AN ABSTRACT OF THE THESIS OF

Joseph G. Umhoefer for the degree of Master of Science in Mathematics
presented on March 14, 2016.
Title: Interpolation Schemes for Two Dimensional Flow with Applications

Abstract approved: _____

Malgorzata S. Peszyńska

In this thesis we study a numerical analysis problem motivated by the need to simulate an event such as an oil spill in a deep water environment. Numerical simulation can help to mitigate the disastrous effects of such events by aiding the management of risk assessment and recovery efforts.

However, an accurate simulation of the physical processes involved in the oil spill requires highly sophisticated and accurate numerical models. Equally important is that such a simulation needs to have accurate hydrodynamics data which may either come from observations or from some other computation simulator which predicts the flow of water near the area involved in the spill. In this thesis we discuss a particular technical problem involved with proper interpretation and use of hydrodynamic data.

In numerical analysis, it is often necessary to approximate a given function or interpolate from discrete data. One may have discrete data from sampling or due to solving a partial differential equation at discrete points but require information between the nodes. The motivation for this investigation of interpolation on scattered data is to recreate a smooth function from hydrodynamic data. In other words, we will discuss algorithms that provide a smooth field from given discrete data. The Blowout and Spill Occurrence Model (BLOSOM) developed by the Department of Energy's National Energy Technology Laboratory models hydrocarbon release events from the sea floor to the final fate of the oil. The generated smooth field could be used in such

a model, potentially improving the predicted outcomes.

The errors in prediction of the fate of oil arise, of course, from multiple sources. We study the errors due to the interpolation scheme applied. One particular aspect is also associated with whether the interpolated velocities have non-physical characteristics, specifically whether the interpolated velocities are conservative, given that the true velocities are. Ultimately, we achieve good results using radial basis function interpolation, but the scale of the problem needs to be considered further, as the large data sets in use may make the problem intractable.

Interpolation Schemes for Two Dimensional Flow with Applications

by
Joseph G. Umhoefer

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Presented March 14, 2016
Commencement June 2016

Master of Science thesis of Joseph G. Umhoefer presented on March 14, 2016

APPROVED:

Major Professor, representing Mathematics

Chair of the Department of Mathematics

Dean of the Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

Joseph G. Umhoefer, Author

ACKNOWLEDGEMENTS

I must first thank my advisor, Dr. Małgorzata Peszyńska, whose patience, support, and guidance made this work possible.

I am indebted to Kelly Rose and Lawrence Sim for providing me with the opportunity to participate in the development of the Blowout and Spill Occurrence Model as an Oak Ridge Institute for Science and Education (ORISE) Fellow. This opportunity inspired the work herein. And I am also grateful to the rest of the BLOSOM team, who made work such an enjoyable experience.

This work began as part of National Energy Technology Laboratory (NETL) research for the Department of Energy's (DOE) Complementary Research Program under section 999 of the Energy Policy Act of 2005 and the Bureau of Safety and Environmental Enforcement (BSEE), U.S. Department of the Interior, Washington, D.C., under agreement E14PG00045.

Finally, I try to convey my love for my family, though I know the words will fall short. For my parents, who support all my enterprises. For my siblings, whose achievements continue to set the bar ever higher. And for Bob Moczynski, who keeps me focused on the bigger picture.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

Page

LIST OF FIGURES

LIST OF FIGURES (Continued)

LIST OF FIGURES (Continued)

For Michael Curtis Umhoefer

Interpolation Schemes for Two Dimensional Flow with Applications

1 Introduction

In this thesis we study a numerical analysis problem motivated by the need to simulate an event such as an oil spill in a deep water environment. Oil spills such as Deepwater Horizon are catastrophic events that can lead to economic losses in the tourism and seafood industries, and a significant decrease in ecological health [1].

Numerical simulation can help to mitigate the disastrous effects of such events by aiding the management of risk assessment and recovery efforts. In particular, accurate simulation of the fate of spilled oil can help identify high risk locations and guide use of containment booms.

However, an accurate simulation of the physical processes involved in the oil spill requires highly sophisticated and accurate numerical models. Equally important is that such a simulation needs to have accurate hydrodynamics data which may come from either observations or from some other computational simulator which predicts the flow of water near the area involved in the spill.

In this thesis we discuss a particular technical problem involved with proper interpretation and use of hydrodynamic data. Assuming that the hydrodynamic data is given at a finite set of nodes

$$\Delta = \{x_i\}_{i=1}^{N}, \tag{1.1}$$

our goal is to discuss algorithms that interpolate this data for the needs of a Lagrangian transport model. We also study possible errors that arise and methods to quantify them.

In numerical analysis, it is often necessary to approximate a given function or interpolate from discrete data. One may have discrete data from sampling or due

to solving a partial differential equation at discrete points, but require information between the nodes. Another possibility is that a function may be known, but is computationally expensive to evaluate, so a simpler approximation is desired. Some methods for resolving these issues are interpolation with polynomials, inverse distance weighting schemes, and radial basis functions.

In this thesis we will consider fluid velocity data

$$(u_i, v_i),\ 1 \le i \le N \tag{1.2}$$

given at (1.1), that are associated either with a rectangular grid or are scattered, i.e. this data could come from observations or an unstructured grid. We will study the interpolation schemes that are naturally applicable for each type of data. The most sophisticated method presented is radial basis function interpolation.

The motivation for this investigation of interpolation on scattered data is to recreate a smooth function representing hydrodynamic data. In other words, we will discuss algorithms that provide a smooth field, $(\tilde{u}, \tilde{v})$ from the data (1.2).

The Blowout and Spill Occurrence Model (BLOSOM) developed by the Department of Energy's National Energy Technology Laboratory models hydrocarbon release events from the sea floor to the final fate of the oil. Some examples of oil fate are beaching, evaporation and degradation. Since the model is designed to use many varieties of hydrodynamic data and hydrodynamic data has no single standard format, BLOSOM needs to be flexible enough to interpolate data that may arrive on different sorts of meshes or unmeshed.

To account for different types of meshes, or to use scattered experimental data, one might use a meshfree interpolation scheme. In contrast, traditional polynomial interpolation requires a well defined structured or unstructured grid which is fixed. One of the advantages of inverse distance weighting or radial basis functions is that they are not tied to any particular grid.

In this thesis we introduce the traditional polynomial interpolation as well as the two aforementioned meshfree interpolation schemes. Then we evaluate the use of

these schemes on three test problems. The first test problem is the Hagen-Poiseuille problem, which has unidirectional flow in a channel. The second test problem is uniform flow through a channel. The last test problem presented is a variation of the lid-driven cavity problem. For each of these problems, we study the interpolation error in the velocity. We also evaluate the error of the quantity of interest. This quantity is the result of Lagrangian transport schemes which would use the interpolated velocities.

The errors in prediction of the fate of oil arise, of course, from multiple sources. Some errors are inherent errors in the velocity data, e.g. hydrodynamic simulators do not always produce divergence free flow [6], but some are entirely due to the interpolation scheme applied. One particular aspect is also associated with whether the interpolated velocities have the physical characteristics of the true velocities. The true velocities are known to be conservative, i.e., they satisfy the divergence free property,

$$\nabla \cdot (u, v) = 0.$$

Unfortunately, the mass conservation property is not, in general, satisfied by approximations $(\tilde{u}, \tilde{v})$, which may have substantial consequences on the quality of the predicted transport solutions.

The outline of this thesis is as follows. In Chapter 2 we provide an introduction to problems considered and discuss ways to correct the interpolated velocity field, $(\tilde{u}, \tilde{v})$, and to post-process it to create a conservative velocity field $(\hat{u}, \hat{v})$. We also introduce examples of velocity fields that will be used as test problems; this includes the Hagen-Poiseuille flow, uniform flow and the lid-driven cavity problem. In Chapter 3 we provide the mathematical setup for the process of interpolation. In Chapter 4 we introduce the oil spill problem, and overview the BLOSOM model. We also give details of the interpolation model in BLOSOM. In Chapter 5 we present three examples of how the interpolation techniques introduced in Chapter 3 can be applied to modeling velocity fields.

2 Introduction to fluid flow and transport models needed for oil fate simulations

In this chapter we introduce some of the notation used throughout the thesis, as well as some background for the problems considered. We first discuss incompressible flow and a potential pitfall in the numerical interpolation thereof. We then describe the Lagrangian transport method, i.e., how to solve the transport equation with the method of characteristics, and how to calculate numerical solutions based on the characteristic curves. Lastly, we introduce the test problems and some auxiliary notions.

2.1 Incompressible flow

Consider a two dimensional, steady state fluid flow problem with eastward velocity, u, and northward velocity, v. Let $u, v : \Omega \to \mathbb{R}$, where $\Omega \subset \mathbb{R}^2$ is the physical domain of interest.

The fluid is incompressible if the velocity is divergence free, i.e.

$$\nabla \cdot (u, v) = 0. \tag{2.1}$$

In numerical simulations, a method that provides velocity approximations, $(\tilde{u}, \tilde{v})$ to (u, v), that preserve the divergence free property is called conservative [6].

Let the concentration of spilled oil at $(x, y, t) \in \Omega \times [0, \infty)$ be denoted by $c(x, y, t)$. Note that $c \in [0, 1]$ is a dimensionless ratio. With a velocity field given by (u, v), conservation of mass with no sources or sinks tells us that this concentration is advected with velocity $(u(x, y, t), v(x, y, t))$ according to the following partial differential equation,

$$\frac{\partial c}{\partial t} + \nabla \cdot (c(u, v)) = 0, \ t > 0, \ (x, y) \in \Omega. \tag{2.2}$$

Then, by Equation (2.1), we can calculate from Equation (2.2) that

$$\frac{\partial c}{\partial t} + \nabla c \cdot (u, v) = 0. \tag{2.3}$$

The method of characteristics can then be used to solve Equation (2.3) for given initial values. We discuss this in Section 2.2.

If a numerical method produces velocities $(\tilde{u}, \tilde{v})$ that are not conservative, i.e. $\nabla \cdot (\tilde{u}, \tilde{v}) \neq 0$, we get

$$\frac{\partial c}{\partial t} + \nabla c \cdot (\tilde{u}, \tilde{v}) = -c \nabla \cdot (\tilde{u}, \tilde{v}). \tag{2.4}$$

Applying method of characteristics to this equation without accounting for the right hand side amounts to a non-physical source or sink for the concentration, which will lead to non-physical solutions and is a numerical artifact.

The numerical artifact which arises if $\nabla \cdot (\tilde{u}, \tilde{v}) \neq 0$ can be introduced by an interpolation scheme or may even come directly from the hydrodynamic data, as many hydrodynamics models used in practice do not provide conservative velocities [6].

2.1.1 Creating a mass conservative velocity field on a mesh

One can correct a velocity field if it is to be used in a grid-based transport solver.

A projection method for post-processing hydrodynamic data on a mesh is given in [5]. Implementing this process would allow an oil spill model to reshape the data in a way that either preserves or creates mass conservation. One reason to reshape the hydrodynamic data is to adjust it for use with a particular interpolation scheme, such as a rectangular grid for use with tensor product interpolation.

We first describe the process in [5]. Suppose the hydrodynamic data is given on an unstructured mesh of a region $\Omega \subset \mathbb{R}^2$ with boundary $\partial\Omega$. Let $\mathbf{U} = (u, v)$ and suppose that we have Dirichlet boundary conditions on the normal velocity on a

portion of the boundary, Ω_1,

$$\mathbf{U} \cdot \nu = g, \quad \text{on } \partial\Omega_1,$$

where ν is the outward pointing unit normal.

For a given mesh on Ω, let h^0 be the mesh parameter and V_{h^0} be the finite dimensional subspace of interpolating functions. Let $\mathbf{U}_{h^0} \in V_{h^0}$, such that $\mathbf{U}_{h^0} \approx \mathbf{U}$ and $\nabla \cdot \mathbf{U}_{h^0} \neq 0$, i.e. $\mathbf{U}_{h^0}$ is an approximation to the true velocities and is not divergence free.

Then, on a new mesh of Ω, let h^n be the mesh parameter and V_{h^n} be the finite dimensional subspace of interpolating functions. We want to find the interpolant $\mathbf{U}_{h^n} \in V_{h^n}$ such that $\mathbf{U}_{h^n}$ is a good approximation to $\mathbf{U}_{h^0}$ and such that $\mathbf{U}_{h^n}$ is mass conservative element-by-element. By finding $\mathbf{U}_{h^n}$ such that the conservation is element-by-element, we are solving a local conservation problem, so this process may be applied to only a subset of the domain Ω.

Thus, for our steady state, incompressible problem, we want to have

$$\begin{cases} \nabla \cdot \mathbf{U}_{h^n} = 0 & \text{in } \Omega, \\ \mathbf{U}_{h^n} \cdot \nu = g & \text{on } \partial\Omega_1. \end{cases} \tag{2.5}$$

The trick is how to find U_{h^n}. Let $\mathcal{P}_{h^n}\mathbf{U}_{h^0}$ be the L^2 projection of $\mathbf{U}_{h^0}$ into V_{h^n}, i.e., it is the best approximation in V_{h^n} of $\mathbf{U}_{h^0}$ as measured by the L^2 norm, i.e., $\|\mathcal{P}_{h^n}\mathbf{U}_{h^0} - \mathbf{U}_{h^0}\|_{L^2} \leq \|\mathbf{u}_{h^n} - \mathbf{U}_{h^0}\|_{L^2}$ for all $\mathbf{u}_{h^n} \in V_{h^n}$. In [5] it was proposed that the new velocity, $\mathbf{U}_{h^n}$, is the sum of $\mathcal{P}_{h^n}\mathbf{U}_{h^0}$ and a correction term $\Gamma_{h^n} \in V_{h^n}$ that needs to be calculated, i.e.

$$\mathbf{U}_{h^n} = \mathcal{P}_{h^n}\mathbf{U}_{h^0} + \Gamma_{h^n}. \tag{2.6}$$

To find a problem satisfied by Γ_{h^n}, we substitute Equation (2.6) into Equation (2.5), producing the following boundary value problem with the newly defined

source terms $\tilde{f}$ and $\tilde{g}$:

$$\begin{cases} \nabla \cdot \Gamma_{h^n} = -\nabla \cdot \mathcal{P}_{h^n} \mathbf{U}_{h^0} \equiv \tilde{f} & \text{in } \Omega, \\ \Gamma_{h^n} \cdot \nu = g - \mathcal{P}_{h^n} \mathbf{U}_{h^0} \cdot \nu \equiv \tilde{g} & \text{on } \partial\Omega_1. \end{cases} \tag{2.7}$$

Here, $\tilde{f}$ and $\tilde{g}$ are calculated from the original approximation, $\mathbf{U}_{h^0}$, to the true velocities, $\mathbf{U}$.

Next, we suppose that the correction term, Γ_{h^n} is a the gradient of a scalar function, ϕ_{h^n}, that can be thought of as a pseudo-pressure. We have

$$\Gamma_{h^n} = -\nabla \phi_{h^n}. \tag{2.8}$$

We substitute Equation (2.8) into the boundary value problem (2.7), to get the new boundary value problem to be solved for ϕ_{h^n},

$$\begin{cases} -\Delta \phi_{h^n} = \tilde{f} & \text{in } \Omega, \\ -\nabla \phi_{h_n} \cdot \nu = \tilde{g} & \text{on } \partial\Omega_1, \\ \phi_{h^n} = 0 & \text{on } \partial\Omega / \partial\Omega_1. \end{cases} \tag{2.9}$$

We observe that Equation (2.10) is a second order, elliptic partial differential equation. Second order, elliptic partial differential equations are steady state differential equations such that when written in the form

$$\sum_{i,j=1}^{n} a_{ij} u_{x_i, x_j} = f(\mathbf{x}, u, \mathbf{p}), \tag{2.10}$$

the matrix (a_{ij}) of its coefficients is positive definite [11, p. 45]. In Equation (2.10), $\mathbf{x} = (x_1, \ldots, x_n)$, $\mathbf{p} = (p_1, \ldots, p_n)$ with $p_j = u_{x_j}$, $j = 1, \ldots, n$.

Next, we want solve the boundary value problem in Equation (2.10) numerically. The method used should approximate the fluxes (Γ) and the pseudo-pressures (ϕ). If the underlying method produces a solution that has continuous fluxes across the element edges, then the numerical solution will be mass conservative element-wise, that is, we want $\Gamma \cdot \nu = -\nabla \phi \cdot \nu$. One such method is the mixed/hybrid finite element method, as used in [5]

2.1.2 Correcting for a non-mass conservative field when no mesh is given

The method described in Section 2.1.1 requires that the domain has a mesh that is being solved on, but not all interpolation schemes require a mesh. So if the method being used is meshfree, a different correction scheme is necessary.

Suppose we have a problem where the divergence of $(\tilde{u}, \tilde{v})$ is nonzero, i.e. $\nabla \cdot (\tilde{u}, \tilde{v}) \neq 0$, and we must correct for the numerical sources or sinks presented in Equation (2.4) without a mesh. Suppose also that we are solving a Lagrangian transport problem. Then we may consider the correction proposed in [6].

For this correction method, we refer to Equation (2.4) to get a new problem with source term $-c\nabla \cdot (\tilde{u}, \tilde{v})$. This problem can then be solved through the method of characteristics which is described in Section 2.2.

The two methods described in Sections 2.1.1 and 2.1.2 for correcting the loss of conservation are not implemented in this thesis. We observe, however, that the interpolation methods presented herein are not in general divergence free. We show in Chapter 5 the divergence of the interpolations on the given test problems.

2.2 Lagrangian transport

Given a velocity field (u, v), one can simulate the transport of particles using either traditional grid based Eulerian methods or using Lagrangian techniques. The latter relies on transport along the characteristics found numerically and produces solutions close to those found analytically by the method of characteristics as described in [11].

For an oil model, there are several reasons why it is advantageous to use Lagrangian methods instead of Eulerian. For a given spill scenario, it is known where the oil originates; a blown wellhead can be considered a point source. So it is known that initially the oil is concentrated in a small region of the domain, thus calculating

Lagrangian transport for the oil parcels could potentially cut down on computational costs relative to a global Eulerian method. As the oil spill evolves, the oil may spread, limiting this advantage.

It may also be the case that the hydrodynamic data being used in the simulation was not designed explicitly for the use of the oil spill model. So, again, if the domain is unnecessarily large, Eulerian techniques could be costly.

