rhs_here -= dist[i, j - 1].Value * uij_1term;uij_1term =
0;neg_y++;break;}case BoundaryCase.PositiveYBoundary:{rhs_here -= dist[i, j + 1].Value * uij1te
rm;rhs_here -= dist[i + surplusi, j + surplusj].Value * usurplusterm;usurplusterm = 0;uij1term
= 0;pos_y++;break;}case BoundaryCase.NegativeXBoundary:{rhs_here -= dist[i - 1, j].Value * ui_1
jterm;ui_1jterm = 0;neg_x++;break;}case BoundaryCase.PositiveXBoundary:{rhs_here -= dist[i + 1,
 j].Value * ui1jterm;rhs_here -= dist[i + surplusi, j + surplusj].Value * usurplusterm;usurplus
term = 0;ui1jterm = 0;pos_x++;break;}case BoundaryCase.ULCorner:{rhs_here -= dist[i, j + 1].Val
ue * uij1term;rhs_here -= dist[i - 1, j].Value * ui_1jterm;rhs_here -= dist[i + surplusi, j + s
urplusj].Value * usurplusterm;uij1term = 0;ui_1jterm = 0;usurplusterm = 0;ul_corner++;break;}ca
se BoundaryCase.LLCorner:{rhs_here -= dist[i - 1, j].Value * ui_1jterm;rhs_here -= dist[i, j -
1].Value * uij_1term;ui_1jterm = 0;uij_1term = 0;ll_corner++;break;}case BoundaryCase.URCorner:
{rhs_here -= dist[i + 1, j].Value * ui1jterm;rhs_here -= dist[i, j + 1].Value * uij1term;rhs_he
re -= dist[i + surplusi, j + surplusj].Value * usurplusterm;ui1jterm = 0;uij1term = 0;usurplust
erm = 0;ur_corner++;break;}case BoundaryCase.LRCorner:{rhs_here -= dist[i, j - 1].Value * uij_1
term;rhs_here -= dist[i + 1, j].Value * ui1jterm;rhs_here -= dist[i + surplusj, j + surplusj].V
alue * usurplusterm;uij_1term = 0;ui1jterm = 0;usurplusterm = 0;lr_corner++;break;}case Boundar
yCase.Body:{body++;break;}}system_matrix[row, map2Dto1D(i - 1, j - 1, m - 1, n - 1)] = -1 * uij
term;if (ui1jterm != 0){system_matrix[row, map2Dto1D(i, j - 1, m - 1, n - 1)] = ui1jterm;}if (u
ij1term != 0){system_matrix[row, map2Dto1D(i - 1, j, m - 1, n - 1)] = uij1term;}if (ui_1jterm !
= 0){system_matrix[row, map2Dto1D(i - 2, j - 1, m - 1, n - 1)] = ui_1jterm;}if (uij_1term != 0)
{system_matrix[row, map2Dto1D(i - 1, j - 2, m - 1, n - 1)] = uij_1term;}if (usurplusterm != 0){
system_matrix[row, map2Dto1D(i - 1 + surplusi, j - 1 + surplusj, m - 1, n - 1)] = usurplusterm;
}RHS[row] = rhs_here;}}if (console_output){Console.WriteLine("System populated.");}Matrix resul
ts = RunExteriorSolve(system_matrix, RHS);for (int i = 0; i < total_nodes; i++){int offset = n
- 2;int J = i % offset;int I = (i - J) / offset;dist.assign_value_at(I + 1, J + 1, results[i]);
}return dist;}public NCGrid_Distribution SolveKaczMarzExt(int maxiterations, double tolerance,
int radius){int m = discretization.Xcount;int n = discretization.Ycount;int total_nodes = (m -
2) * (n - 2);NCGrid_Distribution dist = discretization.Clone();dist.ApplyBoundary(boundary_cond
itions);Matrix system_matrix = new Matrix(total_nodes, total_nodes);Matrix RHS = new Matrix(tot
al_nodes, 1);int neg_y = 0;int pos_y = 0;int neg_x = 0;int pos_x = 0;int body = 0;int ll_corner
 = 0;int lr_corner = 0;int ur_corner = 0;int ul_corner = 0;if (console_output) { Console.WriteL
```

```
ine("Populating linear system..."); }for (int i = 1; i < m - 1; i++){if (console_output && i %
(m - 1) / 13 == 0){Console.WriteLine((100 * i / (m - 1)).ToString() + "%");}for (int j = 1; j <
 n - 1; j++){double rhs_here = 0;//for now, assume zero forcing function.BoundaryCase _case;boo
l interior = !isCloseToBoundary(i, j, m, n, out _case);int surplusi = 1;int surplusj = 1;Matrix
 b = dist.GetTaylorSystemCoeffs(i, j, surplusi, surplusj);double uijterm = 0;double ui1jterm =
0;double uij1term = 0;double ui_1jterm = 0;double uij_1term = 0;double usurplusterm = 0;int row
 = (m - 2) * (i - 1) + j - 1;for (int h = 0; h < 5; h++){ui1jterm += b[h, 0] * lin_operator[h];
uij1term += b[h, 1] * lin_operator[h];ui_1jterm += b[h, 2] * lin_operator[h];uij_1term += b[h,
3] * lin_operator[h];usurplusterm += b[h, 4] * lin_operator[h];double temp_ij = 0;for (int k =
0; k < 5; k++){temp_ij += b[h, k];}uijterm += lin_operator[h] * temp_ij;}switch (_case){case Bo
undaryCase.NegativeYBoundary:{rhs_here -= dist[i, j - 1].Value * uij_1term;uij_1term = 0;neg_y+
+;break;}case BoundaryCase.PositiveYBoundary:{rhs_here -= dist[i, j + 1].Value * uij1term;rhs_h
ere -= dist[i + surplusi, j + surplusj].Value * usurplusterm;usurplusterm = 0;uij1term = 0;pos_
y++;break;}case BoundaryCase.NegativeXBoundary:{rhs_here -= dist[i - 1, j].Value * ui_1jterm;ui
_1jterm = 0;neg_x++;break;}case BoundaryCase.PositiveXBoundary:{rhs_here -= dist[i + 1, j].Valu
e * ui1jterm;rhs_here -= dist[i + surplusi, j + surplusj].Value * usurplusterm;usurplusterm = 0
;ui1jterm = 0;pos_x++;break;}case BoundaryCase.ULCorner:{rhs_here -= dist[i, j + 1].Value * uij
1term;rhs_here -= dist[i - 1, j].Value * ui_1jterm;rhs_here -= dist[i + surplusi, j + surplusj]
.Value * usurplusterm;uij1term = 0;ui_1jterm = 0;usurplusterm = 0;ul_corner++;break;}case Bound
aryCase.LLCorner:{rhs_here -= dist[i - 1, j].Value * ui_1jterm;rhs_here -= dist[i, j - 1].Value
 * uij_1term;ui_1jterm = 0;uij_1term = 0;ll_corner++;break;}case BoundaryCase.URCorner:{rhs_her
e -= dist[i + 1, j].Value * ui1jterm;rhs_here -= dist[i, j + 1].Value * uij1term;rhs_here -= di
st[i + surplusi, j + surplusj].Value * usurplusterm;ui1jterm = 0;uij1term = 0;usurplusterm = 0;
ur_corner++;break;}case BoundaryCase.LRCorner:{rhs_here -= dist[i, j - 1].Value * uij_1term;rhs
_here -= dist[i + 1, j].Value * ui1jterm;rhs_here -= dist[i + surplusj, j + surplusj].Value * u
surplusterm;uij_1term = 0;ui1jterm = 0;usurplusterm = 0;lr_corner++;break;}case BoundaryCase.Bo
dy:{body++;break;}}system_matrix[row, map2Dto1D(i - 1, j - 1, m - 1, n - 1)] = -1 * uijterm;if
(ui1jterm != 0){system_matrix[row, map2Dto1D(i, j - 1, m - 1, n - 1)] = ui1jterm;}if (uij1term
!= 0){system_matrix[row, map2Dto1D(i - 1, j, m - 1, n - 1)] = uij1term;}if (ui_1jterm != 0){sys
tem_matrix[row, map2Dto1D(i - 2, j - 1, m - 1, n - 1)] = ui_1jterm;}if (uij_1term != 0){system_
matrix[row, map2Dto1D(i - 1, j - 2, m - 1, n - 1)] = uij_1term;}if (usurplusterm != 0){system_m
atrix[row, map2Dto1D(i - 1 + surplusi, j - 1 + surplusj, m - 1, n - 1)] = usurplusterm;}RHS[row
] = rhs_here;}}if (console_output){Console.WriteLine("System populated.");}Matrix results = Mat
rix.ext_kaczmarz_radius(system_matrix, RHS, tolerance, maxiterations, radius);for (int i = 0; i
 < total_nodes; i++){int offset = n - 2;int J = i % offset;int I = (i - J) / offset;dist.assign
_value_at(I + 1, J + 1, results[i]);}return dist;}public NCGrid_Distribution Solve(Matrix.Syste
mSolvingScheme scheme){int m = discretization.Xcount;int n = discretization.Ycount;int total_no
des = (m - 2) * (n - 2);NCGrid_Distribution dist = discretization.Clone();dist.ApplyBoundary(bo
undary_conditions);Matrix system_matrix = new Matrix(total_nodes, total_nodes);Matrix RHS = new
 Matrix(total_nodes, 1);int neg_y = 0;int pos_y = 0;int neg_x = 0;int pos_x = 0;int body = 0;in
t ll_corner = 0;int lr_corner = 0;int ur_corner = 0;int ul_corner = 0;if (console_output) { Con
sole.WriteLine("Populating linear system..."); }for (int i = 1; i < m - 1; i++){if (console_out
put && i % (m - 1) / 13 == 0){Console.WriteLine((100 * i / (m - 1)).ToString() + "%");}for (int
 j = 1; j < n - 1; j++){double rhs_here = 0;//for now, assume zero forcing function.BoundaryCas
e _case;bool interior = !isCloseToBoundary(i, j, m, n, out _case);int surplusi = 1;int surplusj
 = 1;Matrix b = dist.GetTaylorSystemCoeffs(i, j, surplusi, surplusj);double uijterm = 0;double
ui1jterm = 0;double uij1term = 0;double ui_1jterm = 0;double uij_1term = 0;double usurplusterm
= 0;int row = (m - 2) * (i - 1) + j - 1;for (int h = 0; h < 5; h++){ui1jterm += b[h, 0] * lin_o
perator[h];uij1term += b[h, 1] * lin_operator[h];ui_1jterm += b[h, 2] * lin_operator[h];uij_1te
rm += b[h, 3] * lin_operator[h];usurplusterm += b[h, 4] * lin_operator[h];double temp_ij = 0;fo
r (int k = 0; k < 5; k++){temp_ij += b[h, k];}uijterm += lin_operator[h] * temp_ij;}switch (_ca
se){case BoundaryCase.NegativeYBoundary:{rhs_here -= dist[i, j - 1].Value * uij_1term;uij_1term
 = 0;neg_y++;break;}case BoundaryCase.PositiveYBoundary:{rhs_here -= dist[i, j + 1].Value * uij
1term;rhs_here -= dist[i + surplusi, j + surplusj].Value * usurplusterm;usurplusterm = 0;uij1te
rm = 0;pos_y++;break;}case BoundaryCase.NegativeXBoundary:{rhs_here -= dist[i - 1, j].Value * u
i_1jterm;ui_1jterm = 0;neg_x++;break;}case BoundaryCase.PositiveXBoundary:{rhs_here -= dist[i +
```

```
 1, j].Value * ui1jterm;rhs_here -= dist[i + surplusi, j + surplusj].Value * usurplusterm;usurp
lusterm = 0;ui1jterm = 0;pos_x++;break;}case BoundaryCase.ULCorner:{rhs_here -= dist[i, j + 1].
Value * uij1term;rhs_here -= dist[i - 1, j].Value * ui_1jterm;rhs_here -= dist[i + surplusi, j
 + surplusj].Value * usurplusterm;uij1term = 0;ui_1jterm = 0;usurplusterm = 0;ul_corner++;break;
}case BoundaryCase.LLCorner:{rhs_here -= dist[i - 1, j].Value * ui_1jterm;rhs_here -= dist[i, j
 - 1].Value * uij_1term;ui_1jterm = 0;uij_1term = 0;ll_corner++;break;}case BoundaryCase.URCorn
er:{rhs_here -= dist[i + 1, j].Value * ui1jterm;rhs_here -= dist[i, j + 1].Value * uij1term;rhs
_here -= dist[i + surplusi, j + surplusj].Value * usurplusterm;ui1jterm = 0;uij1term = 0;usurpl
usterm = 0;ur_corner++;break;}case BoundaryCase.LRCorner:{rhs_here -= dist[i, j - 1].Value * ui
j_1term;rhs_here -= dist[i + 1, j].Value * ui1jterm;rhs_here -= dist[i + surplusj, j + surplusj
].Value * usurplusterm;uij_1term = 0;ui1jterm = 0;usurplusterm = 0;lr_corner++;break;}case Boun
daryCase.Body:{body++;break;}}system_matrix[row, map2Dto1D(i - 1, j - 1, m - 1, n - 1)] = -1 *
uijterm;if (ui1jterm != 0){system_matrix[row, map2Dto1D(i, j - 1, m - 1, n - 1)] = ui1jterm;}if
 (uij1term != 0){system_matrix[row, map2Dto1D(i - 1, j, m - 1, n - 1)] = uij1term;}if (ui_1jter
m != 0){system_matrix[row, map2Dto1D(i - 2, j - 1, m - 1, n - 1)] = ui_1jterm;}if (uij_1term !=
 0){system_matrix[row, map2Dto1D(i - 1, j - 2, m - 1, n - 1)] = uij_1term;}if (usurplusterm !=
0){system_matrix[row, map2Dto1D(i - 1 + surplusi, j - 1 + surplusj, m - 1, n - 1)] = usurpluste
rm;}RHS[row] = rhs_here;}}if (console_output){Console.WriteLine("System populated.");}Matrix re
sults = Matrix.solve_system(system_matrix, RHS, scheme);for (int i = 0; i < total_nodes; i++){i
nt offset = n - 2;int J = i % offset;int I = (i - J) / offset;dist.assign_value_at(I + 1, J + 1
, results[i]);}return dist;}private int maxval(List<int> f){int max = f[0];for (int i = 1; i <
f.Count; i++){if (f[i] > max) { max = f[i]; }}return max;}private void buildsystem(out Matrix s
ystem_matrix, out Matrix RHS){int m = discretization.Xcount;int n = discretization.Ycount;int t
otal_nodes = (m - 2) * (n - 2);NCGrid_Distribution dist = discretization.Clone();dist.ApplyBoun
dary(boundary_conditions);system_matrix = new Matrix(total_nodes, total_nodes);RHS = new Matrix
(total_nodes, 1);int neg_y = 0;int pos_y = 0;int neg_x = 0;int pos_x = 0;int body = 0;int ll_co
rner = 0;int lr_corner = 0;int ur_corner = 0;int ul_corner = 0;if (console_output) { Console.Wr
iteLine("Populating linear system..."); }for (int i = 1; i < m - 1; i++){if (console_output &&
i % (m - 1) / 13 == 0){Console.WriteLine((100 * i / (m - 1)).ToString() + "%");}for (int j = 1;
 j < n - 1; j++){double rhs_here = 0;//for now, assume zero forcing function.BoundaryCase _case
;bool interior = !isCloseToBoundary(i, j, m, n, out _case);int surplusi = 1;int surplusj = 1;Ma
trix b = dist.GetTaylorSystemCoeffs(i, j, surplusi, surplusj);double uijterm = 0;double ui1jter
m = 0;double uij1term = 0;double ui_1jterm = 0;double uij_1term = 0;double usurplusterm = 0;int
 row = (m - 2) * (i - 1) + j - 1;for (int h = 0; h < 5; h++){ui1jterm += b[h, 0] * lin_operator
[h];uij1term += b[h, 1] * lin_operator[h];ui_1jterm += b[h, 2] * lin_operator[h];uij_1term += b
[h, 3] * lin_operator[h];usurplusterm += b[h, 4] * lin_operator[h];double temp_ij = 0;for (int
k = 0; k < 5; k++){temp_ij += b[h, k];}uijterm += lin_operator[h] * temp_ij;}switch (_case){cas
e BoundaryCase.NegativeYBoundary:{rhs_here -= dist[i, j - 1].Value * uij_1term;uij_1term = 0;ne
g_y++;break;}case BoundaryCase.PositiveYBoundary:{rhs_here -= dist[i, j + 1].Value * uij1term;r
hs_here -= dist[i + surplusi, j + surplusj].Value * usurplusterm;usurplusterm = 0;uij1term = 0;
pos_y++;break;}case BoundaryCase.NegativeXBoundary:{rhs_here -= dist[i - 1, j].Value * ui_1jter
m;ui_1jterm = 0;neg_x++;break;}case BoundaryCase.PositiveXBoundary:{rhs_here -= dist[i + 1, j].
Value * ui1jterm;rhs_here -= dist[i + surplusi, j + surplusj].Value * usurplusterm;usurplusterm
 = 0;ui1jterm = 0;pos_x++;break;}case BoundaryCase.ULCorner:{rhs_here -= dist[i, j + 1].Value *
 uij1term;rhs_here -= dist[i - 1, j].Value * ui_1jterm;rhs_here -= dist[i + surplusi, j + surpl
usj].Value * usurplusterm;uij1term = 0;ui_1jterm = 0;usurplusterm = 0;ul_corner++;break;}case B
oundaryCase.LLCorner:{rhs_here -= dist[i - 1, j].Value * ui_1jterm;rhs_here -= dist[i, j - 1].V
alue * uij_1term;ui_1jterm = 0;uij_1term = 0;ll_corner++;break;}case BoundaryCase.URCorner:{rhs
_here -= dist[i + 1, j].Value * ui1jterm;rhs_here -= dist[i, j + 1].Value * uij1term;rhs_here -
= dist[i + surplusi, j + surplusj].Value * usurplusterm;ui1jterm = 0;uij1term = 0;usurplusterm
= 0;ur_corner++;break;}case BoundaryCase.LRCorner:{rhs_here -= dist[i, j - 1].Value * uij_1term
;rhs_here -= dist[i + 1, j].Value * ui1jterm;rhs_here -= dist[i + surplusj, j + surplusj].Value
 * usurplusterm;uij_1term = 0;ui1jterm = 0;usurplusterm = 0;lr_corner++;break;}case BoundaryCas
e.Body:{body++;break;}}system_matrix[row, map2Dto1D(i - 1, j - 1, m - 1, n - 1)] = -1 * uijterm
;if (ui1jterm != 0){system_matrix[row, map2Dto1D(i, j - 1, m - 1, n - 1)] = ui1jterm;}if (uij1t
erm != 0){system_matrix[row, map2Dto1D(i - 1, j, m - 1, n - 1)] = uij1term;}if (ui_1jterm != 0)
```

```
{system_matrix[row, map2Dto1D(i - 2, j - 1, m - 1, n - 1)] = ui_1jterm;}if (uij_1term != 0){sys
tem_matrix[row, map2Dto1D(i - 1, j - 2, m - 1, n - 1)] = uij_1term;}if (usurplusterm != 0){syst
em_matrix[row, map2Dto1D(i - 1 + surplusi, j - 1 + surplusj, m - 1, n - 1)] = usurplusterm;}RHS
[row] = rhs_here;}}if (console_output){Console.WriteLine("System populated.");}}public NCGrid_D
istribution SolveSRDD(){Matrix system_matrix, RHS;NCGrid_Distribution dist = discretization.Clo
ne();dist.ApplyBoundary(boundary_conditions);buildsystem(out system_matrix, out RHS);Matrix res
ults = RunExteriorSolve(system_matrix, RHS);for (int i = 0; i < RHS.Rows; i++){int offset = dis
cretization.Ycount - 2;int J = i % offset;int I = (i - J) / offset;dist.assign_value_at(I + 1,
J + 1, results[i]);}return dist;}private Matrix RunExteriorSolve(Matrix system_matrix, Matrix R
HS){system_matrix.exportToFile(@"C:\Users\Will\Desktop\Folders\MATH435\repo\matrices\sys.csv");
Matrix system = Matrix.ConcatenateRight(system_matrix, RHS);Matrix result = new Matrix(system.R
ows, 1);int m = system.Rows;for (int diagrow = 0; diagrow < m; diagrow++){if (console_output &&
 diagrow % (m/17) == 0){Console.WriteLine(diagrow.ToString() + " of " + m.ToString() + " rows n
ormalized (" + ((100*diagrow)/m).ToString() + "%).");}double scl1 = 1 / system[diagrow, diagrow
];system.RowScale(diagrow, scl1);for (int i = 0; i < m; i++){if (i != diagrow && system[i, diag
row] != 0){double scl2 = -1 * system[i, diagrow];system.RowAdd(i, diagrow, scl2);}}//foreach (i
nt i in targets[diagrow])//{//    double scl2 = -1 * system[i, diagrow];//    system.RowAdd(i,
diagrow, scl2);//}}for (int i = 0; i < m; i++){result[i] = system[i, m];}return result;}public
NCGrid_Distribution solve_iterative_morph(int max_morph_count, double size){NCGrid_Distribution
 soln = Solve();for (int i = 0; i < max_morph_count - 1; i++){soln.QuickSketch("soln-" + i.ToSt
ring());soln.ApplyMeshMorphGA(size);discretization = soln.Clone();soln = Solve();}return soln;}
public NCGrid_Distribution SolveSlow(){int m = discretization.Xcount;int n = discretization.Yco
unt;int total_nodes = (m - 2) * (n - 2);//NCGrid_Distribution dist = new NCGrid_Distribution(bo
undary_conditions.Bounds, discretization.Xcount, discretization.Ycount);NCGrid_Distribution dis
t = discretization.Clone();dist.ApplyBoundary(boundary_conditions);Matrix system_matrix = new M
atrix(total_nodes,total_nodes);Matrix RHS = new Matrix(total_nodes, 1);int neg_y = 0;int pos_y
= 0;int neg_x = 0;int pos_x = 0;int body = 0;int ll_corner = 0;int lr_corner = 0;int ur_corner
= 0;int ul_corner = 0;List<string> indices = new List<string>();if (console_output) { Console.W
riteLine("Populating linear system..."); }for (int i = 1; i < m-1; i++){if (console_output && i
 % (m-1)/13 == 0){Console.WriteLine((100 * i / (m - 1)).ToString() + "%");}for (int j = 1; j <
n-1; j++){double rhs_here = 0;indices.Add(i.ToString() + "," + j.ToString());//for now, assume
zero forcing function.BoundaryCase _case;bool interior = !isCloseToBoundary(i, j, m, n, out _ca
se);int surplusi = 1;int surplusj = 1;Matrix b = dist.GetTaylorSystemCoeffs(i, j, surplusi, sur
plusj);double uijterm = 0;double ui1jterm = 0;double uij1term = 0;double ui_1jterm = 0;double u
ij_1term = 0;double usurplusterm = 0;int row = (m - 2) * (i - 1) + j - 1;for (int h = 0; h < 5;
 h++){ui1jterm += b[h, 0] * lin_operator[h];uij1term += b[h, 1] * lin_operator[h];ui_1jterm +=
b[h, 2] * lin_operator[h];uij_1term += b[h, 3] * lin_operator[h];usurplusterm += b[h, 4] * lin_
operator[h];double temp_ij = 0;for (int k = 0; k < 5; k++){temp_ij += b[h, k];}uijterm += lin_o
perator[h] * temp_ij;}switch (_case){case BoundaryCase.NegativeYBoundary:{rhs_here -= dist[i, j
 - 1].Value * uij_1term;uij_1term = 0;neg_y++;break;}case BoundaryCase.PositiveYBoundary:{rhs_h
ere -= dist[i, j + 1].Value * uij1term;rhs_here -= dist[i + surplusi, j + surplusj].Value * usu
rplusterm;usurplusterm = 0;uij1term = 0;pos_y++;break;}case BoundaryCase.NegativeXBoundary:{rhs
_here -= dist[i - 1, j].Value * ui_1jterm;ui_1jterm = 0;neg_x++;break;}case BoundaryCase.Positi
veXBoundary:{rhs_here -= dist[i + 1, j].Value * ui1jterm;rhs_here -= dist[i + surplusi, j + sur
plusj].Value * usurplusterm;usurplusterm = 0;ui1jterm = 0;pos_x++;break;}case BoundaryCase.ULCo
rner:{rhs_here -= dist[i, j + 1].Value * uij1term;rhs_here -= dist[i - 1, j].Value * ui_1jterm;
rhs_here -= dist[i + surplusi, j + surplusj].Value * usurplusterm;uij1term = 0;ui_1jterm = 0;us
urplusterm = 0;ul_corner++;break;}case BoundaryCase.LLCorner:{rhs_here -= dist[i - 1, j].Value
* ui_1jterm;rhs_here -= dist[i, j - 1].Value * uij_1term;ui_1jterm = 0;uij_1term = 0;ll_corner+
+;break;}case BoundaryCase.URCorner:{rhs_here -= dist[i + 1, j].Value * ui1jterm;rhs_here -= di
st[i, j + 1].Value * uij1term;rhs_here -= dist[i + surplusi, j + surplusj].Value * usurplusterm
;ui1jterm = 0;uij1term = 0;usurplusterm = 0;ur_corner++;break;}case BoundaryCase.LRCorner:{rhs_
here -= dist[i, j - 1].Value * uij_1term;rhs_here -= dist[i + 1, j].Value * ui1jterm;rhs_here -
= dist[i + surplusj, j + surplusj].Value * usurplusterm;uij_1term = 0;ui1jterm = 0;usurplusterm
 = 0;lr_corner++;break;}case BoundaryCase.Body:{body++;break;}}system_matrix[row, map2Dto1D(i -
 1, j - 1, m - 1, n - 1)] = -1*uijterm;if (ui1jterm != 0){system_matrix[row, map2Dto1D(i, j - 1
```

```
, m - 1, n - 1)] = ui1jterm;}if (uij1term != 0){system_matrix[row, map2Dto1D(i-1, j, m - 1, n -
 1)] = uij1term;}if (ui_1jterm != 0){system_matrix[row, map2Dto1D(i-2, j - 1, m - 1, n - 1)] =
ui_1jterm;}if (uij_1term != 0){system_matrix[row, map2Dto1D(i - 1, j - 2, m - 1, n - 1)] = uij_
1term;}if (usurplusterm != 0){system_matrix[row, map2Dto1D(i - 1 + surplusi, j - 1 + surplusj,
m - 1, n - 1)] = usurplusterm;}RHS[row] = rhs_here;}}if (console_output){Console.WriteLine("Sys
tem populated.");}Matrix results = Matrix.solve_system(system_matrix, RHS, Matrix.SystemSolving
Scheme.Basic, true, true);for (int i = 0; i < total_nodes; i++){int offset = n - 2;int J = i %
offset;int I = (i-J)/offset;dist.assign_value_at(I+1,J+1, results[i]);}return dist;}private boo
l isCloseToBoundary(int i, int j, int xcount, int ycount, out BoundaryCase _case){bool onTop =
j == ycount - 2;bool onBottom = j == 1;bool onLeft = i == 1;bool onRight = i == xcount - 2;_cas
e = BoundaryCase.Body;if (onBottom && !onRight && !onLeft) { _case = BoundaryCase.NegativeYBoun
dary; }if (onBottom && onRight) { _case = BoundaryCase.LRCorner; }if (onBottom && onLeft) { _ca
se = BoundaryCase.LLCorner; }if (!onBottom&&!onTop&&onLeft) { _case = BoundaryCase.NegativeXBou
ndary; }if (onTop && !onRight && !onLeft) { _case = BoundaryCase.PositiveYBoundary; }if (onTop
&& onRight) { _case = BoundaryCase.URCorner; }if (onTop && onLeft) { _case = BoundaryCase.ULCor
ner; }if (!onBottom && !onTop && onRight) { _case = BoundaryCase.PositiveXBoundary; }return onT
op || onBottom || onLeft || onRight;}private int map2Dto1D(int i, int j, int icount, int jcount
){return i * (jcount-1) + j;}private int[] map1Dto2D(int q, int icount, int jcount){int j = q &
 jcount;int i = (q - j) / jcount;int[] output = { i, j };return output;}}}/* Copyright (c) 2018
 William van Noordt */using System;using System.Collections.Generic;using System.Linq;using Sys
tem.Text;using System.Threading.Tasks;using System.IO;using System.Drawing;using AForge.Video.V
FW;namespace m_435_NCAMR{class DataSet{protected string absolute_set_location;protected const s
tring video_default_repo = @"C:\Users\Will\Desktop\Folders\MATH435\repo\animations";protected c
onst string default_source = @"C:\Users\Will\Desktop\Folders\MATH435\repo\datasets";protected D
ataSetInfo info;public string AbsoluteDataSetLocation{get { return absolute_set_location; }set
{ absolute_set_location = value; }}protected static string bufferint(int n, int size){string l
= string.Empty;for (int i = 0; i < size + 1 - n.ToString().Length; i++){l += "0";}l += n.ToStri
ng();return l;}public virtual void create_animation(string output_title, bool enable_console_ou
tput, bool using_full_path, bool allow_overwrite, bool use_floating_color_scale, DistributionSk
etchSettings S){string absolute_video_output_path = video_default_repo + "\\" + output_title +
".avi";if (using_full_path) { absolute_video_output_path = output_title; }if (File.Exists(absol
ute_video_output_path)&&!allow_overwrite) { throw new Exception("Error: File exists and overwri
te permission not granted."); }Directory.CreateDirectory(AbsoluteDataSetLocation + "\\images");
string[] allfiles = Directory.GetFiles(absolute_set_location);List<string> distrib_files = new
List<string>();int q = 0;foreach (string i in allfiles){if (i.EndsWith(".dist")){distrib_files.
Add(i);NCGrid_Distribution n = NCGrid_Distribution.from_file(i, true);DistributionSketch2D sk =
 new DistributionSketch2D(n, S);if (!use_floating_color_scale){sk.override_colormap_limits(info
);}sk.CreateSketch(true);sk.SaveImage(AbsoluteDataSetLocation + "\\images\\render" + bufferint(
q++, 5) + ".bmp", true);}}AVIWriter V = new AVIWriter("wmv3");string[] filenames = Directory.Ge
tFiles(AbsoluteDataSetLocation + "\\images");List<string> bmpnames = new List<string>();foreach
 (string i in filenames){if (i.EndsWith(".bmp")) { bmpnames.Add(i); }}if (bmpnames.Count != 0){
Bitmap first = (Bitmap)Bitmap.FromFile(bmpnames[0]);V.Open(absolute_video_output_path, first.Wi
dth, first.Height);V.FrameRate = 50;V.AddFrame(first);int ct = bmpnames.Count;for (int i = 1; i
 < bmpnames.Count; i++){Bitmap butt = (Bitmap)Bitmap.FromFile(bmpnames[i]);if (enable_console_o
utput) { Console.WriteLine(i.ToString() + " of " + ct.ToString() + " frames stacked" + "(" + (i
 * 100 / ct).ToString() + "%)"); }V.AddFrame(butt);butt.Dispose();}V.Close();first.Dispose();if
 (enable_console_output) { Console.WriteLine("AVI successfully created on " + DateTime.Now.ToSt
ring() + " in directory " + absolute_video_output_path); }string[] imagenames = Directory.GetFi
les(AbsoluteDataSetLocation + "\\images");foreach (string g in imagenames){File.Delete(g);}Dire
ctory.Delete(AbsoluteDataSetLocation + "\\images");}else{Console.WriteLine("No bitmap images fo
und in the current directory.");}}public DataSet(string title, bool using_full_path, bool enabl
e_console_output){absolute_set_location = default_source + "\\" + title;if (using_full_path) {
absolute_set_location = title; }info = new DataSetInfo(Directory.GetFiles(absolute_set_location
), enable_console_output);}public static DataSet FirstAvailable(bool enable_console_output){str
ing[] allnames = Directory.GetDirectories(default_source);if (allnames.Length != 0){return new
DataSet(allnames[0], true, enable_console_output);}else{throw new Exception("No datasets availa
```