2.2.1 Method of characteristics

Let $u(x,y), v(x,y)$ be the eastward and northward velocities, respectively, and $c(x,y,t) \in [0,1]$ be concentration of oil. We consider the transport problem in the domain $\Omega \subset \mathbb{R}^2$, $t \geq 0$ given by Equation (2.2), and the additional initial condition,

$$\begin{cases} \frac{\partial c}{\partial t} + \nabla \cdot (c(u,v)) = 0, \ t > 0, \ (x,y) \in \Omega, \\ c(x_0(s), y_0(s), 0) = c_0(s), \end{cases} \tag{2.11}$$

where $s = (s_1, s_2)$. We want to solve this problem with the method of characteristics, i.e., we want to find characteristic curves $(x(\tau), y(\tau), c(\tau))$ that pass through $(x_0(s), y_0(s), c_0(s))$ when $\tau = 0$. First, we will expand the terms and simplify, assuming that the flow is divergence free, i.e. $\nabla \cdot (u,v) = 0$. We have

$$\frac{\partial c}{\partial t} + \nabla c \cdot (u,v) + c\nabla \cdot (u,v) = \frac{\partial c}{\partial t} + \nabla c \cdot (u,v) = \frac{\partial c}{\partial t} + \frac{\partial c}{\partial x}u + \frac{\partial c}{\partial y}v = 0.$$

Thus the Problem (2.11) is equivalent to

$$\begin{cases} \frac{\partial c}{\partial t} + u(x,y)\frac{\partial c}{\partial x} + v(x,y)\frac{\partial c}{\partial y} = 0, \\ c(x_0(s), y_0(s), 0) = c_0(s). \end{cases} \tag{2.12}$$

Now, by Theorem 2-1 in [11, p. 25], if $u(x,y)$ and $v(x,y)$ are continuously differentiable in Ω, the curve of initial conditions $(x_0(s), y_0(s), c_0(s)) \in \Omega \times [0,1]$ is continuously differentiable, and if

$$\frac{\partial(x,y,t)}{\partial(\tau, s_1, s_2)} \neq 0,$$

then there exists a unique continuously differentiable solution to Problem (2.12) for $0 \le t \le T$.

To construct this solution through the method of characteristics, we solve the following system of ordinary differential equations,

$$\begin{cases} \frac{dt}{d\tau} = 1, \\ \frac{dx}{d\tau} = u(x,y), \\ \frac{dy}{d\tau} = v(x,y), \\ \frac{dc}{d\tau} = 0. \end{cases} \tag{2.13}$$

We solve these ordinary differential equations to find the trajectories. In the case of constant u and v we get

$$\begin{cases} t = \tau + t_0(s), \\ x = u\tau + x_0(s), \\ y = v\tau + y_0(s), \\ c = c_0(x_0(s), y_0(s)). \end{cases} \tag{2.14}$$

Let $t_0 = 0$, solve for x_0, y_0 and we substitute to find

$$c(x,y,t) = c_0(x - ut, y - vt).$$

Thus the solution to Equation (2.12) is constant along the characteristic curves. Solving this system creates a surface of solutions based on the curve of initial conditions $(x_0(s), y_0(s), c_0(s))$.

For non-constant u and v, the solution $c(x,y,t)$ is still constant along the characteristic curves, but the characteristic curves derived from Equation (2.13) must be solved numerically; this is explained below.

2.2.2 Runge-Kutta methods

The trajectories of the particles in Equation (2.13) can be computed numerically using any variety of ODE solvers, such as one from the Runge-Kutta family.

In this thesis, we use a multistage Runge-Kutta method which evaluates the velocities at multiple locations for each time step.

If the velocities used to calculate the streamlines are expensive to find, then a possible improvement in computation cost could be found by using a multistep method. If, for example, the velocities are being approximated by a function with support Ω in a function space of dimension N, then the method must sum the evaluation of N basis functions. A multistage method will have to evaluate the basis functions repeatedly for each time step. A multistep function would store the evaluation for a given time step to be used in future time steps. Changing methods could maintain the accuracy, but one could reduce computation cost on finding function values [15, p. 130].

The method used for the test problems to solve the initial value problems (IVP) was the four stage, fourth order Runge-Kutta method as developed in [15, p. 126]. Since the velocities in the test problems considered are steady, the system is autonomous. To solve Equation (2.13) with this method, we first let $t_0(s) = 0$, so $t = \tau$. We then use the Runge-Kutta method with $\frac{dx}{d\tau} = u(x, y)$ and $\frac{dy}{d\tau} = v(x, y)$.

Let X^n, Y^n be the numerical solution at time step n, $t_n = n\Delta t$, and $k = \Delta t$. Then define

$$
\begin{aligned}
L_1 &= X^n & M_1 &= Y^n \\
L_2 &= X^n + \frac{1}{2}ku(L_1, M_1) & M_2 &= Y^n + \frac{1}{2}kv(L_1, M_1) \\
L_3 &= X^n + \frac{1}{2}ku(L_2, M_2) & M_3 &= Y^n + \frac{1}{2}kv(L_2, M_2) \\
L_4 &= X^n + ku(L_3, M_3) & M_4 &= Y^n + kv(L_3, M_3)
\end{aligned}
$$

$$
\begin{aligned}
X^{n+1} &= X^n + \frac{k}{6}\left[u(L_1, M_1) + 2u(L_2, M_2) + 2u(L_3, M_3) + u(L_4, M_4)\right], \\
Y^{n+1} &= Y^n + \frac{k}{6}\left[v(L_1, M_1) + 2v\left(L_2, M_2\right) + 2v\left(L_3, M_3\right) + v(L_4, M_4)\right].
\end{aligned}
$$

This four stage, fourth order Runge-Kutta method calculates the streamlines for Equation (2.13). For an implementation example see Appendix A.

2.3 Hagen-Poiseuille flow

The Hagen-Poiseuille equation is a physical law that describes the pressure drop of flow in a long uniformly cylindrical tube [14]. The flow is in the axial, x, direction of the tube and there is no flow in the radial, y direction. We assume that the fluid is Newtonian and incompressible and that the flow is laminar and steady state with no-slip boundary conditions. Under these assumptions, one can derive the parabolic velocity profile from the Navier-Stokes equations. By Equation (16.13) in [14, p. 798], we have

$$u(x,y) = -\frac{1}{4\mu}\frac{\partial p}{\partial x}(R^2 - y^2),\ y \in [0,1] \tag{2.15}$$
$$v(x,y) = 0$$

where R is the radius of the tube, μ is the viscosity and the pressure gradient $\frac{\partial p}{\partial x}$ is constant since we have fully developed flow. A cartoon of the flow is shown in Figure 2.1.

For our test problem, we'll look at the domain $[0,1]\times[0,1]$ and choose relative parameters $\frac{1}{\mu}\frac{\partial p}{\partial x} = 10$, so that flow over time $T = 1$ gives appropriate results, i.e. the transport results remain in the domain. So Equation (2.15) becomes

$$u(y) = \frac{5}{2}\big((0.5)^2 - (y-0.5)^2\big),\ y \in [0,1]. \tag{2.16}$$

2.4 The lid-driven cavity problem

The lid-driven cavity is a standard two-dimensional test problem for numerical methods in fluid dynamics. In [20], a variation of this problem is introduced by specifying a particular velocity, $u(x,1)$, along the top boundary. The variation in [20] eliminates the top corners as possible singularities, but at a cost of introducing a source term. It also has the advantage of having a known analytic solution.

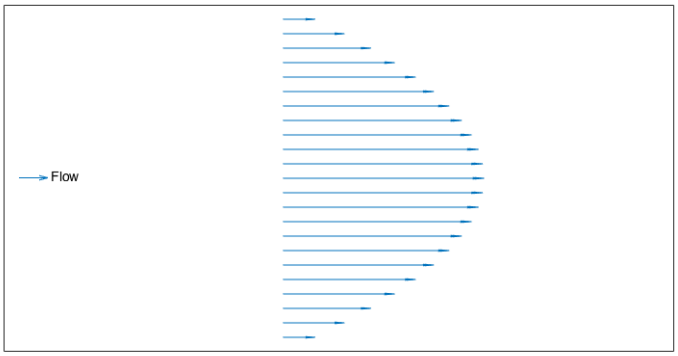

FIGURE 2.1: Visualization of the Hagen-Poiseuille flow.

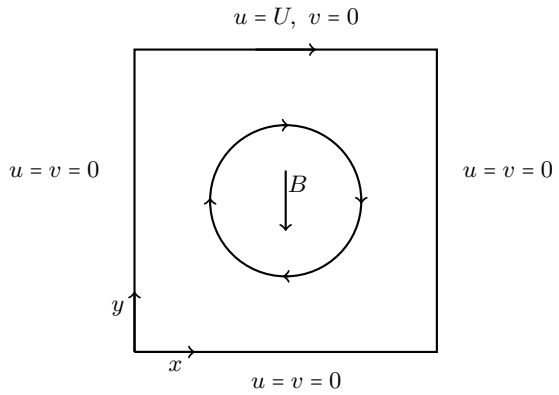

FIGURE 2.2: A diagram of the lid-driven cavity problem.

Consider a unit square, with Dirichlet boundary conditions on the sides $y = 0, x = 0, x = 1$. On the top of the box, $y = 1$, the boundary is moving with horizontal velocity, $u(x, 1)$, see Figure 2.2.

Let $\mathbf{u} = (u, v)$. Then the governing equations for this problem are

$$\begin{aligned} \nabla \cdot \mathbf{u} &= 0, \\ \mathbf{u} \cdot \nabla u &= \frac{1}{\mathrm{Re}} \Delta u - \frac{\partial p}{\partial x}, \\ \mathbf{u} \cdot \nabla v &= \frac{1}{\mathrm{Re}} \Delta v - \frac{\partial p}{\partial y} - B(x, y, Re) \end{aligned}$$

where B is a body force. Along the top surface

$$u(x,1) = 16(x^4 - 2x^3 + x^2), \quad x \in [0,1]$$

The exact solution to this problem is known to be

$$u(x,y) = 8(x^4 - 2x^3 + x^2)(4y^3 - 2y),$$
$$v(x,y) = -8(4x^3 - 6x^2 + 2x)(y^4 - y^2).$$

The exact solution of this problem is shown in Figure 2.3.

2.5 Low discrepancy sets

For many of the examples in this thesis, we will use a particular set of scattered data nodes called a low discrepancy set. We follow [7] and introduce some definitions.

The first objects we need are subsets of the unit square. We consider intervals of the torus $\mathbb{T}^k = \mathbb{R}^k / \mathbb{Z}^k$, that has edges aligned with the coordinate axes. Then, for an interval that extends past the boundary of the unit square, it merely continues back at the axis. We will count the points in one of these given intervals with a function A and compare it to the measure of the interval.

First, we let $J = [a_1, b_1) \times \cdots \times [a_k, b_k) \subseteq \mathbb{R}^k$, a rectangle in the k dimensional space $\mathbb{R}^k$, with $0 < b_i - a_i \leq 1$ for $1 \leq i \leq k$. Then, for the torus $\mathbb{T}^k = \mathbb{R}^k / \mathbb{Z}^k$, we have the interval $I = J/\mathbb{Z}^k$. Let μ_k be Lebesgue measure on $\mathbb{R}^k$. Then, for any interval $I \subseteq \mathbb{R}^k / \mathbb{Z}^k$ and sequence $\{x_n\}_{n=1}^{\infty}$, $x_n \in \mathbb{R}^k$, let $A(I, N, x_n)$ be the number of points, x_n, $1 \leq n \leq N$, for which $\{x_n\} \in I$. That is,

$$A(I, N, x_n) = \sum_{n=1}^{N} \Xi_I(x_n),$$

where Ξ_I is the characteristic function of I.

We can use definition 1.5 in [7, p. 4]:

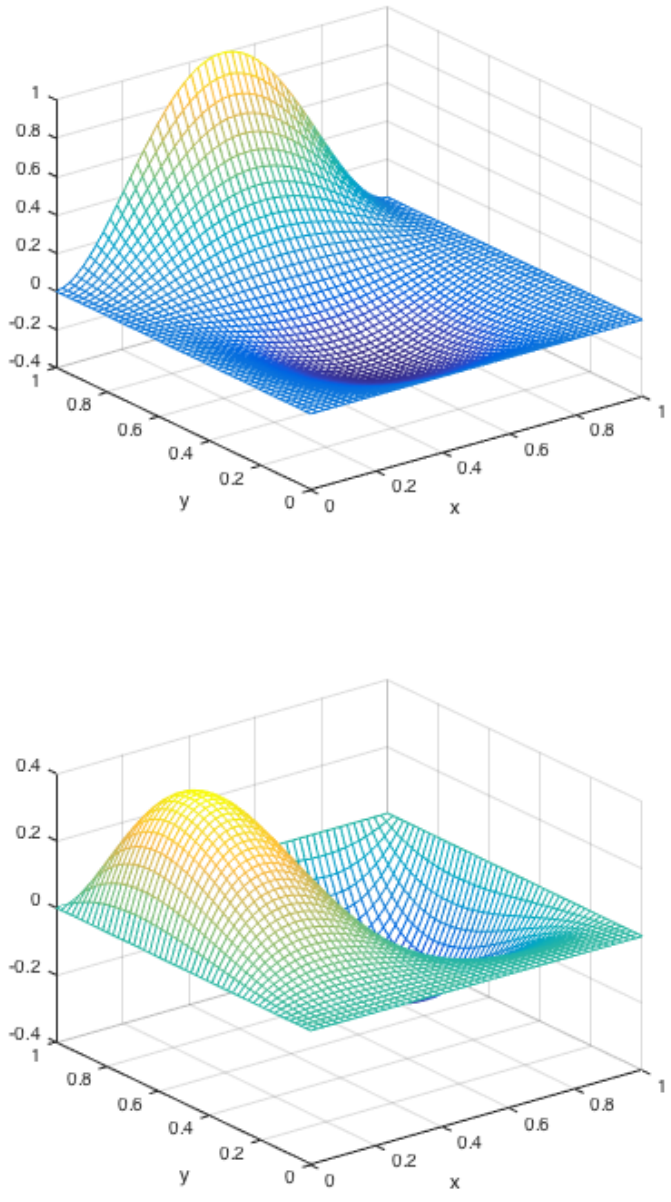

FIGURE 2.3: The exact solutions, u, v to the lid-driven cavity problem, defined in Section 2.4. Velocity u on the top and velocity v on the bottom.

Definition 2.5.1. *Let* $\{x_n\}_{n=1}^N$ *be a finite sequence in* $\mathbb{R}^k$*. Then, define the* ***discrepancy****,* D_n *as*

$$D_N = \sup_{I \subseteq \mathbb{T}^k} \left| \frac{A(I, N, x_n)}{N} - \mu_k(I) \right|. \tag{2.17}$$

Equation 2.17 implies that for a low discrepancy set, the proportion of a sequence in a set A is roughly equal to the measure of A.

To generate sets of low-discrepancy nodes, we use the sequence defined in [12]. Let $b_1, \ldots, b_{d-1}$ be coprime positive integers, $b_i \geq 1,\ 1 \leq i \leq d-1$. For each $n \leq N$, write n in it's b_i-ary representation

$$n = a_0 + a_1 b_i + \cdots a_m b_i^m.$$

Then write the digits in reverse order, following a point. Call this function g_{b_i},

$$g_{b_i}(n) = a_0 b_i^{-1} + a_1 b_i^{-2} + \cdots + a_m b_i^{-m-1}. \tag{2.18}$$

Then the d-dimensional Hammersley set of N nodes is defined by

$$x_n = (g_{b_1}(n), \ldots, g_{b_{d_{-1}}}(n), \frac{n}{N}), \quad 1 \leq n \leq N.$$

From [12], we know this set has discrepancy

$$D_N \leq (\log N)^{d-1} \prod_{i=1}^{d-1} \left(\frac{3b_i - 2}{\log b_i} \right).$$

To generate the sequences, we use the `Hammersley` library for MATLAB, provided by [13].

3 Background of interpolation

In this section we introduce several interpolation schemes. We begin with polynomial interpolation in one dimension, then proceed to multivariate interpolation with tensor products. We then present two meshfree methods, inverse distance weighting and radial basis function interpolation.

3.1 Interpolation in one spatial dimension

A natural starting point is to look at interpolation in one spatial dimension. Consider a finite set, Δ, of nodes on $\mathbb{R}$, on which one has data they wish to interpolate. In our development of polynomial interpolation and piecewise polynomial interpolation we follow [2]. We will start with polynomial interpolation.

3.1.1 Polynomial interpolation

First, we will develop global polynomial interpolation through the use of Lagrange polynomials, where **global** means a single polynomial approximating the data over the entire domain. Let $f : \mathbb{R} \to \mathbb{R}$ by given and let $\{x_i\}$ be a set of distinct points in $\mathbb{R}$ for $0 \le i \le N$, and define $y_i = f(x_i)$. Then there exists a unique polynomial, $p(x)$, of degree at most N such that $p(x_i) = y_i$ for $0 \le i \le N$. Define

$$p(x) \equiv \sum_{i=0}^{N} y_i l_i(x)$$

where $l_i(x)$ are the **Lagrange Interpolating Polynomials**, defined as

$$\begin{aligned} l_i(x) &\equiv \frac{(x-x_0)(x-x_1)\cdots(x-x_{i-1})(x-x_{i+1})\cdots(x-x_N)}{(x_i-x_0)(x_i-x_1)\cdots(x_i-x_{i-1})(x_i-x_{i+1})\cdots(x_i-x_N)} \\ &= \prod_{\substack{j=0 \\ j\neq i}}^{N} \frac{x-x_j}{x_i-x_j}. \end{aligned}$$

The polynomials $l_i(x)$ satisfy the Kronecker delta property, $l_i(x_j) = \delta_{ij}$, i.e. $l_i(x_j) = 1$ for $i = j$ and $l_i(x_j) = 0$ otherwise. With this we find that

$$p(x_j) = \sum_{i=0}^{N} y_i l_i(x_j) = \sum_{i=0}^{N} y_i \delta_{ij} = y_j.$$

Since each polynomial $l_i(x)$ has degree N, the interpolating polynomial has degree at most N.

To see that the polynomial $p(x)$ is unique, consider the polynomial q, such that q also has degree at most N and $q(x_i) = y_i$ for $0 \le i \le N$. Then, for the polynomial $r(x) = p(x) - q(x)$ we find $r(x_i) = p(x_i) - q(x_i) = y_i - y_i = 0$ for $0 \le i \le N$. So r has $N+1$ zeros but has degree at most N, so, by the fundamental theorem of algebra [23, p.51], we conclude that $r(x) \equiv 0$. Hence, $p = q$.

3.1.1.1 Error estimates for polynomial interpolation

First, we define how we measure error. Let $f : [a, b] \to \mathbb{R}$ and $\|\cdot\|_\infty$ denote the sup-norm, i.e. $\|f\|_\infty = \sup_{x\in[a,b]} |f(x)|$. Let $\|\cdot\|_{L^2}$ denote the standard L^2-norm, i.e. $\|f\|_{L^2} = \left(\int_a^b |f|^2\right)^{1/2}$.

Let $f : [a, b] \to \mathbb{R}$ be a given function interpolated by p_N at the nodes $\{x_i\}_{i=0}^N$. If $f \in C^{N+1}([a, b])$, then

$$f(x) - p_N(x) = \frac{f^{(N+1)}(\xi)}{(N+1)!} w(x) \tag{3.1}$$

for some $\xi \in [a, b]$ and $w(x) = \prod_{i=0}^{N}(x - x_i)$. Then we have

$$\|f - p_N\|_\infty \le \frac{\|f^{(N+1)}\|_\infty}{(N+1)!} |w(x)|. \tag{3.2}$$

The proof of the estimate (3.2) follows [2, 3]

Proof of (3.1). Define the error term $R_N(x) = f(x) - p_N(x)$. For fixed $x \ne x_i$, $0 \le i \le N$, let $Y(t) = R_N(t) - \frac{R_N(x)}{w(x)} w(t)$. We have that $Y(x) = R_N(x) - \frac{R_N(x)}{w(x)} w(x) = 0$ and that $Y(x_i) = R_N(x_i) + \frac{R_N(x)}{w(x)} w(x_i) = 0$ for $0 \le i \le N$. Thus Y is a polynomial with $N + 2$ zeros in $[a, b]$.

By repeated use of Rolle's theorem, we find that $Y^{(N+1)}(t)$ has 1 root, $\xi \in [a,b]$. Since p_N is a polynomial of degree at most n, $p_N^{(N+1)} = 0$, so $R_N^{(N+1)}(x) = f^{(N+1)}(x)$. Since $w(t)$ is a monic polynomial of degree N, $w^{(N+1)}(t) = (N+1)!$. Then $Y^{(N+1)}(t) = f^{(N+1)}(t) - \frac{R_n(x)}{w(x)}(N+1)!$. But $Y^{(N+1)}(\xi) = 0$, so

$$f^{(N+1)}(\xi) - \frac{R_n(x)}{w(x)}(N+1)! = 0$$

□

Consider the case where the interpolation points are uniformly spaced, so $x_{i+1} - x_i = h = \frac{b-a}{N}$ and $x_i = a + ih$, $0 \le i \le N$. Since distance is invariant under translation, we may suppose $a = 0$. Then we have a correspondence between the set $\Delta = \{x - x_i\}_{i=0}^N$ and $\overline{\Delta} = \{h/2, h, 2h, \ldots, Nh\}$, where for exactly one i, $|x - x_i| < h/2$, then for the next nearest point, x_j, $|x - x_j| < h$. We can continue in this manner, so each entry in Δ has a unique corresponding bound in $\overline{\Delta}$.

Consider the node x_i such that $0 < x - x_i < h$. Define, $g(x) = |(x - x_i)(x - x_{i+1})| = (x - ih)((i+1)h - x)$ on $[ih, (i+1)h]$. Then $g'(x) = -2x + (2i+1)h$, and the critical point of $g(x)$ is $x = ih + h/2$. So $g(x) \le (ih + h/2 - ih)((i+1)h - ih + h/2) = h/4$. Note that these two terms correspond to the values $h/2, h \in \overline{\Delta}$. Then we have,

$$|w(x)| \le \prod_{j=0}^{N} |x - x_j| \le \frac{h}{4} \prod_{\substack{j=0 \\ j \ne i, i+1}}^{N} jh \le \frac{h}{4} \prod_{j=1}^{N} jh = \frac{N! h^{N+1}}{4}. \tag{3.3}$$

Combining the result in Equation (3.3) with the prior estimate for the sup-norm error in Equation (3.2), we find

$$\|f - p_N\|_\infty \le \frac{\|f^{(N+1)}\|_\infty}{4(N+1)} h^{N+1}.$$

One way to get a tighter bound on polynomial interpolation is to adjust the interpolation points to minimize the oscillation, $\max |w(x)|$. Define the Chebyshev nodes as the set $\{\cos(\frac{2(N-i)+1}{2N+2}\pi)\}_{i=0}^N$, which takes values in $[-1, 1]$. By [24, p. 152], it can be shown that $\min_{\{x_i\}_{i=0}^N} \max_{x \in [-1,1]} |w(x)| = 2^{-n}$. This minimum is achieved by

the Chebyshev nodes. When interpolating on Chebyshev nodes in the interval $[-1,1]$, the interpolation estimate (3.2) becomes

$$\|f - p_N\|_\infty \leq \frac{\|f^{N+1}\|_\infty}{2^N(N+1)!}.$$

Unfortunately, it is the case that $\|f - p_N\|_\infty$ need not converge to 0 as $N \to \infty$ if general nodes are used.

3.1.1.2 Polynomial interpolation of the Runge function

While the Weierstrass approximation theorem states that every continuous function defined on a closed interval can be approximated as closely as desired by a polynomial, it does not prescribe how. A naive placement of uniformly spaced roots can lead to significant errors, so other methods have been developed. Chebyshev nodes are just one such improvement.

We now provide the classical example to illustrate the difficulties of interpolation with higher order polynomials. Consider the Runge function, $f(x) = \frac{1}{1+\alpha x^2},\ \alpha > 0$. We will use $\alpha = 25$. Let p_N be the Lagrange interpolating polynomial of degree N, with nodes $\{x_i\}_{i=0}^N,\ x_i = -1 + i\frac{2}{N}$, i.e. the nodes are uniformly spaced through the domain $[-1,1]$, with $h = \Delta x = x_i - x_{i-1} = \frac{2}{N}$. Then our error bound is

$$\|f - p_N\|_\infty \leq \frac{\|f^{(N+1)}\|_\infty}{4(N+1)}\left(\frac{2}{N}\right)^{N+1} = \frac{2^{N-1}}{(N+1)N^{N+1}}\|f^{(N+1)}\|_\infty.$$

It is the case that as we refine the step size, h, and correspondingly increase the degree, N, of our approximating polynomial, the bound on the Runge function gets worse. This is due to two terms, $\|f^{(N+1)}\|_\infty$ and $|w(x)|$, in the estimate (3.2) that grow without bound as $N \to \infty$. While the coefficient $\frac{2^{N-1}}{(N+1)N^{N+1}} \to 0$ as $N \to \infty$, we find that $\|f^{(N+1)}\|_\infty$ grows faster.

However, if we approximate f with Chebyshev nodes, the error estimate is

$$\|f - p_N\|_\infty \leq \frac{1}{2^N(N+1)!}\|f^{(N+1)}\|_\infty.$$

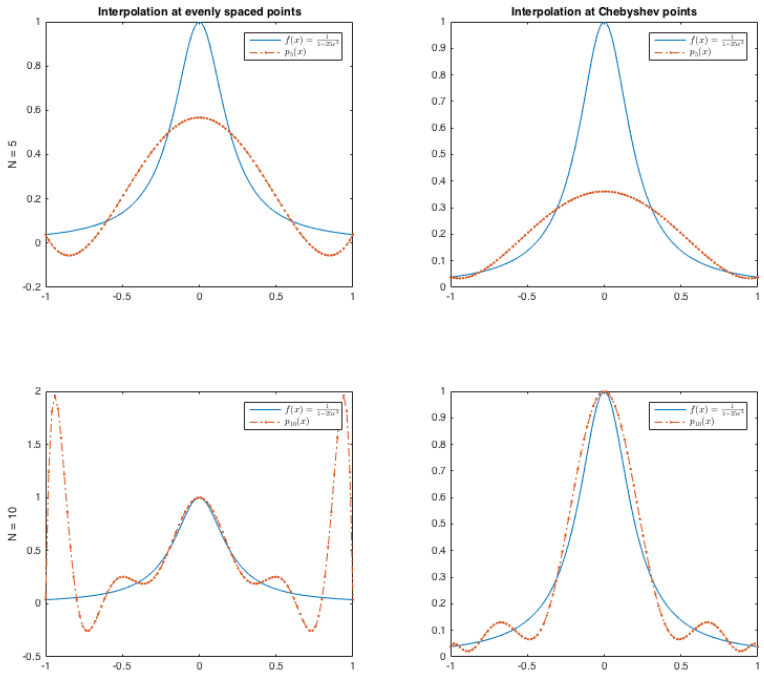

FIGURE 3.1: Polynomial interpolation of the Runge function, with $\alpha = 25$. Left: uniformly spaced nodes. Right: Chebyshev nodes. Top: N = 5. Bottom: N = 10.

Presented in Figure 3.1 are the results of interpolating f, with both uniformly spaced nodes and Chebyshev nodes, where the degree of interpolating polynomial is $N = 5$ or $N = 10$. We also present in Table 3.1 the error of each interpolation, as measured by sup-norm and L^2-norm. What we find is that when using uniformly spaced nodes as interpolation points, the approximation actually gets worse as N increases from 5 to 10, in both sup-norm and L^2-norm error. However, when using the Chebyshev nodes, our approximation error decreases in both norms as N increases.

3.1.2 Piecewise polynomial interpolation

In our example of interpolating the Runge function with a global polynomial with uniformly spaced nodes, we had the counterintuitive result that more data lead to a worse interpolation. One way to improve our polynomial interpolation and take

	$N = 5$		$N = 10$	
	$\|\|\cdot\|\|_\infty$	$\|\|\cdot\|\|_{L^2}$	$\|\|\cdot\|\|_\infty$	$\|\|\cdot\|\|_{L^2}$
$h = 2/N$	0.43	0.21	1.92	0.87
Chebyshev nodes	0.64	0.30	0.13	0.09

TABLE 3.1: Error results of polynomial interpolation of the Runge function

advantage of finer data is with piecewise polynomials. Here, we will partition the domain into subintervals with the number of nodes in each subinterval based on what degree of interpolating polynomial we wish to use.

For example, if we want to interpolate a function, f, with a piecewise quadratic, p_2, partition the domain into $[x_0, x_2]$, $[x_2, x_4]$, $\ldots$, $[x_{N-2}, x_N]$. Each of these intervals is called an **element.** With 2 nodes in each element of the domain, the local interpolating polynomial will be a unique affine function. With 3 nodes in each element, the local interpolating polynomial will be quadratic, and so on. For a given number of nodes, N, we must have that N is divisible by the degree of the interpolating polynomials.

Define a piecewise Lagrange basis function for each node, so the interpolating basis still has the same dimension as the number of nodes. If a node is on the boundary of an element, the corresponding basis function has value on each bordering element and is 0 otherwise. For nodes in the interior of an element, the basis function has value only inside the element. For example, consider piecewise quadratic Lagrange polynomials on the interval $[x_i, x_{i+2}]$. Then we have

$$l_i(x) = \begin{cases} \frac{(x-x_{i-1})(x-x_{i-2})}{(x_i-x_{i-1})(x_i-x_{i-2})}, & x \in [x_{i-2}, x_i] \\ \frac{(x-x_{i+1})(x-x_{i+2})}{(x_i-x_{i+1})(x_i-x_{i+2})}, & x \in [x_i, x_{i+2}] \\ 0, & \text{otherwise} \end{cases},$$

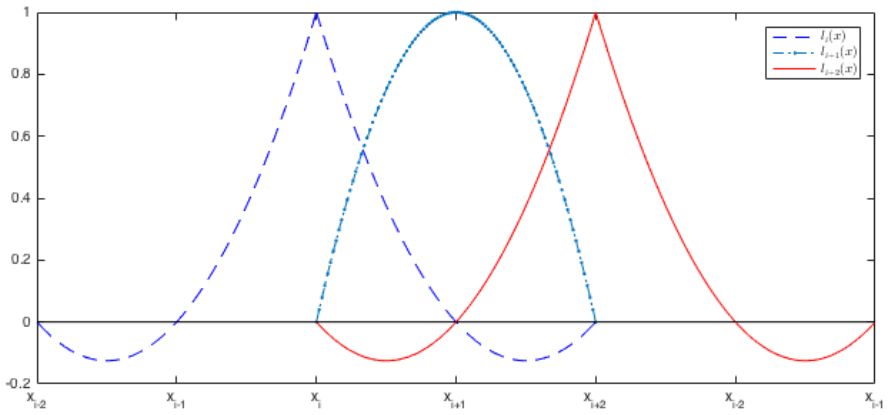

FIGURE 3.2: The quadratic piecewise Lagrange basis functions l_i, l_{i+1}, l_{i+2} on the interval $[x_{i-2}, x_{i+2}]$.

$$l_{i+1}(x) = \begin{cases} \frac{(x-x_i)(x-x_{i+2})}{(x_{i+1}-x_i)(x_{i+1}-x_{i+2})}, & x \in [x_i, x_{i+2}] \\ 0, & \text{otherwise} \end{cases}.$$

These functions are shown in Figure 3.2. Note that for interpolation with non-constant polynomials, at x_i, $0 \le i \le N$, we have $p(x_i) = \sum_{i=0}^{N} y_i l_i(x_i) = y_i$. In particular, this holds for the boundary nodes, so $p \in C^0([a,b])$.

3.1.2.1 Error estimates for piecewise polynomial interpolation

The error estimates for piecewise polynomial interpolation will start with Equation (3.2) from the error estimates of global polynomial interpolation. Here, though, we don't need the interpolated function, f, to be in $C^{N+1}([a,b])$. We can relax the conditions so that for piecewise interpolation with polynomials of degree n, we have $f \in C^{n+1}$. So on each element, with interpolating function p_n, Equation (3.2) tells us that

$$\|f - p_n\|_\infty \le \frac{\|f^{(n+1)}\|_\infty}{(n+1)!}|w(x)|, \tag{3.4}$$

so again we need to find a bound on $|w(x)|$. We follow [2] to develop an estimate. Let $x \in [x_i, x_{i+n}]$. Then, for some $a \in [0, n]$ we have $x = x_i + ah$. Then,

$$\begin{aligned} |w(x)| &= |(x - x_i)\cdots(x - x_{i+n})| \\ &= |(x_i + ah - x_i)\cdots(x_i + ah - (x_i + nh))| \\ &= |(ah)\cdots(ah - nh)| \\ &= |a\cdots(a-n)|h^{n+1}. \end{aligned} \tag{3.5}$$

But from Equation (3.5) we see that $\prod_{i=0}^{n}(a-i)$ is a polynomial in a with compact support $[0, n]$, and so it is bounded. Let $\prod_{i=0}^{n}(a-i) < C$. $\prod_{i=0}^{n}(a-i)$ is dependent on the interpolating degree n, but not on the step size h, so C is independent of h. So, $w(x)$ is $O(h^{n+1})$. Combining this result with Equation (3.4) shows

$$\|f - p_n\|_\infty \leq \frac{Ch^{n+1}}{(n+1)!}\|f^{(n+1)}\|_\infty. \tag{3.6}$$

3.1.2.2 Piecewise polynomial interpolation of the Runge function

Figure 3.3 shows how the Runge function can be approximated with piecewise linear polynomials and piecewise quadratics, with $N = 4,\ 8$ uniformly spaced nodes. One can see a clear improvement in the interpolation over the global polynomials, even with fewer data points.

3.2 Multivariate interpolation on a grid

Suppose that we wanted to interpolate data given in more than one spatial dimension. There are, of course, many reasons for wanting to do so; say, interpolating measurements taken at different weather stations. Of the multitude of ways in which one can do multivariate interpolation, the ones introduced here will be tensor product, inverse distance weighting, and radial basis function.

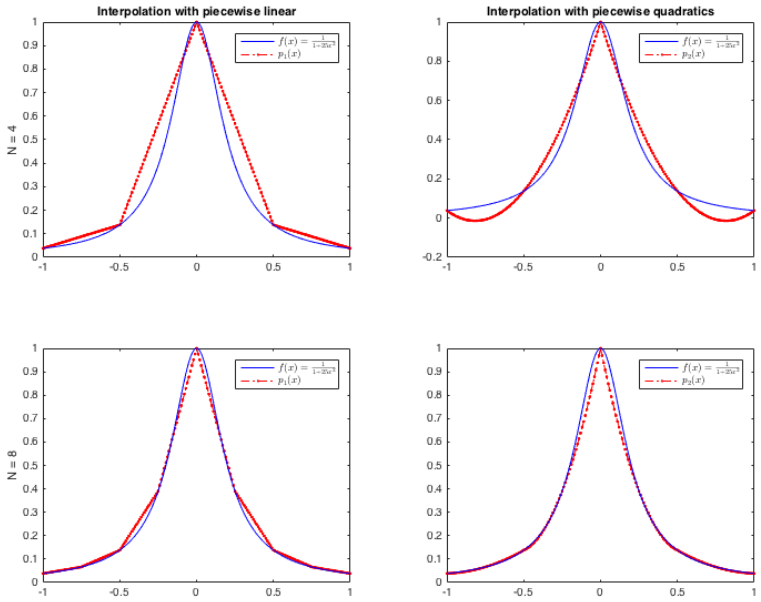

FIGURE 3.3: Piecewise polynomial interpolation of the Runge function. Left: piecewise linear. Right: piecewise quadratic.

3.2.1 Tensor product interpolation

Tensor product interpolation is a natural extension of the polynomial interpolation covered in Section 3.1. I will develop the interpolation method in two dimensions, but the ideas scale readily to any space of arbitrary dimension, given appropriate nodes [2, p. 56].

Let $\Delta_x = \{x_0, x_1, \ldots, x_N\}$ and $\Delta_y = \{y_0, y_1, \ldots, y_M\}$ be partitions of $[a, b] \subset \mathbb{R}$ and $[c, d] \subset \mathbb{R}$, respectively. We want to consider their cartesian product $\Delta_x \times \Delta_y = \{(x_i, y_j) : 0 \leq i \leq N,\ 0 \leq j \leq M\} \subset \mathbb{R}^2$. Then for a function $f : [a, b] \times [c, d] \to \mathbb{R}$, we want to find an interpolant, p_n, such that $p_n(x_i, y_j) = f(x_i, y_j)$, $0 \leq i \leq M,\ 0 \leq j \leq N$, and p_n is a bivariate piecewise polynomial with maximal degree n.

As a basis for the interpolant, we will take ideas from piecewise Langrange interpolation. Let $\{\psi_i^n\}_{i=0}^N$ be the set of piecewise Lagrange interpolating functions on Δ_x and let $\{\phi_j^m\}_{j=0}^M$ be the set of piecewise Lagrange interpolating functions on Δ_y,

where n and m are the maximum degree of the interpolating functions. Note that n need not equal m. Then the basis functions for the tensor product interpolating functions are $\Pi_{n,m} = \{\psi_i^n \phi_j^m : 0 \le i \le N, 0 \le j \le M\}$. Note that $\Pi_{n,m}$ contains all linear combinations

$$\sum_{i=0}^{N} \sum_{j=0}^{M} c_{i,j} \psi_i(x) \phi_j(y).$$

The basis function for $\Pi_{1,1}$ corresponding to node (x_i, y_j) is shown in Figure 3.4. For $\Pi_{1,1}$, the interpolating functions are linear along constant x or y and quadratic otherwise.

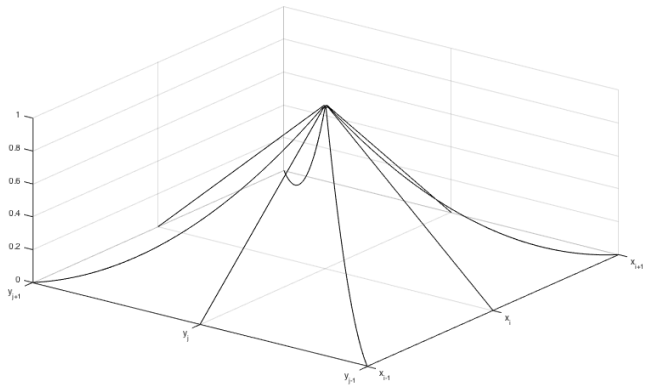

FIGURE 3.4: The basis function $\pi_{i,j}(x) = \psi_i^1(x)\phi_j^1(x)$.

3.2.1.1 Error estimates for tensor product interpolation

To find the error estimates in the sup-norm, we follow the discussion in [2, p. 56]. First, we need a defintion.