```
ble.");}}}}/* Copyright (c) 2018 William van Noordt */using System;using System.Collections.Gen
eric;using System.Linq;using System.Text;using System.Threading.Tasks;using System.IO.Compressi
on;using System.IO;namespace m_435_NCAMR{static class DataSetBuilder{private static string info
_file_tite = "info.inf";private static string default_repo = @"C:\Users\Will\Desktop\Folders\MA
TH435\repo\datasets";private static string default_source = @"C:\Users\Will\Desktop\Folders\MAT
H435\repo\distributions";public static void build_from_default_repo(string output_title, bool u
sing_full_path, bool enable_console_output){string absolute_source_path = default_source;string
 absolute_output_path = default_repo + "\\" + output_title;if (using_full_path) { absolute_sour
ce_path = output_title; }string[] all_absolute_names = Directory.GetFiles(absolute_source_path)
;//will need interpolation/regression code.DataSetInfo info = new DataSetInfo(all_absolute_name
s, enable_console_output);Directory.CreateDirectory(absolute_output_path);int n = 0;foreach(str
ing i in all_absolute_names){if (enable_console_output) { Console.WriteLine("Processing "+ (++n
).ToString() + " of " + all_absolute_names.Length.ToString() + " files." ); }File.Copy(i, absol
ute_output_path + "\\" + rip_file_title(i));File.Delete(i);}info.write_to_info_file(absolute_ou
tput_path + "\\" + info_file_tite);}private static string rip_file_title(string fullpath){retur
n fullpath.Split('\\').Last();}public static void build_by_keyword(string keyword, string path)
{}}}/* Copyright (c) 2018 William van Noordt */using System;using System.Collections.Generic;us
ing System.Linq;using System.Text;using System.Threading.Tasks;using System.IO;namespace m_435_
NCAMR{class DataSetInfo{private int distrib_count;private double universal_max, universal_min;p
ublic int DistributionCount{get { return distrib_count; }}public double UniversalMax{get { retu
rn universal_max; }}public double UniversalMin{get { return universal_min; }}public DataSetInfo
(string[] fullfilenames, bool enable_console_output){DateTime then = DateTime.Now;universal_min
 = double.MaxValue;universal_max = double.MinValue;List<string> all_dist = new List<string>();f
oreach (string p in fullfilenames){if (p.EndsWith(".dist")){all_dist.Add(p);}}distrib_count = a
ll_dist.Count;for (int i = 0; i < distrib_count; i++){using (StreamReader r = new StreamReader(
all_dist[i])){string first_line = r.ReadLine();string[] spt = first_line.Split(':');double temp
max, tempmin;if (!(double.TryParse(spt[6], out tempmax)&& double.TryParse(spt[8], out tempmin))
) { throw new Exception("File improperly formatted."); }if (tempmax > universal_max) { universa
l_max = tempmax; }if (tempmin < universal_min) { universal_min = tempmin; }}}DateTime now = Dat
eTime.Now;if (enable_console_output){Console.WriteLine("DataSetInfo generated in " + (now - the
n).Milliseconds.ToString() + " milliseconds");Console.WriteLine(this.ToString());}}public overr
ide string ToString(){//incomplete: will need interpolation coefficientsreturn "dist_count:" +
distrib_count + ":max_value:" + universal_max.ToString() + ":min_value:" + universal_min.ToStri
ng();}public void write_to_info_file(string fullpath){string[] contents = { this.ToString() };F
ile.WriteAllLines(fullpath, contents);}}}/* Copyright (c) 2018 William van Noordt */using Syste
m;using System.Collections.Generic;using System.Linq;using System.Text;using System.Threading.T
asks;using System.Drawing;using ColorMapping;namespace m_435_NCAMR{enum SketchMode{EMPTY,GRID_D
ISTRIBUTION,GRID_ONLY,DISTRIBUTION_ONLY}class DistributionSketch2D{protected const string defau
lt_repo = @"C:\Users\Will\Desktop\Folders\MATH435\repo\images";protected DistributionSketchSett
ings settings;protected float override_max;protected float override_min;protected bool use_over
riden_limits = false;//parametersprotected float MAIN_SUBJECT_LOWER_LEFT_X_2D;protected float M
AIN_SUBJECT_LOWER_LEFT_Y_2D;protected float MAIN_SUBJECT_UPPER_RIGHT_X_2D;protected float MAIN_
SUBJECT_UPPER_RIGHT_Y_2D;//constantsprotected const float FIGURETITLE_Y = 0.04f;protected const
 float FIGURETITLE_X = 0.1f;protected const float SQUARE_BORDER_THIN = 0.07f;protected const fl
oat SQUARE_BORDER_THICK = 0.130f;protected const float GRIDLINE_THICKNESS = 2.3f;protected cons
t float X_AXIS_TITLE_X = 0.5f;protected const float Y_AXIS_TITLE_Y = 0.5f;protected float Y_AXI
S_TITLE_X = 0.31f * SQUARE_BORDER_THICK;protected const float X_AXIS_TITLE_Y = 0.5f * SQUARE_BO
RDER_THICK;protected const float LABEL_OFFSET_Y = -0.056f;protected const float HEATBOX_BORDER_
LEFT = 0.84f;protected const float HEATBOX_BORDER_RIGHT = 0.97f;protected static float HEATBOX_
BORDER_BOTTOM = 0.35f;protected float HEATBOX_BORDER_TOP = 1 - HEATBOX_BORDER_BOTTOM;protected
 const float HEATBOX_LL_X = 0.85f;protected float HEATBOX_UR_X = 0.96f;protected static float HE
ATBOX_LL_Y = 0.36f;protected float HEATBOX_UR_Y = 1 - HEATBOX_LL_Y;protected float HEATMAP_LABE
LS_X = 0.87f;protected static float HEATMAP_LABELS_YU = 0.33f;protected static float HEATMAP_LA
BELS_YL = 0.64f;protected float X_LABEL_BASE_Y = 0.94f * SQUARE_BORDER_THICK;protected float X_
LABEL_BASE_X = 1.0f * SQUARE_BORDER_THICK;protected float Y_LABEL_BASE_Y = 1.0f * SQUARE_BORDER
_THICK;protected float Y_LABEL_BASE_X = 0.54f * SQUARE_BORDER_THICK;protected float VALUE_TEXT_
```

```
PROPORTION = 40f / 2000f;protected float TITLE_TEXT_PROPORTION = 65f / 2000f;protected const fl
oat THIN_PEN = 1.5f;protected const float THICK_PEN = 2.4f;protected Pen THICK_GRID_PEN = new P
en(Color.Black, THICK_PEN);protected Pen THIN_GRID_PEN = new Pen(Color.Black, THIN_PEN);protect
ed Color BACKGROUND_COLOR = Color.White;protected Color TEXTCOLOR = Color.Black;//calculated re
lativesprotected float grid_xmin;protected float grid_xmax;protected float grid_ymin;protected
float grid_ymax;protected float pix_per_unit_x;protected float pix_per_unit_y;protected Bitmap
canvas;protected int x_pixel_count, y_pixel_count;protected NCGrid_Distribution subject;public
SketchMode Sketch_Mode{get { return settings.Mode; }set { settings.Mode = value; }}public Bitma
p Canvas{get { return canvas; }}public DistributionSketch2D(NCGrid_Distribution _subject, Distr
ibutionSketchSettings S){settings = S;subject = _subject;canvas = new Bitmap(S.ImageWidth, S.Im
ageHeight);x_pixel_count = canvas.Width;y_pixel_count = canvas.Height;get_positions();}public v
irtual void CreateSketch(bool allow_console_output){using ( Graphics drawer = Graphics.FromImag
e(canvas)){drawer.Clear(BACKGROUND_COLOR);bool no_dt = false;DateTime then = DateTime.Now;switc
h (settings.Mode){case SketchMode.EMPTY: { if (allow_console_output) { Console.WriteLine("Warni
ng: Sketchmode set to \"EMPTY\". No drawing operations executed."); no_dt = true; } break; }cas
e SketchMode.GRID_ONLY:{draw_grid(drawer);break;}case SketchMode.DISTRIBUTION_ONLY:{draw_distri
b(drawer);break;}case SketchMode.GRID_DISTRIBUTION:{draw_distrib(drawer);draw_grid(drawer);brea
k;}}if (settings.HasAxisValues) { draw_axis_values(drawer); }if (settings.HasAxisTitles) { draw
_axis_titles(drawer); }if (settings.HasGridlines) { draw_gridlines(drawer); }if (settings.HasHe
atmap) { draw_heatmap_box(drawer); }DateTime now = DateTime.Now;draw_fig_title(drawer);if (allo
w_console_output && !no_dt) { Console.WriteLine("Sketch generated in " + (now - then).Milliseco
nds.ToString() + " milliseconds."); }}}protected void get_positions(){if (settings.HasAxisValue
s || settings.HasAxisTitles){MAIN_SUBJECT_LOWER_LEFT_X_2D = SQUARE_BORDER_THICK;MAIN_SUBJECT_LO
WER_LEFT_Y_2D = SQUARE_BORDER_THICK;MAIN_SUBJECT_UPPER_RIGHT_X_2D = 1 - SQUARE_BORDER_THICK;MAI
N_SUBJECT_UPPER_RIGHT_Y_2D = 1 - SQUARE_BORDER_THICK;}else{MAIN_SUBJECT_LOWER_LEFT_X_2D = SQUAR
E_BORDER_THIN;MAIN_SUBJECT_LOWER_LEFT_Y_2D = SQUARE_BORDER_THIN;MAIN_SUBJECT_UPPER_RIGHT_X_2D =
 1 - SQUARE_BORDER_THIN;MAIN_SUBJECT_UPPER_RIGHT_Y_2D = 1 - SQUARE_BORDER_THIN;}grid_xmin = MAI
N_SUBJECT_LOWER_LEFT_X_2D * (float)x_pixel_count;grid_xmax = MAIN_SUBJECT_UPPER_RIGHT_X_2D * (f
loat)x_pixel_count;grid_ymin = MAIN_SUBJECT_LOWER_LEFT_Y_2D * (float)y_pixel_count;grid_ymax =
MAIN_SUBJECT_UPPER_RIGHT_Y_2D * (float)y_pixel_count;pix_per_unit_x = (grid_xmax - grid_xmin) /
 (float)(subject.Xmax - subject.Xmin);pix_per_unit_y = (grid_ymax - grid_ymin) / (float)(subjec
t.Ymax - subject.Ymin);}protected void draw_axis_values(Graphics drawer){Brush textbrush = new
SolidBrush(TEXTCOLOR);float xlabel_abs_pos_x = ((float)x_pixel_count * X_LABEL_BASE_X);float xl
abel_abs_increment = (((1 - 2 * (Y_LABEL_BASE_Y)) * (float)x_pixel_count) / (float)(settings.XL
abelCount-1));float ylabel_abs_pos_y = ((float)y_pixel_count * (Y_LABEL_BASE_X));float ylabel_a
bs_increment = (((1 - 2 * Y_LABEL_BASE_Y) * (float)y_pixel_count) / (float)(settings.YLabelCoun
t-1));float xlabel_abs_pos_y = (X_LABEL_BASE_Y * (float)x_pixel_count);float ylabel_abs_pos_x =
 ((Y_LABEL_BASE_X-LABEL_OFFSET_Y) * (float)y_pixel_count);double delta_x = (subject.Xmax - subj
ect.Xmin) / (settings.XLabelCount - 1);double delta_y = (subject.Ymax - subject.Ymin) / (settin
gs.YLabelCount - 1);for (int i = 0; i < settings.XLabelCount; i++){drawer.DrawString(truncate_s
tring((subject.Xmin + i * delta_x).ToString(),4), new Font("arial", VALUE_TEXT_PROPORTION * set
tings.ImageHeight), textbrush, new PointF(xlabel_abs_pos_x + i * xlabel_abs_increment, settings
.ImageHeight - xlabel_abs_pos_y));}for (int j = 0;  j< settings.YLabelCount; j++){drawer.DrawSt
ring(truncate_string((subject.Ymax - j * delta_y).ToString(),4), new Font("arial", VALUE_TEXT_P
ROPORTION * settings.ImageHeight), textbrush, new PointF(ylabel_abs_pos_y, ylabel_abs_pos_x + j
*ylabel_abs_increment));}}protected void draw_heatmap_box(Graphics drawer){float abspos_hb_ll_x
 = HEATBOX_LL_X * settings.ImageWidth;float abspos_hb_ll_y = HEATBOX_LL_Y * settings.ImageHeigh
t;float abspos_hb_ur_x = HEATBOX_UR_X * settings.ImageWidth;float abspos_hb_ur_y = HEATBOX_UR_Y
 * settings.ImageHeight;float abspos_border_upper = HEATBOX_BORDER_TOP * settings.ImageHeight;f
loat abspos_border_lower = HEATBOX_BORDER_BOTTOM * settings.ImageHeight;float abspos_border_lef
t = HEATBOX_BORDER_LEFT * settings.ImageWidth;float abspos_border_right = HEATBOX_BORDER_RIGHT
* settings.ImageWidth;PointF[] border ={new PointF(abspos_border_left, abspos_border_upper),new
 PointF(abspos_border_left, abspos_border_lower),new PointF(abspos_border_right, abspos_border_
lower),new PointF(abspos_border_right, abspos_border_upper)};Brush borderbrush = new SolidBrush
(Color.White);drawer.FillPolygon(borderbrush, border);int heatmap_pixel_count = (int)Math.Abs((
abspos_hb_ll_y - abspos_hb_ur_y));for (int i = 0; i <= heatmap_pixel_count; i++){Pen drawingpen
```

```
;if (settings.UseCustomColorMap) { drawingpen = new Pen(ColorMap.MapColorMultiple(settings.Colo
rMap, (double)i, 0, (double)heatmap_pixel_count),1.0f); }else { drawingpen = new Pen(ColorMap.M
apRainbowColor((float)i, (float)heatmap_pixel_count, 0),1.0f); }PointF left = new PointF(abspos
_hb_ll_x, abspos_hb_ur_y - i);PointF right = new PointF(abspos_hb_ur_x, abspos_hb_ur_y - i);dra
wer.DrawLine(drawingpen, left, right);}float abs_upper_label_y = HEATMAP_LABELS_YU * settings.I
mageHeight;float abs_lower_label_y = HEATMAP_LABELS_YL * settings.ImageHeight;float abs_label_x
 = HEATMAP_LABELS_X * settings.ImageWidth;string upper_label, lower_label;if (use_overriden_lim
its){upper_label = truncate_string(override_max.ToString(),4);lower_label = truncate_string(ove
rride_min.ToString(), 4);}else{upper_label = truncate_string(subject.MaxValue.ToString(), 4);lo
wer_label = truncate_string(subject.MinValue.ToString(), 4);}Brush textbrush = new SolidBrush(T
EXTCOLOR);drawer.DrawString(upper_label, new Font("arial", VALUE_TEXT_PROPORTION * settings.Ima
geHeight), textbrush, new PointF(abs_label_x, abs_upper_label_y));drawer.DrawString(lower_label
, new Font("arial", VALUE_TEXT_PROPORTION * settings.ImageHeight), textbrush, new PointF(abs_la
bel_x, abs_lower_label_y));}protected void draw_gridlines(Graphics drawer){Pen gridlinepen = ne
w Pen(Color.FromArgb(130, 0, 0, 0), GRIDLINE_THICKNESS);NCAMRNode[,] gridpoints = new NCAMRNode
[settings.XGridlineCount, settings.YGridlineCount];double coordinate_dx = (subject.Xmax - subje
ct.Xmin) / (settings.XGridlineCount - 1);double coordinate_dy = (subject.Ymax - subject.Ymin) /
 (settings.YGridlineCount - 1);for (int i = 0; i < settings.XGridlineCount; i++){for (int j = 0
; j < settings.YGridlineCount; j++){gridpoints[i, j] = new NCAMRNode(subject.Xmin + i * coordin
ate_dx, subject.Ymin + j * coordinate_dy, 0);}}for (int i = 0; i < settings.XGridlineCount; i++
){for (int j = 0; j < settings.YGridlineCount - 1; j++){//thick grid pen temporarydrawer.DrawLi
ne(gridlinepen, mapGridNode(gridpoints[i, j]), mapGridNode(gridpoints[i, j + 1]));}}for (int i
= 0; i < settings.YGridlineCount; i++){for (int j = 0; j < settings.XGridlineCount - 1; j++){//
thick grid pen temporarydrawer.DrawLine(gridlinepen, mapGridNode(gridpoints[j+1, i]), mapGridNo
de(gridpoints[j, i]));}}}protected void draw_axis_titles(Graphics drawer){float x_title_abs_pos
_x = X_AXIS_TITLE_X * settings.ImageWidth;float x_title_abs_pos_y = X_AXIS_TITLE_Y * settings.I
mageWidth;float y_title_abs_pos_x = Y_AXIS_TITLE_X * settings.ImageHeight;float y_title_abs_pos
_y = Y_AXIS_TITLE_Y * settings.ImageHeight;Brush textbrush = new SolidBrush(TEXTCOLOR);drawer.D
rawString(settings.HorizontalTitle, new Font("arial", TITLE_TEXT_PROPORTION * settings.ImageHei
ght), textbrush, new PointF(x_title_abs_pos_x, settings.ImageHeight - x_title_abs_pos_y));drawe
r.DrawString(settings.VerticalTitle, new Font("arial", TITLE_TEXT_PROPORTION * settings.ImageHe
ight), textbrush, new PointF(y_title_abs_pos_x, y_title_abs_pos_y));}protected void draw_grid(G
raphics drawer){for (int i = 0; i < subject.Xcount - 1; i++){drawer.DrawLine(THICK_GRID_PEN, ma
pGridNode(subject[i, subject.Ycount - 1]), mapGridNode(subject[i + 1, subject.Ycount - 1]));dra
wer.DrawLine(THICK_GRID_PEN, mapGridNode(subject[i, 0]), mapGridNode(subject[i + 1, 0]));}for (
int j = 0; j < subject.Ycount - 1; j++){drawer.DrawLine(THICK_GRID_PEN, mapGridNode(subject[sub
ject.Xcount - 1, j]), mapGridNode(subject[subject.Xcount - 1, j + 1]));drawer.DrawLine(THICK_GR
ID_PEN, mapGridNode(subject[0, j]), mapGridNode(subject[0, j + 1]));}for (int i = 0; i < subjec
t.Xcount - 1; i++){for (int j = 0; j < subject.Xcount - 1; j++){drawer.DrawLine(THIN_GRID_PEN,
mapGridNode(subject[i, j]), mapGridNode(subject[i + 1, j]));drawer.DrawLine(THIN_GRID_PEN, mapG
ridNode(subject[i, j]), mapGridNode(subject[i, j+1]));}}}public void update_label_y_pos(double
d){Y_AXIS_TITLE_X = (float)d;}protected void draw_distrib(Graphics drawer){double min = subject
.MinValue;double max = subject.MaxValue;if (use_overriden_limits){max = override_max;min = over
ride_min;}for (int i = 0; i < subject.Xcount - 1; i++){for (int j = 0; j < subject.Ycount - 1;
j++){NCAMRNode[] nodes = { subject[i, j], subject[i+1, j], subject[i+1, j+1], subject[i, j+1] }
;double zavg = 0.25d * (nodes[0].Value+ nodes[1].Value+ nodes[2].Value+ nodes[3].Value);if (Mat
h.Abs(max - min) < 0.001d){max = 1;min = 0;zavg = 0;}SolidBrush B;if (settings.UseCustomColorMa
p){B = new SolidBrush(ColorMap.MapColorMultiple(settings.ColorMap, zavg, min, max));}else{B = n
ew SolidBrush(ColorMap.MapRainbowColor((float)zavg, (float)max, (float)min));}PointF[] pts = ne
w PointF[4];for (int k = 0; k < 4; k++){pts[k] = mapGridNode(nodes[k]);}drawer.FillPolygon(B, p
ts);}}}protected PointF mapGridNode(NCAMRNode toMap){return new PointF(grid_xmin + pix_per_unit
_x * (float)(toMap.X - subject.Xmin), grid_ymin + pix_per_unit_y * (float)((subject.Ymax - toMa
p.Y)));}public void SaveImage(string title, bool using_full_path){string path = default_repo +
"\\" + title + ".bmp";if (using_full_path) { path = title; }canvas.Save(path);}public void over
ride_colormap_limits(float new_min, float new_max){use_overriden_limits = true;override_max = n
ew_max;override_min = new_min;}public void override_colormap_limits(DataSetInfo D){use_override
```

```
n_limits = true;override_max = (float)D.UniversalMax;override_min = (float)D.UniversalMin;}publ
ic void reset_colormap_limits(){use_overriden_limits = false;}public void SetSketchSettings(Dis
tributionSketchSettings S){settings = S;get_positions();}protected void draw_fig_title(Graphics
 drawer){float absx = settings.ImageWidth * FIGURETITLE_X;float absy = settings.ImageHeight * F
IGURETITLE_Y;Brush textbrush = new SolidBrush(TEXTCOLOR);drawer.DrawString(settings.FigureTitle
, new Font("arial", 1.9f*VALUE_TEXT_PROPORTION * settings.ImageHeight), textbrush, new PointF(a
bsx, absy));}protected string truncate_string(string input, int number){string supplementary =
string.Empty;if (input.Contains("E")){int idx = input.LastIndexOf('E');for (int i = idx; i < in
put.Length; i++){supplementary += input[i];}}if (input.Length <= number) return input;else{stri
ng output = string.Empty;for (int i = 0; i < number;  i++){output += input[i];}return output+su
pplementary;}}}}/* Copyright (c) 2018 William van Noordt */using System;using System.Collection
s.Generic;using System.Linq;using System.Text;using System.Threading.Tasks;using System.Drawing
;using _3DSimple;using ColorMapping;namespace m_435_NCAMR{class DistributionSketch3DBasic{prote
cted const string default_repo = @"C:\Users\Will\Desktop\Folders\MATH435\repo\images";private N
CGrid_Distribution subject;Bitmap output;public NCGrid_Distribution Subject{get { return subjec
t; }set { subject = value; }}public DistributionSketch3DBasic(NCGrid_Distribution sub){subject
= sub;output = new Bitmap(2000, 2000);}public void SaveImage(string title, bool using_full_path
){string path = default_repo + "\\" + title + ".bmp";if (using_full_path) { path = title; }outp
ut.Save(path);}public void GenerateSketch(){using (var g = Graphics.FromImage(output)){Point og
 = new Point(output.Width / 2, output.Height / 2);Point3 light = new Point3(subject.Bounds.Xmin
+0.5*(subject.Bounds.Xmax-subject.Bounds.Xmin), subject.Bounds.Ymin + 0.5 * (subject.Bounds.Yma
x - subject.Bounds.Ymin), 1.5*subject.MaxValue);double camZ = 45;double camXY = 45;double luste
r = 0.9;double zoom = 100;double dx = (subject.Bounds.Xmax - subject.Bounds.Xmin) / (subject.Xc
ount - 1);double dy = (subject.Bounds.Ymax - subject.Bounds.Ymin) / (subject.Ycount - 1);g.Clea
r(Color.DarkGray);for (int i = 0; i < subject.Xcount-1; i++){for (int j = 0; j < subject.Ycount
-1; j++){double k = Math.PI / 180;Point3 P00 = subject[i, j].toPoint();Point3 P01 = subject[i,
j + 1].toPoint();Point3 P10 = subject[i + 1, j].toPoint();Point3 P11 = subject[i + 1, j + 1].to
Point();Vector3 V1 = new Vector3(P00, P11);Vector3 V2 = new Vector3(P10, P01);float cosZ = (flo
at)Math.Cos(camZ * k);float cosXY = (float)Math.Cos(camXY * k);float sinZ = (float)Math.Sin(cam
Z * k);float sinXY = (float)Math.Sin(camXY * k);Vector3 vw = new Vector3(cosZ * cosXY, cosZ * s
inXY, sinZ);Vector3 Normal = V1 % V2;double zavg = 0.25 * (P00.Z + P10.Z + P01.Z + P11.Z);Point
 a00 = transform(P00, camZ, camXY, zoom, og);Point a01 = transform(P01, camZ, camXY, zoom, og);
Point a10 = transform(P10, camZ, camXY, zoom, og);Point a11 = transform(P11, camZ, camXY, zoom,
 og);Point[] pts = new Point[4];pts[0] = a00;pts[1] = a01;pts[2] = a11;pts[3] = a10;Vector3 lit
eVector = new Vector3(P00, light);double ux2 = ((P10.Z - P00.Z) / dx) * ((P10.Z - P00.Z) / dx);
double uy2 = ((P01.Z - P00.Z) / dy) * ((P01.Z - P00.Z) / dy);double strain = 0.5 * ((Math.Sqrt(
1 + ux2) - 1) + (Math.Sqrt(1 + uy2) - 1));SolidBrush B = new SolidBrush(ShadeSingle(Color.Red,
liteVector, Normal, camXY, camZ, luster));g.FillPolygon(B, pts);}}g.Dispose();}}Point transform
(Point3 objectLocation, double cameraZ, double cameraXY, double zoom, Point form_origin){camera
XY = 90 - cameraXY;double conv = Math.PI / 180;double cos_z = Math.Cos(cameraZ * conv);double s
in_z = Math.Sin(cameraZ * conv);double cos_xy = Math.Cos(cameraXY * conv);double sin_xy = Math.
Sin(cameraXY * conv);int px = (int)(((objectLocation.Y * sin_xy) - (objectLocation.X * cos_xy))
 * zoom);int py = (int)(((objectLocation.Z * cos_z) - (objectLocation.X * sin_xy * sin_z) - (ob
jectLocation.Y * sin_z * cos_xy)) * zoom);return new Point(form_origin.X + px, form_origin.Y -
py);}private Color ShadeSingle(Color C, Vector3 light, Vector3 Normal, double camXY, double cam
Z, double luster){double lus_thresh = luster * 0.1;if (lus_thresh > 1) { lus_thresh = 1; }Vecto
r3 N = Normal.unit();if (N.K < 0) { N = -1 * N; }Vector3 L = light.unit();double k = Math.PI /
180;float cosZ = (float)Math.Cos(camZ * k);float cosXY = (float)Math.Cos(camXY * k);float sinZ
= (float)Math.Sin(camZ * k);float sinXY = (float)Math.Sin(camXY * k);Vector3 vw = new Vector3(c
osZ * cosXY, cosZ * sinXY, sinZ);double mu = ((2 * (float)(L * N) * N) - L) * vw;if (mu < lus_t
hresh) { return ColorMap.MapGradient(Color.FromArgb(255, 10, 10, 16), C, mu, -1, lus_thresh); }
int r = (int)(lus_thresh * 255);return ColorMap.MapGradient(C, Color.FromArgb(255, r, r, r), mu
, lus_thresh, 1);}}}/* Copyright (c) 2018 William van Noordt */using System;using System.Collec
tions.Generic;using System.Linq;using System.Text;using System.Threading.Tasks;using System.Dra
wing;namespace m_435_NCAMR{class DistributionSketchSettings{private string fig_title = string.E
mpty;private SketchMode _mode;private const int DEFAULTIMAGESIZE = 2000;private const int DEFAU
```