Definition 3.2.1. *Let* $\Delta_i = \{x_0^{(i)}, \ldots, x_{N_i}^{(i)}\}$ *be a partition of* $[a_i, b_i]$ *and* $\Pi_{n_i}(\Delta_i) = span\{l_0^i, \ldots, l_{N_i}^i\}$*, the space of piecewise Lagrange interpolating polynomials of maximum degree* n_i *on* $[a_i, b_i]$*. Then the* ***interpolatory projections*** *of* $f : [a_1, b_1] \times \cdots \times [a_M, b_M] \to \mathbb{R}$*, where* f *is* (n_i+1)*-times differentiable on* $[a_i, b_i]$*,* $\pi_1 : C^{n_1+1}([a_1, b_1]) \to$

$\Pi_{n_1}(\Delta_1),\ \ldots,\ \pi_M : C^{n_M+1}([a_M, b_M]) \to \Pi_{n_M}(\Delta_M)$ *are defined as*

$$(\pi_1 f)(x^{(1)}, \ldots, x^{(M)}) \coloneqq \sum_{i=0}^{N_1} f(x_i^{(1)}, x^{(2)}, \ldots, x^{(M)}) l_i^1(x^{(1)})$$

$$\vdots$$

$$(\pi_M f)(x^{(1)}, \ldots, x^{(M)}) \coloneqq \sum_{i=0}^{N_M} f(x^{(1)}, \ldots, x^{(M-1)}, x_i^{(M)}) l_i^M(x^{(M)})$$

For example, let $f(\cdot, y) \in C^3([a, b])$ and $f(x, \cdot) \in C^2([c, d])$, be interpolated, respectively, with piecewise quadratic and piecewise linear functions on $\Delta_1 = \{x_0, \ldots, x_N\}$, a partition of $[a, b]$, and $\Delta_2 = \{y_0, \ldots, y_M\}$, a partition of $[c, d]$. Then $\pi_1 : C^3([a, b]) \to \Pi_2(\Delta_1)$ and $\pi_2 : C^2([c, d]) \to \Pi_1(\Delta_2)$, where

$$(\pi_1 f)(x, y) = \sum_{i=0}^{N} f(x_i, y) l_i^1(x)$$
$$(\pi_2 f)(x, y) = \sum_{i=0}^{M} f(x, y_i) l_i^2(y).$$

Then, if we compose the projections, we find

$$(\pi_1 \pi_2 f)(x, y) = \sum_{i=0}^{N} \sum_{j=0}^{M} f(x_i, y_j) l_i^1(x) l_j^2(y). \tag{3.7}$$

We use the fact that Equation (3.7) is exactly the tensor product interpolation of f to find the error estimates. Recognizing that the interpolary projections are one dimensional piecewise polynomial interpolations, we use the estimate in Equation (3.6) to find $\|f - \pi_i f\| \le C_i h_i^{n_i}$, where again, n_i is the degree of the piecewise interpolating polynomial, $f \in C^{n_i+1}([a_i, b_i])$ and h_i is the step size of the grid in the i^{th} direction.

The following error estimate will be provided for a function on two dimensions, but is easily generalized to higher dimensions through mathematical induction. The generalization is omitted since it adds little, while the notation obfuscates the point. This result is Theorem 1.13 in [2].

Theorem 3.2.1. *Let Δ_1 and Δ_2 be partitions of $[a, b]$ and $[c, d]$ respectively, with step sizes h_1 and h_2. Let $\Pi(\Delta_1)$ and $\Pi(\Delta_2)$ be the piecewise polynomial spaces and*

π_1, π_2 their associated interpolatory projections. Suppose that $f(\cdot, y) \in C^p([a,b])$ and $f(x,\cdot) \in C^q([c,d])$, $f : [a,b] \times [c,d] \to \mathbb{R}$, so we have

$$\|f(\cdot,y) - \pi_1 f(\cdot,y)\|_\infty \le C_1 h_1^p, \qquad \|f(x,\cdot) - \pi_2 f(x,\cdot)\|_\infty \le C_2 h_2^q.$$

Then the tensor product interpolant, $\hat{f} = \pi_1\pi_2 f$ has the error estimate

$$\|f - \hat{f}\|_\infty \le C \max\{h_1^p, h_2^q\} \tag{3.8}$$

Proof. We use the identity $\hat{f} = \pi_1\pi_2 f$ and the triangle inequality to show

$$\begin{aligned}
\|f - \hat{f}\|_\infty &\le \|(f - \pi_1 f) + (\pi_1 f - \pi_1\pi_2 f) - (f - \pi_2 f) + (f - \pi_2 f)\|_\infty \\
&\le \|f - \pi_1 f\|_\infty + \|\pi_1(f - \pi_2 f) - (f - \pi_2 f)\|_\infty + \|f - \pi_2 f\|_\infty \\
&\le C_1 h_1^p + C_1 h_1^p + C_2 h_2^q \\
&\le C \max\{h_1^p, h_2^q\}
\end{aligned}$$

□

It is important to note that this estimate shows that the order of the tensor product method is equal to the least precise one-dimensional interpolation.

3.2.1.2 Bilinear interpolation in the lid-driven cavity problem

As an example of interpolation on a two dimensional grid, we'll consider a sourced lid-driven cavity problem. **Bilinear** interpolation is the tensor product interpolation in the case where products of piecewise linear functions are used as the basis of the interpolation. The Π_1 interpolation of this function is shown in Figure 3.5.

3.3 Meshfree interpolation methods

We will now consider meshfree methods for solving multivariate interpolation problems. Some of the advantages of working with meshfree methods are that they

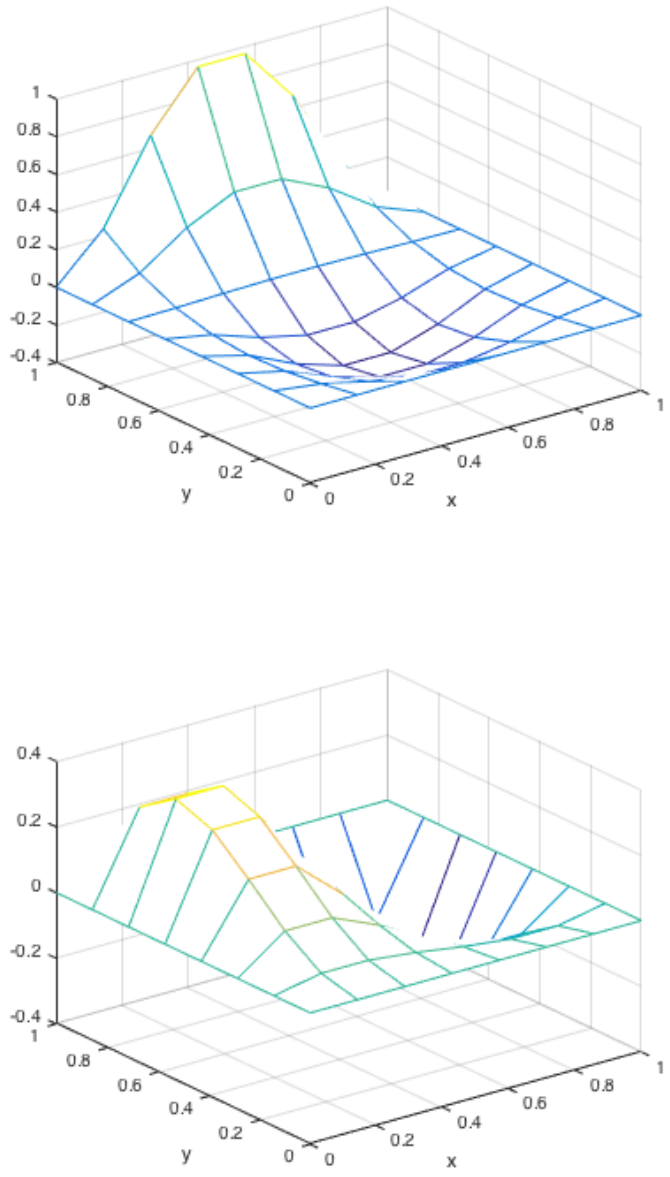

FIGURE 3.5: The bilinearly interpolated velocity of the lid-driven cavity problem. Compare with exact solutions shown in Figure 2.3. Top: $\hat{u}(x, y)$. Bottom: $\hat{v}(x, y)$.

are not restricted by the dimension of the domain and that geometric changes (free surfaces, deformations, etc.) to the domain are easily accounted for.

We can define a broad problem called **scattered data interpolation**, as found in [9, p. 3].

Definition 3.3.1 (Scattered data interpolation)**.** *Given data* $(\boldsymbol{x}_j, y_j)$, $j = 1, \ldots, N$, *with* $\boldsymbol{x}_j \in \mathbb{R}^d$, $y_j \in \mathbb{R}$, *we want to find a continuous function* p_n *such that* $p_n(\boldsymbol{x}_j) = y_j$, $j = 1, \ldots, N$.

We first introduce inverse distance weighting methods for solving the scattered data interpolation problem, then proceed to radial basis function interpolation.

3.3.1 Inverse distance weighting schemes

One limitation of using tensor product interpolation is that it requires a regular grid. Since these grids are not always available, more flexible schemes are necessary. Suppose instead we have a grid with scattered data. In this case, one may use an inverse distance weighting scheme for interpolation.

Inverse distance methods use a weighted average of known data to calculate the interpolated values. For a given location $\mathbf{x}$, the nearer a node, $\mathbf{x}_i$, is to $\mathbf{x}$, the more heavily the method weights the corresponding contribution the interpolated value.

Consider the case in $\mathbb{R}^2$. Let $\Delta = \{(x_i, y_i)\}$, $1 \leq i \leq N$ be scattered data points such that $f(x_i, y_i) = z_i$ is known. Let d be a metric on $\mathbb{R}^2$, with $d_i(x, y) = d((x_i, y_i), (x, y))$, and $W_i(x, y) = \frac{1}{d_i(x,y)}$. Then, as defined in [19], the general scheme for calculating the interpolant $\hat{f}$ via inverse distance weighting is

$$\hat{f}(x, y) = \begin{cases} \frac{\sum_1^N W_i(x,y) z_i}{\sum_1^N W_i(x,y)}, & d_i(x, y) \neq 0 \ \forall i \\ z_i, & d_i(x, y) = 0, \text{ for some } i \end{cases}.$$

It is clear that $\hat{f}$ is globally continuous, but evaluating $\hat{f}$ can be expensive when N is large. We consider a modification of this method that loses continuity, but, by

focusing on nearby nodes, maintains accurate interpolation and lowers the cost of interpolation.

In [10] a Modified Quadratic Shepard method was introduced, with metric d defined as Euclidean distance. It defines the interpolant, $\hat{f}$,

$$\hat{f}(x,y) = \frac{\sum_1^N W_i(x,y)Q_i(x,y)}{\sum_1^N W_i(x,y)}, \tag{3.9}$$

where Q_i are bivariate quadratic functions locally approximating $f(x_i, y_i)$. The weights are defined as

$$W_i(x,y) = \left(\frac{(R_w - d_i)^+}{R_w d_i}\right)^2, \tag{3.10}$$

where R_w is a fixed distance and

$$(R_w - d_i)^+ = \begin{cases} R_w - d_i, & \text{if } R_w > d_i, \\ 0, & \text{if } R_w < d_i. \end{cases}$$

This creates a maximum distance for which nodes are used to calculate the interpolated value. Any node outside that maximum distance has a weight of 0. This method can be shown to have quadratic accuracy [17].

Another version of this method was discussed in [17], the Modified Shepard's Method (MSM). In the MSM, the **radius of influence**, R_w, was varied with the interpolation point, (x, y), so that the interpolation at (x, y) would be derived from a fixed number of nodes, N_w. Let $B_x(r)$ be a ball of radius r centered at x. Then, for interpolation at a given point (x, y), we have $N_w = |\Delta \cap B_{(x,y)}(R_w)|$, where R_w is varied so N_w remains constant. For the MSM, the interpolant in Equation (3.9) becomes

$$\hat{f}(x,y) = \frac{\sum_1^{N_w} W_i(x,y)Q_i(x,y)}{\sum_1^{N_w} W_i(x,y)}. \tag{3.11}$$

The weights remain as given in Equation (3.10). In Figure 3.6, we show the radius of influence, R_w, for a given point. In this example $N_w = 8$.

The benefits of limiting R_w to maintain a constant N_w includes the fact that we can reduce the cost of evaluating the interpolation by removing nodes that have a small impact on the result; nodes that are further away have smaller weights. If N_w were too small, we would likely get poor results.

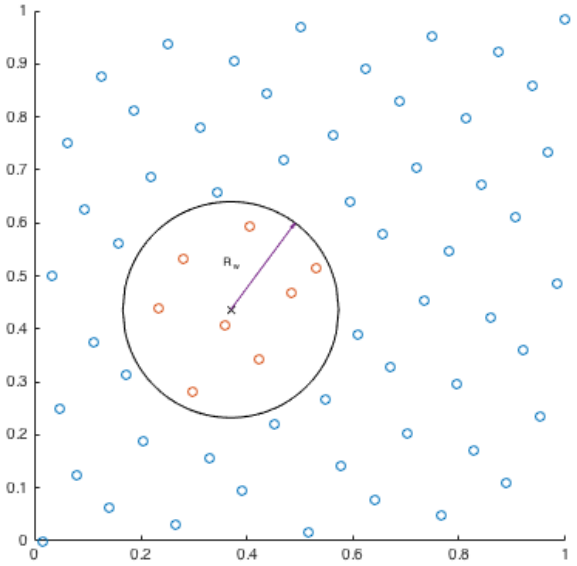

FIGURE 3.6: For interpolation at the point marked x, the radius R_w is varied, so that the number of data points used is constant. For the pictured example, the number of data points is $N_w = 8$.

The implementation tested in [17] was as accurate as the comparable triangle based methods. This method more readily scales up to 3 or more independent variables.

If we use for the bivariate function Q_i the piecewise constant nearest neighbor, i.e., $Q_i(x, y) = \{z_i : d_i(x, y) < d_j(x, y) \forall i \neq j\}$, then we can show that if the number of nodes used in the interpolation is 1, i.e. $N_w = 1$, the method reduces to nearest neighbor interpolation. Nearest neighbor interpolation assigns an interpolated value equal to that of the value at the closest node, so $\hat{f}(x, y) = z_i$ where $d_i(x, y) < d_j(x, y)$ for all $i \neq j$. For implementation of a nearest neighbor method the case where $d_i(x, y) =$

$d_j(x,y)$ for some i and j would need to be resolved, e.g. either through some means for choosing between z_i and z_j or averaging.

Consider Equation (3.9), a simple calculation shows that we have nearest neighbor interpolation when $N_w = 1$.

$$\hat{f}(x,y) = \frac{\sum_1^1 W_i(x,y)Q_i(x,y)}{\sum_1^1 W_i(x,y)} = \frac{W_1(x,y)Q_1(x,y)}{W_1(x,y)} = Q_1(x,y).$$

3.3.1.1 Inverse distance weighted interpolation of the lid-driven cavity problem

In this section we revisit the lid-driven cavity problem described in in 2.4.

This time we will use a set of points that are scattered. Specifically, we will use 64 Hammersley nodes, as described in Section 2.5.

The interpolation is done on an uniformly spaced grid on $[0,1]$, with 50 nodes in each the x and y direction. We want to determine how many nodes, N_w, should lie within the radius of influence to give the best interpolation results. To define what interpolation results are best, we consider the root-mean-square error (RMS-error), as given in [9, p. 10], Equation (1.5),

$$\text{RMS-error} = \sqrt{\frac{1}{M}\sum_{j=1}^{M}[\hat{f}(\xi_j) - f(\xi_j)]^2} = \frac{1}{\sqrt{M}}\|\hat{f} - f\|_{l^2},$$

where $\hat{f}$ is the interpolated solution to a function, f, the **interpolation points** are ξ_j, $1 \le j \le M$, M be the number of points at which the interpolation was calculated and $\|\cdot\|_{l^2}$ be the Euclidean 2-norm. In one dimension, this is close to the so called grid-norm.

If N_w is too large, then we may find that the interpolation is being influenced by data that isn't relevant due to the data's distance from the interpolation point. If N_w is too small, the interpolation may be missing significant nearby data.

Since the lid-driven cavity problem has a vector valued solution, there may be a disagreement as to what N_w is optimal. Recall that (u,v) is the velocity, and let $\hat{u}$ and $\hat{v}$ be the respective MSM interpolation defined in Equation (3.11). For each value

of N_w, we define a Total error, given in Equation (3.12), that combines the RMS error for the interpolation of each $\hat{u}$ and $\hat{v}$. Define

$$\begin{aligned}\text{Total error} &= \left\| \frac{\|\hat{u}-u\|_{l^2}}{\sqrt{M}} + \frac{\|\hat{v}-v\|_{l^2}}{\sqrt{M}} \right\|_{l^2} \\ &= \frac{1}{\sqrt{M}} \sqrt{\sum_{j=1}^{M} [\hat{u}(\xi_j)-u(\xi_j)]^2 + [\hat{v}(\xi_j)-v(\xi_j)]^2} \\ &= \frac{1}{\sqrt{M}} \|(\hat{u},\hat{v})-(u,v)\|_{l^2}. \end{aligned} \tag{3.12}$$

We calculated the total error for $N_w = 2, \ldots, 19$. The error is shown in Figure 3.7. One can see that the minimum error occurs at $N_w = 8$. The result of the interpolation with $N_w = 8$ is shown in Figure 3.8. Then, in Figure 3.9, we see that setting $N_w = 1$ produces the nearest neighbor interpolation.

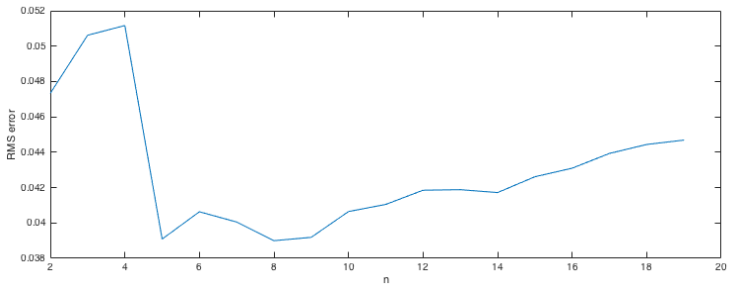

FIGURE 3.7: Total errors for interpolating the lid-driven cavity problem with modified Shepard's method interpolation where $N_w = 2, \ldots, 19$ and 64 nodes were used.