```
LTLABELCOUNT = 5;private bool has_axis_values, has_info, has_axis_titles, use_special_colormap,
 has_heatmap, has_cartesian_gridlines;private int x_cart_gridlines, y_cart_gridlines;private in
t x_label_count, y_label_count, image_width, image_height;private List<Color> special_map;priva
te string horizontal_title, vertical_title;public string FigureTitle{get { return fig_title; }}
public bool HasGridlines{get { return has_cartesian_gridlines; }set { has_cartesian_gridlines =
 value; }}public int XGridlineCount{get { return x_cart_gridlines; }set { x_cart_gridlines = va
lue; }}public int YGridlineCount{get { return y_cart_gridlines; }set { y_cart_gridlines = value
; }}public string HorizontalTitle{get { return horizontal_title; }set { horizontal_title = valu
e; }}public string VerticalTitle{get { return vertical_title; }set { vertical_title = value; }}
public List<Color> ColorMap{get { return special_map; }}public int ImageWidth{get { return imag
e_width; }set { image_width = value; }}public int ImageHeight{get { return image_height; }set {
 image_height = value; }}public SketchMode Mode{get { return _mode; }set { _mode = value; }}pub
lic bool HasAxisValues{get { return has_axis_values; }set { has_axis_values = value; }}public b
ool HasHeatmap{get { return has_heatmap; }set { has_heatmap = value; }}public bool HasInfo{get
 { return has_info; }set { has_info = value; }}public bool HasAxisTitles{get { return has_axis_t
itles; }set { has_axis_titles = value; }}public bool UseCustomColorMap{get { return use_special
_colormap; }}public int XLabelCount{get { return x_label_count; }set { x_label_count = value; }
}public int YLabelCount{get { return y_label_count; }set { y_label_count = value; }}public Dist
ributionSketchSettings(SketchMode skmode, int imagesize){_mode = skmode;has_axis_values = false
;has_axis_titles = false;has_heatmap = false;use_special_colormap = false;has_info = false;has_
cartesian_gridlines = false;x_cart_gridlines = 10;y_cart_gridlines = 10;x_label_count = 5;y_lab
el_count = 5;image_height = imagesize;image_width = imagesize;horizontal_title = "x";vertical_t
itle = "y";}public DistributionSketchSettings(SketchMode skmode, bool hasaxisvalues, bool hasax
istitles, bool hasheatmap, bool hasinfo, bool hascartesians, int xgridcount, int ygridcount, in
t xlabelcount, int ylabelcount, int imagesize){_mode = skmode;has_axis_values = hasaxisvalues;h
as_axis_titles = hasaxistitles;has_heatmap = hasheatmap;use_special_colormap = false;has_info =
 hasinfo;has_cartesian_gridlines = false;x_cart_gridlines = xgridcount;y_cart_gridlines = ygrid
count;x_label_count = xlabelcount;y_label_count = ylabelcount;image_height = imagesize;image_wi
dth = imagesize;horizontal_title = "x";vertical_title = "y";}public void SetColormap(List<Color
> colors){use_special_colormap = true;special_map = colors;}public void revert_simple(){has_axi
s_titles = false;has_axis_values = false;has_cartesian_gridlines = false;has_heatmap = false;ha
s_info = false;}public void SetColormap(DefaultColorMaps t){use_special_colormap = true;List<Co
lor> the_colors = new List<Color>();switch(t){case DefaultColorMaps.AMERICAN:{the_colors.Add(Co
lor.Blue);the_colors.Add(Color.LightBlue);the_colors.Add(Color.White);the_colors.Add(Color.Pink
);the_colors.Add(Color.Red);break;}case DefaultColorMaps.RWBB:{the_colors.Add(Color.Blue);the_c
olors.Add(Color.DarkBlue);the_colors.Add(Color.Black);the_colors.Add(Color.Gray);the_colors.Add
(Color.LightGray);the_colors.Add(Color.White);the_colors.Add(Color.LightPink);the_colors.Add(Co
lor.Pink);the_colors.Add(Color.Red);break;}case DefaultColorMaps.GREYSCALE:{the_colors.Add(Colo
r.Black);the_colors.Add(Color.DarkGray);the_colors.Add(Color.Gray);the_colors.Add(Color.LightGr
ay);the_colors.Add(Color.White);break;}case DefaultColorMaps.RED:{the_colors.Add(Color.Red);the
_colors.Add(Color.OrangeRed);the_colors.Add(Color.Orange);the_colors.Add(Color.Yellow);the_colo
rs.Add(Color.LightYellow);break;}case DefaultColorMaps.REV_AMERICAN:{the_colors.Add(Color.Red);
the_colors.Add(Color.Pink);the_colors.Add(Color.White);the_colors.Add(Color.LightBlue);the_colo
rs.Add(Color.Blue);break;}}special_map = the_colors;}public void RevokeColorMap(){use_special_c
olormap = false;}public enum DefaultColorMaps{AMERICAN, RWBB, GREYSCALE, RED,  REV_AMERICAN}pub
lic void SetFigureTitle(string figtitle){fig_title = figtitle;}public static DistributionSketch
Settings Basic(){return new DistributionSketchSettings(SketchMode.DISTRIBUTION_ONLY, false, fal
se, false, false, false, DEFAULTLABELCOUNT, DEFAULTLABELCOUNT, 10, 10, DEFAULTIMAGESIZE);}publi
c static DistributionSketchSettings Fancy(){return new DistributionSketchSettings(SketchMode.GR
ID_DISTRIBUTION, true, true, true, true, true, DEFAULTLABELCOUNT, DEFAULTLABELCOUNT, 10, 10, DE
FAULTIMAGESIZE);}}}/* Copyright (c) 2018 William van Noordt */using System;using System.Collect
ions.Generic;using System.Linq;using System.Text;using System.Threading.Tasks;using System.IO;u
sing _3DSimple;namespace m_435_NCAMR{static class ErrorEstimation{public static NCGrid_Distribu
tion CompareNiceClean(NCGrid_Distribution test, string vbs_test_function){string title = "temp"
;string path = Paths.DistributionRepo + "\\" + title + ".dist";ExactFunctionGeneratorVB2D.Gener
ateFunction(vbs_test_function, test, title);NCGrid_Distribution true_dist = NCGrid_Distribution
```

```
.from_file(title, false);File.Delete(path);NCGrid_Distribution dif = test - true_dist;dif.force
_extrema_update();return dif;}public static double NormDifference(NCGrid_Distribution A, NCGrid
_Distribution B){NCGrid_Distribution newgrid = new NCGrid_Distribution(A.Bounds, A.Xcount+1, A.
Ycount+1);int n = newgrid.Ycount;int m = newgrid.Xcount;double xmin = newgrid.Xmin;double xmax
= newgrid.Xmax;double ymin = newgrid.Ymin;double ymax = newgrid.Ymax;NCAMRNode ll = new NCAMRNo
de(xmin, ymin, 0);NCAMRNode lr = new NCAMRNode(xmax, ymin, 0);NCAMRNode ul = new NCAMRNode(xmin
, ymax, 0);NCAMRNode ur = new NCAMRNode(xmax, ymax, 0);newgrid[0, 0] = ll;newgrid[0, n-1] = ul;
newgrid[m-1, 0] = lr;newgrid[m-1, n-1] = ur;for (int i = 0; i < A.Xcount-1; i++){newgrid[i + 1,
 0] = xyavg(A[i, 0], A[i + 1, 0]);newgrid[i + 1, A.Ycount] = xyavg(A[i, A.Ycount - 1], A[i + 1,
 A.Ycount - 1]);}for (int j = 0; j < A.Ycount-1; j++){newgrid[0, j + 1] = xyavg(A[0, j], A[0, j
 + 1]);newgrid[A.Xcount, j + 1] = xyavg(A[A.Xcount - 1, j], A[A.Xcount - 1, j + 1]);}for (int i
 = 0; i < A.Xcount-1; i++){for (int j = 0; j < A.Ycount-1; j++){newgrid[i + 1, j + 1] = xyavg(A
[i, j], A[i + 1, j], A[i, j + 1], A[i + 1, j + 1]);}}double accumulator = 0;for (int i = 0; i <
 m-1; i++){for (int j = 0; j < n-1; j++){NCAMRNode[] nodes ={newgrid[i,j],newgrid[i+1,j],newgri
d[i+1,j+1],newgrid[i,j+1]};double[] x = new double[nodes.Length];double[] y = new double[nodes.
Length];for (int z = 0; z < nodes.Length; z++){x[z] = nodes[z].X;y[z] = nodes[z].Y;}accumulator
 += quad_area(x, y)*(A[i,j].Value-B[i,j].Value)*(A[i,j].Value-B[i,j].Value);}}return Math.Sqrt(
accumulator);}private static double area_of_influence_region(NCGrid_Distribution A, int i, int
j){int imax = A.Xcount - 1;int jmax = A.Ycount - 1;double[] x = new double[4];double[] y = new
double[4];NCAMRNode[] grid ={A.GetFake(i-1,j+1),A.GetFake(i,j+1),A.GetFake(i+1,j+1),A.GetFake(i
-1,j),A.GetFake(i,j),A.GetFake(i+1,j),A.GetFake(i-1,j-1),A.GetFake(i,j-1),A.GetFake(i+1,j-1)};N
CAMRNode[] poly ={xyavg(grid[3], grid[4], grid[6], grid[7]),xyavg(grid[4], grid[5], grid[7], gr
id[8]),xyavg(grid[1], grid[2], grid[4], grid[5]),xyavg(grid[0], grid[1], grid[3], grid[4])};for
 (int z = 0; z < poly.Length; z++){x[z] = poly[z].X;y[z] = poly[z].Y;}return quad_area(x, y);}p
rivate static NCAMRNode xyavg(params NCAMRNode[] nodes){int count = nodes.Length;double xacc =
0;double yacc = 0;for (int i = 0; i < count; i++){xacc += nodes[i].X;yacc += nodes[i].Y;}return
 new NCAMRNode(xacc / count, yacc / count, 0);}private static double quad_area(double[] x_clock
wise, double[] y_clockwise){Point3[] poly ={new Point3(x_clockwise[0], y_clockwise[0], 0),new P
oint3(x_clockwise[1], y_clockwise[1], 0),new Point3(x_clockwise[2], y_clockwise[2], 0),new Poin
t3(x_clockwise[3], y_clockwise[3], 0)};Vector3 A1 = new Vector3(poly[0], poly[1]);Vector3 A2 =
new Vector3(poly[0], poly[3]);Vector3 B1 = new Vector3(poly[2], poly[1]);Vector3 B2 = new Vecto
r3(poly[2], poly[3]);return 0.5 * ((A1 % A2).Norm + (B1 % B2).Norm);}}}/* Copyright (c) 2018 Wi
lliam van Noordt */using System;using System.Collections.Generic;using System.Linq;using System
.Text;using System.Threading.Tasks;using System.Diagnostics;using System.Threading;using System
.IO;namespace m_435_NCAMR{static class ExactFunctionGeneratorVB2D{private static string CSCRIPT
 = @"C:\Windows\SysWOW64\cscript.exe";private static string script_template = @"C:\Users\Will\D
esktop\Folders\MATH435\repo\visual-basic-templates\function-generator-template.vbs";private sta
tic string script_disc_template = @"C:\Users\Will\Desktop\Folders\MATH435\repo\visual-basic-tem
plates\function-generator-discretization-template.vbs";public static void GenerateFunction(stri
ng vb_syntax_eval, RBounds2D bounds, int xcount, int ycount, string name){string[] vbsraw = Fil
e.ReadAllLines(script_template);vbsraw[0] = String.Format(vbsraw[0], ycount);vbsraw[1] = String
.Format(vbsraw[1], xcount);vbsraw[2] = String.Format(vbsraw[2], name);vbsraw[3] = String.Format
(vbsraw[3], bounds.Xmin);vbsraw[4] = String.Format(vbsraw[4], bounds.Xmax);vbsraw[5] = String.F
ormat(vbsraw[5], bounds.Ymin);vbsraw[6] = String.Format(vbsraw[6], bounds.Ymax);vbsraw[14] = St
ring.Format(vbsraw[14], (2 + xcount * ycount));vbsraw[25] = String.Format(vbsraw[25], vb_syntax
_eval);vbsraw[50] = String.Format(vbsraw[50], (2 + xcount * ycount));string newname = @"C:\User
s\Will\Desktop\Folders\MATH435\repo\visual-basic-templates\TEMPORARY.vbs";File.WriteAllLines(ne
wname, vbsraw);run_script(newname);}private static void run_script(string scriptname_absolute){
Process vb_script = Process.Start(CSCRIPT, scriptname_absolute);while (!vb_script.HasExited){Th
read.Sleep(10);}File.Delete(scriptname_absolute);}public static void quickPlot(string vb_eval,
string title, RBounds2D bounds){int n = 100;GenerateFunction(vb_eval, bounds, n, n, title);NCGr
id_Distribution p = NCGrid_Distribution.from_file(title, false);File.Delete(Paths.DistributionR
epo + "\\" + title + ".dist");DistributionSketchSettings S = DistributionSketchSettings.Fancy()
;S.HasHeatmap = true;S.HasInfo = false;S.Mode = SketchMode.DISTRIBUTION_ONLY;DistributionSketch
2D sk = new DistributionSketch2D(p, S);sk.CreateSketch(false);sk.SaveImage(title, false);}publi
c static void quickPlot(string vb_eval, string title){quickPlot(vb_eval, title, RBounds2D.Squar
```

```
e(10));}public static NCGrid_Distribution GenerateFunctionToGrid(string vb_eval, NCGrid_Distrib
ution discretization){string absolute_temp_path = Paths.DistributionRepo + "\\temp.dist";Genera
teFunction(vb_eval, discretization, "temp");NCGrid_Distribution output = NCGrid_Distribution.fr
om_file("temp", false);File.Delete(absolute_temp_path);return output;}public static void Genera
teFunction(string vb_syntax_eval, NCGrid_Distribution discretization, string name){discretizati
on.WriteToFile(@"C:\Users\Will\Desktop\Folders\MATH435\repo\visual-basic-templates\TEMPORARY.di
st", false, true, true);string[] vbsraw = File.ReadAllLines(script_disc_template);int fileconst
 = discretization.Xcount * discretization.Ycount + 2;vbsraw[0] = String.Format(vbsraw[0], discr
etization.Ycount);vbsraw[1] = String.Format(vbsraw[1], discretization.Xcount);vbsraw[16] = Stri
ng.Format(vbsraw[16], fileconst);vbsraw[13] = String.Format(vbsraw[13], fileconst);vbsraw[2] =
String.Format(vbsraw[2], name);vbsraw[28] = String.Format(vbsraw[28], vb_syntax_eval);string ne
wname = @"C:\Users\Will\Desktop\Folders\MATH435\repo\visual-basic-templates\TEMPORARY.vbs";File
.WriteAllLines(newname, vbsraw);run_script(newname);File.Delete(@"C:\Users\Will\Desktop\Folders
\MATH435\repo\visual-basic-templates\TEMPORARY.dist");}}}/* Copyright (c) 2018 William van Noor
dt */using System;using System.Collections.Generic;using System.Linq;using System.Text;using Sy
stem.Threading.Tasks;namespace m_435_NCAMR{class FluidProperties{private double rho, mu, nu;pub
lic double Density{get { return rho; }}public double DynamicViscosity{get { return mu; }}public
 double KinematicViscosity{get { return nu; }}public FluidProperties(double _density, double dy
namic_viscosity){mu = dynamic_viscosity;rho = _density;nu = mu / rho;}}}/* Copyright (c) 2018 W
illiam van Noordt */using System;using System.Collections.Generic;using System.Linq;using Syste
m.Text;using System.Threading.Tasks;using System.IO;using System.Drawing;using AForge.Video.VFW
;using _3DSimple;namespace m_435_NCAMR{class FollowedDataSet : DataSet{public FollowedDataSet(s
tring title, bool using_full_path, bool enable_console_output) : base(title, using_full_path, e
nable_console_output){}public new static FollowedDataSet FirstAvailable(bool enable_console_out
put){string[] allnames = Directory.GetDirectories(default_source);if (allnames.Length != 0){ret
urn new FollowedDataSet(allnames[0], true, enable_console_output);}else{throw new Exception("No
 datasets available.");}}public override void create_animation(string output_title, bool enable
_console_output, bool using_full_path, bool allow_overwrite, bool use_floating_color_scale, Dis
tributionSketchSettings S){string absolute_video_output_path = video_default_repo + "\\" + outp
ut_title + ".avi";if (using_full_path) { absolute_video_output_path = output_title; }if (File.E
xists(absolute_video_output_path) && !allow_overwrite) { throw new Exception("Error: File exist
s and overwrite permission not granted."); }Directory.CreateDirectory(AbsoluteDataSetLocation +
 "\\images");string[] allfiles = Directory.GetFiles(absolute_set_location);List<string> distrib
_files = new List<string>();int q = 0;foreach (string i in allfiles){if (i.EndsWith(".dist")){d
istrib_files.Add(i);NCGrid_Distribution n = NCGrid_Distribution.from_file(i, true);NCGrid_Distr
ibution following_grid = new NCGrid_Distribution(n.Bounds, n.Xcount, n.Ycount);int border = 1;f
or (int ii = border; ii < following_grid.Xcount- border; ii++){for (int jj = border; jj < follo
wing_grid.Ycount- border; jj++){double deltax = (n.Xmax - n.Xmin) / (n.Xcount - 1);double delta
y = (n.Ymax - n.Ymin) / (n.Ycount - 1);double[] stencil ={n[ii-1,jj-1].Value,n[ii,jj-1].Value,n
[ii+1,jj-1].Value,n[ii-1,jj].Value,n[ii,jj].Value,n[ii+1,jj].Value,n[ii-1,jj+1].Value,n[ii,jj+1
].Value,n[ii+1,jj+1].Value,};double ux = (stencil[5] - stencil[3]) / (2 * deltax);double uy = (
stencil[7] - stencil[1]) / (2 * deltay);double uxx = (stencil[5]+stencil[3] - (2*stencil[4])) /
 (deltax * deltax);double uyy = (stencil[7] + stencil[1] - (2 * stencil[4])) / (deltay * deltay
);double uxy = (stencil[8]+stencil[0]-stencil[6]-stencil[2])/(4*deltax*deltay);double dx = 0.06
 * ((uxx*ux)+(uxy*uy));double dy = 0.06 * ((uyy*uy)+(ux*uxy));Vector3 move = new Vector3(dx, dy
, 0);while ((following_grid[ii, jj] + move).X > n.Bounds.Xmax || (following_grid[ii, jj] + move
).X < n.Bounds.Xmin || (following_grid[ii, jj] + move).Y > n.Bounds.Ymax || (following_grid[ii,
 jj] + move).Y < n.Bounds.Ymin){dx = 0.5 * dx;dy = 0.5 * dy;move = new Vector3(dx, dy, 0);}foll
owing_grid[ii, jj] = following_grid[ii, jj] + move;}}HybridDistributionSketch2D sk = new Hybrid
DistributionSketch2D(n, S);if (!use_floating_color_scale){sk.override_colormap_limits(info);}sk
.set_superimposed_grid(following_grid);sk.CreateSketch(true);sk.SaveImage(AbsoluteDataSetLocati
on + "\\images\\render" + bufferint(q++, 5) + ".bmp", true);}}AVIWriter V = new AVIWriter("wmv3
");string[] filenames = Directory.GetFiles(AbsoluteDataSetLocation + "\\images");List<string> b
mpnames = new List<string>();foreach (string i in filenames){if (i.EndsWith(".bmp")) { bmpnames
.Add(i); }}if (bmpnames.Count != 0){Bitmap first = (Bitmap)Bitmap.FromFile(bmpnames[0]);V.Open(
absolute_video_output_path, first.Width, first.Height);V.FrameRate = 50;V.AddFrame(first);int c
```

```
t = bmpnames.Count;for (int i = 1; i < bmpnames.Count; i++){Bitmap butt = (Bitmap)Bitmap.FromFi
le(bmpnames[i]);if (enable_console_output) { Console.WriteLine(i.ToString() + " of " + ct.ToStr
ing() + " frames stacked" + "(" + (i * 100 / ct).ToString() + "%)"); }V.AddFrame(butt);butt.Dis
pose();}V.Close();first.Dispose();if (enable_console_output) { Console.WriteLine("AVI successfu
lly created on " + DateTime.Now.ToString() + " in directory " + absolute_video_output_path); }s
tring[] imagenames = Directory.GetFiles(AbsoluteDataSetLocation + "\\images");foreach (string g
 in imagenames){File.Delete(g);}Directory.Delete(AbsoluteDataSetLocation + "\\images");}else{Co
nsole.WriteLine("No bitmap images found in the current directory.");}}}}}/* Copyright (c) 2018 W
illiam van Noordt */using System;using System.Collections.Generic;using System.Linq;using Syste
m.Text;using System.Threading.Tasks;using System.Drawing;namespace m_435_NCAMR{class HybridDist
ributionSketch2D : DistributionSketch2D{private NCGrid_Distribution superimpose_grid;private bo
ol has_superimposed;public HybridDistributionSketch2D(NCGrid_Distribution subject, Distribution
SketchSettings S) : base(subject, S){has_superimposed = false;}public void set_superimposed_gri
d(NCGrid_Distribution _super){has_superimposed = true;superimpose_grid = _super;}public void re
voke_superimposed_grid(){has_superimposed = false;}public override void CreateSketch(bool allow
_console_output){settings.Mode = SketchMode.DISTRIBUTION_ONLY;base.CreateSketch(allow_console_o
utput);if (has_superimposed){using (Graphics drawer = Graphics.FromImage(canvas)){for (int i =
0; i < superimpose_grid.Xcount - 1; i++){drawer.DrawLine(THICK_GRID_PEN, mapGridNode(superimpos
e_grid[i, superimpose_grid.Ycount - 1]), mapGridNode(superimpose_grid[i + 1, superimpose_grid.Y
count - 1]));drawer.DrawLine(THICK_GRID_PEN, mapGridNode(superimpose_grid[i, 0]), mapGridNode(s
uperimpose_grid[i + 1, 0]));}for (int j = 0; j < superimpose_grid.Ycount - 1; j++){drawer.DrawL
ine(THICK_GRID_PEN, mapGridNode(superimpose_grid[superimpose_grid.Xcount - 1, j]), mapGridNode(
superimpose_grid[superimpose_grid.Xcount - 1, j + 1]));drawer.DrawLine(THICK_GRID_PEN, mapGridN
ode(superimpose_grid[0, j]), mapGridNode(superimpose_grid[0, j + 1]));}for (int i = 0; i < supe
rimpose_grid.Xcount - 1; i++){for (int j = 0; j < superimpose_grid.Xcount - 1; j++){drawer.Draw
Line(THIN_GRID_PEN, mapGridNode(superimpose_grid[i, j]), mapGridNode(superimpose_grid[i + 1, j]
));drawer.DrawLine(THIN_GRID_PEN, mapGridNode(superimpose_grid[i, j]), mapGridNode(superimpose_
grid[i, j + 1]));}}if (settings.HasHeatmap){draw_heatmap_box(drawer);}}}}}}/* Copyright (c) 201
8 William van Noordt */using System;using System.Collections.Generic;using System.Linq;using Sy
stem.Text;using System.Threading.Tasks;namespace m_435_NCAMR{class LinearOperatorOrder2{public
enum OperatorType{Parabolic,Hyperbolic,Elliptic}OperatorType type;double a, a_x, a_y, a_xx, a_y
y, a_xy;public OperatorType Classification{get { return type; }}public double A{get { return a;
 }set { a = value; determine_type(); }}public double Ax{get { return a_x; }set { a_x = value; d
etermine_type(); }}public double Ay{get { return a_y; }set { a_y = value; determine_type(); }}p
ublic double Axx{get { return a_xx; }set { a_xx = value; determine_type(); }}public double Ayy{
get { return a_yy; }set { a_yy = value; determine_type(); }}public double Axy{get { return a_xy
; }set { a_xy = value; determine_type(); }}public LinearOperatorOrder2(double[] coeffs_6x){if (
coeffs_6x.Length != 6) { throw new Exception("Error: Please provide exactly 6 coefficients.");
}a = coeffs_6x[0];a_x = coeffs_6x[1];a_y = coeffs_6x[2];a_xx = coeffs_6x[3];a_yy = coeffs_6x[4]
;a_xy = coeffs_6x[5];determine_type();}public LinearOperatorOrder2(double _a, double _a_x, doub
le _a_y, double _a_xx, double _a_yy, double _a_xy){a = _a;a_x = _a_x;a_y = _a_y;a_xx = _a_xx;a_
yy = _a_yy;a_xy = _a_xy;determine_type();}private void determine_type(){double z = (a_xy * a_xy
) - (4 * a_xx * a_yy);if (Math.Abs(z) < 1E-10) { type = OperatorType.Parabolic; }else if (z > 0
) { type = OperatorType.Hyperbolic; }else { type = OperatorType.Elliptic; }}public static Linea
rOperatorOrder2 Laplace = new LinearOperatorOrder2(0, 0, 0, 1, 1, 0);public double this[int i]{
get{switch (i){case -1:{return a;}case 0:{return a_x;}case 1:{return a_y;}case 2:{return a_xx;}
case 3:{return a_yy;}case 4:{return a_xy;}default:{throw new Exception("Index out of range.");}
}}set{switch (i){case -1:{a = value;break;}case 0:{a_x = value;break;}case 1:{a_y = value;break
;}case 2:{a_xx = value;break;}case 3:{a_yy = value;break;}case 4:{a_xy = value;break;}default:{
throw new Exception("Index out of range.");}}}}}}/* Copyright (c) 2018 William van Noordt */usi
ng System;using System.Collections.Generic;using System.Linq;using System.Text;using System.Thr
eading.Tasks;using _3DSimple;using System.IO;namespace m_435_NCAMR{class Matrix{private const d
ouble JUST_ABOUT_ZERO = 1E-10;static int zero_count_parameter = 3;private int rows, columns;pri
vate double[] contents;private double determinant;private bool isSquare;private Matrix _inverse
 = null;private Matrix _ref = null;private Matrix _rref = null;private Matrix _transpose = null
;public Matrix Inverse{get{if (this.isSquare){return _inverse ?? internalInverse(this);}else{th
```