3.3.2 Radial basis function interpolation

We now turn to a different method for solving the scattered data interpolation problem. This development follows [9]. Here, our solution will use the common approach of constructing the approximating function by a linear combination of functions, B_k,

$$p(\mathbf{x}) = \sum_{k=1}^{N} c_k B_k(\mathbf{x}), \ \mathbf{x} \in \mathbb{R}^d. \tag{3.13}$$

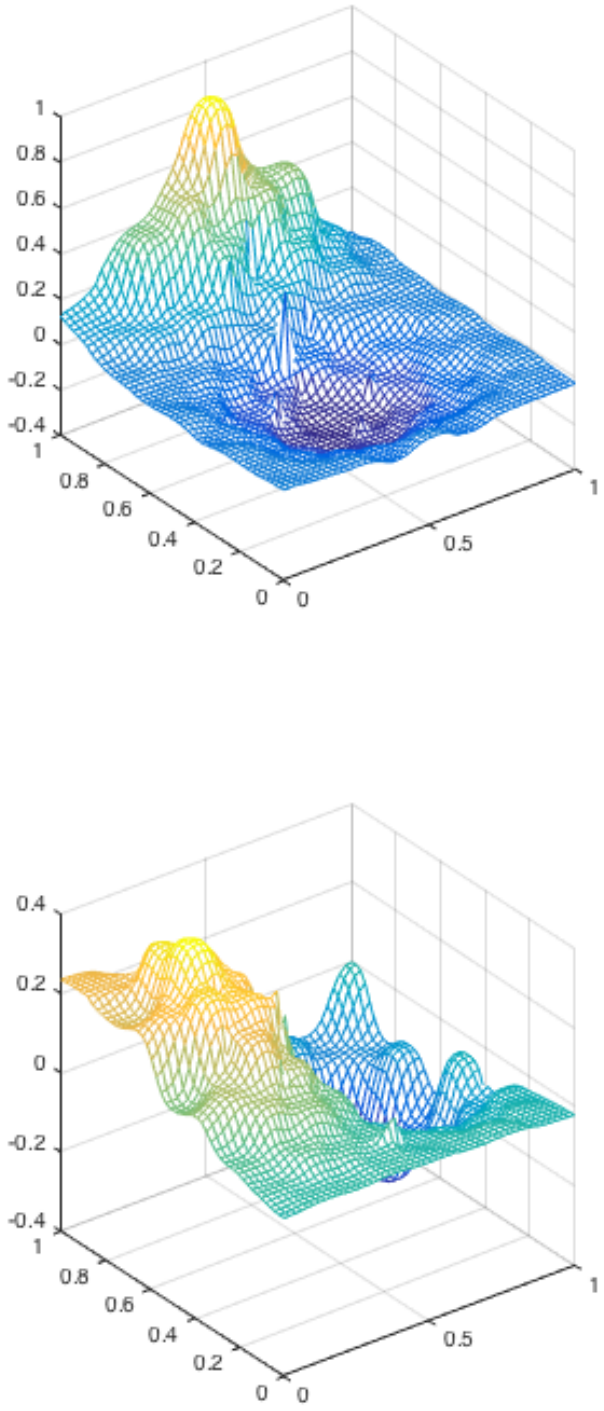

FIGURE 3.8: The optimized MSM interpolation of the driven cavity problem with 64 nodes, here $N_w = 8$. (a) shows velocity u, and (b) shows velocity v.

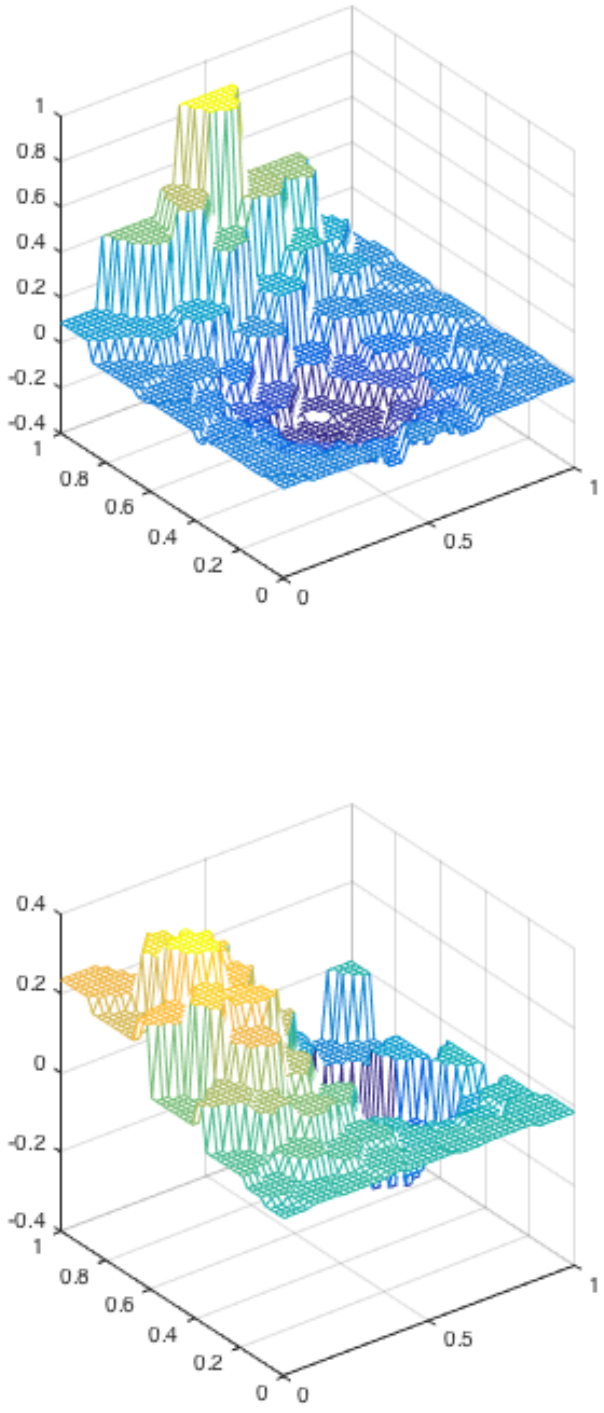

FIGURE 3.9: When using MSM with R_w varied so N_w = 1, the method reduces to nearest neighbor interpolation. (a) shows velocity u, and (b) shows velocity v.

With this approach, solving the interpolation problem is equivalent to solving the system of linear equations

$$A\mathbf{c} = \mathbf{y}, \tag{3.14}$$

where A is the interpolation matrix, $A_{jk} = B_k(\mathbf{x}_j)$, $1 \le j,k \le N$, $\mathbf{c} = [c_1, \dots c_N]^T$, and $\mathbf{y} = [y_1, \dots, y_N]^T$. This system has a unique solution exactly when A is non-singular.

The matrix A is defined by the value of the basis functions, B_k, at the points $\mathbf{x}_j$, $1 \le j,k \le N$. There does not exist a set of predetermined basis functions that will produce non-singular A for all possible data nodes $\mathbf{x}_j$, $1 \le j \le N$ [9, p. 4]. This problem is solved by defining the basis functions relative to the data locations.

A radial basis function is a basis function that is symmetric about its origin. For example, define $B_k(\mathbf{x}) = \|\mathbf{x}-\mathbf{x}_k\|_2$, for $x \in \mathbb{R}$, see Figure 3.10. Each basis function B_k is symmetric about its origin, $\mathbf{x}_k$. With this set of radial basis functions, Equation (3.14) becomes

$$\begin{pmatrix} \|\mathbf{x}_1 - \mathbf{x}_1\|_2 & \|\mathbf{x}_1 - \mathbf{x}_2\|_2 & \cdots & \|\mathbf{x}_1 - \mathbf{x}_N\|_2 \\ \|\mathbf{x}_2 - \mathbf{x}_1\|_2 & \|\mathbf{x}_2 - \mathbf{x}_2\|_2 & \cdots & \|\mathbf{x}_2 - \mathbf{x}_N\|_2 \\ \vdots & \vdots & & \vdots \\ \|\mathbf{x}_N - \mathbf{x}_1\|_2 & \|\mathbf{x}_N - \mathbf{x}_2\|_2 & \cdots & \|\mathbf{x}_N - \mathbf{x}_N\|_2 \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_N \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix}. \tag{3.15}$$

It is known that Euclidean distance matrices in $\mathbb{R}^d$ are non-singular [9, p. 77], so there exists a unique solution for $\mathbf{c}$.

It is important to note that the matrix in Equation (3.15) is dense. This density is not an artifact of the choice of B_k as basis functions, but is a general trait for radial basis functions. As the problem grows and N increases, the computational cost of solving Equation (3.14) increases significantly. Solving Equation (3.15) for large scale systems may require sophisticated techniques, such as radial basis functions with compact support (to lessen the density of the matrix), or taking advantage of the symmetric structure and the specific radial basis functions used.

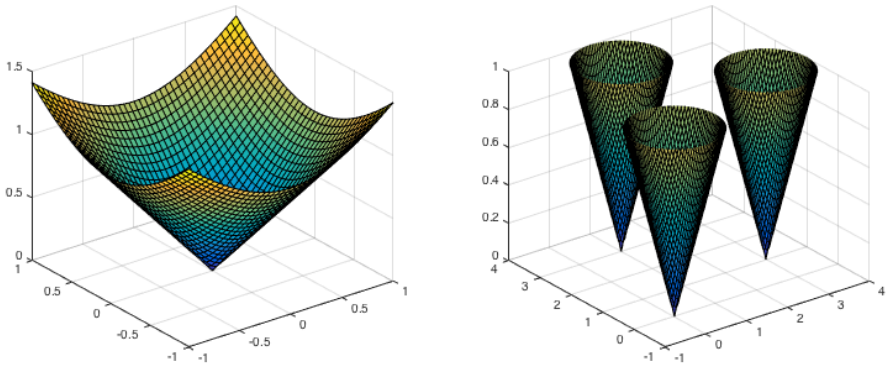

FIGURE 3.10: Left, the Radial Basis Function $B_k(\mathbf{x}) = |\mathbf{x} - \mathbf{x}_k|$, where $\mathbf{x}_k = 0$. Right, the same Radial Basis Function repeated with three different centers on a single domain.

3.3.2.1 Radial basis functions

While the Euclidean distance matrix, such as in Equation (3.15), is easy to use for finding interpolants, it suffers from limited accuracy and smoothness [9, p. 17]. By composing the distance function with various smooth univariate functions, we can take advantage of the distance matrix, while shoring up its weaknesses.

Definition 3.3.2. *A function $\Phi : \mathbb{R}^d \to \mathbb{R}$ is called **radial** provided there exists a **univariate** function $\phi : [0, \infty) \to \mathbb{R}$ such that*

$$\Phi(\boldsymbol{x}) = \phi(r), \qquad \text{where} \quad r = ||\boldsymbol{x}||,$$

and $\|\cdot\|$ is some norm on $\mathbb{R}^d$ – usually the Euclidean norm.

A familiar example of a radial function is the Gaussian, $\phi^\epsilon(r) = e^{-\epsilon r^2}$, $r \in \mathbb{R}$, where $\epsilon > 0$ in the superscript denotes a parameter. We can compose this function with the Euclidean distance function, $\|\cdot\|$, and center it about the point $\mathbf{x}_j$ to get the radially symmetric function

$$\Phi_k^\epsilon(\mathbf{x}) = \phi^\epsilon(||\mathbf{x} - \mathbf{x}_k||) = e^{-\epsilon^2 ||\mathbf{x} - \mathbf{x}_k||^2}, \ \mathbf{x} \in \mathbb{R}^d. \tag{3.16}$$

The parameter ϵ plays a strong role in how local the basis functions are. Small values of ϵ flatten out this radial function, while large values of ϵ cause a peak in the

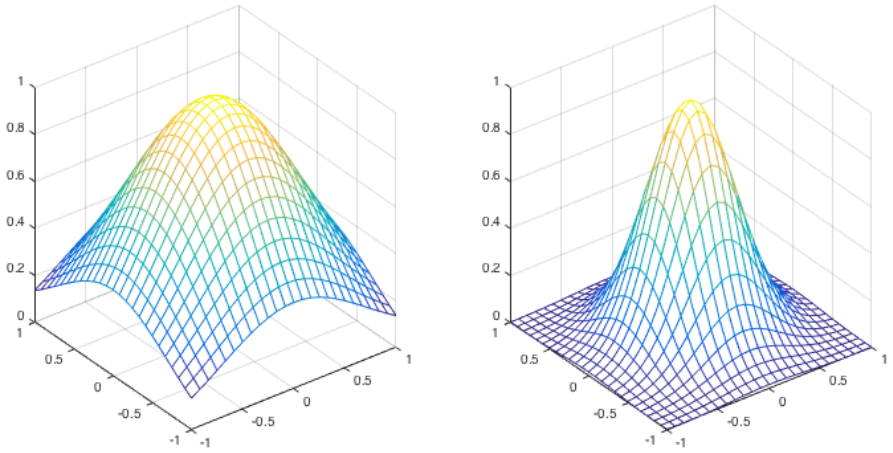

FIGURE 3.11: Examples of the Gaussian RBF given in Equation 3.16 with $\epsilon = 1$ (left) and $\epsilon = 2$ (right)

center. This parameter has a significant affect on both the accuracy and numerical stability of the interpolation problem [9, p. 18]. Figure 3.11 shows the radial function $\Phi^\epsilon(\mathbf{x}) = e^{-\epsilon||\mathbf{x}||^2}$, $\mathbf{x} \in [-1, 1]^2$.

With the Gaussian in mind, we can update our interpolation function in Equation (3.13), to get

$$p^\epsilon(\mathbf{x}) = \sum_{k=1}^{N} c_k \Phi_k^\epsilon(\mathbf{x}), \ \ \mathbf{x} \in \mathbb{R}^d,$$

and our interpolation matrix A in Equation (3.14) becomes

$$\begin{pmatrix} \phi^\epsilon(||\mathbf{x}_1 - \mathbf{x}_1||) & \phi^\epsilon(||\mathbf{x}_1 - \mathbf{x}_2||) & \cdots & \phi^\epsilon(||\mathbf{x}_1 - \mathbf{x}_N||) \\ \phi^\epsilon(||\mathbf{x}_2 - \mathbf{x}_1||) & \phi^\epsilon(||\mathbf{x}_2 - \mathbf{x}_2||) & \cdots & \phi^\epsilon(||\mathbf{x}_2 - \mathbf{x}_N||) \\ \vdots & \vdots & & \vdots \\ \phi^\epsilon(||\mathbf{x}_N - \mathbf{x}_1||) & \phi^\epsilon(||\mathbf{x}_N - \mathbf{x}_2||) & \cdots & \phi^\epsilon(||\mathbf{x}_N - \mathbf{x}_N||) \end{pmatrix}. \tag{3.17}$$

An important next step is to determine for which functions ϕ^ϵ the matrix A is invertible. As of 2007, no one had characterized the class of all basis functions that generate non-singular matrices for any set of distinct data sites [9, p. 27]. What is sufficient is strictly positive definite functions, as given by definition 3.2 in [9, p.28]

Definition 3.3.3. *A complex-valued continuous function* $\Phi : \mathbb{R}^d \to \mathbb{C}$ *is called* ***positive definite*** *on* $\mathbb{R}^d$ *if*

$$\sum_{j=1}^{N}\sum_{k=1}^{N} c_j\overline{c_k}\Phi(\boldsymbol{x}_j - \boldsymbol{x}_k) \geq 0$$

for any N *pairwise different points* $x_1, \ldots, x_N \in \mathbb{R}^d$, *and* $\boldsymbol{c} = [c_1, \ldots, c_N]^T \in \mathbb{C}^N$.

For details on how strictly positive definite functions correspond to positive definite matrices, see [9, Ch. 3].

The test problems in this paper are solved with the strictly positive definite radial function given in Equation (3.16), the Gaussian radial basis function. The radial basis function shown in Figure 3.10, $B_k(\mathbf{x}) = ||\mathbf{x} - \mathbf{x}_k||_2$ is not strictly positive definite.

3.3.2.2 Radial basis function interpolation of Franke's function

As an example, let us consider the Franke function, known as a hard test case for interpolation. Define

$$\begin{aligned} f(x_1, x_2) = \frac{3}{4}e^{-((9x_1-2)^2+(9x_2-2)^2)/4} + \frac{3}{4}e^{-((9x_1+1)^2/49-(9x_2+1)/10)} \\ +\frac{1}{2}e^{-((9x_1-7)^2+(9x_2-3)^2)/4} - \frac{1}{5}e^{-((9x_1-4)^2+(9x_2-7)^2)} \end{aligned} \tag{3.18}$$

on the domain $[0,1]^2$.

We use the Gaussian radial basis functions with $\epsilon = 3.8$, so

$$\Phi_k^{3.8}(\mathbf{x}) = e^{-3.8^2||\mathbf{x}-\mathbf{x}_k||^2},\ \mathbf{x} \in [0,1]^2,\ 1 \leq k \leq 50.$$

The data will be sampled at 50 Hammersley points, Δ. Let $p^{3.8}(x_1, x_2)$ be the radial basis function interpolation to $f(x_1, x_2)$ at Δ.

Figure 3.12 shows the exact function, $f(x_1, x_2)$ in the top left, the interpolated solution, $p^{3.8}(x_1, x_2)$ in the top right, the nodes in the bottom left and the value $|f - p^{3.8}|$ in the bottom right.

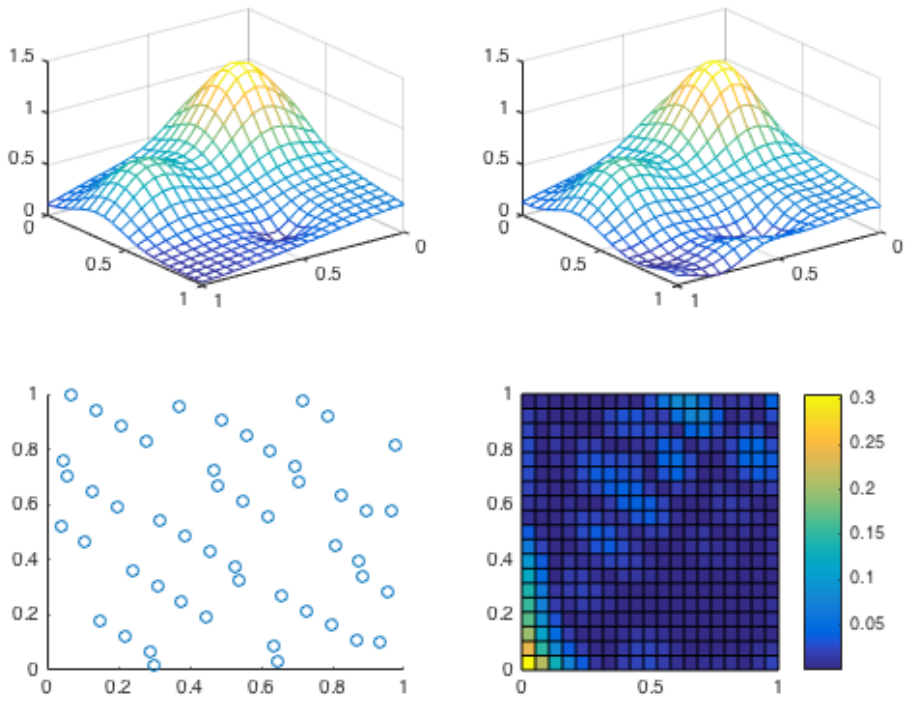

FIGURE 3.12: Gaussian radial basis function interpolation of the Franke function given in Equation 3.18. Top left: the exact function. Top right: The Gaussian RBF interpolation with parameter $\epsilon = 3.8$. Bottom left: the location of the nodes. Bottom right: the sup-norm error of the interpolation

4 Blowout and spill occurrence model

The Blowout and Spill Occurrence Model (BLOSOM) is a modeling suite developed by the Department of Energy's National Energy Technology Laboratory (NETL). It is a complete and stand-alone modeling package designed to simulate the trajectory and fate of oil spilled from the sea floor. BLOSOM uses several different components, each suited for a different aspect of the model. The Jet/Plume Model is used for the initial buoyant jet and plume, while the Spill/Transport and Weathering Models handle the longterm movement and degradation of the oil. The Conversion Model serves to transition between the Jet/Plume and Spill/Transport models. The Crude Oil and Gas/Hydrates Models provide characteristics of the crude oil and gases. The Ambient/Hydrodynamic Handler reads ocean data and provides some correlations and interpolations of its own [21].

4.1 Validation of the model

There have been a number of attempts to validate BLOSOM. Amongst these efforts was participation in a comparison study hosted by the American Petroleum Institute (API). The focus of the study was on plume dynamics and droplet sizes, and how they reacted to dispersants [22]. This work compared the oil spill model predictions from five models, given a prototype subsea blowout. The results provided by BLOSOM compared favorably with the results of the other four models used in the study.