```
row new ArgumentException("Error: Matrix is not square.");}}}public int Dimension{get { return
rows * columns; }}public bool isSingular{get { return this.Determinant == 0; }}public double De
terminant{get{if (!isSquare) { throw new ArgumentException("Matrix not square."); }else{if (det
erminant == double.PositiveInfinity){determinant = basicDet(this);return determinant;}else{retu
rn determinant;}}}}public int Rows{get { return rows; }}public int Columns{get { return columns
; }}public bool IsSquare{get { return isSquare; }}public enum SystemSolvingScheme{Basic,SortToD
iagonal,Kaczmarz}public Matrix(int _rows, int _columns){isSquare = _rows == _columns;determinan
t = double.PositiveInfinity;contents = new double[_rows * _columns];rows = _rows;columns = _col
umns;for (int i = 0; i < _rows; i++){for (int j = 0; j < columns; j++){contents[j + (i * column
s)] = 0;}}}public Matrix(int rank){isSquare = true;determinant = double.PositiveInfinity;conten
ts = new double[rank * rank];rows = rank;columns = rank;for (int i = 0; i < rank; i++){for (int
 j = 0; j < rank; j++){contents[j + (i * columns)] = 0;}}}public Matrix(int _rows, int _columns
, params double[] cts){isSquare = _rows == _columns;determinant = double.PositiveInfinity;conte
nts = new double[_rows * _columns];rows = _rows;columns = _columns;for (int i = 0; i < _rows; i
++){for (int j = 0; j < columns; j++){contents[j + (i * columns)] = cts[j + (i * columns)];}}}p
ublic static Matrix eliminateEntry(Matrix A, int row, int col){Matrix outMat = new Matrix(A.row
s - 1, A.Columns - 1);int colOffset = 0;for (int i = 0; i < A.Columns - 1; i++){int rowOffset =
 0;if (i == col) { colOffset = 1; }for (int j = 0; j < A.Rows - 1; j++){if (j == row) { rowOffs
et = 1; }outMat[i, j] = A[i + colOffset, j + rowOffset];}}return outMat;}/// <summary>/// "angl
e" in degrees/// </summary>public static Matrix RotationMatrix3D(double angle, Vector3 axis){Ve
ctor3 u = axis.unit();double aangle = angle;double cos = Math.Cos(aangle);double sin = Math.Sin
(aangle);double onemns = 1 - cos;double[] C = new double[9];C[0] = cos + (u.I * u.I * onemns);C
[1] = (u.I * u.J * onemns) - (u.K * sin);C[2] = (u.I * u.K * onemns) + (u.J * sin);C[3] = (u.J
* u.I * onemns) + (u.K * sin);C[4] = cos + (u.J * u.J * onemns);C[5] = (u.J * u.K * onemns) - (
u.I * sin);C[6] = (u.K * u.I * onemns) - (u.J * sin);C[7] = (u.K * u.J * onemns) + (u.I * sin);
C[8] = cos + (u.K * u.K * onemns);return new Matrix(3, 3, C);}public double this[int row, int c
ol]{get { int index = col + (row * columns); return contents[index]; }set{if (row >= rows) { th
row new IndexOutOfRangeException("Row index out of range: exceeded maximum value."); }if (col >
= columns) { throw new IndexOutOfRangeException("Column index out of range: exceeded maximum va
lue."); }if (row < 0) { throw new IndexOutOfRangeException("Row index out of range: exceeded mi
nimum value."); }if (col < 0) { throw new IndexOutOfRangeException("Column index out of range:
exceeded minimum value."); }double inter = value;int index = col + (row * columns);contents[ind
ex] = inter;clearDependent();}}public static Matrix ConcatenateRight(Matrix A, Matrix B){if (A.
rows != B.rows){throw new Exception("Error: Inconsistent dimensions.");}int rs = A.rows;int fir
stcols = A.columns;int secondcols = B.columns;int totalcols = firstcols + secondcols;Matrix out
put = new Matrix(rs, totalcols);for (int i = 0; i < rs; i++){for (int j = 0; j < firstcols; j++
){output[i, j] = A[i, j];}for (int j = firstcols; j < totalcols; j++){output[i, j] = B[i, j-fir
stcols];}}return output;}public double this[int index]{get{if (rows == 1 || columns == 1){retur
n contents[index];}else{throw new Exception("Error: Matrix is not a row or column vector.");}}s
et{if (rows == 1 || columns == 1){contents[index] = value;clearDependent();}else{throw new Exce
ption("Error: Matrix is not a row or column vector.");}}}private double basicDet(Matrix A){if (
A.Rows == 1 && A.Columns == 1){return A[0, 0];}else{double acc = 0;double sgn = 1;for (int i =
0; i < A.columns; i++){acc += A[i, 0] * sgn * basicDet(Matrix.eliminateEntry(A, 0, i));sgn *= -
1;}return acc;}}public static Matrix Identity(int m){Matrix I = new Matrix(m);for (int i = 0; i
 < m; i++){I[i, i] = 1;}return I;}public Matrix Clone(){Matrix X = new Matrix(rows, columns);fo
r (int i = 0; i < rows; i++){for (int j = 0; j < columns; j++){X[i, j] = this[i, j];}}return X;
}private Matrix internalInverse(Matrix A){//will need modification to switch around rows if fir
st diagonal entry is zero.int m = A.rows;Matrix invs = Matrix.Identity(m);Matrix clone = this.C
lone();for (int diagrow = 0; diagrow < m; diagrow++){double scl1 = 1 / clone[diagrow, diagrow];
invs.RowScale(diagrow, scl1);clone.RowScale(diagrow, scl1);for (int i = 0; i < m; i++){if (i !=
 diagrow && clone[i, diagrow] != 0){double scl2 = -1 * clone[i, diagrow];invs.RowAdd(i, diagrow
, scl2);clone.RowAdd(i, diagrow, scl2);}}}_inverse = invs;return invs;}private Matrix internalI
nverse(Matrix A, bool doLatexOutput, string path, bool approximmateZero){List<string> contents
= new List<string>();int m = A.rows;contents.Add("Let $M = " + this.Latexoutput(approximmateZer
o) + "$. Then, the following operations are used to invert $M$ to find $M^{-1}$:");Matrix clone
 = this;Matrix invs = Matrix.Identity(m);contents.Add("\\begin{equation*}");contents.Add(clone.
```

```
Latexoutput(approximmateZero));contents.Add("\\longleftrightarrow");contents.Add(invs.Latexoutput(approximmateZero));contents.Add("\\end{equation*}");for (int diagrow = 0; diagrow < m; diagrow++){double scl1 = 1 / clone[diagrow, diagrow];invs.RowScale(diagrow, scl1);clone.RowScale(diagrow, scl1);contents.Add("\\begin{equation*}");contents.Add(clone.Latexoutput(approximmateZero));contents.Add("\\longleftrightarrow");contents.Add(invs.Latexoutput(approximmateZero));contents.Add("\\end{equation*}");for (int i = 0; i < m; i++){if (i != diagrow){double scl2 = -1 * clone[i, diagrow];invs.RowAdd(i, diagrow, scl2);clone.RowAdd(i, diagrow, scl2);}}contents.Add("\\begin{equation*}");contents.Add(clone.Latexoutput(approximmateZero));contents.Add("\\longleftrightarrow");contents.Add(invs.Latexoutput(approximmateZero));contents.Add("\\end{equation*}");}_inverse = invs;contents.Add("Therefore, $M^{-1} = " + _inverse.Latexoutput(approximmateZero) + "$.");File.WriteAllLines(path, contents.ToArray());Console.WriteLine("Latex file output.");return invs;}public void RowScale(int row_zer_bas, double scale){for (int j = 0; j < columns; j++){this[row_zer_bas, j] *= scale;}}private static bool isZero(double D){return Math.Abs(D) < JUST_ABOUT_ZERO;}public void RowAdd(int row_add_to, int row_add_from, double prescale){for (int j = 0; j < columns; j++){this[row_add_to, j] += (prescale * this[row_add_from, j]);}}public static Matrix operator +(Matrix A, Matrix B){if (A.Rows != B.Rows || A.Columns != B.Columns){throw new ArgumentException("Error: matrix dimensions must agree.");}else{Matrix C = new Matrix(A.rows, B.Columns);for (int i = 0; i < A.Rows; i++){for (int j = 0; j < B.columns; j++){C[i, j] = A[i, j] + B[i, j];}}return C;}}private static double vec_inner_prod(Matrix A, Matrix B){double acc = 0;for (int i = 0; i < A.Dimension; i++){acc += A[i] * B[i];}return acc;}public static Matrix operator -(Matrix A, Matrix B){if (A.Rows != B.Rows || A.Columns != B.Columns){throw new ArgumentException("Error: matrix dimensions must agree.");}else{Matrix C = new Matrix(A.rows, B.Columns);for (int i = 0; i < A.Rows; i++){for (int j = 0; j < B.columns; j++){C[i, j] = A[i, j] - B[i, j];}}return C;}}public static Matrix operator *(Matrix A, Matrix B){if (A.Columns != B.Rows){throw new ArgumentException("Error: Matrix dimensions are incompatible with multiplication");}else{Matrix C = new Matrix(A.Rows, B.Columns);for (int i = 0; i < C.Rows; i++){for (int j = 0; j < C.Columns; j++){double acc = 0;for (int y = 0; y < B.Rows; y++){acc += A[i, y] * B[y, j];}C[i, j] = acc;}}return C;}}private void clearDependent(){determinant = double.PositiveInfinity;_inverse = null;_ref = null;_rref = null;_transpose = null;}/// <summary>/// Solves an N by N system of equations, with right-hand N-vecctor./// </summary>/// <param name="coeffs">Must be a square matrix</param>/// <param name="rhs">Right hand side</param>/// <returns></returns>public static Matrix solve_system(Matrix coeffs, Matrix rhs){return solve_system(coeffs, rhs, SystemSolvingScheme.Basic, true, false);}/// <summary>/// Solves an N by N system of equations, with right-hand N-vecctor./// </summary>/// <param name="coeffs">Must be a square matrix</param>/// <param name="rhs">Right hand side</param>/// <param name="scheme">Indicates the scheme used to solve the system</param>/// <returns></returns>public static Matrix solve_system(Matrix coeffs, Matrix rhs, SystemSolvingScheme scheme){return solve_system(coeffs, rhs, scheme, true, false);}/// <summary>/// Solves an N by N system of equations, with right-hand N-vecctor./// </summary>/// <param name="coeffs">Must be a square matrix</param>/// <param name="rhs">Right hand side</param>/// <param name="suppress_determinant_validation">Used to supress determinant validation, lower computing time, but risks error</param>/// <returns></returns>public static Matrix solve_system(Matrix coeffs, Matrix rhs, bool suppress_determinant_validation){return solve_system(coeffs, rhs, SystemSolvingScheme.Basic, suppress_determinant_validation, false);}/// <summary>/// Solves an N by N system of equations, with right-hand N-vecctor./// </summary>/// <param name="coeffs">Must be a square matrix</param>/// <param name="rhs">Right hand side</param>/// <param name="scheme">Indicates the scheme used to solve the system</param>/// <param name="suppress_determinant_validation">Used to supress determinant validation, lower computing time, but risks error</param>/// <returns></returns>public static Matrix solve_system(Matrix coeffs, Matrix rhs, SystemSolvingScheme scheme, bool suppress_determinant_validation, bool enable_console_output){if (!coeffs.IsSquare || (rhs.Columns != 1) || (rhs.Rows != coeffs.Rows)){throw new ArgumentException("System has incompatible dimensions.");}if (!suppress_determinant_validation){if (coeffs.isSingular){throw new ArgumentException("Coefficient matrix is singular.");}}switch (scheme){case SystemSolvingScheme.Basic:{return coeffs.Inverse * rhs;}case SystemSolvingScheme.SortToDiagonal:{return solve_using_sort_diagonal(coeffs, rhs, true);}case SystemSolvingScheme.Kaczmarz:{return kaczmarz_solve(coeffs, rhs, 100, 1E-5);}default:{throw new ArgumentException("Invalid scheme");}}}private static Matrix solve_using_sort_diagonal(Matrix coeffs, Matrix rhs, bool enable_con
```

```
sole_output){int rws = coeffs.rows;int cols = coeffs.columns + 1;Matrix augment = new Matrix(rw
s, cols + 1);if (enable_console_output) { Console.WriteLine("Augmenting..."); }for (int i = 0;
i < rws; i++){if (enable_console_output && i%(rws/13) == 0){Console.WriteLine(((100 * i) / rws)
.ToString() + "%");}for (int j = 0; j < cols - 1; j++){augment[i, j] = coeffs[i, j];}augment[i,
 cols - 1] = rhs[i];augment[i, cols] = 0;}for (int cur_row = 0; cur_row < augment.rows; cur_row
++){bool zero = true;for (int cur_col = 0; cur_col < augment.columns && zero; cur_col++){zero =
 isZero(augment[cur_row, cur_col]);if (zero){augment[cur_row, cols] = cur_col + 1;}}}sort_by_la
st(augment);int m = augment.rows;int n = augment.columns - 1;if (enable_console_output) { Conso
le.WriteLine("Solving..."); }for (int diagrow = 0; diagrow < m; diagrow++){if (enable_console_o
utput && diagrow%(m/13) == 0){Console.WriteLine(((100*diagrow)/m).ToString()+"%");}double scl1
= 1 / augment[diagrow, diagrow];augment.RowScale(diagrow, scl1);int zerocount = 0;for (int i =
0; i < m && zerocount < zero_count_parameter; i++){if (Math.Abs(augment[i, diagrow]) < 1E-10) {
 zerocount++; }if (i != diagrow){double scl2 = -1 * augment[i, diagrow];augment.RowAdd(i, diagr
ow, scl2);}}}double[] result = new double[augment.rows];for (int i = 0; i < augment.rows; i++){
result[i] = augment[i, augment.columns - 2];}return new Matrix(augment.rows, 1, result);}public
 void RowSwap(int row, int other_row){for (int i = 0; i < this.columns; i++){double med = this[
other_row, i];this[other_row, i] = this[row, i];this[row, i] = med;}}public double sumrow(int r
ow){double sum = 0;for (int i = 0; i < columns; i++){sum += this[row, i];}return sum;}public vo
id exportToFile(string path){string filetype = path.Split('.')[1];switch (filetype){case "csv":
{List<string> filestuff = new List<string>();for (int i = 0; i < rows; i++){string currentline
= string.Empty;currentline += this[i, 0].ToString();for (int j = 1; j < columns; j++){currentli
ne += "," + this[i, j].ToString();}filestuff.Add(currentline);}File.WriteAllLines(path, filestu
ff.ToArray());break;}case "txt":{File.WriteAllLines(path, this.ToStrings());break;}default:{thr
ow new Exception("Error: Invalid filetype.");}}}public string[] ToStrings(){List<string> sts =
new List<string>();int shelfsize = 8;int trunc = 6;for (int i = 0; i < this.Rows; i++){string l
ine = string.Empty;for (int j = 0; j < this.Columns; j++){string entry = this[i, j].ToString();
string tr_entry = entry.Substring(0, Math.Min(trunc, entry.Length));line += tr_entry.PadRight(s
helfsize);}sts.Add(line);}return sts.ToArray();}public string[] ToStrings(int _trunc, int _shel
fsize){List<string> sts = new List<string>();int shelfsize = _shelfsize;int trunc = _trunc;for
(int i = 0; i < this.Rows; i++){string line = string.Empty;for (int j = 0; j < this.Columns; j+
+){string entry = this[i, j].ToString();string tr_entry = entry.Substring(0, Math.Min(trunc, en
try.Length));line += tr_entry.PadRight(shelfsize);}sts.Add(line);}return sts.ToArray();}private
 static void sort_by_last(Matrix M){int last = M.rows - 1;quicksort(M, 0, last);}private static
 void quicksort(Matrix M, int lo, int hi){if (lo < hi){int pi = partition(M, lo, hi);quicksort(
M, lo, pi - 1);quicksort(M, pi + 1, hi);}}private static int partition(Matrix M, int lo, int hi
){int lst = M.columns - 1;int pivot = (int)M[hi, lst];int i = (lo - 1);for (int j = lo; j <= hi
 - 1; j++){if (M[j, lst] <= pivot){i++;M.RowSwap(i, j);}}M.RowSwap(i + 1, hi);return (i + 1);}p
ublic string Latexoutput(bool approximate_zeros){string output = "\\begin{bmatrix}";for (int j
= 0; j < rows; j++){output += ((float)(this[j, 0])).ToString();for (int i = 1; i < columns; i++
){if (approximate_zeros && Math.Abs(this[j, i]) < 1E-9){output += "&" + "0";}else{output += "&"
 + ((float)(this[j, i])).ToString();}}output += "\\" + "\\";}return output + "\\end{bmatrix}";}
public static Matrix ext_kaczmarz_radius(Matrix A, Matrix b, double tolerance, int maxiteration
s, int radius){Matrix xk = Matrix.constvec(b.Rows, 1);int n = b.Rows;int k = 0;Matrix xprev = x
k.Clone();while (k <= maxiterations){double complete_residual = Matrix.VecNorm(A * xk - b);for
(int i = 0; i < xk.Dimension; i++){Matrix ai = A.RowVector(i);double prod = vec_prod_radius(ai,
 xk, radius, i);double top = prod - b[i];double mag = Matrix.VecNorm(ai);double bottom = mag *
mag;double scale = top / bottom;Matrix subtract = scale * ai;xprev = xk.Clone();xk = xk - subtr
act.Transpose();}double residual = Matrix.VecNorm(xk - xprev) / Matrix.VecNorm(xk);Console.Writ
eLine(complete_residual.ToString() + "," + residual.ToString());if (residual < tolerance){break
;}k++;}return xk;}private static double vec_prod_radius(Matrix a, Matrix b, int assumedradius,
int center){int bottom = Math.Max(0, center - assumedradius);int top = Math.Min(b.Dimension - 1
, center + assumedradius);double acc = 0;for (int i = bottom; i < top; i++){acc += a[i] * b[i];
}return acc;}public static Matrix FromCsv(string path){if (!path.EndsWith(".csv")) { throw new
Exception("Invalid file type."); }List<double[]> cts = new List<double[]>();string[] stuff = Fi
le.ReadAllLines(path);int rows = stuff.Length;int columns = stuff[0].Split(',').Length;Matrix M
 = new Matrix(rows, columns);for (int i = 0; i < rows; i++){string[] line = stuff[i].Split(',')
```

```
;for (int j = 0; j < columns; j++){double entry;if (!double.TryParse(line[j], out entry)){throw
 new Exception("Error: invalid entry at " + i.ToString() + ", " + j.ToString());}else{M[i, j] =
 entry;}}}return M;}public Matrix WriteInverseToFile(string path){return internalInverse(this,
true, path, true);}public Matrix Transpose(){Matrix output = new Matrix(this.columns, this.rows
);for (int i = 0; i < output.rows; i++){for (int j = 0; j < output.Columns; j++){output[i, j] =
 this[j, i];}}return output;}private static Matrix kaczmarz_solve(Matrix A, Matrix b, int maxit
erations, double tolerance){//initial guessMatrix xk = Matrix.constvec(b.Rows, 1);int n = b.Row
s;int k = 0;Matrix xprev = xk.Clone();while (k <= maxiterations){double complete_residual = Mat
rix.VecNorm(A * xk - b);for (int i = 0; i < xk.Dimension; i++){Matrix ai = A.RowVector(i);doubl
e prod = Matrix.vec_inner_prod(ai, xk);double top = prod - b[i];double mag = Matrix.VecNorm(ai)
;double bottom = mag * mag;double scale = top / bottom;Matrix subtract = scale * ai;xprev = xk.
Clone();xk = xk - subtract.Transpose();}double residual = Matrix.VecNorm(xk - xprev) / Matrix.V
ecNorm(xk);Console.WriteLine(complete_residual.ToString() + "," + residual.ToString());if (resi
dual < tolerance){break;}k++;}return xk;}public static Matrix operator*(double scale, Matrix A)
{Matrix output = new Matrix(A.rows, A.columns);for (int i = 0; i < A.rows; i++){for (int j = 0;
 j < A.columns; j++){output[i, j] = scale * A[i, j];}}return output;}public static double VecNo
rm(Matrix vectormatrix){double acc = 0;for (int i = 0; i < vectormatrix.Dimension; i++){acc +=
vectormatrix[i] * vectormatrix[i];}return Math.Sqrt(acc);}public Matrix ColumnVector(int i){Mat
rix output = new Matrix(rows, 1);for (int z = 0; z < rows; z++){output[z] = this[z, i];}return
output;}public Matrix RowVector(int i){Matrix output = new Matrix(1, columns);for (int z = 0; z
 < columns; z++){output[z] = this[i, z];}return output;}private static Matrix constvec(int rowd
imension, double val){Matrix z = new Matrix(rowdimension, 1);for (int i = 0; i < rowdimension;
i++){z[i] = val;}return z;}}}/* Copyright (c) 2018 William van Noordt */using System;using Syst
em.Collections.Generic;using System.Linq;using System.Text;using System.Threading.Tasks;using S
ystem.IO;namespace m_435_NCAMR{class NavierStokesProblemSingleFrame{private int problemsize;pri
vate Matrix system_matrix;private NCGrid_Distribution prev_u, prev_v, u_star, v_star, discretiz
ation;private NCGrid_Distribution[] solution;public NCGrid_Distribution[] Solution { get { retu
rn solution; } }private Matrix current_solution_raw;private Matrix rhs;private FluidProperties
 f_props;int eq_nodes_x;int eq_nodes_y;double deltat;private BoundaryConditions u_boundary_condi
tions, v_boundary_conditions, p_boundary_conditions;public NavierStokesProblemSingleFrame(NCGri
d_Distribution ustar, NCGrid_Distribution vstar, NCGrid_Distribution previous_solution_u, NCGri
d_Distribution previous_solution_v, BoundaryConditions u, BoundaryConditions v, BoundaryConditi
ons p, FluidProperties props, double dt){deltat = dt;f_props = props;u_star = ustar;v_star = vs
tar;discretization = previous_solution_u.Clone();prev_u = previous_solution_u;prev_v = previous
_solution_v;u_boundary_conditions = u;v_boundary_conditions = v;p_boundary_conditions = p;probl
emsize = 3*(discretization.Xcount - 2) * (discretization.Ycount - 2);system_matrix = new Matrix
(problemsize);rhs = new Matrix(problemsize, 1);eq_nodes_x = discretization.Xcount - 2;eq_nodes_
y = discretization.Ycount - 2;}public NCGrid_Distribution[] Solve(){build_system();for (int i =
 0; i < 2; i++)//number of refinements = 3 for now{current_solution_raw = Matrix.solve_system(s
ystem_matrix, rhs,Matrix.SystemSolvingScheme.Kaczmarz);solution = make_from_matrix(current_solu
tion_raw);NCGrid_Distribution magnitude = NCGrid_Distribution.MakeMagnitude(solution[1], soluti
on[2]);magnitude.ApplyMeshMorphGA(0.2);solution[0].MimicMorph(magnitude);solution[1].MimicMorph
(magnitude);solution[2].MimicMorph(magnitude);discretization.MimicMorph(magnitude);u_star = sol
ution[1].Clone();v_star = solution[2].Clone();build_system();}current_solution_raw = Matrix.sol
ve_system(system_matrix, rhs, Matrix.SystemSolvingScheme.Kaczmarz);solution = make_from_matrix(
current_solution_raw);solution[0].MimicMorph(discretization);solution[1].MimicMorph(discretizat
ion);solution[2].MimicMorph(discretization);return solution;}private void build_system(){double
 rho = f_props.Density;double nu = f_props.KinematicViscosity;//interiorfor (int i_global = 1;
i_global < discretization.Xcount - 1; i_global++){for (int j_global = 1; j_global < discretizat
ion.Ycount - 1; j_global++){int i_local = i_global - 1;int j_local = j_global - 1;int current_r
ow_cluster = j_local + (i_local * eq_nodes_y);int force_x = 3 * current_row_cluster;int force_y
 = force_x + 1;int continuity = force_x + 2;Matrix b = discretization.GetTaylorSystemCoeffs(i_g
lobal, j_global, 1, 1);double[] sums = new double[5];for (int q = 0; q < 5; q++){sums[q] = 0;}f
or (int q = 0; q < 5; q++){sums[0] += b[0, q];sums[1] += b[1, q];sums[2] += b[2, q];sums[3] +=
b[3, q];sums[4] += b[4, q];}double[] P = {-1*sums[0]/f_props.Density,b[0, 0] / rho,b[0, 1] / rh
o,b[0, 2] / rho,b[0, 3] / rho,b[0, 4] / rho};double[] Q = {(-1*u_star[i_global,j_global].Value*
```

```
sums[0]) - (v_star[i_global,j_global].Value*sums[1]) + (nu*sums[2])+(nu*sums[3]) - (1/deltat),(
u_star[i_global,j_global].Value*b[0,0]) + (v_star[i_global,j_global].Value*b[1,0])-(nu*b[2,0])-
(nu*b[3,0]),(u_star[i_global,j_global].Value*b[0,1]) + (v_star[i_global,j_global].Value*b[1,1])
-(nu*b[2,1])-(nu*b[3,1]),(u_star[i_global,j_global].Value*b[0,2]) + (v_star[i_global,j_global].
Value*b[1,2])-(nu*b[2,2])-(nu*b[3,2]),(u_star[i_global,j_global].Value*b[0,3]) + (v_star[i_glob
al,j_global].Value*b[1,3])-(nu*b[2,3])-(nu*b[3,3]),(u_star[i_global,j_global].Value*b[0,4]) + (
v_star[i_global,j_global].Value*b[1,4])-(nu*b[2,4])-(nu*b[3,4])};double[] R = {-1*sums[1]/f_pro
ps.Density,b[1, 0] / rho,b[1, 1] / rho,b[1, 2] / rho,b[1, 3] / rho,b[1, 4] / rho};double[] S =
Q;double[] T = {sums[0],b[0,0],b[0,1],b[0,2],b[0,3],b[0,4]};double[] W = {sums[1],b[1,0],b[1,1]
,b[1,2],b[1,3],b[1,4]};//build pressure'int rowbase = 3 * current_row_cluster;if (i_global > 1
&& i_global < discretization.Xcount - 2 && j_global > 1 && j_global < discretization.Ycount - 2
){map_interior_node_eqs(rowbase, i_local, j_local, P, Q, R, S, T, W);}if (i_global == 1 && j_gl
obal != 1 && j_global != discretization.Ycount - 2){map_left_node_eqs(rowbase, i_local, j_local
, P, Q, R, S, T, W);}if (i_global == discretization.Xcount - 2 && j_global != 1 && j_global !=
discretization.Ycount - 2){map_right_node_eqs(rowbase, i_local, j_local, P, Q, R, S, T, W);}if
(j_global == 1 && i_global != 1 && i_global != discretization.Xcount - 2){map_lower_node_eqs(ro
wbase, i_local, j_local, P, Q, R, S, T, W);}if (j_global == discretization.Ycount - 2 && i_glob
al != 1 && i_global != discretization.Xcount - 2){map_upper_node_eqs(rowbase, i_local, j_local,
 P, Q, R, S, T, W);}if (j_global == 1 && i_global == 1){map_llcorner_node_eqs(rowbase, i_local,
 j_local, P, Q, R, S, T, W);}if (j_global == 1 && i_global == discretization.Xcount - 2){map_lr
corner_node_eqs(rowbase, i_local, j_local, P, Q, R, S, T, W);}if (j_global == discretization.Yc
ount - 2 && i_global == 1){map_ulcorner_node_eqs(rowbase, i_local, j_local, P, Q, R, S, T, W);}
if (j_global == discretization.Ycount - 2 && i_global == discretization.Xcount - 2){map_urcorne
r_node_eqs(rowbase, i_local, j_local, P, Q, R, S, T, W);}}}}private NCGrid_Distribution[] make_
from_matrix(Matrix doink){NCGrid_Distribution pout = prev_u.Clone();NCGrid_Distribution uout =
prev_u.Clone();NCGrid_Distribution vout = prev_u.Clone();pout.ApplyBoundary(p_boundary_conditio
ns);uout.ApplyBoundary(u_boundary_conditions);vout.ApplyBoundary(v_boundary_conditions);for (in
t i = 0; i < doink.Rows; i+= 3){double pij = doink[i];double uij = doink[i+1];double vij = doin
k[i+2];int[] pcoords = inverse_map_var(i, 'p');int[] ucoords = inverse_map_var(i+1, 'u');int[]
vcoords = inverse_map_var(i+2, 'v');pout.assign_value_at(pcoords[0], pcoords[1], pij);uout.assi
gn_value_at(ucoords[0], ucoords[1], uij);vout.assign_value_at(vcoords[0], vcoords[1], vij);}NCG
rid_Distribution[] output = new NCGrid_Distribution[3];output[0] = pout;output[1] = uout;output
[2] = vout;return output;}private void map_right_node_eqs(int rowbase, int i_local, int j_local
, double[] P, double[] Q, double[] R, double[] S, double[] T, double[] W){int global_i = i_loca
l + 1;int global_j = j_local + 1;rhs[rowbase] = (-1 / deltat) * prev_u[i_local + 1, j_local + 1
].Value;rhs[rowbase + 1] = (-1 / deltat) * prev_v[i_local + 1, j_local + 1].Value;rhs[rowbase +
 2] = 0;//x-momentumsystem_matrix[rowbase, map_var(i_local, j_local, 'p')] = P[0];//system_matr
ix[rowbase, map_var(i_local + 1, j_local, 'p')] = P[1];system_matrix[rowbase, map_var(i_local,
j_local + 1, 'p')] = P[2];system_matrix[rowbase, map_var(i_local - 1, j_local, 'p')] = P[3];sys
tem_matrix[rowbase, map_var(i_local, j_local - 1, 'p')] = P[4];//system_matrix[rowbase, map_var
(i_local + 1, j_local + 1, 'p')] = P[5];system_matrix[rowbase, map_var(i_local, j_local, 'u')]
= Q[0];//system_matrix[rowbase, map_var(i_local + 1, j_local, 'u')] = Q[1];system_matrix[rowbas
e, map_var(i_local, j_local + 1, 'u')] = Q[2];system_matrix[rowbase, map_var(i_local - 1, j_loc
al, 'u')] = Q[3];system_matrix[rowbase, map_var(i_local, j_local - 1, 'u')] = Q[4];//system_mat
rix[rowbase, map_var(i_local + 1, j_local + 1, 'u')] = Q[5];rhs[rowbase] -= p_boundary_conditio
ns[global_j, BoundaryConditions.Direction.Positive_X] * P[1];rhs[rowbase] -= p_boundary_conditi
ons[global_j + 1, BoundaryConditions.Direction.Positive_X] * P[5];rhs[rowbase] -= u_boundary_co
nditions[global_j, BoundaryConditions.Direction.Positive_X] * Q[1];rhs[rowbase] -= u_boundary_c
onditions[global_j + 1, BoundaryConditions.Direction.Positive_X] * Q[5];//y-momentumsystem_matr
ix[rowbase + 1, map_var(i_local, j_local, 'p')] = R[0];//system_matrix[rowbase + 1, map_var(i_l
ocal + 1, j_local, 'p')] = R[1];system_matrix[rowbase + 1, map_var(i_local, j_local + 1, 'p')]
= R[2];system_matrix[rowbase + 1, map_var(i_local - 1, j_local, 'p')] = R[3];system_matrix[rowb
ase + 1, map_var(i_local, j_local - 1, 'p')] = R[4];//system_matrix[rowbase + 1, map_var(i_loca
l + 1, j_local + 1, 'p')] = R[5];system_matrix[rowbase + 1, map_var(i_local, j_local, 'v')] = S
[0];//system_matrix[rowbase + 1, map_var(i_local + 1, j_local, 'v')] = S[1];system_matrix[rowba
se + 1, map_var(i_local, j_local + 1, 'v')] = S[2];system_matrix[rowbase + 1, map_var(i_local -
```

```
 1, j_local, 'v')] = S[3];system_matrix[rowbase + 1, map_var(i_local, j_local - 1, 'v')] = S[4]
;//system_matrix[rowbase + 1, map_var(i_local + 1, j_local + 1, 'v')] = S[5];rhs[rowbase + 1] -
= p_boundary_conditions[global_j, BoundaryConditions.Direction.Positive_X] * R[1];rhs[rowbase +
 1] -= p_boundary_conditions[global_j + 1, BoundaryConditions.Direction.Positive_X] * R[5];rhs[
rowbase + 1] -= v_boundary_conditions[global_j, BoundaryConditions.Direction.Positive_X] * R[1]
;rhs[rowbase + 1] -= v_boundary_conditions[global_j + 1, BoundaryConditions.Direction.Positive_
X] * R[5];//continuitysystem_matrix[rowbase + 2, map_var(i_local, j_local, 'u')] = T[0];//syste
m_matrix[rowbase + 2, map_var(i_local + 1, j_local, 'u')] = T[1];system_matrix[rowbase + 2, map
_var(i_local, j_local + 1, 'u')] = T[2];system_matrix[rowbase + 2, map_var(i_local - 1, j_local
, 'u')] = T[3];system_matrix[rowbase + 2, map_var(i_local, j_local - 1, 'u')] = T[4];//system_m
atrix[rowbase + 2, map_var(i_local + 1, j_local + 1, 'u')] = T[5];system_matrix[rowbase + 2, ma
p_var(i_local, j_local, 'v')] = W[0];//system_matrix[rowbase + 2, map_var(i_local + 1, j_local,
 'v')] = W[1];system_matrix[rowbase + 2, map_var(i_local, j_local + 1, 'v')] = W[2];system_matr
ix[rowbase + 2, map_var(i_local - 1, j_local, 'v')] = W[3];system_matrix[rowbase + 2, map_var(i
_local, j_local - 1, 'v')] = W[4];//system_matrix[rowbase + 2, map_var(i_local + 1, j_local + 1
, 'v')] = W[5];rhs[rowbase + 2] -= u_boundary_conditions[global_j, BoundaryConditions.Direction
.Positive_X] * T[1];rhs[rowbase + 2] -= u_boundary_conditions[global_j + 1, BoundaryConditions.
Direction.Positive_X] * T[5];rhs[rowbase + 2] -= v_boundary_conditions[global_j, BoundaryCondit
ions.Direction.Positive_X] * W[1];rhs[rowbase + 2] -= v_boundary_conditions[global_j + 1, Bound
aryConditions.Direction.Positive_X] * W[5];}private void map_upper_node_eqs(int rowbase, int i_
local, int j_local, double[] P, double[] Q, double[] R, double[] S, double[] T, double[] W){int
 global_i = i_local + 1;int global_j = j_local + 1;rhs[rowbase] = (-1 / deltat) * prev_u[i_loca
l + 1, j_local + 1].Value;rhs[rowbase + 1] = (-1 / deltat) * prev_v[i_local + 1, j_local + 1].V
alue;rhs[rowbase + 2] = 0;//x-momentumsystem_matrix[rowbase, map_var(i_local, j_local, 'p')] =
P[0];system_matrix[rowbase, map_var(i_local + 1, j_local, 'p')] = P[1];//system_matrix[rowbase,
 map_var(i_local, j_local + 1, 'p')] = P[2];//system_matrix[rowbase, map_var(i_local - 1, j_loc
al, 'p')] = P[3];system_matrix[rowbase, map_var(i_local, j_local - 1, 'p')] = P[4];//system_mat
rix[rowbase, map_var(i_local + 1, j_local + 1, 'p')] = P[5];//system_matrix[rowbase, map_var(i_
local, j_local, 'u')] = Q[0];system_matrix[rowbase, map_var(i_local + 1, j_local, 'u')] = Q[1];
//system_matrix[rowbase, map_var(i_local, j_local + 1, 'u')] = Q[2];//system_matrix[rowbase, ma
p_var(i_local - 1, j_local, 'u')] = Q[3];system_matrix[rowbase, map_var(i_local, j_local - 1, '
u')] = Q[4];//system_matrix[rowbase, map_var(i_local + 1, j_local + 1, 'u')] = Q[5];//rhs[rowba
se] -= p_boundary_conditions[global_i, BoundaryConditions.Direction.Positive_Y] * P[2];rhs[rowb
ase] -= p_boundary_conditions[global_i + 1, BoundaryConditions.Direction.Positive_Y] * P[5];rhs
[rowbase] -= u_boundary_conditions[global_i, BoundaryConditions.Direction.Positive_Y] * Q[2];rh
s[rowbase] -= u_boundary_conditions[global_i + 1, BoundaryConditions.Direction.Positive_Y] * Q[
5];//y-momentumsystem_matrix[rowbase + 1, map_var(i_local, j_local, 'p')] = R[0];system_matrix[
rowbase + 1, map_var(i_local + 1, j_local, 'p')] = R[1];//system_matrix[rowbase + 1, map_var(i_
local, j_local + 1, 'p')] = R[2];//system_matrix[rowbase + 1, map_var(i_local - 1, j_local, 'p'
)] = R[3];system_matrix[rowbase + 1, map_var(i_local, j_local - 1, 'p')] = R[4];//system_matrix
[rowbase + 1, map_var(i_local + 1, j_local + 1, 'p')] = R[5];//system_matrix[rowbase + 1, map_v
ar(i_local, j_local, 'v')] = S[0];system_matrix[rowbase + 1, map_var(i_local + 1, j_local, 'v')
] = S[1];//system_matrix[rowbase + 1, map_var(i_local, j_local + 1, 'v')] = S[2];//system_matri
x[rowbase + 1, map_var(i_local - 1, j_local, 'v')] = S[3];system_matrix[rowbase + 1, map_var(i_
local, j_local - 1, 'v')] = S[4];//system_matrix[rowbase + 1, map_var(i_local + 1, j_local + 1,
 'v')] = S[5];//rhs[rowbase + 1] -= p_boundary_conditions[global_i, BoundaryConditions.Directio
n.Positive_Y] * R[2];rhs[rowbase + 1] -= p_boundary_conditions[global_i + 1, BoundaryConditions
.Direction.Positive_Y] * R[5];rhs[rowbase + 1] -= v_boundary_conditions[global_i, BoundaryCondi
tions.Direction.Positive_Y] * R[2];rhs[rowbase + 1] -= v_boundary_conditions[global_i + 1, Boun
daryConditions.Direction.Positive_Y] * R[5];//continuitysystem_matrix[rowbase + 2, map_var(i_lo
cal, j_local, 'u')] = T[0];system_matrix[rowbase + 2, map_var(i_local + 1, j_local, 'u')] = T[1
];//system_matrix[rowbase + 2, map_var(i_local, j_local + 1, 'u')] = T[2];//system_matrix[rowba
se + 2, map_var(i_local - 1, j_local, 'u')] = T[3];system_matrix[rowbase + 2, map_var(i_local,
j_local - 1, 'u')] = T[4];//system_matrix[rowbase + 2, map_var(i_local + 1, j_local + 1, 'u')]
= T[5];//system_matrix[rowbase + 2, map_var(i_local, j_local, 'v')] = W[0];system_matrix[rowbas
e + 2, map_var(i_local + 1, j_local, 'v')] = W[1];//system_matrix[rowbase + 2, map_var(i_local,
```

```
 j_local + 1, 'v')] = W[2];//system_matrix[rowbase + 2, map_var(i_local - 1, j_local, 'v')] = W
[3];system_matrix[rowbase + 2, map_var(i_local, j_local - 1, 'v')] = W[4];//system_matrix[rowba
se + 2, map_var(i_local + 1, j_local + 1, 'v')] = W[5];//rhs[rowbase + 2] -= u_boundary_conditi
ons[global_i, BoundaryConditions.Direction.Positive_Y] * T[2];rhs[rowbase + 2] -= u_boundary_co
nditions[global_i + 1, BoundaryConditions.Direction.Positive_Y] * T[5];rhs[rowbase + 2] -= v_bo
undary_conditions[global_i, BoundaryConditions.Direction.Positive_Y] * W[2];rhs[rowbase + 2] -=
 v_boundary_conditions[global_i + 1, BoundaryConditions.Direction.Positive_Y] * W[5];}private v
oid map_lower_node_eqs(int rowbase, int i_local, int j_local, double[] P, double[] Q, double[]
R, double[] S, double[] T, double[] W){int global_i = i_local + 1;int global_j = j_local + 1;rh
s[rowbase] = (-1 / deltat) * prev_u[i_local + 1, j_local + 1].Value;rhs[rowbase + 1] = (-1 / de
ltat) * prev_v[i_local + 1, j_local + 1].Value;rhs[rowbase + 2] = 0;//x-momentumsystem_matrix[r
owbase, map_var(i_local, j_local, 'p')] = P[0];system_matrix[rowbase, map_var(i_local + 1, j_lo
cal, 'p')] = P[1];system_matrix[rowbase, map_var(i_local, j_local + 1, 'p')] = P[2];system_matr
ix[rowbase, map_var(i_local - 1, j_local, 'p')] = P[3];//system_matrix[rowbase, map_var(i_local
, j_local - 1, 'p')] = P[4];system_matrix[rowbase, map_var(i_local + 1, j_local + 1, 'p')] = P[
5]; //problem heresystem_matrix[rowbase, map_var(i_local, j_local, 'u')] = Q[0];system_matrix[r
owbase, map_var(i_local + 1, j_local, 'u')] = Q[1];system_matrix[rowbase, map_var(i_local, j_lo
cal + 1, 'u')] = Q[2];system_matrix[rowbase, map_var(i_local - 1, j_local, 'u')] = Q[3];//syste
m_matrix[rowbase, map_var(i_local, j_local - 1, 'u')] = Q[4];system_matrix[rowbase, map_var(i_l
ocal + 1, j_local + 1, 'u')] = Q[5];rhs[rowbase] -= p_boundary_conditions[global_i, BoundaryCon
ditions.Direction.Negative_Y] * P[4];rhs[rowbase] -= u_boundary_conditions[global_i, BoundaryCo
nditions.Direction.Negative_Y] * Q[4];//y-momentumsystem_matrix[rowbase + 1, map_var(i_local, j
_local, 'p')] = R[0];system_matrix[rowbase + 1, map_var(i_local + 1, j_local, 'p')] = R[1];syst
em_matrix[rowbase + 1, map_var(i_local, j_local + 1, 'p')] = R[2];system_matrix[rowbase + 1, ma
p_var(i_local - 1, j_local, 'p')] = R[3];//system_matrix[rowbase + 1, map_var(i_local, j_local
- 1, 'p')] = R[4];system_matrix[rowbase + 1, map_var(i_local + 1, j_local + 1, 'p')] = R[5];sys
tem_matrix[rowbase + 1, map_var(i_local, j_local, 'v')] = S[0];system_matrix[rowbase + 1, map_v
ar(i_local + 1, j_local, 'v')] = S[1];system_matrix[rowbase + 1, map_var(i_local, j_local + 1,
'v')] = S[2];system_matrix[rowbase + 1, map_var(i_local - 1, j_local, 'v')] = S[3];//system_mat
rix[rowbase + 1, map_var(i_local, j_local - 1, 'v')] = S[4];system_matrix[rowbase + 1, map_var(
i_local + 1, j_local + 1, 'v')] = S[5];rhs[rowbase+1] -= p_boundary_conditions[global_i, Bounda
ryConditions.Direction.Negative_Y] * R[4];rhs[rowbase+1] -= v_boundary_conditions[global_i, Bou
ndaryConditions.Direction.Negative_Y] * S[4];//continuitysystem_matrix[rowbase + 2, map_var(i_l
ocal, j_local, 'u')] = T[0];system_matrix[rowbase + 2, map_var(i_local + 1, j_local, 'u')] = T[
1];system_matrix[rowbase + 2, map_var(i_local, j_local + 1, 'u')] = T[2];system_matrix[rowbase
+ 2, map_var(i_local - 1, j_local, 'u')] = T[3];//system_matrix[rowbase + 2, map_var(i_local, j
_local - 1, 'u')] = T[4];system_matrix[rowbase + 2, map_var(i_local + 1, j_local + 1, 'u')] = T
[5];system_matrix[rowbase + 2, map_var(i_local, j_local, 'v')] = W[0];system_matrix[rowbase + 2
, map_var(i_local + 1, j_local, 'v')] = W[1];system_matrix[rowbase + 2, map_var(i_local, j_loca
l + 1, 'v')] = W[2];system_matrix[rowbase + 2, map_var(i_local - 1, j_local, 'v')] = W[3];//sys
tem_matrix[rowbase + 2, map_var(i_local, j_local - 1, 'v')] = W[4];system_matrix[rowbase + 2, m
ap_var(i_local + 1, j_local + 1, 'v')] = W[5];rhs[rowbase+2] -= u_boundary_conditions[global_i,
 BoundaryConditions.Direction.Negative_Y] * T[4];rhs[rowbase+2] -= v_boundary_conditions[global
_i, BoundaryConditions.Direction.Negative_Y] * W[4];}private void map_left_node_eqs(int rowbase
, int i_local, int j_local, double[] P, double[] Q, double[] R, double[] S, double[] T, double[
] W){int global_i = i_local + 1;int global_j = j_local + 1;rhs[rowbase] = (-1 / deltat) * prev_
u[i_local + 1, j_local + 1].Value;rhs[rowbase + 1] = (-1 / deltat) * prev_v[i_local + 1, j_loca
l + 1].Value;rhs[rowbase + 2] = 0;//x-momentumsystem_matrix[rowbase, map_var(i_local, j_local,
'p')] = P[0];system_matrix[rowbase, map_var(i_local + 1, j_local, 'p')] = P[1];system_matrix[ro
wbase, map_var(i_local, j_local + 1, 'p')] = P[2];//system_matrix[rowbase, map_var(i_local - 1,
 j_local, 'p')] = P[3];system_matrix[rowbase, map_var(i_local, j_local - 1, 'p')] = P[4];system
_matrix[rowbase, map_var(i_local + 1, j_local + 1, 'p')] = P[5];system_matrix[rowbase, map_var(
i_local, j_local, 'u')] = Q[0];system_matrix[rowbase, map_var(i_local + 1, j_local, 'u')] = Q[1
];system_matrix[rowbase, map_var(i_local, j_local + 1, 'u')] = Q[2];//system_matrix[rowbase, ma
p_var(i_local - 1, j_local, 'u')] = Q[3];system_matrix[rowbase, map_var(i_local, j_local - 1, '
u')] = Q[4];system_matrix[rowbase, map_var(i_local + 1, j_local + 1, 'u')] = Q[5];rhs[rowbase]
```

```
-= p_boundary_conditions[global_j, BoundaryConditions.Direction.Negative_X] * P[3];rhs[rowbase]
 -= u_boundary_conditions[global_j, BoundaryConditions.Direction.Negative_X] * Q[3];//y-momentu
msystem_matrix[rowbase + 1, map_var(i_local, j_local, 'p')] = R[0];system_matrix[rowbase + 1, m
ap_var(i_local + 1, j_local, 'p')] = R[1];system_matrix[rowbase + 1, map_var(i_local, j_local +
 1, 'p')] = R[2];//system_matrix[rowbase + 1, map_var(i_local - 1, j_local, 'p')] = R[3];system
_matrix[rowbase + 1, map_var(i_local, j_local - 1, 'p')] = R[4];system_matrix[rowbase + 1, map_
var(i_local + 1, j_local + 1, 'p')] = R[5];system_matrix[rowbase + 1, map_var(i_local, j_local,
 'v')] = S[0];system_matrix[rowbase + 1, map_var(i_local + 1, j_local, 'v')] = S[1];system_matr
ix[rowbase + 1, map_var(i_local, j_local + 1, 'v')] = S[2];//system_matrix[rowbase + 1, map_var
(i_local - 1, j_local, 'v')] = S[3];system_matrix[rowbase + 1, map_var(i_local, j_local - 1, 'v
')] = S[4];system_matrix[rowbase + 1, map_var(i_local + 1, j_local + 1, 'v')] = S[5];rhs[rowbas
e + 1] -= p_boundary_conditions[global_j, BoundaryConditions.Direction.Negative_X] * R[3];rhs[r
owbase + 1] -= v_boundary_conditions[global_j, BoundaryConditions.Direction.Negative_X] * S[3];
//continuitysystem_matrix[rowbase + 2, map_var(i_local, j_local, 'u')] = T[0];system_matrix[row
base + 2, map_var(i_local + 1, j_local, 'u')] = T[1];system_matrix[rowbase + 2, map_var(i_local
, j_local + 1, 'u')] = T[2];//system_matrix[rowbase + 2, map_var(i_local - 1, j_local, 'u')] =
T[3];system_matrix[rowbase + 2, map_var(i_local, j_local - 1, 'u')] = T[4];system_matrix[rowbas
e + 2, map_var(i_local + 1, j_local + 1, 'u')] = T[5];system_matrix[rowbase + 2, map_var(i_loca
l, j_local, 'v')] = W[0];system_matrix[rowbase + 2, map_var(i_local + 1, j_local, 'v')] = W[1];
system_matrix[rowbase + 2, map_var(i_local, j_local + 1, 'v')] = W[2];//system_matrix[rowbase +
 2, map_var(i_local - 1, j_local, 'v')] = W[3];system_matrix[rowbase + 2, map_var(i_local, j_lo
cal - 1, 'v')] = W[4];system_matrix[rowbase + 2, map_var(i_local + 1, j_local + 1, 'v')] = W[5]
;rhs[rowbase + 2] -= u_boundary_conditions[global_j, BoundaryConditions.Direction.Negative_X] *
 T[3];rhs[rowbase + 2] -= v_boundary_conditions[global_j, BoundaryConditions.Direction.Negative
_X] * W[3];}private void map_ulcorner_node_eqs(int rowbase, int i_local, int j_local, double[]
P, double[] Q, double[] R, double[] S, double[] T, double[] W){int global_i = i_local + 1;int g
lobal_j = j_local + 1;rhs[rowbase] = (-1 / deltat) * prev_u[i_local + 1, j_local + 1].Value;rhs
[rowbase + 1] = (-1 / deltat) * prev_v[i_local + 1, j_local + 1].Value;rhs[rowbase + 2] = 0;//x
-momentumsystem_matrix[rowbase, map_var(i_local, j_local, 'p')] = P[0];system_matrix[rowbase, m
ap_var(i_local + 1, j_local, 'p')] = P[1];//system_matrix[rowbase, map_var(i_local, j_local + 1
, 'p')] = P[2];//system_matrix[rowbase, map_var(i_local - 1, j_local, 'p')] = P[3];system_matri
x[rowbase, map_var(i_local, j_local - 1, 'p')] = P[4];//system_matrix[rowbase, map_var(i_local
+ 1, j_local + 1, 'p')] = P[5];system_matrix[rowbase, map_var(i_local, j_local, 'u')] = Q[0];sy
stem_matrix[rowbase, map_var(i_local + 1, j_local, 'u')] = Q[1];//system_matrix[rowbase, map_va
r(i_local, j_local + 1, 'u')] = Q[2];//system_matrix[rowbase, map_var(i_local - 1, j_local, 'u'
)] = Q[3];system_matrix[rowbase, map_var(i_local, j_local - 1, 'u')] = Q[4];//system_matrix[row
base, map_var(i_local + 1, j_local + 1, 'u')] = Q[5];rhs[rowbase] -= p_boundary_conditions[glob
al_i, BoundaryConditions.Direction.Positive_Y] * P[1];rhs[rowbase] -= p_boundary_conditions[glo
bal_i + 1, BoundaryConditions.Direction.Positive_Y] * P[5];rhs[rowbase] -= p_boundary_condition
s[global_j, BoundaryConditions.Direction.Negative_X] * P[3];rhs[rowbase] -= u_boundary_conditio
ns[global_i, BoundaryConditions.Direction.Positive_Y] * Q[1];rhs[rowbase] -= u_boundary_conditi
ons[global_i + 1, BoundaryConditions.Direction.Positive_Y] * Q[5];rhs[rowbase] -= u_boundary_co
nditions[global_j, BoundaryConditions.Direction.Negative_X] *Q[3];//y-momentumsystem_matrix[row
base + 1, map_var(i_local, j_local, 'p')] = R[0];system_matrix[rowbase + 1, map_var(i_local + 1
, j_local, 'p')] = R[1];//system_matrix[rowbase + 1, map_var(i_local, j_local + 1, 'p')] = R[2]
;//system_matrix[rowbase + 1, map_var(i_local - 1, j_local, 'p')] = R[3];system_matrix[rowbase
+ 1, map_var(i_local, j_local - 1, 'p')] = R[4];//system_matrix[rowbase + 1, map_var(i_local +
1, j_local + 1, 'p')] = R[5];system_matrix[rowbase + 1, map_var(i_local, j_local, 'v')] = S[0];
system_matrix[rowbase + 1, map_var(i_local + 1, j_local, 'v')] = S[1];//system_matrix[rowbase +
 1, map_var(i_local, j_local + 1, 'v')] = S[2];//system_matrix[rowbase + 1, map_var(i_local - 1
, j_local, 'v')] = S[3];system_matrix[rowbase + 1, map_var(i_local, j_local - 1, 'v')] = S[4];/
/system_matrix[rowbase + 1, map_var(i_local + 1, j_local + 1, 'v')] = S[5];rhs[rowbase+1] -= p_
boundary_conditions[global_i, BoundaryConditions.Direction.Positive_Y] * R[1];rhs[rowbase+1] -=
 p_boundary_conditions[global_i + 1, BoundaryConditions.Direction.Positive_Y] * R[5];rhs[rowbas
e+1] -= p_boundary_conditions[global_j, BoundaryConditions.Direction.Negative_X] * R[3];rhs[row
base+1] -= v_boundary_conditions[global_i, BoundaryConditions.Direction.Positive_Y] * S[1];rhs[
```

```
rowbase+1] -= v_boundary_conditions[global_i + 1, BoundaryConditions.Direction.Positive_Y] * S[
5];rhs[rowbase+1] -= v_boundary_conditions[global_j, BoundaryConditions.Direction.Negative_X] *
 S[3];//continuitysystem_matrix[rowbase + 2, map_var(i_local, j_local, 'u')] = T[0];system_matr
ix[rowbase + 2, map_var(i_local + 1, j_local, 'u')] = T[1];//system_matrix[rowbase + 2, map_var
(i_local, j_local + 1, 'u')] = T[2];//system_matrix[rowbase + 2, map_var(i_local - 1, j_local,
'u')] = T[3];system_matrix[rowbase + 2, map_var(i_local, j_local - 1, 'u')] = T[4];//system_mat
rix[rowbase + 2, map_var(i_local + 1, j_local + 1, 'u')] = T[5];system_matrix[rowbase + 2, map_
var(i_local, j_local, 'v')] = W[0];system_matrix[rowbase + 2, map_var(i_local + 1, j_local, 'v'
)] = W[1];//system_matrix[rowbase + 2, map_var(i_local, j_local + 1, 'v')] = W[2];//system_matr
ix[rowbase + 2, map_var(i_local - 1, j_local, 'v')] = W[3];system_matrix[rowbase + 2, map_var(i
_local, j_local - 1, 'v')] = W[4];//system_matrix[rowbase + 2, map_var(i_local + 1, j_local + 1
, 'v')] = W[5];rhs[rowbase+2] -= u_boundary_conditions[global_i, BoundaryConditions.Direction.P
ositive_Y] * T[1];rhs[rowbase+2] -= u_boundary_conditions[global_i + 1, BoundaryConditions.Dire
ction.Positive_Y] * T[5];rhs[rowbase+2] -= u_boundary_conditions[global_j, BoundaryConditions.D
irection.Negative_X] * T[3];rhs[rowbase+2] -= v_boundary_conditions[global_i, BoundaryCondition
s.Direction.Positive_Y] * W[1];rhs[rowbase+2] -= v_boundary_conditions[global_i + 1, BoundaryCo
nditions.Direction.Positive_Y] * W[5];rhs[rowbase+2] -= v_boundary_conditions[global_j, Boundar
yConditions.Direction.Negative_X] * W[3];}private void map_urcorner_node_eqs(int rowbase, int i
_local, int j_local, double[] P, double[] Q, double[] R, double[] S, double[] T, double[] W){in
t global_i = i_local + 1;int global_j = j_local + 1;rhs[rowbase] = (-1 / deltat) * prev_u[i_loc
al + 1, j_local + 1].Value;rhs[rowbase + 1] = (-1 / deltat) * prev_v[i_local + 1, j_local + 1].
Value;rhs[rowbase + 2] = 0;//x-momentumsystem_matrix[rowbase, map_var(i_local, j_local, 'p')] =
 P[0];//system_matrix[rowbase, map_var(i_local + 1, j_local, 'p')] = P[1];//system_matrix[rowba
se, map_var(i_local, j_local + 1, 'p')] = P[2];system_matrix[rowbase, map_var(i_local - 1, j_lo
cal, 'p')] = P[3];system_matrix[rowbase, map_var(i_local, j_local - 1, 'p')] = P[4];//system_ma
trix[rowbase, map_var(i_local + 1, j_local + 1, 'p')] = P[5];system_matrix[rowbase, map_var(i_l
ocal, j_local, 'u')] = Q[0];//system_matrix[rowbase, map_var(i_local + 1, j_local, 'u')] = Q[1]
;//system_matrix[rowbase, map_var(i_local, j_local + 1, 'u')] = Q[2];system_matrix[rowbase, map
_var(i_local - 1, j_local, 'u')] = Q[3];system_matrix[rowbase, map_var(i_local, j_local - 1, 'u
')] = Q[4];//system_matrix[rowbase, map_var(i_local + 1, j_local + 1, 'u')] = Q[5];rhs[rowbase]
 -= p_boundary_conditions[global_i, BoundaryConditions.Direction.Positive_Y] * P[1];rhs[rowbase
] -= p_boundary_conditions[global_i + 1, BoundaryConditions.Direction.Positive_Y] * P[5];rhs[ro
wbase] -= p_boundary_conditions[global_j, BoundaryConditions.Direction.Positive_X] * P[2];rhs[r
owbase] -= u_boundary_conditions[global_i, BoundaryConditions.Direction.Positive_Y] * Q[1];rhs[
rowbase] -= u_boundary_conditions[global_i + 1, BoundaryConditions.Direction.Positive_Y] * Q[5]
;rhs[rowbase] -= u_boundary_conditions[global_j, BoundaryConditions.Direction.Positive_X] * Q[2
];//y-momentumsystem_matrix[rowbase + 1, map_var(i_local, j_local, 'p')] = R[0];//system_matrix
[rowbase + 1, map_var(i_local + 1, j_local, 'p')] = R[1];//system_matrix[rowbase + 1, map_var(i
_local, j_local + 1, 'p')] = R[2];system_matrix[rowbase + 1, map_var(i_local - 1, j_local, 'p')
] = R[3];system_matrix[rowbase + 1, map_var(i_local, j_local - 1, 'p')] = R[4];//system_matrix[
rowbase + 1, map_var(i_local + 1, j_local + 1, 'p')] = R[5];system_matrix[rowbase + 1, map_var(
i_local, j_local, 'v')] = S[0];//system_matrix[rowbase + 1, map_var(i_local + 1, j_local, 'v')]
 = S[1];//system_matrix[rowbase + 1, map_var(i_local, j_local + 1, 'v')] = S[2];system_matrix[r
owbase + 1, map_var(i_local - 1, j_local, 'v')] = S[3];system_matrix[rowbase + 1, map_var(i_loc
al, j_local - 1, 'v')] = S[4];//system_matrix[rowbase + 1, map_var(i_local + 1, j_local + 1, 'v
')] = S[5];rhs[rowbase + 1] -= p_boundary_conditions[global_i, BoundaryConditions.Direction.Pos
itive_Y] * R[1];rhs[rowbase + 1] -= p_boundary_conditions[global_i + 1, BoundaryConditions.Dire
ction.Positive_Y] * R[5];rhs[rowbase + 1] -= p_boundary_conditions[global_j, BoundaryConditions
.Direction.Positive_X] * R[2];rhs[rowbase + 1] -= v_boundary_conditions[global_i, BoundaryCondi
tions.Direction.Positive_Y] * S[1];rhs[rowbase + 1] -= v_boundary_conditions[global_i + 1, Boun
daryConditions.Direction.Positive_Y] * S[5];rhs[rowbase + 1] -= v_boundary_conditions[global_j,
 BoundaryConditions.Direction.Positive_X] * S[2];//continuitysystem_matrix[rowbase + 2, map_var
(i_local, j_local, 'u')] = T[0];//system_matrix[rowbase + 2, map_var(i_local + 1, j_local, 'u')
] = T[1];//system_matrix[rowbase + 2, map_var(i_local, j_local + 1, 'u')] = T[2];system_matrix[
rowbase + 2, map_var(i_local - 1, j_local, 'u')] = T[3];system_matrix[rowbase + 2, map_var(i_lo
cal, j_local - 1, 'u')] = T[4];//system_matrix[rowbase + 2, map_var(i_local + 1, j_local + 1, '
```