Another way that BLOSOM was able to validate its models was against experiments held in the North Sea. In these experiments, oil was released from the sea floor and ambient conditions measured, while observing the fate of the oil. These experiments provide data to compare to model results in the area near the oil release. BLOSOM's simulation results match well with observations on the terminal level of

the jet and widths of the jet and plume. As oil leaves the blowout source, the high pressure causes the oil to behave as a momentum driven **jet**. As the oil advances it entrains water, slowing and expanding until it becomes a **plume**. As a plume, the oil's movement is driven by buoyancy and hydrodynamic conditions.

In [21, p. 5], a jet coefficient, j, was introduced in equation (1). This coefficient exists to modify the the jet as it reaches terminal level, making it behave more as a plume. Within the jet coefficient, there is a "fitting parameter," a, determined by matching results of simulations against the North Sea field experiments. Since this parameter was found empirically, it would be beneficial to simulate against other experiments to determine the sensitivity of a to the local conditions of the North Sea.

The most significant deviation between the model and observations in the North Sea field experiment is the horizontal displacement of the released oil. It was posited in [21, p. 17] that this could be due to coarse current data collection; perhaps the transport could be improved through stronger interpolation methods.

BLOSOM was also tested against the Deepwater Horizon Spill. The tests were not initiated at the blowout origin (spatial or temporal), but instead started on May 6, 2010, using satellite data to place observed oil and adding oil at the blowout location. Two simulations were run for 14 days, each on a different operational hydrodynamic model. Both simulations captured the existence of the distinct "tiger-tail" feature that existed in the real spill. The models varied in the smaller details of the oil movement, but did well to replicate the major features observed in event. The variance in the results of the two simulations showed that BLOSOM can have significant sensitivities to the hydrodynamic data, emphasizing the importance of BLOSOM's capability to use a large variety of hydrodynamic data sets, i.e. so BLOSOM can take an ensemble approach to modeling events [21, p. 18].

4.2 Risk assessment

BLOSOM is currently being used as a tool for assessing risks associated with future hydrocarbon harvesting. This use is being driven by a project of the Department of Interior's Bureau of Safety and Environmental Enforcement. BLOSOM's simulations are helping to evaluate the risks associated with spills in the offshore Arctic, the Salish Sea and offshore California [8]. This broad use, again, shows the importance of BLOSOM being able to use a variety of hydrodynamic data sets.

The National Energy Technology Laboratory's Office of Research and Development is developing a suite of tools that can be used together in these risk assessment efforts. The integrated assessment modeling (IAM) system is comprised of models like BLOSOM, tools such as cumulative spatial impact layers (CSIL), as well as data and approaches being collected and developed by NETL. An explanation of how all these parts can work together to better understand the risks of developing offshore drilling in the Arctic can be found in [18].

In [16], a measure of economic risk given certain oil spills is developed for the Gulf of Mexico. First, along the coast, local populations, assets and community vulnerability was assessed. Then, for a select set of locations and spill conditions, BLOSOM ran simulations. The spill extent as modeled by BLOSOM was cross-referenced with the local conditions, allowing an assessment of the impact a spill may have. This work helps to give context to the damages different spills may cause.

4.3 Hydrodynamics and transport

The discussion of meshfree methods for interpolation is motivated by their potential use in a model such as BLOSOM. In some cases, hydrodynamic data is available on rectangularly gridded data sets. One example of this is the Naval Oceanographic Office's ocean prediction system, based on the Navy Coastal Ocean Model. In this situation, BLOSOM uses bilinear interpolation, as discussed in Section 3.2.1.

In many cases, the data is not rectangular. For this, BLOSOM cannot use bilinear interpolation. Some examples of ocean models that are not restricted to rectangular data are the unstructured grid Finite-Volume, primitive equation Community Ocean Model (FVCOM) and HYbrid Coordinate Ocean Model (HYCOM).

An unstructured grid is a mesh covering using simple shapes, but with an irregular pattern. Unstructured grids come in many varieties; they can be composed of irregular triangles, quadrilaterals, pentagons, etc. in 2D, and have more variation in 3D. In 3D, the shapes may possible be extended either as prisms or (in the case of triangles) as tetrahedra. FVCOM in particular uses an unstructured grid. This motivated the developers of BLOSOM to use a flexible interpolation scheme that could work with all varieties of unstructured grids.

4.3.1 Transport in BLOSOM

A **parcel** is an oil-based object of interest, either a control-volume in the Jet/Plume Model or point-mass in the Spill/Transport Model. Each parcel is given a single spatial coordinate, (x, y, z), where x and y are distance from a static reference point and z is depth from the sea surface. For the context of this physical problem and the test cases to follow, we adjust our notation to (x, y, z) in lieu of the (x_1, x_2, x_3) notation we used in earlier chapters.

To calculate the movement of a parcel, a request for ambient conditions is sent to the Hydrodynamic Handler. The Jet/Plume Model and Spill/Transport Model have their own methods for determining evolution, but their data requests are the same. This request is for data at the position of the parcel, (x, y, z); the requested data depends on what is available in the hydrodynamic data set, but in general this data consists of velocity u, v, temperature, salinity, bathymetry, wind velocity, water surface elevation and more. It is this information at a single point that is used to calculate evolution, in both transport and weathering.

In the Jet/Plume Model, the parcel's evolutionary process is a complicated

mixture of momentum and buoyancy based flow, advection due to and entrainment of ambient fluid, and losses due to dissolution and gas separation. In the Spill/Transport Model, movement is calculated with a Lagrangian specification of the flow field [21, p. 10]. Then, advancing the position of individual parcels, such as oil slicks or subsurface droplet clouds, can be done with a method such as forward Euler or a Runge-Kutta method.

4.3.2 Data interpolation in BLOSOM

The approach to interpolation in BLOSOM is to interpolate linearly in the vertical direction (depth) and with either bilinear interpolation or inverse distance weighting at a fixed depth. Bilinear interpolation is used when the hydrodynamic grids are structured and inverse distance weighting is used when they are unstructured [21, p. 14]

One way that BLOSOM could interpolate data for a given parcel at (x, y) is with a MSM, as described in 3.3.1. Given the large size of hydrodynamic data sets, one could restrict this method to the local nodes, as in [17]. For an unstructured grid, connectivity of the nodes is known. Thus, if one finds the nearest node there would be a set of nearby nodes or elements which contribute to the interpolation. BLOSOM uses a quad-tree algorithm for sorting through nodes and elements in unstructured grids [21, p. 14].

For an unstructured mesh, a **node** is a vetex of the shape used in tessellating the domain. An **element** is the face of each shape. Since unstructured grids have no definite shape, it would be cumbersome to develop interpolation methods that treat each shape as a unique case. Therefore, on unstructured meshes it may be ideal to use meshfree methods. Meshfree methods could also be advantageous in three dimensional velocity fields, as another layer of complexity is introduced by hydrodynamic models variability in using prisms or shapes such as tetrahedra for three-dimensional meshing.

One major advantage of using unstructured grids is that the elements can have

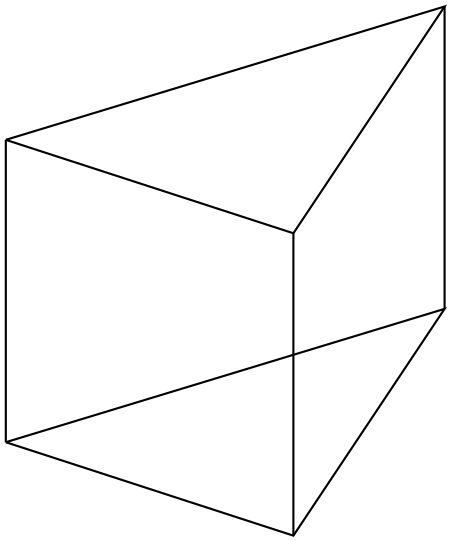

FIGURE 4.1: An example of an element used in the FVCOM models.

varying sizes, based on where higher resolution is needed. For example, varying element sizes allows the mesh to more accurately follow the coastline or continue into estuaries. This suggests that using the MSM in [17], where the radius of influence changes to include a static number of nodes may be beneficial.

We overview how one might implement the MSM with hydrodynamic data provided by an FVCOM model. We consult the FVCOM manual, [4], for the specifics on how the data is solved and arranged. The mesh used by FVCOM is triangular prisms, shown in Figure 4.1, a form of unstructured gridding. The data that the FVCOM models calculate is given either at the nodes or at the center of mass of the elements. If the data is a scalar value, like temperature or salinity, then the data is given on a node. If the data is vector valued, like velocity, then the data is given at the element center.

Because BLOSOM is a 3 dimensional simulator, the hydrodynamic data used is often in three spatial dimensions. In the data sets that BLOSOM uses, the data can be thought of as columns. Let $\hat{\Delta}$ be a two dimensional set of nodes with locations (x_i, y_i), $1 \leq i \leq M$. Then the full set of nodes, Δ, have locations (x_i, y_i, z_j), $1 \leq j \leq D$, $1 \leq i \leq M$, where D is the number of layers in the vertical direction.

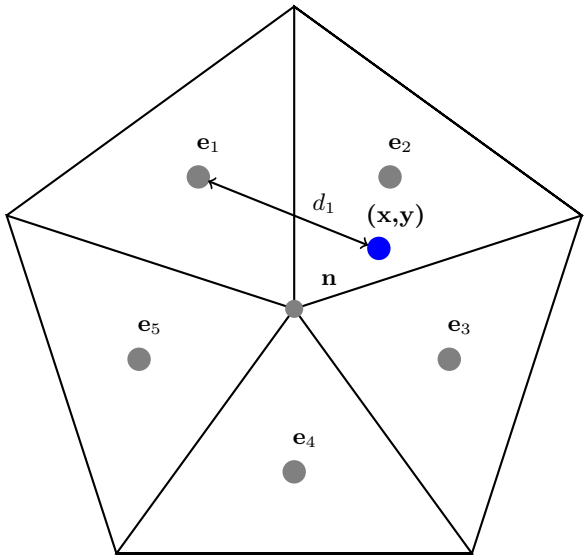

FIGURE 4.2: A point (x, y), with nearest node, n, and five adjacent triangular elements.

Suppose that we want to interpolate at the point (x, y, z). Since the interpolations over depth are linear, we need the interpolated values at (x, y, z_j), (x, y, z_{j+1}), where $z_j < z < z_{j+1}$.

Since velocity is vector valued, the data is given at the element centers. For a given position (x, y) we cannot know what element we are in, so we use the search tree to find the nearest node, n. Then, define N as the number of elements that use node n as a vertex. For FVCOM grids, $N \leq 8$. Let these elements be those within the radius on influence. See Figure 4.2 for an example with $N = 5$.

Next we need to calculate the weights for each element. Let e_i, $1 \leq i \leq N$ be the center of mass for each element. Let $r_i = d(e_i, (x, y))$ be the Euclidean distance from e_i to (x, y). Define $R = \max_i\{r_i\}$. Then the weights are

$$w_i = \left(\frac{R - r_i}{R r_i}\right)^2, \quad 1 \leq i \leq N.$$

Let $(u_{i,j}, v_{i,j})$ be the velocities at e_i corresponding to depth z_j. Let $(\hat{u}_j(x, y), \hat{v}_j(x, y))$ be the interpolated velocities at the given point, (x, y, z_j). We calculate $(\hat{u}_j(x, y), \hat{v}_j(x, y))$

by

$$\hat{u}_j = \frac{\sum_i^N w_i u_{i,j}}{\sum_i^N w_i}, \qquad \hat{v}_j = \frac{\sum_i^N w_i v_{i,j}}{\sum_i^N w_i}.$$

After $(\hat{u}_j, \hat{v}_j)$ are found for all j, these values are linearly interpolated between z_j and z_{j+1} to find the interpolated values $\hat{u}(x, y, z)$ and $\hat{v}(x, y, z)$. These are

$$\hat{u}(x, y, z) = \frac{z_{j+1} - z}{z_{j+1} - z_j}\hat{u}_{j+1} + \frac{z - z_j}{z_{j+1} - z_j}\hat{u}_j \qquad \hat{v}(x, y, z) = \frac{z_{j+1} - z}{z_{j+1} - z_j}\hat{v}_{j+1} + \frac{z - z_j}{z_{j+1} - z_j}\hat{v}_j.$$

5 Results and Examples

In this chapter, we use three test problems to evaluate the MSM and radial basis function interpolation with Gaussian basis functions. The test problems we use are interpolation of Hagen-Poiseuille flow, uniform flow and the lid-driven cavity problem. We evaluate the methods through measuring error in a manner similar to the so called grid norm and through Lagrangian transport outcomes.

To calculate the streamlines, we use a four stage, fourth order Runge-Kutta method, as described in Section 2.1.

5.1 Interpolation of the Hagen-Poiseuille flow

The goal of the this test problem is to evaluate how the methods would do interpolating flow in a channel. This test problem mimics a hypothetical flow in a channel when data is being collected from buoys. The 6 nodes chosen ad-hoc for interpolation are shown in Figure 5.1.

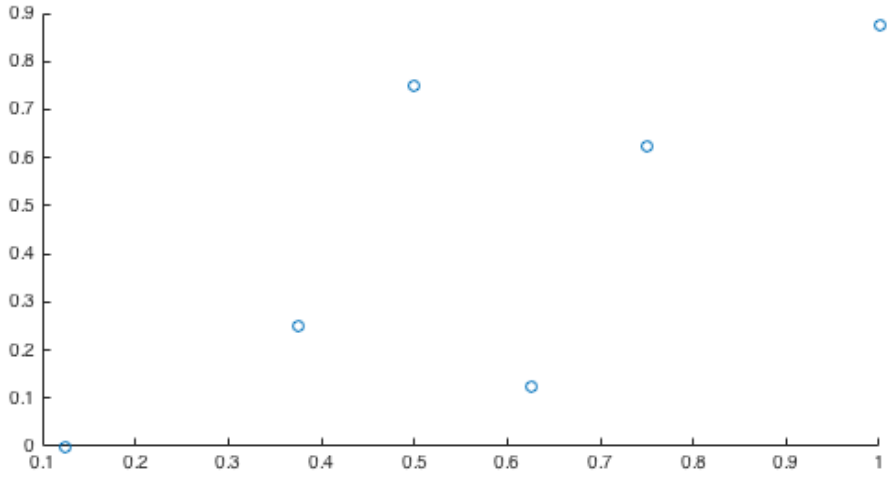

FIGURE 5.1: The interpolation nodes used in the Hagen-Poiseuille test flow.

For this test problem, we use initial positions of $(0.25, i\Delta y)$, $0 \leq i \leq 24$ with $\Delta y = 1/24$. Because $v(x, y) = 0$, the Lagrangian transport will be exclusively in the x direction. Rather than observing the path followed from each initial position, we show the final position after a fixed time, $T = 1$.

When solving for transport with analytic velocities, for a fixed height y the velocity is constant, as given by Equation (2.16). So, the position, $x(t, y)$, can be calculated simply by $x(1, y) = u(y) + x(0) = u(y) + .25$.

To approximate the final position using the interpolation methods, we use a Runge-Kutta method, as described in Section 2.2.2. For the MSM interpolation we use all 6 nodes to interpolate the velocity. For the Gaussian radial basis function interpolation we use $\epsilon = 3$ unless otherwise noted.

Figure 5.2 shows the flow transport after time $T = 1$. We see the expected parabolic profile in the solution found using analytic velocities. Both the MSM and the radial basis function methods fail to maintain the no-slip boundary conditions. By Equation (2.16), the velocity on the boundaries is 0, so we expect there to be no transport along the boundary. The two results are distinct, however, in the shape of their profile. The radial basis function maintains the parabolic shape, where the MSM does not, with significant transport along the top boundary.

Next we consider other parameters, ϵ. In Figure 5.3 we show the Gaussian radial basis function with parameter $\epsilon = .25$. We test the radial basis function interpolation, measuring the RMS error as described in Section 3.3.1.1 over different values of the parameter $\epsilon \in [0.25, 15]$. The RMS values as a function of ϵ were roughly decreasing as $\epsilon \searrow 0$, so it's possible that $\epsilon < 0.25$ may produce smaller RMS errors than $\epsilon = 0.25$.

We see good agreement between the profile shapes of transport. One significant feature in the transport solution with radial basis function interpolated velocities is that on the top boundary, $y = 1$, we see transport in the backwards direction. This is an obvious concern, as flow was unidirectional.

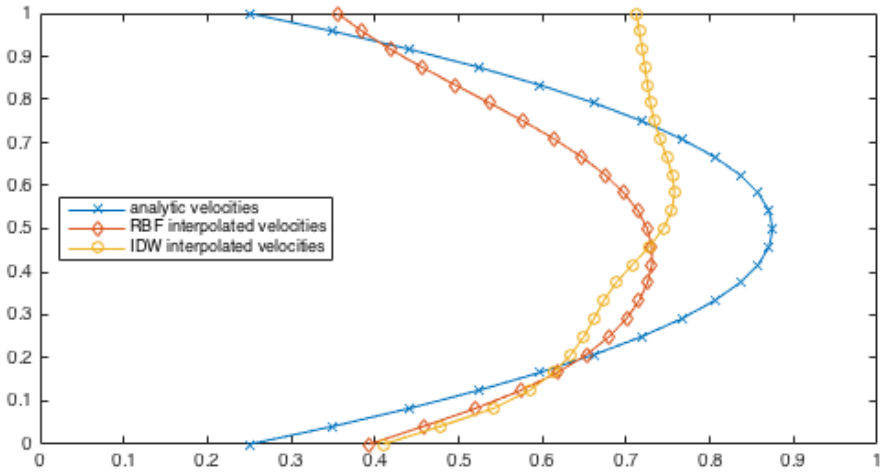

FIGURE 5.2: Transport results on the Hagen-Poiseuille flow, using analytic velocities, MSM interpolated velocities and Gaussian radial basis function interpolated velocities.

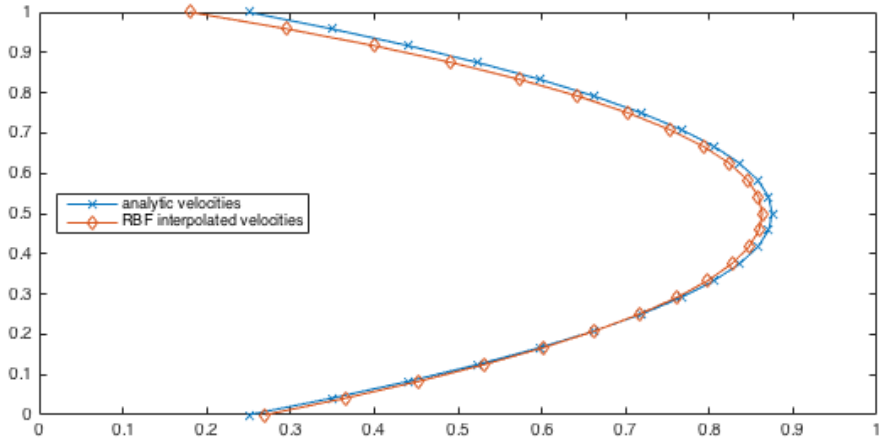

FIGURE 5.3: Gaussian radial basis function interpolation of the Hagen-Poiseuille flow with parameter $\epsilon = .25$

5.2 Interpolation of uniform, steady flow in a channel

In this section we consider uniform flow through a region, $u(x, y) = c \in \mathbb{R}$, $v(x, y) = 0$, for $(x, y) \in [0, 1]^2$. This situation could arise when the flow is not steady state or subject to no-slip boundary conditions, as assumed for the Hagen-Poiseuille flow. The characteristic curves, from Equation (2.14), are $x(t) = ct + x(0)$, and $y = y_0 \in [0, 1]$. Figure 5.4 shows the results of the transport simulations.