```
u')] = T[5];system_matrix[rowbase + 2, map_var(i_local, j_local, 'v')] = W[0];//system_matrix[r
owbase + 2, map_var(i_local + 1, j_local, 'v')] = W[1];//system_matrix[rowbase + 2, map_var(i_l
ocal, j_local + 1, 'v')] = W[2];system_matrix[rowbase + 2, map_var(i_local - 1, j_local, 'v')]
= W[3];system_matrix[rowbase + 2, map_var(i_local, j_local - 1, 'v')] = W[4];//system_matrix[ro
wbase + 2, map_var(i_local + 1, j_local + 1, 'v')] = W[5];rhs[rowbase + 2] -= u_boundary_condit
ions[global_i, BoundaryConditions.Direction.Positive_Y] * T[1];rhs[rowbase + 2] -= u_boundary_c
onditions[global_i + 1, BoundaryConditions.Direction.Positive_Y] * T[5];rhs[rowbase + 2] -= u_b
oundary_conditions[global_j, BoundaryConditions.Direction.Positive_X] * T[2];rhs[rowbase + 2] -
= v_boundary_conditions[global_i, BoundaryConditions.Direction.Positive_Y] * W[1];rhs[rowbase +
 2] -= v_boundary_conditions[global_i + 1, BoundaryConditions.Direction.Positive_Y] * W[5];rhs[
rowbase + 2] -= v_boundary_conditions[global_j, BoundaryConditions.Direction.Positive_X] * W[2]
;}private void map_llcorner_node_eqs(int rowbase, int i_local, int j_local, double[] P, double[
] Q, double[] R, double[] S, double[] T, double[] W){int global_i = i_local + 1;int global_j =
j_local + 1;rhs[rowbase] = (-1 / deltat) * prev_u[i_local + 1, j_local + 1].Value;rhs[rowbase +
 1] = (-1 / deltat) * prev_v[i_local + 1, j_local + 1].Value;rhs[rowbase + 2] = 0;//x-momentums
ystem_matrix[rowbase, map_var(i_local, j_local, 'p')] = P[0];system_matrix[rowbase, map_var(i_l
ocal + 1, j_local, 'p')] = P[1];system_matrix[rowbase, map_var(i_local, j_local + 1, 'p')] = P[
2];//system_matrix[rowbase, map_var(i_local - 1, j_local, 'p')] = P[3];//system_matrix[rowbase,
 map_var(i_local, j_local - 1, 'p')] = P[4];system_matrix[rowbase, map_var(i_local + 1, j_local
 + 1, 'p')] = P[5];system_matrix[rowbase, map_var(i_local, j_local, 'u')] = Q[0];system_matrix[
rowbase, map_var(i_local + 1, j_local, 'u')] = Q[1];system_matrix[rowbase, map_var(i_local, j_l
ocal + 1, 'u')] = Q[2];//system_matrix[rowbase, map_var(i_local - 1, j_local, 'u')] = Q[3];//sy
stem_matrix[rowbase, map_var(i_local, j_local - 1, 'u')] = Q[4];system_matrix[rowbase, map_var(
i_local + 1, j_local + 1, 'u')] = Q[5];rhs[rowbase] -= p_boundary_conditions[global_i, Boundary
Conditions.Direction.Negative_Y] * P[4];rhs[rowbase] -= p_boundary_conditions[global_j, Boundar
yConditions.Direction.Negative_X] * P[3];rhs[rowbase] -= u_boundary_conditions[global_i, Bounda
ryConditions.Direction.Negative_Y] * Q[4];rhs[rowbase] -= u_boundary_conditions[global_j, Bound
aryConditions.Direction.Negative_X] * Q[3];//y-momentumsystem_matrix[rowbase + 1, map_var(i_loc
al, j_local, 'p')] = R[0];system_matrix[rowbase + 1, map_var(i_local + 1, j_local, 'p')] = R[1]
;system_matrix[rowbase + 1, map_var(i_local, j_local + 1, 'p')] = R[2];//system_matrix[rowbase
+ 1, map_var(i_local - 1, j_local, 'p')] = R[3];//system_matrix[rowbase + 1, map_var(i_local, j
_local - 1, 'p')] = R[4];system_matrix[rowbase + 1, map_var(i_local + 1, j_local + 1, 'p')] = R
[5];system_matrix[rowbase + 1, map_var(i_local, j_local, 'v')] = S[0];system_matrix[rowbase + 1
, map_var(i_local + 1, j_local, 'v')] = S[1];system_matrix[rowbase + 1, map_var(i_local, j_loca
l + 1, 'v')] = S[2];//system_matrix[rowbase + 1, map_var(i_local - 1, j_local, 'v')] = S[3];//s
ystem_matrix[rowbase + 1, map_var(i_local, j_local - 1, 'v')] = S[4];system_matrix[rowbase + 1,
 map_var(i_local + 1, j_local + 1, 'v')] = S[5];rhs[rowbase + 1] -= p_boundary_conditions[globa
l_i, BoundaryConditions.Direction.Negative_Y] * R[4];rhs[rowbase + 1] -= p_boundary_conditions[
global_j, BoundaryConditions.Direction.Negative_X] * R[3];rhs[rowbase + 1] -= v_boundary_condit
ions[global_i, BoundaryConditions.Direction.Negative_Y] * S[4];rhs[rowbase + 1] -= v_boundary_c
onditions[global_j, BoundaryConditions.Direction.Negative_X] * S[3];//continuitysystem_matrix[r
owbase + 2, map_var(i_local, j_local, 'u')] = T[0];system_matrix[rowbase + 2, map_var(i_local +
 1, j_local, 'u')] = T[1];system_matrix[rowbase + 2, map_var(i_local, j_local + 1, 'u')] = T[2]
;//system_matrix[rowbase + 2, map_var(i_local - 1, j_local, 'u')] = T[3];//system_matrix[rowbas
e + 2, map_var(i_local, j_local - 1, 'u')] = T[4];system_matrix[rowbase + 2, map_var(i_local +
1, j_local + 1, 'u')] = T[5];system_matrix[rowbase + 2, map_var(i_local, j_local, 'v')] = W[0];
system_matrix[rowbase + 2, map_var(i_local + 1, j_local, 'v')] = W[1];system_matrix[rowbase + 2
, map_var(i_local, j_local + 1, 'v')] = W[2];//system_matrix[rowbase + 2, map_var(i_local - 1,
j_local, 'v')] = W[3];//system_matrix[rowbase + 2, map_var(i_local, j_local - 1, 'v')] = W[4];s
ystem_matrix[rowbase + 2, map_var(i_local + 1, j_local + 1, 'v')] = W[5];rhs[rowbase + 2] -= u_
boundary_conditions[global_i, BoundaryConditions.Direction.Negative_Y] * T[4];rhs[rowbase + 2]
-= u_boundary_conditions[global_j, BoundaryConditions.Direction.Negative_X] * T[3];rhs[rowbase
+ 2] -= v_boundary_conditions[global_i, BoundaryConditions.Direction.Negative_Y] * W[4];rhs[row
base + 2] -= v_boundary_conditions[global_j, BoundaryConditions.Direction.Negative_X] * W[3];}p
rivate void map_lrcorner_node_eqs(int rowbase, int i_local, int j_local, double[] P, double[] Q
, double[] R, double[] S, double[] T, double[] W){int global_i = i_local + 1;int global_j = j_l
```

```
ocal + 1;rhs[rowbase] = (-1 / deltat) * prev_u[i_local + 1, j_local + 1].Value;rhs[rowbase + 1]
 = (-1 / deltat) * prev_v[i_local + 1, j_local + 1].Value;rhs[rowbase + 2] = 0;//x-momentumsyst
em_matrix[rowbase, map_var(i_local, j_local, 'p')] = P[0];//system_matrix[rowbase, map_var(i_lo
cal + 1, j_local, 'p')] = P[1];system_matrix[rowbase, map_var(i_local, j_local + 1, 'p')] = P[2
];system_matrix[rowbase, map_var(i_local - 1, j_local, 'p')] = P[3];//system_matrix[rowbase, ma
p_var(i_local, j_local - 1, 'p')] = P[4];//system_matrix[rowbase, map_var(i_local + 1, j_local
+ 1, 'p')] = P[5];system_matrix[rowbase, map_var(i_local, j_local, 'u')] = Q[0];//system_matrix
[rowbase, map_var(i_local + 1, j_local, 'u')] = Q[1];system_matrix[rowbase, map_var(i_local, j_
local + 1, 'u')] = Q[2];system_matrix[rowbase, map_var(i_local - 1, j_local, 'u')] = Q[3];//sys
tem_matrix[rowbase, map_var(i_local, j_local - 1, 'u')] = Q[4];//system_matrix[rowbase, map_var
(i_local + 1, j_local + 1, 'u')] = Q[5];rhs[rowbase] -= p_boundary_conditions[global_i, Boundar
yConditions.Direction.Negative_Y] * P[4];rhs[rowbase] -= p_boundary_conditions[global_j, Bounda
ryConditions.Direction.Positive_X] * P[1];rhs[rowbase] -= p_boundary_conditions[global_j+1, Bou
ndaryConditions.Direction.Positive_X] * P[5];rhs[rowbase] -= u_boundary_conditions[global_i, Bo
undaryConditions.Direction.Negative_Y] * Q[4];rhs[rowbase] -= u_boundary_conditions[global_j, B
oundaryConditions.Direction.Positive_X] * Q[1];rhs[rowbase] -= u_boundary_conditions[global_j+1
, BoundaryConditions.Direction.Positive_X] * Q[5];//y-momentumsystem_matrix[rowbase + 1, map_va
r(i_local, j_local, 'p')] = R[0];//system_matrix[rowbase + 1, map_var(i_local + 1, j_local, 'p'
)] = R[1];system_matrix[rowbase + 1, map_var(i_local, j_local + 1, 'p')] = R[2];system_matrix[r
owbase + 1, map_var(i_local - 1, j_local, 'p')] = R[3];//system_matrix[rowbase + 1, map_var(i_l
ocal, j_local - 1, 'p')] = R[4];//system_matrix[rowbase + 1, map_var(i_local + 1, j_local + 1,
'p')] = R[5];system_matrix[rowbase + 1, map_var(i_local, j_local, 'v')] = S[0];//system_matrix[
rowbase + 1, map_var(i_local + 1, j_local, 'v')] = S[1];system_matrix[rowbase + 1, map_var(i_lo
cal, j_local + 1, 'v')] = S[2];system_matrix[rowbase + 1, map_var(i_local - 1, j_local, 'v')] =
 S[3];//system_matrix[rowbase + 1, map_var(i_local, j_local - 1, 'v')] = S[4];//system_matrix[r
owbase + 1, map_var(i_local + 1, j_local + 1, 'v')] = S[5];rhs[rowbase + 1] -= p_boundary_condi
tions[global_i, BoundaryConditions.Direction.Negative_Y] * R[4];rhs[rowbase + 1] -= p_boundary_
conditions[global_j, BoundaryConditions.Direction.Positive_X] * R[1];rhs[rowbase + 1] -= p_boun
dary_conditions[global_j+1, BoundaryConditions.Direction.Positive_X] * R[5];rhs[rowbase + 1] -=
 v_boundary_conditions[global_i, BoundaryConditions.Direction.Negative_Y] * S[4];rhs[rowbase +
1] -= v_boundary_conditions[global_j, BoundaryConditions.Direction.Positive_X] * S[1];rhs[rowba
se + 1] -= v_boundary_conditions[global_j+1, BoundaryConditions.Direction.Positive_X] * S[5];//
continuitysystem_matrix[rowbase + 2, map_var(i_local, j_local, 'u')] = T[0];//system_matrix[row
base + 2, map_var(i_local + 1, j_local, 'u')] = T[1];system_matrix[rowbase + 2, map_var(i_local
, j_local + 1, 'u')] = T[2];system_matrix[rowbase + 2, map_var(i_local - 1, j_local, 'u')] = T[
3];//system_matrix[rowbase + 2, map_var(i_local, j_local - 1, 'u')] = T[4];//system_matrix[rowb
ase + 2, map_var(i_local + 1, j_local + 1, 'u')] = T[5];system_matrix[rowbase + 2, map_var(i_lo
cal, j_local, 'v')] = W[0];//system_matrix[rowbase + 2, map_var(i_local + 1, j_local, 'v')] = W
[1];system_matrix[rowbase + 2, map_var(i_local, j_local + 1, 'v')] = W[2];system_matrix[rowbase
 + 2, map_var(i_local - 1, j_local, 'v')] = W[3];//system_matrix[rowbase + 2, map_var(i_local,
j_local - 1, 'v')] = W[4];//system_matrix[rowbase + 2, map_var(i_local + 1, j_local + 1, 'v')]
= W[5];rhs[rowbase + 2] -= u_boundary_conditions[global_i, BoundaryConditions.Direction.Negativ
e_Y] * T[4];rhs[rowbase + 2] -= u_boundary_conditions[global_j, BoundaryConditions.Direction.Po
sitive_X] * T[1];rhs[rowbase + 2] -= u_boundary_conditions[global_j+1, BoundaryConditions.Direc
tion.Positive_X] * T[5];rhs[rowbase + 2] -= v_boundary_conditions[global_i, BoundaryConditions.
Direction.Negative_Y] * W[4];rhs[rowbase + 2] -= v_boundary_conditions[global_j, BoundaryCondit
ions.Direction.Positive_X] * W[1];rhs[rowbase + 2] -= v_boundary_conditions[global_j+1, Boundar
yConditions.Direction.Positive_X] * W[5];}private void map_interior_node_eqs(int rowbase, int i
_local, int j_local, double[] P, double[] Q, double [] R, double[] S, double[] T, double[] W){/
/x-momentumsystem_matrix[rowbase, map_var(i_local, j_local, 'p')] = P[0];system_matrix[rowbase,
 map_var(i_local + 1, j_local, 'p')] = P[1];system_matrix[rowbase, map_var(i_local, j_local + 1
, 'p')] = P[2];system_matrix[rowbase, map_var(i_local - 1, j_local, 'p')] = P[3];system_matrix[
rowbase, map_var(i_local, j_local - 1, 'p')] = P[4];system_matrix[rowbase, map_var(i_local + 1,
 j_local + 1, 'p')] = P[5];system_matrix[rowbase, map_var(i_local, j_local, 'u')] = Q[0];system
_matrix[rowbase, map_var(i_local + 1, j_local, 'u')] = Q[1];system_matrix[rowbase, map_var(i_lo
cal, j_local + 1, 'u')] = Q[2];system_matrix[rowbase, map_var(i_local - 1, j_local, 'u')] = Q[3
```

```
];system_matrix[rowbase, map_var(i_local, j_local - 1, 'u')] = Q[4];system_matrix[rowbase, map_
var(i_local + 1, j_local + 1, 'u')] = Q[5];//y-momentumsystem_matrix[rowbase + 1, map_var(i_loc
al, j_local, 'p')] = R[0];system_matrix[rowbase + 1, map_var(i_local + 1, j_local, 'p')] = R[1]
;system_matrix[rowbase + 1, map_var(i_local, j_local + 1, 'p')] = R[2];system_matrix[rowbase +
1, map_var(i_local - 1, j_local, 'p')] = R[3];system_matrix[rowbase + 1, map_var(i_local, j_loc
al - 1, 'p')] = R[4];system_matrix[rowbase + 1, map_var(i_local + 1, j_local + 1, 'p')] = R[5];
system_matrix[rowbase + 1, map_var(i_local, j_local, 'v')] = S[0];system_matrix[rowbase + 1, ma
p_var(i_local + 1, j_local, 'v')] = S[1];system_matrix[rowbase + 1, map_var(i_local, j_local +
1, 'v')] = S[2];system_matrix[rowbase + 1, map_var(i_local - 1, j_local, 'v')] = S[3];system_ma
trix[rowbase + 1, map_var(i_local, j_local - 1, 'v')] = S[4];system_matrix[rowbase + 1, map_var
(i_local + 1, j_local + 1, 'v')] = S[5];//continuitysystem_matrix[rowbase + 2, map_var(i_local,
 j_local, 'u')] = T[0];system_matrix[rowbase + 2, map_var(i_local + 1, j_local, 'u')] = T[1];sy
stem_matrix[rowbase + 2, map_var(i_local, j_local + 1, 'u')] = T[2];system_matrix[rowbase + 2,
map_var(i_local - 1, j_local, 'u')] = T[3];system_matrix[rowbase + 2, map_var(i_local, j_local
- 1, 'u')] = T[4];system_matrix[rowbase + 2, map_var(i_local + 1, j_local + 1, 'u')] = T[5];sys
tem_matrix[rowbase + 2, map_var(i_local, j_local, 'v')] = W[0];system_matrix[rowbase + 2, map_v
ar(i_local + 1, j_local, 'v')] = W[1];system_matrix[rowbase + 2, map_var(i_local, j_local + 1,
'v')] = W[2];system_matrix[rowbase + 2, map_var(i_local - 1, j_local, 'v')] = W[3];system_matri
x[rowbase + 2, map_var(i_local, j_local - 1, 'v')] = W[4];system_matrix[rowbase + 2, map_var(i_
local + 1, j_local + 1, 'v')] = W[5];rhs[rowbase] = (-1 / deltat) * prev_u[i_local+1, j_local+1
].Value;rhs[rowbase+1] = (-1 / deltat) * prev_v[i_local+1, j_local+1].Value;rhs[rowbase+2] = 0;
}private int map_var(int local_i, int local_j, char symbol){int local_offset = 0;switch (symbol
){case 'p':{local_offset = 0;break;}case 'u':{local_offset = 1;break;}case 'v':{local_offset =
2;break;}}int block = (local_j) + (eq_nodes_y * (local_i));int y =  (3 * block) + local_offset;
return y;}private int[] inverse_map_var(int index, char symbol){int local_offset = 0;switch (sy
mbol){case 'p':{local_offset = 0;break;}case 'u':{local_offset = 1;break;}case 'v':{local_offse
t = 2;break;}}int block = (index - local_offset) / 3;int local_j = block % eq_nodes_y;int local
_i = (block - local_j) / eq_nodes_y;int[] output = new int[2];output[0] = local_i + 1;output[1]
 = local_j + 1;return output;}}}/* Copyright (c) 2018 William van Noordt */using System;using S
ystem.Collections.Generic;using System.Linq;using System.Text;using System.IO;using System.Thre
ading.Tasks;namespace m_435_NCAMR{class NavierStokesTransient{int time_steps;NCGrid_Distributio
n current_ustar, current_vstar, initial_u, initial_v;BoundaryConditions p_boundary, u_boundary,
 v_boundary;bool enable_console_output = true;double deltat;FluidProperties properties;public b
ool ConsoleOutputEnabled{get { return enable_console_output; }set { enable_console_output = val
ue; }}public NavierStokesTransient(int timesteps, NCGrid_Distribution u_initial, NCGrid_Distrib
ution v_initial, BoundaryConditions p, BoundaryConditions u, BoundaryConditions v, FluidPropert
ies props, double dt){deltat = dt;properties = props;current_ustar = new NCGrid_Distribution(u_
initial.Bounds, u_initial.Xcount, u_initial.Ycount);current_vstar = new NCGrid_Distribution(v_i
nitial.Bounds, v_initial.Xcount, v_initial.Ycount);current_ustar.SetConstant(1);current_vstar.S
etConstant(1);time_steps = timesteps;p_boundary = p;u_boundary = u;v_boundary = v;initial_u = u
_initial;initial_v = v_initial;}public void Solve(){Solve("solution");}public void Solve(string
 title) // return p,u,v{string path = Paths.NavSolutionsRepo + "//" + title;if (Directory.Exist
s(path)){int n = 0;while (Directory.Exists(path + n.ToString())){n++;}path += n.ToString();}NCG
rid_Distribution[] last;Directory.CreateDirectory(path);Directory.CreateDirectory(path + "\\p")
;Directory.CreateDirectory(path + "\\u");Directory.CreateDirectory(path + "\\v");double t = 0;N
avierStokesProblemSingleFrame frame = new NavierStokesProblemSingleFrame(current_ustar, current
_vstar, initial_u, initial_v, p_boundary, u_boundary, v_boundary, properties, deltat);NCGrid_Di
stribution[] sols = frame.Solve();sols[0].WriteToFile(path + "\\p\\" + title + "-pressure-" + b
ufferint(0) + ".dist", false, true, true);sols[1].WriteToFile(path + "\\u\\" + title + "-x-vel-
" + bufferint(0) + ".dist", false, true, true);sols[2].WriteToFile(path + "\\v\\" + title + "-y
-vel-" + bufferint(0) + ".dist", false, true, true);last = sols;for (int i = 0; i < time_steps;
 i++){t += deltat;NavierStokesProblemSingleFrame newframe = new NavierStokesProblemSingleFrame(
last[1], last[2], last[1], last[2], u_boundary, v_boundary, p_boundary, properties, deltat);NCG
rid_Distribution[] solsnew = newframe.Solve();solsnew[0].WriteToFile(path + "\\p\\" + title + "
-pressure-" + bufferint(i+1) + ".dist", false, true, true);solsnew[1].WriteToFile(path + "\\u\\
" + title + "-x-vel-" + bufferint(i+1) + ".dist", false, true, true);solsnew[2].WriteToFile(pat
```

```
h + "\\v\\" + title + "-y-vel-" + bufferint(i+1) + ".dist", false, true, true);if (enable_conso
le_output){Console.ForegroundColor = ConsoleColor.Green;Console.WriteLine("Step " + i.ToString(
) + " of " + time_steps.ToString() + " complete." );Console.ForegroundColor = ConsoleColor.Gray
;}last[0] = solsnew[0].Clone();last[1] = solsnew[1].Clone();last[2] = solsnew[2].Clone();}}priv
ate string bufferint(int t){int numzeros = time_steps.ToString().Length - t.ToString().Length;s
tring output = string.Empty;for (int i = 0; i < numzeros; i++){output += "0";}output += t.ToStr
ing();return output;}}}/* Copyright (c) 2018 William van Noordt */using System;using System.Col
lections.Generic;using System.Linq;using System.Text;using System.Threading.Tasks;using _3DSimp
le;namespace m_435_NCAMR{class NCAMRNode{double node_value, x, y;public NCAMRNode(double _x, do
uble _y, double _val){node_value = _val;x = _x;y = _y;}public double Value{get { return node_va
lue; }set { node_value = value; }}public double X{get { return x; }set { x = value; }}public do
uble Y{get { return y; }set { y = value; }}public Point3 toPoint(){return new Point3(x, y, node
_value);}public static NCAMRNode operator +(NCAMRNode root, Vector3 increment){return new NCAMR
Node(root.X + increment.I, root.Y + increment.J, root.Value);}public static NCAMRNode operator
-(NCAMRNode root, Vector3 increment){return new NCAMRNode(root.X - increment.I, root.Y - increm
ent.J, root.Value);}public NCAMRNode Clone(){NCAMRNode b = new NCAMRNode(x, y, node_value);retu
rn b;}public override string ToString(){return "node:" + x.ToString() + ":" + y.ToString() + ":
" + node_value.ToString();}public static NCAMRNode fromstring(string node_string){double _x, _y
, _val;string[] ar = node_string.Split(':');if(!(double.TryParse(ar[1], out _x) && double.TryPa
rse(ar[2], out _y) && double.TryParse(ar[3], out _val))) { throw new ArgumentException("String
improperly formatted."); }else { return new NCAMRNode(_x, _y, _val); }}}}/* Copyright (c) 2018
William van Noordt */using System;using System.Collections.Generic;using System.Linq;using Syst
em.Text;using System.Threading.Tasks;using System.IO;using _3DSimple;namespace m_435_NCAMR{clas
s NCGrid_Distribution{private Random R = new Random();private RBounds2D bounds;private double d
x, dy;private const int additional_file_lines = 2;private int x_node_count, y_node_count;privat
e double minval, maxval;private NCAMRNode[,] nodes;public RBounds2D Bounds{get { return bounds;
 }}public enum DerivativeEstimationMode{WEIGHTFUNCTION,GFDM,TAYLOR_SURPLUS_UPPER_RIGHT,NAIVE}pu
blic enum SurplusNodeAccessingMode{UPPER_RIGHT,RANDOM,MINIMAL_DISTANCE}public double MaxValue{g
et { return maxval; }}public double MinValue{get { return minval; }}public int Xcount{get { ret
urn x_node_count; }}public int Ycount{get { return y_node_count; }}public double Xmin{get { ret
urn bounds.Xmin; }set { bounds.Xmin = value; }}public double Xmax{get { return bounds.Xmax; }se
t { bounds.Xmax = value; }}public double Ymin{get { return bounds.Ymin; }set { bounds.Ymin = va
lue; }}public double Ymax{get { return bounds.Ymax; }set { bounds.Ymax = value; }}public static
 NCGrid_Distribution FirstAvailable(){string[] allnames = Directory.GetFiles(Paths.Distribution
Repo);if (allnames.Length != 0){return from_file(allnames[0], true);}else{throw new Exception("
No files available.");}}public double interpolate_at(double x, double y){if (x < bounds.Xmin ||
 x > bounds.Xmax || y < bounds.Ymin || y > bounds.Ymax) { throw new Exception("Error: Extrapola
tion prevented."); }else{int i = get_x_ll_index(x, y);int j = get_y_ll_index(x, y);if (i >= Xco
unt - 1) { i--; }if (j >= Ycount - 1) { j--; }NCAMRNode n1 = nodes[i, j];NCAMRNode n2 = nodes[i
 + 1, j];NCAMRNode n3 = nodes[i + 1, j + 1];NCAMRNode n4 = nodes[i, j + 1];double[] weights ={M
ath.Exp(-sq(x - n1.X) - sq(y - n1.Y)),Math.Exp(-sq(x - n2.X) - sq(y - n2.Y)),Math.Exp(-sq(x - n
3.X) - sq(y - n3.Y)),Math.Exp(-sq(x - n4.X) - sq(y - n4.Y))};double[] vals ={n1.Value,n2.Value,
n3.Value,n4.Value};double acc_top = 0;double acc_bottom = 0;for (int q = 0; q < vals.Length; q+
+){acc_bottom += weights[q];acc_top += weights[q] * vals[q];}return acc_top / acc_bottom;}}priv
ate int get_x_ll_index(double x, double y){int upper = Xcount;int lower = 0;int med = (upper +
lower) / 2;while (upper - lower > 1){if (in_x_polygon_assume_right(med, x, y)){upper = med;med
= (upper + lower) / 2;}else{lower = med;med = (upper + lower) / 2;}}return lower;}private int g
et_y_ll_index(double x, double y){int upper = Ycount;int lower = 0;int med = (upper + lower) /
2;while (upper - lower > 1){if (in_y_polygon_assume_right(med, x, y)){upper = med;med = (upper
+ lower) / 2;}else{lower = med;med = (upper + lower) / 2;}}return lower;}private double sq(doub
le toSq){return toSq * toSq;}private bool in_x_polygon_assume_right(int lowerbound, double x, d
ouble y){bool odd = false;for (int i = 0; i < Ycount; i++){NCAMRNode tracker = nodes[lowerbound
, i];double track_x = tracker.X;double track_y = tracker.Y;if (((track_y > y) != odd) && track_
x > x){odd = !odd;}}return odd;}private bool in_y_polygon_assume_right(int lowerbound, double x
, double y){bool odd = false;for (int i = 0; i < Xcount; i++){NCAMRNode tracker = nodes[i, lowe
rbound];double track_x = tracker.X;double track_y = tracker.Y;if (((track_x > x) != odd) && tra
```

```
ck_y > y){odd = !odd;}}return odd;}public NCGrid_Distribution(RBounds2D _bounds, int xcount, in
t ycount){bounds = _bounds;x_node_count = xcount;y_node_count = ycount;nodes = new NCAMRNode[x_
node_count, y_node_count];generate_nodes(0);minval = 0;maxval = 0;}public NCGrid_Distribution(R
Bounds2D _bounds, int xcount, int ycount, double presetvalue){bounds = _bounds;x_node_count = x
count;y_node_count = ycount;nodes = new NCAMRNode[x_node_count, y_node_count];generate_nodes(pr
esetvalue);minval = presetvalue;maxval = presetvalue;}public void ApplyMeshMorphGA(double size)
{int surplus_i = 1;int surplus_j = 1;for (int i = 1; i < x_node_count-1; i++){for (int j = 1; j
 < y_node_count-1; j++){NCAMRNode[] stencil ={nodes[i,j],nodes[i+1,j],nodes[i,j+1],nodes[i-1,j]
,nodes[i,j-1],nodes[i+surplus_i,j+surplus_j]};TaylorSystem t_system = new TaylorSystem(stencil)
;Matrix Astar = t_system.TaylorCoefficients;double k = 200;double[] kstuff = { k, k, k * k, k *
 k, k * k };for (int c = 0; c < 5; c++){for (int d = 0; d < 5; d++){Astar[d, c] = Astar[d, c] *
 kstuff[c];}}Matrix K = new Matrix(5);K[0, 0] = k;K[1, 1] = k;K[2, 2] = k * k;K[3, 3] = k * k;K
[4, 4] = k * k;Matrix b =  K * Astar.Inverse;Matrix xi = b * t_system.RHS;double ux = xi[0];dou
ble uxx = xi[2];double uy = xi[1];double uyy = xi[3];double uxy = xi[4];double _dx = ((uxx * ux
) + (uxy * uy));double _dy = ((uyy * uy) + (ux * uxy));//normalize to max of gradient?double st
ep_size = size;Vector3 move = new Vector3(step_size * _dx, step_size * _dy, 0);double xnew = (t
his[i, j] + move).X;double ynew = (this[i, j] + move).Y;while (xnew < bounds.Xmin || xnew > bou
nds.Xmax || ynew > bounds.Ymax || ynew < bounds.Ymin || move.Norm > 0.2*(dx+dy)){move = 0.5 * m
ove;xnew = (this[i, j] + move).X;ynew = (this[i, j] + move).Y;}this[i, j] = this[i, j] + move;}
}}public void ApplyMeshMorphGA(int iterations, double size){for (int i = 0; i < iterations; i++
){ApplyMeshMorphGA(size);}}public NCGrid_Distribution Clone(){NCGrid_Distribution n = new NCGri
d_Distribution(bounds, x_node_count, y_node_count);for (int i = 0; i < x_node_count; i++){for (
int j = 0; j < y_node_count; j++){n.assign_value_at(i, j, this[i, j].Value);n[i, j].X = this[i,
 j].X;n[i, j].Y = this[i, j].Y;}}return n;}private void generate_nodes(double preset_value){dx
= (bounds.Xmax - bounds.Xmin) / (x_node_count - 1);dy = (bounds.Ymax - bounds.Ymin) / (y_node_c
ount - 1);minval = preset_value;maxval = preset_value;for (int i = 0; i < x_node_count; i++){fo
r (int j = 0; j < y_node_count; j++){NCAMRNode newnode = new NCAMRNode(bounds.Xmin + i * dx, bo
unds.Ymin + j * dy, preset_value);nodes[i, j] = newnode;}}}public void WriteToFile(string title
){bool okaytowrite = !File.Exists(Paths.DistributionRepo + "\\" + title + ".dist");if (okaytowr
ite) { WriteToFile(title, false, false, false); }else{int n = 0;okaytowrite = !File.Exists(Path
s.DistributionRepo + "\\" + title + n.ToString() + ".dist");while (!okaytowrite){n++;okaytowrit
e = !File.Exists(Paths.DistributionRepo + "\\" + title + n.ToString() + ".dist");}WriteToFile(t
itle + n.ToString(), false, false, false);}}public void WriteToFile(string title, bool enable_c
onsole_output, bool overwrite, bool using_full_path){string path = Paths.DistributionRepo + "\\
" + title + ".dist";if (using_full_path) { path = title; }if (File.Exists(path) && !overwrite)
{ throw new Exception("File exists, overwriting permission not explicitly granted"); }DateTime
then = DateTime.Now;string[] contents = new string[x_node_count * y_node_count + additional_fil
e_lines];contents[0] = bounds.ToString() + ":max_value:" + maxval.ToString() + ":min_value:" +
minval.ToString();contents[1] = "Nx:" + x_node_count.ToString() + ":Ny:" + y_node_count.ToStrin
g();for (int i = 0; i < x_node_count; i++){for (int j = 0; j < y_node_count; j++){contents[addi
tional_file_lines + i + x_node_count * j] = nodes[i, j].ToString();}}File.WriteAllLines(path, c
ontents);DateTime now = DateTime.Now;if (enable_console_output) { Console.WriteLine("File outpu
t in " + (now - then).Milliseconds.ToString() + " milliseconds."); }}public Vector3 estimate_nu
m_graident(double x, double y, DerivativeEstimationMode mode){//will need to add in edge cases.
 Ignore for now.switch (mode){case DerivativeEstimationMode.GFDM:{return Vector3.Zero;}case Der
ivativeEstimationMode.WEIGHTFUNCTION:{double delta_x = (bounds.Xmax - bounds.Xmin) / (Xcount -
1);double delta_y = (bounds.Ymax - bounds.Ymin) / (Ycount - 1);double dzdx = (this.interpolate_
at(x + delta_x, y) - this.interpolate_at(x - delta_x, y)) / (2 * delta_x);double dzdy = (this.i
nterpolate_at(x, y + delta_y) - this.interpolate_at(x, y - delta_y)) / (2 * delta_x);return new
 Vector3(dzdx, dzdy, 0);}default:{return Vector3.Zero;}}}public Vector3 estimate_num_graident(i
nt i, int j, DerivativeEstimationMode mode){//will need to add in edge cases. Ignore for now.sw
itch (mode){case DerivativeEstimationMode.GFDM:{//lacks implementationreturn Vector3.Zero;}case
 DerivativeEstimationMode.TAYLOR_SURPLUS_UPPER_RIGHT:{//clean this up???NCAMRNode[] stencil ={n
odes[i,j],nodes[i+1,j],nodes[i,j+1],nodes[i-1,j],nodes[i,j-1],nodes[i+1,j+1]};double[] deltas_u
 ={stencil[1].Value-stencil[0].Value,stencil[2].Value-stencil[0].Value,stencil[3].Value-stencil
[0].Value,stencil[4].Value-stencil[0].Value,stencil[5].Value-stencil[0].Value};double[] deltas_
```

```
x ={stencil[1].X-stencil[0].X,stencil[2].X-stencil[0].X,stencil[3].X-stencil[0].X,stencil[4].X-
stencil[0].X,stencil[5].X-stencil[0].X};double[] deltas_y ={stencil[1].Y-stencil[0].Y,stencil[2
].Y-stencil[0].Y,stencil[3].Y-stencil[0].Y,stencil[4].Y-stencil[0].Y,stencil[5].Y-stencil[0].Y}
;double[] matrix_contents ={deltas_x[0], deltas_y[0], 0.5*sq(deltas_x[0]), 0.5*sq(deltas_x[0]),
 deltas_x[0]*deltas_y[0],deltas_x[1], deltas_y[1], 0.5*sq(deltas_x[1]), 0.5*sq(deltas_x[1]), de
ltas_x[1]*deltas_y[1],deltas_x[2], deltas_y[2], 0.5*sq(deltas_x[2]), 0.5*sq(deltas_x[2]), delta
s_x[2]*deltas_y[2],deltas_x[3], deltas_y[3], 0.5*sq(deltas_x[3]), 0.5*sq(deltas_x[3]), deltas_x
[3]*deltas_y[3],deltas_x[4], deltas_y[4], 0.5*sq(deltas_x[4]), 0.5*sq(deltas_x[4]), deltas_x[4]
*deltas_y[4]};Matrix delta = new Matrix(5, 1);return Vector3.Zero;}case DerivativeEstimationMod
e.WEIGHTFUNCTION:{return Vector3.Zero;}case DerivativeEstimationMode.NAIVE:{double delta_x = (b
ounds.Xmax - bounds.Xmin) / (Xcount - 1);double delta_y = (bounds.Ymax - bounds.Ymin) / (Ycount
 - 1);NCAMRNode node = nodes[i, j];double dzdx = (this.interpolate_at(node.X + delta_x, node.Y)
 - this.interpolate_at(node.X - delta_x, node.Y)) / (2 * delta_x);double dzdy = (this.interpola
te_at(node.X, node.Y + delta_y) - this.interpolate_at(node.X, node.Y - delta_y)) / (2 * delta_x
);return new Vector3(dzdx, dzdy, 0);}default:{return Vector3.Zero;}}}public NCAMRNode this[int
i, int j]{get { return nodes[i, j]; }set{nodes[i, j] = value;if (value.Value > maxval){maxval =
 value.Value;}if (value.Value < minval){minval = value.Value;}}}public void QuickSketch(string
title){force_extrema_update();DistributionSketchSettings S = DistributionSketchSettings.Fancy()
;DistributionSketch2D f = new DistributionSketch2D(this, S);f.CreateSketch(false);bool oktowrit
e = !File.Exists(Paths.ImageRepo + "\\" + title + ".bmp");if (oktowrite){f.SaveImage(title, fal
se);}else{int n = 0;oktowrite = !File.Exists(Paths.ImageRepo + "\\" + title + n.ToString() + ".
bmp");while (!oktowrite){n++;oktowrite = !File.Exists(Paths.ImageRepo + "\\" + title + n.ToStri
ng() + ".bmp");}f.SaveImage(title + n.ToString(), false);}}public void QuickSketch(string title
, bool using_full_path){force_extrema_update();DistributionSketchSettings S = DistributionSketc
hSettings.Fancy();DistributionSketch2D f = new DistributionSketch2D(this, S);f.CreateSketch(fal
se);f.SaveImage(title, using_full_path);}public void force_extrema_update(){maxval = double.Neg
ativeInfinity;minval = double.PositiveInfinity;for (int i = 0; i < Xcount; i++){for (int j = 0;
 j < Ycount; j++){if (this[i, j].Value > maxval) { maxval = this[i, j].Value; }if (this[i, j].V
alue < minval) { minval = this[i, j].Value; }}}}public static NCGrid_Distribution MakeMagnitude
(NCGrid_Distribution A, NCGrid_Distribution B){NCGrid_Distribution output = A.Clone();for (int
i = 0; i < A.Xcount; i++){for (int j = 0; j < B.Ycount; j++){double x = A[i, j].Value;double y
= B[i, j].Value;output.assign_value_at(i, j, Math.Sqrt((x * x) + (y * y)));}}output.force_extre
ma_update();return output;}public void MimicMorph(NCGrid_Distribution template){for (int i = 0;
 i < x_node_count; i++){for (int j = 0; j < y_node_count; j++){this[i, j].X = template[i, j].X;
this[i, j].Y = template[i, j].Y;}}}public static NCGrid_Distribution from_file(string title, bo
ol using_full_path){string path = Paths.DistributionRepo + "\\" + title + ".dist";if (using_ful
l_path) { path = title; }string[] contents = File.ReadAllLines(path);int length = contents.Leng
th;RBounds2D new_bounds = RBounds2D.fromstring(contents[0]);string[] numbers = contents[1].Spli
t(':');int nx, ny;if (!(int.TryParse(numbers[1], out nx) && int.TryParse(numbers[3], out ny)))
{ throw new Exception("File format error"); }NCGrid_Distribution created = new NCGrid_Distribut
ion(new_bounds, nx, ny);for (int q = additional_file_lines; q < contents.Length; q++){int qprim
e = q - additional_file_lines;int i = qprime % nx;int j = (qprime - i) / nx;created[i, j] = NCA
MRNode.fromstring(contents[q]);}return created;}public void SetConstant(double constant){maxval
 = constant;minval = constant;for (int i = 0; i < x_node_count; i++){for (int j = 0; j < y_node
_count; j++){this[i, j].Value = constant;}}}public Matrix GetTaylorSystemCoeffs(int i, int j, i
nt surplus_i, int surplus_j){NCAMRNode[] stencil ={nodes[i,j],nodes[i+1,j],nodes[i,j+1],nodes[i
-1,j],nodes[i,j-1],nodes[i+surplus_i,j+surplus_j]};TaylorSystem t_system = new TaylorSystem(ste
ncil);Matrix Astar = t_system.TaylorCoefficients;double k = 20;double[] kstuff = { k, k, k * k,
 k * k, k * k };for (int c = 0; c < 5; c++){for (int d = 0; d < 5; d++){Astar[d, c] = Astar[d,
c] * kstuff[c];}}Matrix K = new Matrix(5);K[0, 0] = k;K[1, 1] = k;K[2, 2] = k * k;K[3, 3] = k *
 k;K[4, 4] = k * k;return K * Astar.Inverse;}public Matrix EstimateSecondDerivs(int i, int j, S
urplusNodeAccessingMode mode){int surplus_i = 1;int surplus_j = 1;switch (mode){case SurplusNod
eAccessingMode.RANDOM:{surplus_i = 1 - 2 * R.Next(0, 1);surplus_j = 1 - 2 * R.Next(0, 1);break;
}}NCAMRNode[] stencil ={nodes[i,j],nodes[i+1,j],nodes[i,j+1],nodes[i-1,j],nodes[i,j-1],nodes[i+
surplus_i,j+surplus_j]};TaylorSystem t_system = new TaylorSystem(stencil);Matrix Astar = t_syst
em.TaylorCoefficients;double k = 9999;double[] kstuff = { k, k, k * k, k * k, k * k };for (int
```

```
c = 0; c < 5; c++){for (int d = 0; d < 5; d++){Astar[d, c] = Astar[d, c] * kstuff[c];}}Matrix K
 = new Matrix(5);K[0, 0] = k;K[1, 1] = k;K[2, 2] = k * k;K[3, 3] = k * k;K[4, 4] = k * k;return
 K * Astar.Inverse * t_system.RHS;}public void WriteAllDerivs(string title, bool enable_console
_output, bool using_full_path, bool allow_overwrite){DateTime Then = DateTime.Now;NCGrid_Distri
bution this_x = new NCGrid_Distribution(bounds, this.Xcount, this.Ycount);NCGrid_Distribution t
his_y = new NCGrid_Distribution(bounds, this.Xcount, this.Ycount);NCGrid_Distribution this_xx =
 new NCGrid_Distribution(bounds, this.Xcount, this.Ycount);NCGrid_Distribution this_yy = new NC
Grid_Distribution(bounds, this.Xcount, this.Ycount);NCGrid_Distribution this_xy = new NCGrid_Di
stribution(bounds, this.Xcount, this.Ycount);for (int i = 1; i < this.Xcount - 1; i++){for (int
 j = 1; j < this.Ycount - 1; j++){Matrix xi = EstimateSecondDerivs(i, j, SurplusNodeAccessingMo
de.UPPER_RIGHT);this_x[i, j].Value = xi[0];this_y[i, j].Value = xi[1];this_xx[i, j].Value = xi[
2];this_yy[i, j].Value = xi[3];this_xy[i, j].Value = xi[4];}}DateTime Now = DateTime.Now;string
 title_x = title + "_x";string title_y = title + "_y";string title_xx = title + "_xx";string ti
tle_yy = title + "_yy";string title_xy = title + "_xy";if (using_full_path){string basestring =
 title.Split('.')[0];title_x = basestring + "_x.dist";title_y = basestring + "_y.dist";title_xx
 = basestring + "_xx.dist";title_yy = basestring + "_yy.dist";title_xy = basestring + "_xy.dist
";}this_x.WriteToFile(title_x, enable_console_output, allow_overwrite, using_full_path);this_y.
WriteToFile(title_y, enable_console_output, allow_overwrite, using_full_path);this_xx.WriteToFi
le(title_xx, enable_console_output, allow_overwrite, using_full_path);this_yy.WriteToFile(title
_yy, enable_console_output, allow_overwrite, using_full_path);this_xy.WriteToFile(title_xy, ena
ble_console_output, allow_overwrite, using_full_path);if (enable_console_output) { Console.Writ
eLine((Now - Then).TotalMilliseconds); }}public static NCGrid_Distribution operator +(NCGrid_Di
stribution A, NCGrid_Distribution B){if (A.Xcount == B.Xcount && A.Ycount == B.Ycount){NCGrid_D
istribution C = new NCGrid_Distribution(A.Bounds, A.Xcount, B.Ycount);for (int i = 0; i < A.Xco
unt; i++){for (int j = 0; j < A.Ycount; j++){C[i, j] = A[i, j].Clone();C[i, j].Value = A[i, j].
Value + B[i, j].Value;}}return C;}else{throw new Exception("Dimensions do not agree.");}}public
 static NCGrid_Distribution operator -(NCGrid_Distribution A, NCGrid_Distribution B){if (A.Xcou
nt == B.Xcount && A.Ycount == B.Ycount){NCGrid_Distribution C = new NCGrid_Distribution(A.Bound
s, A.Xcount, B.Ycount);for (int i = 0; i < A.Xcount; i++){for (int j = 0; j < A.Ycount; j++){C[
i, j] = A[i, j].Clone();C[i, j].Value = A[i, j].Value - B[i, j].Value;}}return C;}else{throw ne
w Exception("Dimensions do not agree.");}}public static NCGrid_Distribution operator *(NCGrid_D
istribution A, NCGrid_Distribution B){if (A.Xcount == B.Xcount && A.Ycount == B.Ycount){NCGrid_
Distribution C = new NCGrid_Distribution(A.Bounds, A.Xcount, B.Ycount);for (int i = 0; i < A.Xc
ount; i++){for (int j = 0; j < A.Ycount; j++){C[i, j] = A[i, j].Clone();C[i, j].Value = A[i, j]
.Value * B[i, j].Value;}}return C;}else{throw new Exception("Dimensions do not agree.");}}publi
c static NCGrid_Distribution operator /(NCGrid_Distribution A, NCGrid_Distribution B){if (A.Xco
unt == B.Xcount && A.Ycount == B.Ycount){NCGrid_Distribution C = new NCGrid_Distribution(A.Boun
ds, A.Xcount, B.Ycount);for (int i = 0; i < A.Xcount; i++){for (int j = 0; j < A.Ycount; j++){C
[i, j] = A[i, j].Clone();C[i, j].Value = A[i, j].Value / B[i, j].Value;}}return C;}else{throw n
ew Exception("Dimensions do not agree.");}}public static NCGrid_Distribution operator *(double
 scale, NCGrid_Distribution A){NCGrid_Distribution output = A.Clone();for (int i = 0; i < A.Xcou
nt; i++){for (int j = 0; j < A.Ycount; j++){output.assign_value_at(i, j, scale * A[i, j].Value)
;}}return output;}public static NCGrid_Distribution NiceFunction(string vb_syntax_eval, RBounds
2D bounds, int xcount, int ycount, bool keepfile, string title){ExactFunctionGeneratorVB2D.Gene
rateFunction(vb_syntax_eval, bounds, xcount, ycount, title);NCGrid_Distribution n = NCGrid_Dist
ribution.from_file(title, false);return n;}public static NCGrid_Distribution NiceFunction(strin
g vb_syntax_eval, RBounds2D bounds, int xcount, int ycount){ExactFunctionGeneratorVB2D.Generate
Function(vb_syntax_eval, bounds, xcount, ycount, "temp");NCGrid_Distribution n = NCGrid_Distrib
ution.from_file("temp", false);string to_delete = Paths.DistributionRepo + "\\temp.dist";File.D
elete(to_delete);return n;}public void ApplyBoundary(BoundaryConditions B){if (!check_compatibi
lity(B, this)) { throw new Exception("Incompatible conditions."); }for (int i = 0; i < B.Xcount
; i++){double byn = B[i, BoundaryConditions.Direction.Negative_Y];double byp = B[i, BoundaryCon
ditions.Direction.Positive_Y];this[i, 0].Value = byn;this[i, B.Xcount - 1].Value = byp;if (byp
> maxval) { maxval = byp; }if (byn > maxval) { maxval = byn; }if (byp < minval) { minval = byp;
 }if (byn < minval) { minval = byn; }}for (int j = 0; j < B.Ycount; j++){double bxp = B[j, Boun
daryConditions.Direction.Positive_X];double bxn = B[j, BoundaryConditions.Direction.Negative_X]
```

```
;this[0, j].Value = bxn;this[B.Ycount - 1, j].Value = bxp;if (bxp > maxval) { maxval = bxp; }if
 (bxn > maxval) { maxval = bxn; }if (bxp < minval) { minval = bxp; }if (bxn < minval) { minval
= bxn; }}}public void assign_value_at(int i, int j, double valuein){this[i, j].Value = valuein;
if (valuein > maxval) { maxval = valuein; }if (valuein < minval) { minval = valuein; }}public s
tatic bool check_compatibility(BoundaryConditions B, NCGrid_Distribution C){bool[] stuff ={B.Bo
unds.Xmax == C.Xmax,B.Bounds.Xmin == C.Xmin,B.Bounds.Ymax == C.Ymax,B.Bounds.Ymin == C.Ymin,B.Y
count == C.Ycount,B.Xcount == C.Xcount};bool valid = true;foreach (bool i in stuff){valid = val
id && i;}return valid;}public NCAMRNode GetFake(int i, int j){if (i >= 0 && i <= Xcount-1 && j
>= 0 && j <= Ycount-1){return this[i, j];}else{int imax = Xcount - 1;int jmax = Ycount - 1;doub
le x = double.NegativeInfinity;double y = double.NegativeInfinity;if (i < 0 && j < 0) //ll{x =
Xmin;y = Ymin;}if (i > imax && j < 0) //lr{x = Xmax;y = Ymin;}if (i > imax && j > jmax) //ur{x
= Xmax;y = Ymax;}if (i < 0 && j > jmax) //ul{y = Ymax;x = Xmin;}if (i > imax && j <= jmax && j
>= 0) //px{x = Xmax;y = this[imax, j].Y;}if (i < 0 && j <= jmax && j >= 0){x = Xmin;y = this[0,
 j].Y;}if (j < 0 && i <= imax && i >= 0){x = this[i, 0].X;y = Ymax;}if (j > jmax && i >= 0 && i
 <= imax){x = this[i, jmax].X;y = Ymax;}return new NCAMRNode(x, y, 0);}}}}/* Copyright (c) 2018
 William van Noordt */using System;using System.Collections.Generic;using System.Linq;using Sys
tem.Text;using System.Threading.Tasks;using System.Diagnostics;namespace m_435_NCAMR{static cla
ss OpenImage{//needs workstatic string prg = @"C:\Program Files (x86)\Windows Live\Photo Galler
y\WLXPhotoGallery";static string img_repo = @"C:\Users\Will\Desktop\Folders\MATH435\repo\images
";public static void Show(string title){string fullpath = img_repo + "\\" + title + ".bmp";Proc
ess.Start(prg, fullpath);}public static void Show(string path, bool using_full){Process.Start(p
rg, path);}}}/* Copyright (c) 2018 William van Noordt */using System;using System.Collections.G
eneric;using System.Linq;using System.Text;using System.Threading.Tasks;namespace m_435_NCAMR{s
tatic class Paths{private static string dist_repo = @"C:\Users\Will\Desktop\Folders\MATH435\rep
o\distributions";private static string img_repo = @"C:\Users\Will\Desktop\Folders\MATH435\repo\
images";private static string vb_repo = @"C:\Users\Will\Desktop\Folders\MATH435\repo\visual-bas
ic-templates";private static string mat_repo = @"C:\Users\Will\Desktop\Folders\MATH435\repo\mat
rices";private static string nav_stokes_solutions = @"C:\Users\Will\Desktop\Folders\MATH435\rep
o\n-s-solutions";public static string NavSolutionsRepo{get { return nav_stokes_solutions; }}pub
lic static string DistributionRepo{get { return dist_repo; }}public static string ImageRepo{get
 { return img_repo; }}public static string MatrixRepo{get { return mat_repo; }}}}/* Copyright (
c) 2018 William van Noordt */using System;using System.Collections.Generic;using System.Linq;us
ing System.Text;using System.Threading.Tasks;using System.Drawing;using ColorMapping;using Syst
em.IO;namespace m_435_NCAMR{class Plot2D{protected const string default_repo = @"C:\Users\Will\
Desktop\Folders\MATH435\repo\images";protected Plot2DSettings settings;protected float override
_max;protected float override_min;protected bool use_overriden_limits = false;protected List<Co
lor> HReferenceColors = new List<Color>();protected List<double> HReferenceValues = new List<do
uble>();protected List<Color> VReferenceColors = new List<Color>();protected List<double> VRefe
renceValues = new List<double>();protected List<double[]> xsurplus = new List<double[]>();prote
cted List<double[]> ysurplus = new List<double[]>();protected List<Color> colorsurplus = new Li
st<Color>();//parametersprotected float MAIN_SUBJECT_LOWER_LEFT_X_2D;protected float MAIN_SUBJE
CT_LOWER_LEFT_Y_2D;protected float MAIN_SUBJECT_UPPER_RIGHT_X_2D;protected float MAIN_SUBJECT_U
PPER_RIGHT_Y_2D;//constantsprotected const float FIGURETITLE_Y = 0.04f;protected const float FI
GURETITLE_X = 0.1f;protected const float SQUARE_BORDER_THIN = 0.07f;protected const float SQUAR
E_BORDER_THICK = 0.130f;protected const float GRIDLINE_THICKNESS = 3.3f;protected const float X
_AXIS_TITLE_X = 0.5f;protected const float Y_AXIS_TITLE_Y = 0.5f;protected const float Y_AXIS_T
ITLE_X = 0.1f * SQUARE_BORDER_THICK;protected const float X_AXIS_TITLE_Y = 0.5f * SQUARE_BORDER
_THICK;protected const float LABEL_OFFSET_Y = -0.056f;protected const float HEATBOX_BORDER_LEFT
 = 0.84f;protected const float HEATBOX_BORDER_RIGHT = 0.97f;protected static float HEATBOX_BORD
ER_BOTTOM = 0.35f;protected float HEATBOX_BORDER_TOP = 1 - HEATBOX_BORDER_BOTTOM;protected cons
t float HEATBOX_LL_X = 0.85f;protected float HEATBOX_UR_X = 0.96f;protected static float HEATBO
X_LL_Y = 0.36f;protected float HEATBOX_UR_Y = 1 - HEATBOX_LL_Y;protected float HEATMAP_LABELS_X
 = 0.91f;protected static float HEATMAP_LABELS_YU = 0.33f;protected static float HEATMAP_LABELS
_YL = 0.64f;protected float X_LABEL_BASE_Y = 0.94f * SQUARE_BORDER_THICK;protected float X_LABE
L_BASE_X = 1.0f * SQUARE_BORDER_THICK;protected float Y_LABEL_BASE_Y = 1.0f * SQUARE_BORDER_THI
CK;protected float Y_LABEL_BASE_X = 0.44f * SQUARE_BORDER_THICK;protected float VALUE_TEXT_PROP
```

```
ORTION = 40f / 2000f;protected float TITLE_TEXT_PROPORTION = 65f / 2000f;protected const float
THIN_PEN = 0.8f;protected const float THICK_PEN = 2.4f;protected Pen THICK_GRID_PEN = new Pen(C
olor.Black, THICK_PEN);protected Pen THIN_GRID_PEN = new Pen(Color.Black, THIN_PEN);protected C
olor BACKGROUND_COLOR = Color.White;protected Color TEXTCOLOR = Color.Black;protected Color DEF
AULTCURVECOLOR = Color.Black;RBounds2D sketch_bounds;double[] xvals;double[] yvals;//calculated
 relativesprotected float grid_xmin;protected float grid_xmax;protected float grid_ymin;protect
ed float grid_ymax;protected float pix_per_unit_x;protected float pix_per_unit_y;protected stat
ic Color[] default_colors = {Color.Blue, Color.Red, Color.Orange, Color.Green, Color.Green, Col
or.Yellow, Color.Gray, Color.DarkGoldenrod, Color.DarkGreen};protected Bitmap canvas;protected
int x_pixel_count, y_pixel_count;public Bitmap Canvas{get { return canvas; }}public Plot2D(stri
ng function_VBSYNTAX, Plot2DSettings S){settings = S;throw new Exception("Error: Function not y
et implemented");canvas = new Bitmap(S.ImageWidth, S.ImageHeight);x_pixel_count = canvas.Width;
y_pixel_count = canvas.Height;get_positions();}public Plot2D(double[] x, double[] y, Plot2DSett
ings S){settings = S;if (x.Length != y.Length) { throw new Exception("Error: Array dimensions a
re inconsistent."); }xvals = x;yvals = y;canvas = new Bitmap(S.ImageWidth, S.ImageHeight);x_pix
el_count = canvas.Width;y_pixel_count = canvas.Height;double ymax = double.NegativeInfinity;dou
ble ymin = double.PositiveInfinity;double xmax = double.NegativeInfinity;double xmin = double.P
ositiveInfinity;for (int i = 0; i < x.Length; i++){if (x[i] > xmax) { xmax = x[i]; }if (x[i] <
xmin) { xmin = x[i]; }if (y[i] > ymax) { ymax = y[i]; }if (y[i] < ymin) { ymin = y[i]; }}double
 deltax = xmax - xmin;double deltay = ymax - ymin;double xbar = (xmax + xmin) * 0.5;double ybar
 = (ymax + ymin) * 0.5;double xscl = 1.0;double yscl = 1.0;double newdeltax = xscl * deltax;dou
ble newdeltay = yscl * deltay;xmax = xbar + 0.5 * newdeltax;xmin = xbar - 0.5 * newdeltax;ymax
= ybar + 0.5 * newdeltay;ymin = ybar - 0.5 * newdeltay;sketch_bounds = new RBounds2D(xmin, xmax
, ymin, ymax);get_positions();}public void ScaleCurveY(double scale_factor){for (int i = 0; i <
 yvals.Length; i++){yvals[i] = yvals[i] * scale_factor;sketch_bounds.Ymax *= scale_factor;sketc
h_bounds.Ymin *= scale_factor;}}public virtual void CreateSketch(bool allow_console_output){usi
ng (Graphics drawer = Graphics.FromImage(canvas)){drawer.Clear(BACKGROUND_COLOR);draw_reference
s(drawer);draw_curve(drawer);draw_fig_title(drawer);if (settings.HasAxisTitles) { draw_axis_tit
les(drawer); }if (settings.HasAxisValues) { draw_axis_values(drawer); }if (settings.HasGridline
s) { draw_gridlines(drawer); }}}public void AddHorizontalReference(double val, Color C){HRefere
nceValues.Add(val);HReferenceColors.Add(C);}public void AddVerticalReference(double val, Color
C){VReferenceValues.Add(val);VReferenceColors.Add(C);}protected void draw_references(Graphics d
rawer){for (int i = 0; i < HReferenceValues.Count; i++){if (is_between(HReferenceValues[i], ske
tch_bounds.Ymin, sketch_bounds.Ymax)){Pen currentpen = new Pen(HReferenceColors[i], 6.0f);drawe
r.DrawLine(currentpen, mapXY(sketch_bounds.Xmin, HReferenceValues[i]), mapXY(sketch_bounds.Xmax
, HReferenceValues[i]));}}for (int i = 0; i < VReferenceValues.Count; i++){if (is_between(VRefe
renceValues[i], sketch_bounds.Xmin, sketch_bounds.Xmax)){Pen currentpen = new Pen(VReferenceCol
ors[i], 6.0f);drawer.DrawLine(currentpen, mapXY(VReferenceValues[i], sketch_bounds.Ymin), mapXY
(VReferenceValues[i], sketch_bounds.Ymax));}}}protected bool is_between(double val, double lowe
r, double upper){return (val < upper) && (val > lower);}protected void draw_curve(Graphics draw
er){Pen curvepen = new Pen(Color.Black, 4.5f);List<PointF> curve = new List<PointF>();for (int
i = 0; i < xvals.Length; i++){curve.Add(mapXY(xvals[i], yvals[i]));}for (int i = 0; i < xsurplu
s.Count; i++){List<PointF> curcurve = new List<PointF>();Pen curcurvepen = new Pen(colorsurplus
[i], 4.5f);for (int j = 0; j < xsurplus[i].Length; j++){curcurve.Add(mapXY(xsurplus[i][j], ysur
plus[i][j]));}drawer.DrawLines(curcurvepen, curcurve.ToArray());}drawer.DrawLines(curvepen, cur
ve.ToArray());}public void AddCurve(double[] xs, double[] ys, Color C){xsurplus.Add(xs);ysurplu
s.Add(ys);colorsurplus.Add(C);}protected void get_positions(){if (settings.HasAxisValues || set
tings.HasAxisTitles){MAIN_SUBJECT_LOWER_LEFT_X_2D = SQUARE_BORDER_THICK + settings.WidthOffset;
MAIN_SUBJECT_LOWER_LEFT_Y_2D = SQUARE_BORDER_THICK;MAIN_SUBJECT_UPPER_RIGHT_X_2D = 1 - SQUARE_B
ORDER_THICK + settings.WidthOffset;MAIN_SUBJECT_UPPER_RIGHT_Y_2D = 1 - SQUARE_BORDER_THICK;}els
e{MAIN_SUBJECT_LOWER_LEFT_X_2D = SQUARE_BORDER_THIN;MAIN_SUBJECT_LOWER_LEFT_Y_2D = SQUARE_BORDE
R_THIN;MAIN_SUBJECT_UPPER_RIGHT_X_2D = 1 - SQUARE_BORDER_THIN;MAIN_SUBJECT_UPPER_RIGHT_Y_2D = 1
 - SQUARE_BORDER_THIN;}grid_xmin = MAIN_SUBJECT_LOWER_LEFT_X_2D * (float)x_pixel_count;grid_xma
x = MAIN_SUBJECT_UPPER_RIGHT_X_2D * (float)x_pixel_count;grid_ymin = MAIN_SUBJECT_LOWER_LEFT_Y_
2D * (float)y_pixel_count;grid_ymax = MAIN_SUBJECT_UPPER_RIGHT_Y_2D * (float)y_pixel_count;pix_
per_unit_x = (grid_xmax - grid_xmin) / (float)(sketch_bounds.Xmax - sketch_bounds.Xmin);pix_per
```