For the MSM, the interpolated velocities at time step t_n, as derived from equation (3.9), are

$$\hat{u}(y^n) = \frac{\sum_{i=1}^{N_w} W_i(y^n) Q_i(y^n)}{\sum_{i=1}^{N_w} W_i(y^n)} = c \frac{\sum_{i=1}^{N_w} W_i(y^n)}{\sum_{i=1}^{N_w} W_i(y^n)} = c.$$

Thus, $\hat{u} = u$ and the transport results are exact.

However, the transport calculated with $(\hat{u}, \hat{v})$ given by the Gaussian radial basis function interpolation, $\epsilon = 3$, does not match the expected transport as seen in Figure 5.4. For some values of y, the flow exceeds expectations, at others it lags by over half the expected transport.

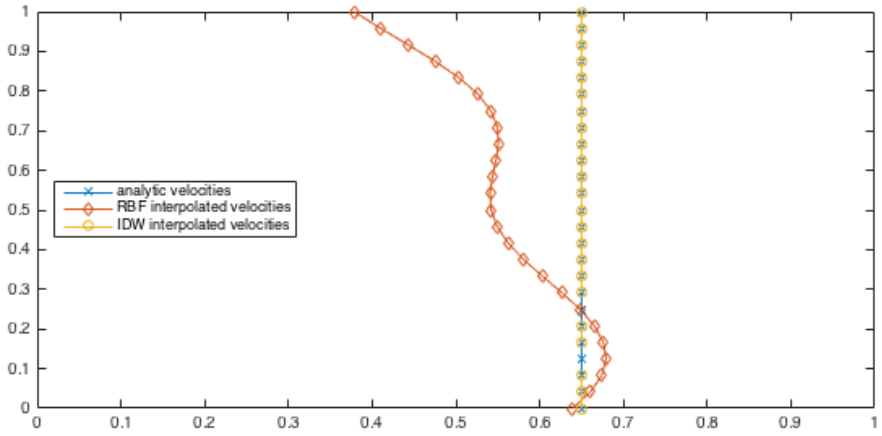

FIGURE 5.4: Transport results on uniform flow in a channel, using analytic velocities, MSM interpolated velocities and Gaussian radial basis function interpolated velocities, $\epsilon = 3$.

We optimize again for parameter ϵ. We found that, again, $\epsilon = 0.25$ minimizes

the RMS error. Then, as shown in Figure 5.5, we get excellent results, where the transport using radial basis function interpolated velocities approximates the exact solution well.

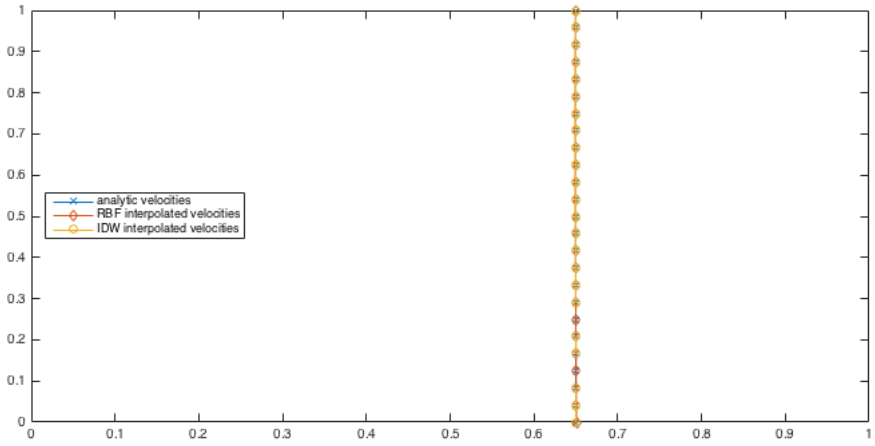

FIGURE 5.5: Transport results on uniform flow in a channel, using analytic velocities, MSM interpolated velocities and Gaussian radial basis function interpolated velocities, $\epsilon = 0.25$.

5.3 Interpolation of the lid-driven cavity solution

In this section, we will look more in depth at the lid-driven cavity problem as defined in Section 3.2.1.2. We use this as our test problem with two-dimensional flow.

For the interpolation nodes, we will use Hammersley nodes, Δ, discussed in Section 2.5, generated using the `Hammersley` library for MATLAB, provided by [13]. We generated sets of $N = 64, 128, 256, 512$, and 1024 nodes for use in the tests. The set 64 of interpolating nodes is shown in Figure 3.6.

The goal in choosing this low discrepancy set, Δ, is to simulate how hydrodynamic data might be handled by BLOSOM. In that situation, the data is provided on a regular, triangular or hexahedral grid. But the shapes need not be congruent or similar, so there is a variation to how the data is spatially laid out. We also

want to consider the situation where the hydrodynamic data is provided by direct measurement, i.e. by readings from buoys.

We want to compare the results of interpolating the solution to the lid-driven cavity problem with both inverse distance weighting and radial basis functions. We use the total root-mean-square error as described in Equation (3.12) and through observing and measuring streamlines.

The inverse distance weighted interpolation is optimized for minimizing total error by varying N_w, the number of nodes in the radius of influence. This optimum value was found to be $N_w = 8$, when using 64 nodes. For the radial basis function interpolation, the interpolation was optimized for parameter ϵ, using a step size of ϵ of 1/4. We found the optimal $\epsilon = 1$ for 64 nodes.

The radial basis function we use for the interpolations is given in Equation (3.16) and repeated here:

$$\Phi_k(\mathbf{x}) = \phi(||\mathbf{x} - \mathbf{x}_k||) = e^{-\epsilon^2||\mathbf{x}-\mathbf{x}_k||^2}, \ \mathbf{x} \in \mathbb{R}^2 . \tag{5.1}$$

Within the class of Gaussian RBF functions used in this interpolation, we can expect that as the number of nodes used increases, the optimal value of ϵ should also increase. Higher values of ϵ lead to more significantly peaked basis functions, as seen in Figure 3.11. Since 64 nodes are sufficiently sparse, a low value for ϵ produces good results.

The errors for different values of ϵ and for different numbers of nodes are shown in Figure 5.6. We can see that as the nodes become denser, there is much inconsistency in what ϵ would be optimal, i.e. the curve is not smooth with a clear minimum. This is due to the fact that we need to solve a problem with a dense matrix.

First, we will compare contour plots of the flow. We present in Figure 5.7, the contours of the exact velocities, the MSM of interpolation with the optimized value $N_w = 8$, as found in Section 3.3.1.1, the Gaussian radial basis function interpolation interpolation with an arbitrary parameter $\epsilon = 7$, and from the optimized Gaussian

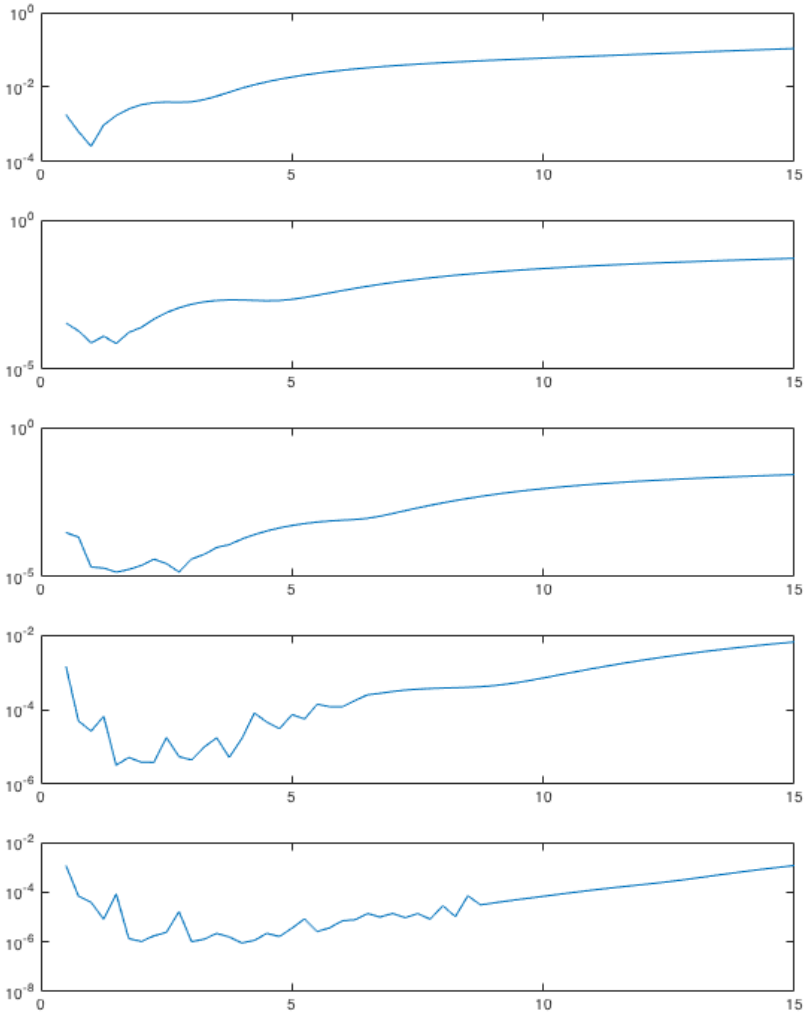

FIGURE 5.6: Total errors for the Gaussian basis function interpolation. ϵ varies over x-axis. From top to bottom, $N = 64, 128, 256, 512, 1024$.

radial basis function interpolation, where $\epsilon = 1$. We can see that the MSM in this case is lacking, compared to the radial basis function interpolated solutions.

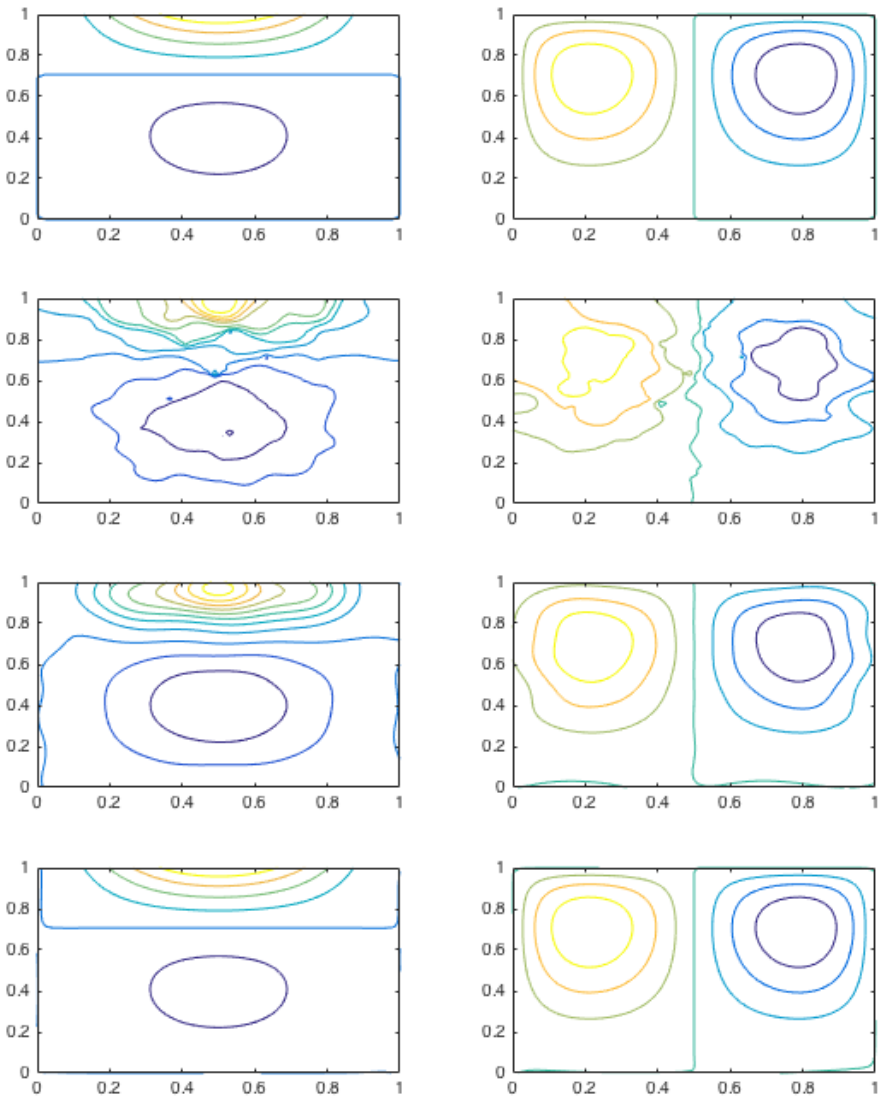

FIGURE 5.7: Contour plots of the lid-driven cavity flow and interpolations of the flow. Left: u, right: v. From top to bottom: Analytic velocities, MSM interpolation, Gaussian radial basis function interpolation with $\epsilon = 7$, and Gaussian radial basis function interpolation with $\epsilon = 1$.

In Figure 5.8 we see the result of following a streamline along a non-optimized radial basis interpolation. When we set $\epsilon = 7$ there is a clear spiral outward from the center of the swirling flow.

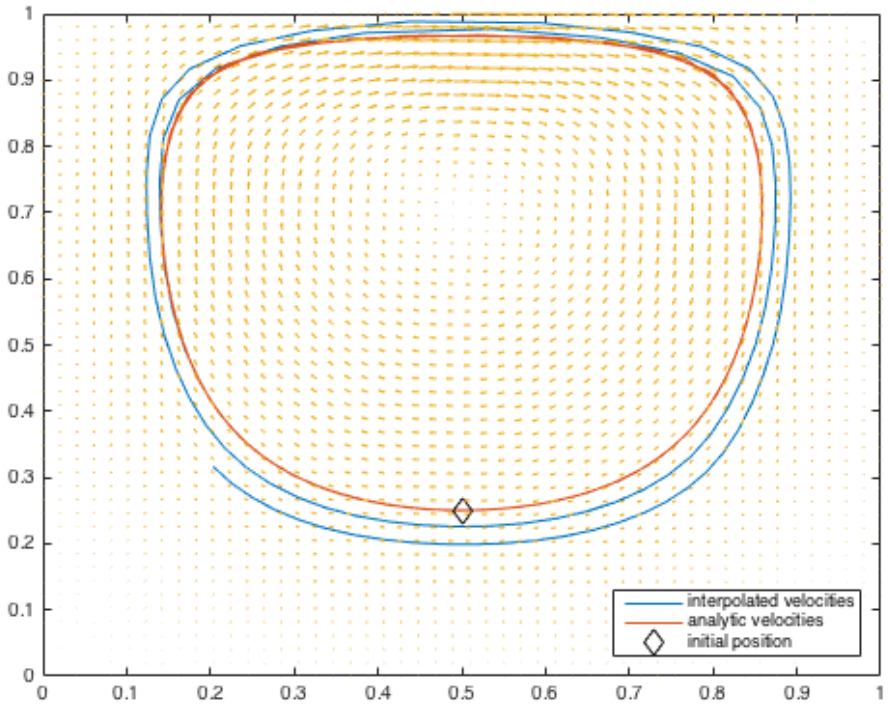

FIGURE 5.8: Radial basis function interpolation of the lid-driven cavity solution with $\epsilon = 7$. Streamlines calculated with the interpolated velocity show an outward spiral.

In Figure 5.9, we have calculated the streamline for the optimized radial basis interpolation, with $\epsilon = 1$. There is excellent agreement between the streamlines generated by the interpolated velocities and the exact velocities.

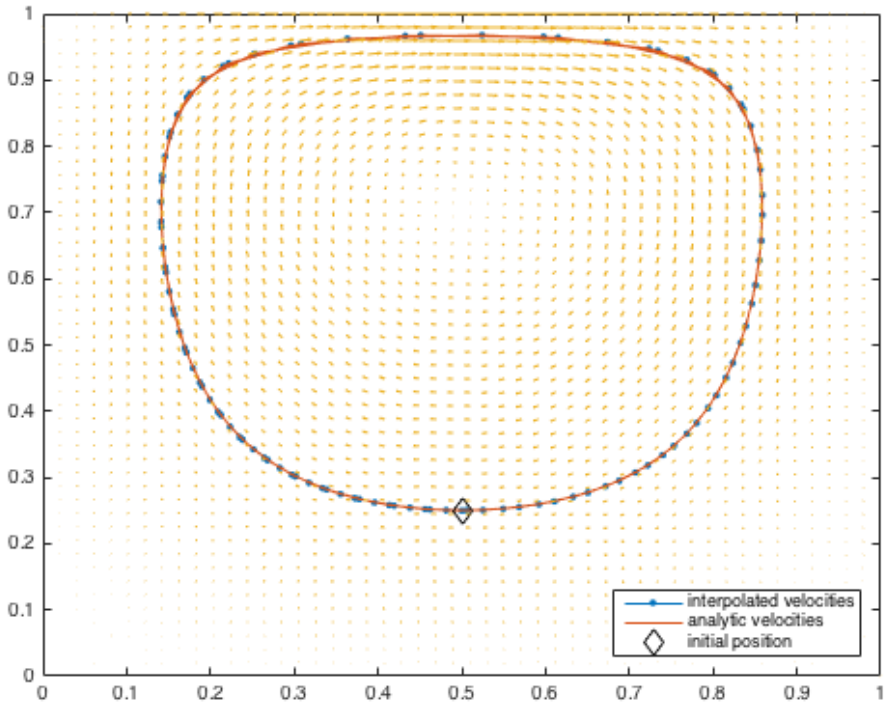

FIGURE 5.9: Radial basis function interpolation of the lid-driven cavity solution with $\epsilon = 1$.

In Figure 5.10, we have calculated the streamline for the MSM interpolation, where $N_w = 8$ was found to minimize the RMS error. We can see inconsistency in the velocity field underlying the streamlines. The streamline also swirls away from the center, even leaving the top of the domain.

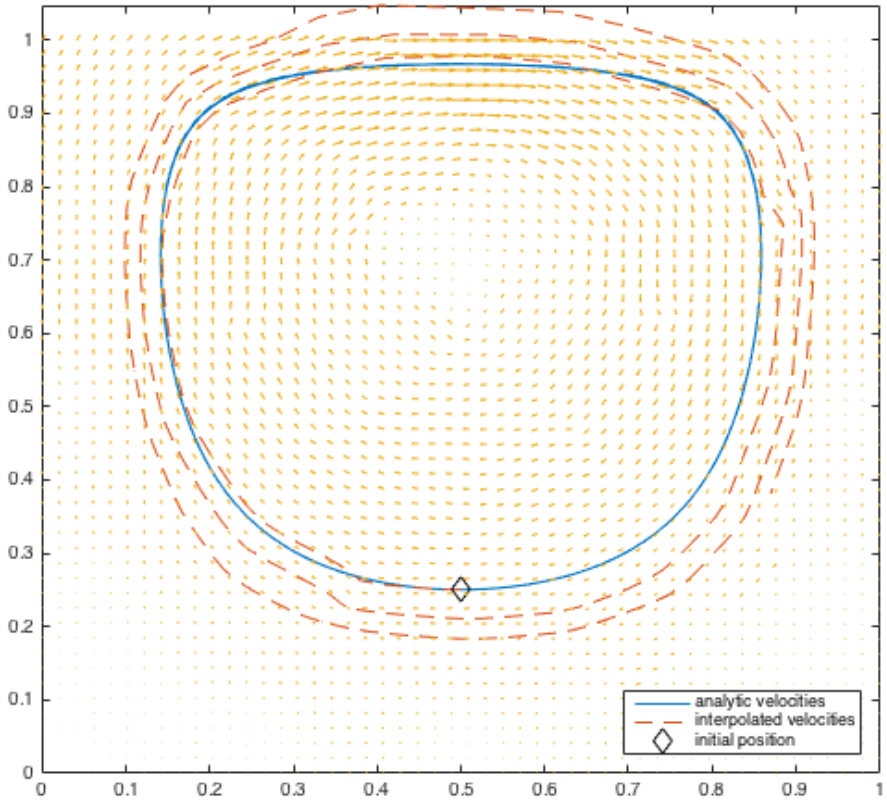

FIGURE 5.10: MSM interpolation of the lid-driven cavity solution with $N_w = 8$.

5.4 Divergence of the radial basis function interpolation

In this section, we investigate whether the Gaussian radial basis function interpolation of the previous three problems, Hagen-Poiseuille flow, uniform flow and the

lid-driven cavity problem, is conservative.

Let the Gaussian radial basis function interpolation of u and v be P_u and P_v, respectively, with parameter $\epsilon > 0$. Assume we have a set with $|\Delta| = N$. Then

$$P_u = \sum_{k=1}^{N} c_k e^{-\epsilon^2((x-x_k)^2+(y-y_k)^2)},$$
$$P_v = \sum_{k=1}^{N} d_k e^{-\epsilon^2((x-x_k)^2+(y-y_k)^2)},$$

for some $c_k, d_k \in \mathbb{R}$, $1 \le k \le N$. But then,

$$\nabla \cdot (P_u, P_v) = \sum_{k=1}^{N} -2\epsilon^2(c_k(x-x_k) + d_k(y-y_k))e^{-\epsilon^2((x-x_k)^2+(y-y_k)^2)}. \tag{5.2}$$

For each example, we calculate the divergence of the interpolation.

5.4.1 Divergence of radial basis function interpolation of the Hagen-Poiseuille flow

Let $\mathbf{u}(x,y) = (u(x,y), v(x,y))$. We first show that the velocity is divergence free.

$$\nabla \cdot \mathbf{u} = \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) = \frac{\partial}{\partial x}u(x,y) = \frac{\partial}{\partial x}\frac{5}{2}\left(\frac{1}{4} - (y-0.5)^2\right) = 0. \tag{5.3}$$

We show the divergence in the case where $\epsilon = 3$ in Figure 5.11. We use $N = 6$ nodes and parameter $\epsilon = 3$. The maximum of the divergence here is significant, reaching a value of about 1.36, relative to $\|(u,v)\|_\infty = 5/8$.

5.4.2 Divergence of radial basis function interpolation of uniform, steady flow

It is obvious that the divergence of uniform flow is 0. The divergence of the Gaussian radial basis function interpolation of the flow is shown in Figure 5.12. We use $N = 6$ nodes and parameter $\epsilon = 3$. Here, the maximum of the divergence is about 0.95, relative to $\|(u,v)\|_\infty = 0.4$.

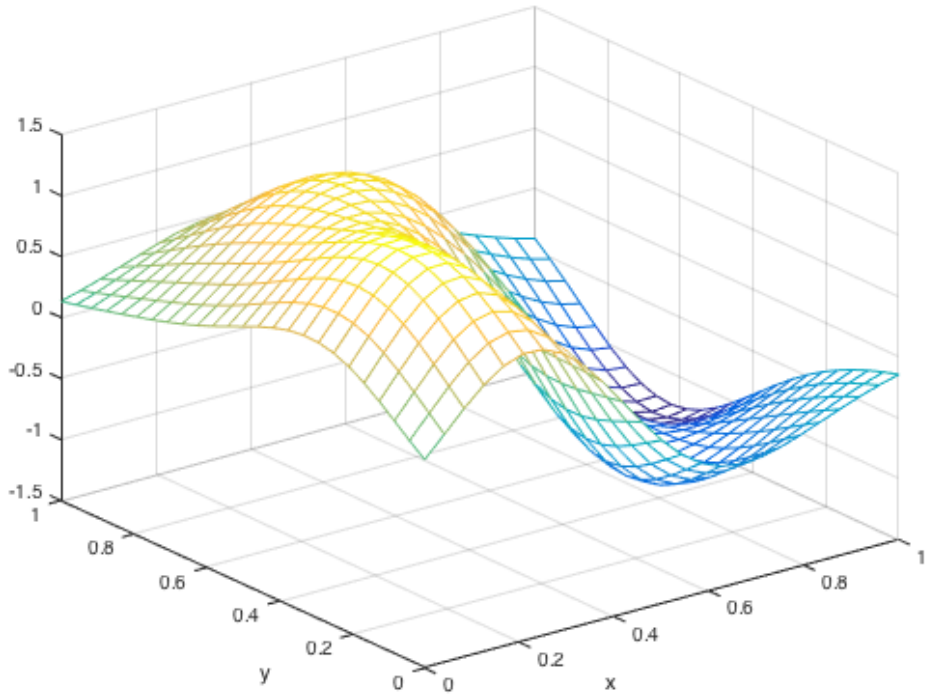

FIGURE 5.11: Divergence of the Gaussian radial basis function interpolation of the Hagen-Poiseuille flow with parameter $\epsilon = 3$. Discussed in Section 5.4.1

5.4.3 Divergence of radial basis function interpolation of the lid-driven cavity flow

The lid-driven cavity flow problem was explicitly chosen so that we could evaluate whether the Gaussian radial basis function interpolation was conservative in two dimensions, in the situation where the flow is conservative. We know that the test problem is divergence free by construction.

In the case where $\epsilon = 1$, we can see in Figure 5.13, that the divergence is non-zero. Here, we're interpolating with $N = 64$ nodes.

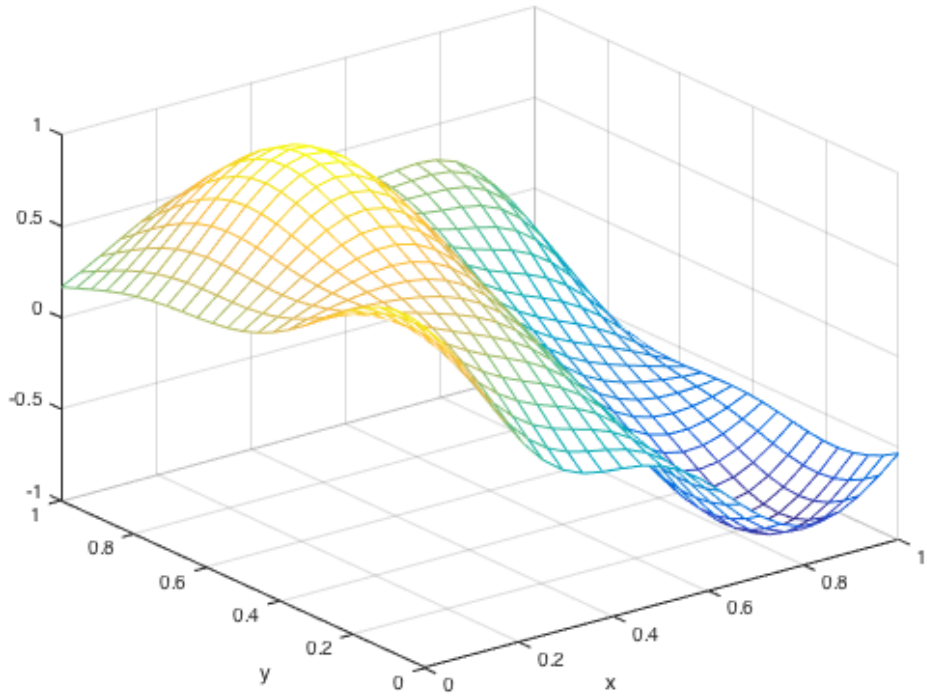

FIGURE 5.12: Divergence of the Gaussian radial basis function interpolation of uniform flow with parameter $\epsilon = 3$. Discussed in Section 5.4.2

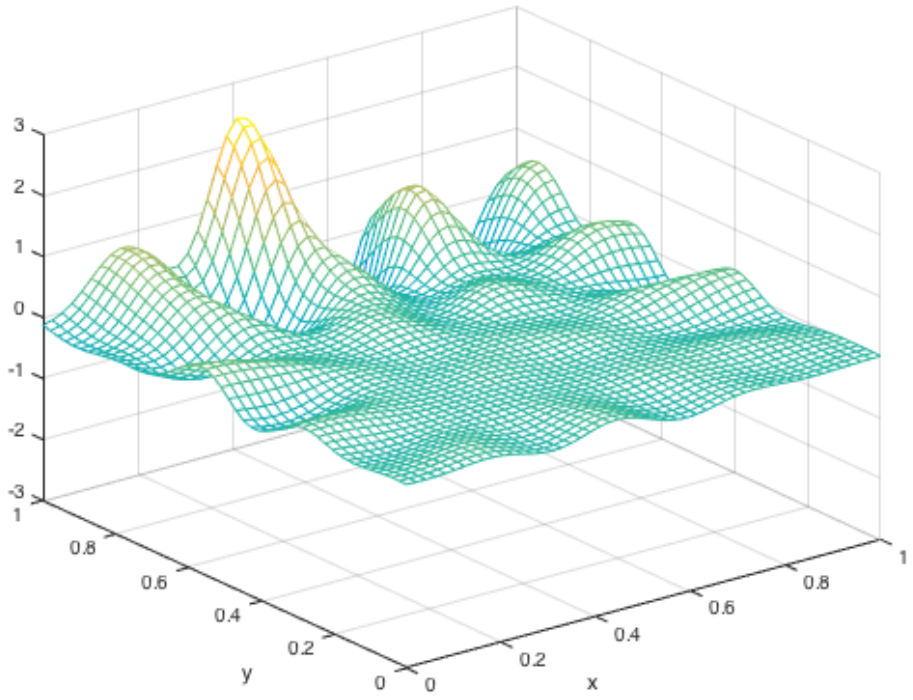

FIGURE 5.13: Divergence of the Gaussian radial basis function interpolation of the lid-driven cavity problem with parameter $\epsilon = 1$.

6 Conclusions

In this thesis we introduced several interpolation schemes, interpolation with polynomials, inverse distance weighting and with Radial Basis Functions. We tested these methods against known test problems, the Hagen-Poiseuille flow, uniform flow and the lid-driven cavity problem. The main motivation was to explore the accuracy and efficiency of these interpolation methods in the context of their possible use in simulators such as BLOSOM.

The results presented show that the Gaussian radial basis function approach presents an improvement over a MSM in accuracy for general flows. However, In the case of constant flow, the MSM calculated the exact solution where the radial basis functions did not. A possible solution is provided in [9, Ch. 6], which focuses on achieving polynomial precision while solving scattered data problems.

The accurate performance of radial basis interpolation comes at a cost in computation time. Solving for the interpolation matrix requires solving for a dense matrix. At the scale of oil spill modeling, $O(100,000)$ nodes, this may be intractable.

The scale of oil spill modeling justifies the use of Lagrangian methods for spill transport. Because the location of the spilled oil may be restricted to a small subset of a given hydrodynamic model, Lagrangian methods allow for focused computational efforts. However, accurate Lagrangian methods require conservative velocity fields or require corrections.

The radial basis interpolation was shown to be non-conservative. This presents possible complications in the form a non-physical solution.

One possible refinement of the radial basis function interpolation could be to create a local method, i.e. use only nearby data to create the smooth interpolated field from radial basis functions with compact support. This could then lend itself to a hybrid method. Using a MSM to predict roughly where the oil may go, then using a targeted radial basis function interpolation of the relevant local conditions.

For the tests in this thesis, the author used MATLAB to code and run the simulations. The author found the challenge of implementing the MSM and the radial basis function interpolations similar. No recommendation can be made for ease of coding at this point.

Further work that could be done in this area, is to evaluate radial basis functions against a 3 dimensional test problem or in a time dependent situation. Investigation will also be needed, of course, to determine how relevant the test problem would be to ocean hydrodynamics. Amongst other behavior, the oceans stratify, perhaps causing errors in the interpolation due to the symmetry in the radial basis functions.

Bibliography

1. Alexandra Adams. Summary of information concerning the ecological and economic impacts of the bp deepwater horizon oil spill disaster. http://www.nrdc.org/energy/gulfspill/files/gulfspill-impacts-summary-IP.pdf, 2015. Accessed: 2016-02-28.

2. Myron Allen III and Eli Isaacson. *Numerical analysis for applied science*. Wiley, New York, 1998.

3. Kendall Atkinson and Weimin Han. *Theoretical Numerical Analysis, A Functional Analysis Framework*. Springer, New York, 2009.

4. Changsheng Chen, Robert C. Beardsley, Geoffrey Cowles, Jianhua Qi, Zhiqang Lai, Guoping Gao, David Stuebe, Qichun Xu, Pengfei Xue, Jianzhong Ge, Rubao Ji, Song Hu, Rucheng Tian, Haosheng Huang, Lunyu Wu, and Huichan Lin. *An Unstructured Grid, Finite-Volume Coastal Ocean Model FVCOM User Manual*. School for Marine Sciences and Technology, University of Massachusetts-Dartmouth, 2011.

5. S. Chippada, C. N. Dawson, M. L. Martínez, and M. F. Wheeler. A projection method for constructing a mass conservative velocity field. *Comput. Methods Appl. Mech. Engrg.*, 157(1-2):1–10, 1998.

6. Clint Dawson. Conservative, shock-capturing transport methods with nonconservative velocity approximations. *Comput. Geosci.*, 3(3-4):205–227 (2000), 1999.

7. Michael Drmota and Robert F Tichy. *Sequences, discrepancies and applications*. Springer, 1997.

8. Elly Earls. Blosom: 'what if' technology rapidly simulates offshore oil spills. `http://www.offshore-technology.com/features/featureblosom-what-if-technology-rapidly-simulates-offshore-oil-spills-4583925/`, 2015. Accessed: 2016-02-12.

9. Gregory E. Fasshauer. *Meshfree approximation methods with MATLAB*, volume 6 of *Interdisciplinary Mathematical Sciences*. World Scientific Publishing Co. Pte. Ltd., Hackensack, NJ, 2007. With 1 CD-ROM (Windows, Macintosh and UNIX).

10. Richard Franke and Greg Nielson. Smooth interpolation of large sets of scattered data. *International Journal for Numerical Methods in Engineering*, 15(11):1691–1704, 1980.

11. R.B. Guenther and J.W. Lee. *Partial Differential Equations of Mathematical Physics and Integral Equations*. Dover books on mathematics. Dover Publications, 1996.

12. J.M. Hammersley and D.C. Handscomb. *Monte Carlo Methods*. Springer Netherlands, 1964.

13. John Burkardt. Hammersley. `https://people.sc.fsu.edu/~jburkardt/m_src/hammersley/hammersley.html`. 2004.

14. Pijush K. Kundu, Ira M. Cohen, and David R. Dowling. *Fluid Mechanics (Fifth Edition)*. Academic Press, Waltham, MA, 2012.

15. R. LeVeque. *Finite Difference Methods for Ordinary and Partial Differential Equations*. Society for Industrial and Applied Mathematics, 2007.

16. J.R. Nelson, T.H. Grubesic, L. Sim, K. Rose, and J. Graham. Approach for assessing coastal vulnerability to oil spills for prevention and readiness using {GIS} and the blowout and spill occurrence model. *Ocean & Coastal Management*, 112:1 – 11, 2015.

17. Robert J. Renka. Multivariate interpolation of large sets of scattered data. *ACM Transactions on Mathematical Software*, 14:139–148, 1988.

18. L. Romeo, K. Rose, J. R. Bauer, C. Disenhof, L. Sim, J. Nelson, C. Thimmisetty, M. Mark-Moser, and A. Barkhurst. Adapting the national energy technology laboratory's offshore integrated assessment modeling approach for the offshore arctic. Technical report, U.S. Department of Energy, National Energy Technology Laboratory, Morgantown, WV, 2015. NETL-TRS-3-2015.

19. Donald Shepard. A two-dimensional interpolation function for irregularly-spaced data. In *Proceedings of the 1968 23rd ACM National Conference*, ACM '68, pages 517–524, New York, NY, USA, 1968. ACM.

20. T. M. Shih, C. H. Tan, and B. C. Hwang. Effects of grid staggering on numerical schemes. *International Journal for Numerical Methods in Fluids*, 9(2):193–212, 1989.

21. L. Sim, J. Graham, K. Rose, R. Duran, J. Nelson, J. Umhoefer, and J. Vielma. Developing a comprehensive deepwater blowout and spill model. Technical report, U.S. Department of Energy, National Energy Technology Laboratory, Albany, OR, 2015. NETL-TRS-9-2015.

22. Scott A. Socolofsky, E. Eric Adams, Michel C. Boufadel, Zachary M. Aman, Øistein Johansen, Wolfgang J. Konkel, David Lindo, Mads N. Madsen, Elizabeth W. North, Claire B. Paris, Dorte Rasmussen, Mark Reed, Petter Rnningen, Lawrence H. Sim, Thomas Uhrenholdt, Karl G. Anderson, Cortis Cooper, and Tim J. Nedwed. Intercomparison of oil spill prediction models for accidental blowout scenarios with and without subsea chemical dispersant injection. *Marine Pollution Bulletin*, 96(12):110 – 126, 2015.

23. Elias M Stein and Rami Shakarchi. *Complex analysis*, volume 2. Princeton University Press, 2010.

24. G. Stewart. *Afternotes on Numerical Analysis*. Society for Industrial and Applied Mathematics, 1996.

APPENDICES

A Code

Here we present a MATLAB implementation for finding a characteristic curve in the lid-driven cavity problem.

Let the number of steps taken be $n = 100$, the final time be $T = 10$ and the initial position of a parcel be $(0.5, 0.25)$. Then, the position of the parcel for each time step is stored in the array `parcel` and calculated by the following code:

```
n = 100; T = 10; k = T/n;
u = @(x,y) 8*(x.^4 -2*x.^3 + x.^2).*(4*y.^3 - 2*y);
v = @(x,y) -8*(4*x.^3 -6*x.^2 + 2*x).*(y.^4 - y.^2);
x1 = 0.5; y1 = 0.25;
parcel = zeros(2,n+1); parcel(1,1) = x1; parcel(2,1) = y1;
for j = 1:n
    U1 = u(x1,y1);
    V1 = v(x1,y1);
    U2 = u(x1 +k/2 *U1, y1 +k/2 *V1);
    V2 = v(x1 +k/2 *U1, y1 +k/2 *V1);
    U3 = u(x1 +k/2 *U2, y1 +k/2 *V2);
    V3 = v(x1 +k/2 *U2, y1 +k/2 *V2);
    U4 = u(x1 +k   *U3, y1 +k   *V3);
    V4 = v(x1 +k   *U3, y1 +k   *V3);
    U = (U1 +2*U2 +2*U3 +U4)/6;
    V = (V1 +2*V2 +2*V3 +V4)/6;
    x1 = x1 +k *U;
    y1 = y1 +k *V;
    parcel(1,j+1)= x1; parcel(2,j+1) = y1;
end
```