```
_unit_y = (grid_ymax - grid_ymin) / (float)(sketch_bounds.Ymax - sketch_bounds.Ymin);}protected
 void draw_axis_values(Graphics drawer){Brush textbrush = new SolidBrush(TEXTCOLOR);float xlabe
l_abs_pos_x = ((float)x_pixel_count * (X_LABEL_BASE_X+settings.WidthOffset));float xlabel_abs_i
ncrement = (((1 - 2 * (Y_LABEL_BASE_Y)) * (float)x_pixel_count) / (float)(settings.XLabelCount
 - 1));float ylabel_abs_pos_y = ((float)y_pixel_count * (Y_LABEL_BASE_X));float ylabel_abs_incre
ment = (((1 - 2 * Y_LABEL_BASE_Y) * (float)y_pixel_count) / (float)(settings.YLabelCount - 1));
float xlabel_abs_pos_y = (X_LABEL_BASE_Y * (float)x_pixel_count);float ylabel_abs_pos_x = ((Y_L
ABEL_BASE_X - LABEL_OFFSET_Y) * (float)y_pixel_count);double delta_x = (sketch_bounds.Xmax - sk
etch_bounds.Xmin) / (settings.XLabelCount - 1);double delta_y = (sketch_bounds.Ymax - sketch_bo
unds.Ymin) / (settings.YLabelCount - 1);for (int i = 0; i < settings.XLabelCount; i++){drawer.D
rawString(truncate_string((sketch_bounds.Xmin + i * delta_x).ToString(), 4), new Font("arial",
settings.FontSize), textbrush, new PointF(xlabel_abs_pos_x + i * xlabel_abs_increment, settings
.ImageHeight - xlabel_abs_pos_y));}for (int j = 0; j < settings.YLabelCount; j++){drawer.DrawSt
ring(truncate_string((sketch_bounds.Ymax - j * delta_y).ToString(), 4), new Font("arial", setti
ngs.FontSize), textbrush, new PointF(ylabel_abs_pos_y, ylabel_abs_pos_x + j * ylabel_abs_increm
ent));}}protected void draw_gridlines(Graphics drawer){Pen gridlinepen = new Pen(Color.FromArgb
(130, 0, 0, 0), GRIDLINE_THICKNESS);NCAMRNode[,] gridpoints = new NCAMRNode[settings.XGridlineC
ount, settings.YGridlineCount];double coordinate_dx = (sketch_bounds.Xmax - sketch_bounds.Xmin)
 / (settings.XGridlineCount - 1);double coordinate_dy = (sketch_bounds.Ymax - sketch_bounds.Ymi
n) / (settings.YGridlineCount - 1);for (int i = 0; i < settings.XGridlineCount; i++){for (int j
 = 0; j < settings.YGridlineCount; j++){gridpoints[i, j] = new NCAMRNode(sketch_bounds.Xmin + i
 * coordinate_dx, sketch_bounds.Ymin + j * coordinate_dy, 0);}}for (int i = 0; i < settings.XGr
idlineCount; i++){for (int j = 0; j < settings.YGridlineCount - 1; j++){//thick grid pen tempor
arydrawer.DrawLine(gridlinepen, mapGridNode(gridpoints[i, j]), mapGridNode(gridpoints[i, j + 1]
));}}for (int i = 0; i < settings.YGridlineCount; i++){for (int j = 0; j < settings.XGridlineCo
unt - 1; j++){//thick grid pen temporarydrawer.DrawLine(gridlinepen, mapGridNode(gridpoints[j +
 1, i]), mapGridNode(gridpoints[j, i]));}}}protected void draw_fig_title(Graphics drawer){float
 absx = settings.ImageWidth * FIGURETITLE_X;float absy = settings.ImageHeight * FIGURETITLE_Y;B
rush textbrush = new SolidBrush(TEXTCOLOR);drawer.DrawString(settings.FigureTitle, new Font("ar
ial", 1.9f * VALUE_TEXT_PROPORTION * settings.ImageHeight), textbrush, new PointF(absx, absy));
}protected void draw_axis_titles(Graphics drawer){float x_title_abs_pos_x = X_AXIS_TITLE_X * se
ttings.ImageWidth;float x_title_abs_pos_y = X_AXIS_TITLE_Y * settings.ImageWidth;float y_title_
abs_pos_x = Y_AXIS_TITLE_X * settings.ImageHeight;float y_title_abs_pos_y = Y_AXIS_TITLE_Y * se
ttings.ImageHeight;Brush textbrush = new SolidBrush(TEXTCOLOR);drawer.DrawString(settings.Horiz
ontalTitle, new Font("arial", TITLE_TEXT_PROPORTION * settings.ImageHeight), textbrush, new Poi
ntF(x_title_abs_pos_x, settings.ImageHeight - x_title_abs_pos_y));drawer.DrawString(settings.Ve
rticalTitle, new Font("arial", TITLE_TEXT_PROPORTION * settings.ImageHeight), textbrush, new Po
intF(y_title_abs_pos_x, y_title_abs_pos_y));}protected PointF mapXY(double x, double y){return
 new PointF(grid_xmin + pix_per_unit_x * (float)(x - sketch_bounds.Xmin), grid_ymin + pix_per_un
it_y * (float)((sketch_bounds.Ymax - y)));}protected PointF mapGridNode(NCAMRNode toMap){return
 new PointF(grid_xmin + pix_per_unit_x * (float)(toMap.X - sketch_bounds.Xmin), grid_ymin + pix
_per_unit_y * (float)((sketch_bounds.Ymax - toMap.Y)));}public static Plot2D ReadCSV(string pat
h, Plot2DSettings S, params int[] yindeces){string[] filestuff = File.ReadAllLines(path);int st
uffcount = filestuff[0].Split(',').Length-1;int[] index = yindeces;if (yindeces.Length == 0){in
dex = new int[stuffcount];for (int i = 0; i < stuffcount; i++){index[i] = i + 1;}}List<double>[
] arrs = new List<double>[index.Length + 1];for (int i = 0; i < arrs.Length; i++){arrs[i] = new
 List<double>();}for (int i = 0; i < filestuff.Length; i++){string[] spt = filestuff[i].Split('
,');double x = Convert.ToDouble(spt[0]);List<double> curdouble = new List<double>();foreach (in
t f in index){curdouble.Add(Convert.ToDouble(spt[f]));}arrs[0].Add(x);for (int q = 1; q <= curd
ouble.Count; q++){arrs[q].Add(curdouble[q - 1]);}}double[] basex = arrs[0].ToArray();double[] b
asey = arrs[1].ToArray();Plot2D plot = new Plot2D(basex, basey, S);for (int i = 2; i < arrs.Len
gth; i++){plot.AddCurve(basex, arrs[i].ToArray(), default_colors[i - 2]);}return plot;}public v
oid SaveImage(string title, bool using_full_path){string path = default_repo + "\\" + title + "
.png";if (using_full_path) { path = title; }Image imgout = (Image)canvas;imgout.Save(path);}pub
lic void override_colormap_limits(float new_min, float new_max){use_overriden_limits = true;ove
rride_max = new_max;override_min = new_min;}public void override_colormap_limits(DataSetInfo D)
```

```
{use_overriden_limits = true;override_max = (float)D.UniversalMax;override_min = (float)D.Unive
rsalMin;}public void reset_colormap_limits(){use_overriden_limits = false;}public void SetSketc
hSettings(Plot2DSettings S){settings = S;get_positions();}protected string truncate_string(stri
ng input, int number){if (sketch_bounds.Xmax-sketch_bounds.Xmin > 0.1 && input.Contains("E-"))
{ return "0"; }if (input.Length <= number) return input;else{string output = string.Empty;for (
int i = 0; i < number; i++){if (!(i == number - 1 && input[i] == '.')){output += input[i];}}ret
urn output;}}public void OverrideBounds(RBounds2D newbounds){sketch_bounds = newbounds;get_posi
tions();}}}/* Copyright (c) 2018 William van Noordt */using System;using System.Collections.Gen
eric;using System.Linq;using System.Text;using System.Threading.Tasks;namespace m_435_NCAMR{cla
ss Plot2DSettings{private const int DEFAULTIMAGESIZE = 2000;private float width_offset = 0.08f;
private string fig_title;private float fontsize = 40f;private const int DEFAULTLABELCOUNT = 5;p
rivate bool has_axis_values, has_axis_titles, has_cartesian_gridlines;private int x_cart_gridli
nes, y_cart_gridlines;private int x_label_count, y_label_count, image_width, image_height;priva
te string horizontal_title, vertical_title;public float WidthOffset{get { return width_offset;
}}public void setWidthOffset(float to_set){width_offset = to_set;}public float FontSize{get { r
eturn fontsize; }}public void setFontSize(float size){fontsize = size;}public string FigureTitl
e{get { return fig_title; }}public void SetFigureTitle(string title){fig_title = title;}public
bool HasGridlines{get { return has_cartesian_gridlines; }set { has_cartesian_gridlines = value;
 }}public int XGridlineCount{get { return x_cart_gridlines; }set { x_cart_gridlines = value; }}
public int YGridlineCount{get { return y_cart_gridlines; }set { y_cart_gridlines = value; }}pub
lic string HorizontalTitle{get { return horizontal_title; }set { horizontal_title = value; }}pu
blic string VerticalTitle{get { return vertical_title; }set { vertical_title = value; }}public
int ImageWidth{get { return image_width; }set { image_width = value; }}public int ImageHeight{g
et { return image_height; }set { image_height = value; }}public bool HasAxisValues{get { return
 has_axis_values; }set { has_axis_values = value; }}public bool HasAxisTitles{get { return has_
axis_titles; }set { has_axis_titles = value; }}public int XLabelCount{get { return x_label_coun
t; }set { x_label_count = value; }}public int YLabelCount{get { return y_label_count; }set { y_
label_count = value; }}public Plot2DSettings(int imagesize){has_axis_values = false;has_axis_ti
tles = false;has_cartesian_gridlines = false;x_cart_gridlines = 10;y_cart_gridlines = 10;x_labe
l_count = 5;y_label_count = 5;image_height = imagesize;image_width = imagesize;horizontal_title
 = "x";vertical_title = "y";}public Plot2DSettings(bool hasaxisvalues, bool hasaxistitles, bool
 hascartesians, int xgridcount, int ygridcount, int xlabelcount, int ylabelcount, int imagesize
){has_axis_values = hasaxisvalues;has_axis_titles = hasaxistitles;has_cartesian_gridlines = has
cartesians;x_cart_gridlines = xgridcount;y_cart_gridlines = ygridcount;x_label_count = xlabelco
unt;y_label_count = ylabelcount;image_height = imagesize;image_width = imagesize;horizontal_tit
le = "x";vertical_title = "y";}public void revert_simple(){has_axis_titles = false;has_axis_val
ues = false;has_cartesian_gridlines = false;}public static Plot2DSettings Fancy(){return new Pl
ot2DSettings(true, true, true, 10, 10, DEFAULTLABELCOUNT, DEFAULTLABELCOUNT, DEFAULTIMAGESIZE);
}}}/* Copyright (c) 2018 William van Noordt */using System;using System.Collections.Generic;usi
ng System.Linq;using System.Text;using System.Threading.Tasks;using System.Drawing;using System
.IO;using System.IO.Compression;using _3DSimple;namespace m_435_NCAMR{class Program{static void
 Main(string[] args){if (args.Length != 0){if (args[0].EndsWith(".dist")){Console.WriteLine("Lo
ading " + args[0]);Console.Write("Enter sketch title: ");string name = Console.ReadLine();NCGri
d_Distribution p = NCGrid_Distribution.from_file(args[0], true);p.QuickSketch(name);}}else{//te
stfunction3();Console.WriteLine("Press enter to begin.");Console.ReadLine();for (int i = 0; i <
 50; i++){Console.WriteLine(i);}Console.WriteLine("Done");Console.ReadLine();}}static void run_
presentation_transient(int numframes){double t = 0;double dt = 0.1;int n = 25;double L = 10;dou
ble H = 10;RBounds2D domain = new RBounds2D(0, L, 0, H);NCGrid_Distribution soln = new NCGrid_D
istribution(domain, n, n);double max1 = 4;double max2 = 3;double max3 = 5;double max4 = 2;doubl
e omega1 = 1;double omega2 = 0.6;double omega3 = 0.2;double omega4 = 1.2;double phi1 = -0.4;dou
ble phi2 = -0.9;double phi3 = 1.3;double phi4 = 0.2;double dx = L / (n - 1);double dy = H / (n
- 1);double refinestep = 0.1;int refinecount = 3;for (int i = 0; i < numframes; i++){BoundaryCo
nditions cond = new BoundaryConditions(soln, BoundaryConditions.BoundaryConditionType.Dirichlet
);t += dt;for (int q = 0; q < n; q++){double x = q * dx;double y = q * dy;cond[q, BoundaryCondi
tions.Direction.Negative_X] = max1 * Math.Sin(4*Math.PI * x / L) * Math.Sin(omega1 * t - phi1);
cond[q, BoundaryConditions.Direction.Negative_Y] = max2 * Math.Sin(2*Math.PI * y / L) * Math.Si
```

```
n(omega2 * t - phi2);cond[q, BoundaryConditions.Direction.Positive_X] = max3 * Math.Sin(3*Math.
PI * x / L) * Math.Sin(omega3 * t - phi3);cond[q, BoundaryConditions.Direction.Positive_Y] = ma
x4 * Math.Sin(5*Math.PI * y / L) * Math.Sin(omega4 * t - phi4);}BVPLinear2D problem = new BVPLi
near2D(cond, LinearOperatorOrder2.Laplace, soln);NCGrid_Distribution sol = problem.SolveKaczMar
zExt(100, 10, 20);sol.ApplyMeshMorphGA(refinestep);for (int z = 0; z < refinecount-1; z++){prob
lem = new BVPLinear2D(cond, LinearOperatorOrder2.Laplace, soln);sol = problem.Solve(Matrix.Syst
emSolvingScheme.Kaczmarz);sol.ApplyMeshMorphGA(refinestep);}sol.WriteToFile("longtransient_" +
i.ToString());sol.QuickSketch("sol_" + bufferint(i, 4));}}static void testfunction(){Console.Wr
iteLine("Press enter to begin, or enter \"c\" to clear repos.");if (Console.ReadLine() == "c"){
RepoManagement.ClearRepo(Paths.DistributionRepo, Paths.ImageRepo);}DateTime then = DateTime.Now
;Console.WriteLine("Solving...");double L = 10;double H = 10;int n = 20;RBounds2D bounds = new
RBounds2D(0, L, 0, H);NCGrid_Distribution dist = new NCGrid_Distribution(bounds, n, n);Boundary
Conditions conditions = new BoundaryConditions(bounds, n, n, BoundaryConditions.BoundaryConditi
onType.Dirichlet);double dx = L / (n - 1);double max = 5;string fxn = string.Format("{0}*Sin({1
}*x/{2})*(Exp({1}*y/{2}) - Exp(-1*{1}*y/{2}))/(Exp({1}*{3}/{2}) - Exp(-1*{1}*{3}/{2}))", max, M
ath.PI, L, H);conditions.SetConstant(0, BoundaryConditions.Direction.Negative_X);conditions.Set
Constant(0, BoundaryConditions.Direction.Negative_Y);conditions.SetConstant(0, BoundaryConditio
ns.Direction.Positive_Y);for (int i = 0; i < n; i++){double x = i * dx;double z = max * Math.Si
n(Math.PI * x / L);conditions[i, BoundaryConditions.Direction.Positive_Y] = z;}int solcount = 5
0;double[] errors = new double[solcount];double[] iteration = new double[solcount];Distribution
SketchSettings S = DistributionSketchSettings.Fancy();S.SetFigureTitle("Temperature Distributio
n");for (int i = 0; i < solcount; i++){Console.WriteLine(i.ToString() + " of " + solcount.ToStr
ing() + " iterations processed.");iteration[i] = i;BVPLinear2D problem = new BVPLinear2D(condit
ions, LinearOperatorOrder2.Laplace, dist);problem.EnableConsoleOutput();NCGrid_Distribution sol
n = problem.SolveSRDD();NCGrid_Distribution analytic = ExactFunctionGeneratorVB2D.GenerateFunct
ionToGrid(fxn, soln);soln.ApplyMeshMorphGA(15);errors[i] = ErrorEstimation.NormDifference(soln,
 analytic);string title = "iterative-" + i.ToString();soln.WriteToFile(title);DistributionSketc
h2D sketch = new DistributionSketch2D(soln, S);dist = soln.Clone();sketch.CreateSketch(true);sk
etch.SaveImage(title + "-plot", false);}List<string> filestuff = new List<string>();for (int i
= 0; i < iteration.Length; i++){filestuff.Add(iteration[i].ToString() + "," + errors[i].ToStrin
g());}File.WriteAllLines(@"C:\Users\Will\Desktop\Folders\MATH435\repo\curves-1d\errors-temp.csv
", filestuff.ToArray());Console.WriteLine("Done.");Console.ReadLine();}static void testfunction
2(){Console.WriteLine("Press enter to begin, or enter \"c\" to clear repos.");if (Console.ReadL
ine() == "c"){RepoManagement.ClearRepo(Paths.DistributionRepo, Paths.ImageRepo);}DateTime then
= DateTime.Now;Console.WriteLine("Solving...");double L = 10;double H = 10;int n = 38;RBounds2D
 bounds = new RBounds2D(0, L, 0, H);NCGrid_Distribution dist = new NCGrid_Distribution(bounds,
n, n);BoundaryConditions conditions = new BoundaryConditions(bounds, n, n, BoundaryConditions.B
oundaryConditionType.Dirichlet);double dx = L / (n - 1);double omega = 4;double max = 8;string
fxn = string.Format("{4}*Sin({0}*{1}*x/{2})*Exp(y/{3})", Math.PI,omega,L,H, max);ExactFunctionG
eneratorVB2D.quickPlot(fxn, "analytic");conditions.SetConstant(0, BoundaryConditions.Direction.
Negative_X);conditions.SetConstant(0, BoundaryConditions.Direction.Positive_X);for (int i = 0;
i < n; i++){double x = i * dx;double z_neg = max * Math.Sin(Math.PI * omega*x / L);double z_pos
 = max * Math.E*Math.Sin(Math.PI * omega*x / L);conditions[i, BoundaryConditions.Direction.Posi
tive_Y] = z_pos;conditions[i, BoundaryConditions.Direction.Negative_Y] = z_neg;}int solcount =
15;LinearOperatorOrder2 op = new LinearOperatorOrder2(0, 0, 0, 1 / (H * H), Math.PI * Math.PI *
 omega * omega / (L * L), 0);double[] errors = new double[solcount];double[] iteration = new do
uble[solcount];DistributionSketchSettings S = DistributionSketchSettings.Fancy();S.SetFigureTit
le("Temperature Distribution");for (int i = 0; i < solcount; i++){Console.WriteLine(i.ToString(
) + " of " + solcount.ToString() + " iterations processed.");iteration[i] = i;BVPLinear2D probl
em = new BVPLinear2D(conditions, op, dist);problem.EnableConsoleOutput();NCGrid_Distribution so
ln = problem.SolveSRDD();NCGrid_Distribution analytic = ExactFunctionGeneratorVB2D.GenerateFunc
tionToGrid(fxn, soln);soln.ApplyMeshMorphGA(15, 0.0000015);errors[i] = ErrorEstimation.NormDiff
erence(soln, analytic);string title = "iterative-" + i.ToString();soln.WriteToFile(title);Distr
ibutionSketch2D sketch = new DistributionSketch2D(soln, S);dist = soln.Clone();sketch.CreateSke
tch(true);sketch.SaveImage(title + "-plot", false);}List<string> filestuff = new List<string>()
;for (int i = 0; i < iteration.Length; i++){filestuff.Add(iteration[i].ToString() + "," + error
```

```
s[i].ToString());}File.WriteAllLines(@"C:\Users\Will\Desktop\Folders\MATH435\repo\curves-1d\err
ors-temp.csv", filestuff.ToArray());Console.WriteLine("Done.");Console.ReadLine();}static void
testfunction3(){Console.WriteLine("Press enter to begin, or enter \"c\" to clear repos.");if (C
onsole.ReadLine() == "c"){RepoManagement.ClearRepo(Paths.DistributionRepo, Paths.ImageRepo);}Da
teTime then = DateTime.Now;Console.WriteLine("Solving...");double L = 10;double H = 10;int n =
26;RBounds2D bounds = new RBounds2D(0, L, 0, H);NCGrid_Distribution dist = new NCGrid_Distribut
ion(bounds, n, n);BoundaryConditions conditions = new BoundaryConditions(bounds, n, n, Boundary
Conditions.BoundaryConditionType.Dirichlet);double dx = L / (n - 1);double omega = 4;double max
 = 8;conditions.SetConstant(0, BoundaryConditions.Direction.Negative_X);conditions.SetConstant(
0, BoundaryConditions.Direction.Negative_Y);for (int i = 0; i < n; i++){double x = i * dx;doubl
e y = i * dx;double ybound = max / (1 + Math.Exp(-1 * (omega * x / L)));double xbound = max / (
1 + Math.Exp(-1 * (omega * y / H)));conditions[i, BoundaryConditions.Direction.Positive_Y] = yb
ound;conditions[i, BoundaryConditions.Direction.Positive_X] = xbound;}int solcount = 15;LinearO
peratorOrder2 op = LinearOperatorOrder2.Laplace;DistributionSketchSettings S = DistributionSket
chSettings.Fancy();S.SetFigureTitle("Double Logistic Boundary");for (int i = 0; i < solcount; i
++){Console.WriteLine(i.ToString() + " of " + solcount.ToString() + " iterations processed.");B
VPLinear2D problem = new BVPLinear2D(conditions, op, dist);problem.EnableConsoleOutput();NCGrid
_Distribution soln = problem.SolveSRDD();string title = "iterative-" + i.ToString();soln.WriteT
oFile(title);DistributionSketch2D sketch = new DistributionSketch2D(soln, S);dist = soln.Clone(
);sketch.CreateSketch(true);sketch.SaveImage(title + "-plot", false);//dist.ApplyMeshMorphGA(2,
 0.0019);Random R = new Random();for (int h = 1; h < dist.Xcount-1; h++){for (int k = 1; k < di
st.Ycount-1; k++){double ddx = (0.5-R.NextDouble()) * dx*0.6;double ddy = (0.5-R.NextDouble())
* dx*0.6;Vector3 move = new Vector3(ddx, ddy, 0);dist[h, k] = dist[h, k] + move;}}}Console.Writ
eLine("Done.");Console.ReadLine();}public static void printMatrix(Matrix M){int shelfsize = 8;i
nt trunc = 6;for (int i = 0; i < M.Rows; i++){string line = string.Empty;for (int j = 0; j < M.
Columns; j++){string entry = M[i, j].ToString();string tr_entry = entry.Substring(0, Math.Min(t
runc, entry.Length));line += tr_entry.PadRight(shelfsize);}Console.WriteLine(line);}}static str
ing bufferint(int n, int size){string l = string.Empty;for (int i = 0; i < size + 1 - n.ToStrin
g().Length; i++){l += "0";}l += n.ToString();return l;}}}/* Copyright (c) 2018 William van Noor
dt */using System;using System.Collections.Generic;using System.Linq;using System.Text;using Sy
stem.Threading.Tasks;namespace m_435_NCAMR{class RBounds2D{private double xmin, xmax, ymin, yma
x;public double Xmin{get { return xmin; }set { xmin = value; }}public double Xmax{get { return
xmax; }set { xmax = value; }}public double Ymin{get { return ymin; }set { ymin = value; }}publi
c double Ymax{get { return ymax; }set { ymax = value; }}public RBounds2D(double _xmin, double _
xmax, double _ymin, double _ymax){xmin = _xmin;ymin = _ymin;xmax = _xmax;ymax = _ymax;}public o
verride string ToString(){return "bounds:" + xmin.ToString() + ":" + xmax.ToString() + ":" + ym
in.ToString() + ":" + ymax.ToString();}public static RBounds2D fromstring(string source){string
[] ar = source.Split(':');double xn, xx, yn, yx;if (!(double.TryParse(ar[1], out xn) && double.
TryParse(ar[2], out xx) && double.TryParse(ar[3], out yn) && double.TryParse(ar[4], out yx))) {
 throw new ArgumentException("String improperly formatted."); }else { return new RBounds2D(xn,
xx, yn, yx); }}public static RBounds2D Square(double radius){return new RBounds2D(-radius, radi
us, -radius, radius);}}}/* Copyright (c) 2018 William van Noordt */using System;using System.Co
llections.Generic;using System.Linq;using System.Text;using System.Threading.Tasks;using System
.IO;namespace m_435_NCAMR{static class RepoManagement{public static void ClearRepo(params strin
g[] abs_repo_dists){List<string[]> arrs = new List<string[]>();foreach (string i in abs_repo_di
sts){arrs.Add(Directory.GetFiles(i));}foreach (string[] repnames in arrs){foreach (string y in
repnames){File.Delete(y);}}}}}/* Copyright (c) 2018 William van Noordt */using System;using Sys
tem.Collections.Generic;using System.Linq;using System.Text;using System.Threading.Tasks;using
Matrix_v1;namespace m_435_NCAMR{class TaylorSystem{private Matrix taylor_coefs, rhs;public Matr
ix TaylorCoefficients{get { return taylor_coefs; }}public Matrix RHS{get { return rhs; }}public
 TaylorSystem(NCAMRNode[] stencil){BuildTaylorCoeffs(stencil);}private void BuildTaylorCoeffs(N
CAMRNode[] stencil){double[] deltas_u ={stencil[1].Value-stencil[0].Value,stencil[2].Value-sten
cil[0].Value,stencil[3].Value-stencil[0].Value,stencil[4].Value-stencil[0].Value,stencil[5].Val
ue-stencil[0].Value};double[] deltas_x ={stencil[1].X-stencil[0].X,stencil[2].X-stencil[0].X,st
encil[3].X-stencil[0].X,stencil[4].X-stencil[0].X,stencil[5].X-stencil[0].X};double[] deltas_y
={stencil[1].Y-stencil[0].Y,stencil[2].Y-stencil[0].Y,stencil[3].Y-stencil[0].Y,stencil[4].Y-st
```

```
encil[0].Y,stencil[5].Y-stencil[0].Y};double[] matrix_contents ={deltas_x[0], deltas_y[0], 0.5*
sq(deltas_x[0]), 0.5*sq(deltas_y[0]), deltas_x[0]*deltas_y[0],deltas_x[1], deltas_y[1], 0.5*sq(
deltas_x[1]), 0.5*sq(deltas_y[1]), deltas_x[1]*deltas_y[1],deltas_x[2], deltas_y[2], 0.5*sq(del
tas_x[2]), 0.5*sq(deltas_y[2]), deltas_x[2]*deltas_y[2],deltas_x[3], deltas_y[3], 0.5*sq(deltas
_x[3]), 0.5*sq(deltas_y[3]), deltas_x[3]*deltas_y[3],deltas_x[4], deltas_y[4], 0.5*sq(deltas_x[
4]), 0.5*sq(deltas_y[4]), deltas_x[4]*deltas_y[4]};Matrix delta = new Matrix(5, 1, deltas_u);Ma
trix A = new Matrix(5, 5, matrix_contents);taylor_coefs = A;rhs = delta;string[] filestuff = {
taylor_coefs.Latexoutput(true) };}private double sq(double to){return to * to;}}}
```