THE UNIVERSITY OF READING

Mesh Movement

via

Optimal Transportation

Robin Douglas Giddings

This thesis is submitted for the degree of

Doctor of Philosophy

Department of Mathematics

October 2008

Declaration

I confirm that this is my own work and the use of all material from other sources has been properly and fully acknowledged.

Robin Douglas Giddings

Abstract

This thesis develops a new moving mesh method for the Euler equations based on optimal transportation theory. It employs the Geometric Method, a numerical solution procedure for the optimal transport equations originally developed in the context of weather simulation, to generate an unstructured polyhedral mesh with cells of specified areas. One property of the method is that the mesh connectivity changes naturally as the areas change, so that the mesh cannot tangle. Another is that the method is global - change just one area and the entire mesh will adjust to accommodate - allowing rapid adaption to changing conditions. However this sensitivity requires careful smoothing in the presence of discontinuities in the data such as shocks, and is achieved here by the use of novel monitor functions based on weighted least squares errors. Conservation is achieved by the construction of space-time cells and discretisation by a second order extension of Godunov's method which includes a novel limiter. The method is demonstrated on several test problems including some comparable with published solutions and finally another with large deformation, in which the regions of concentrated mesh change topology significantly during the course of the simulation, merging and separating.

Acknowledgements

I would like to thank my supervisors Professor Cullen and Professor Baines for their sage counsel and tireless good-humoured interest in my blundering about. Also thanks to Dr Sweby for agreeing with me that one time.

I am grateful to A.W.E. for funding this work.

Contents

Introduction

The goal of computer simulation is to solve practical problems quickly and accurately. However perfect accuracy is generally impossible due to the presence of various types of error, so it is necessary to understand these in order to gauge the level of accuracy and gain confidence in the simulation. There are three main types of error: modelling, discretisation and round-off, which are now described.

The first step in any simulation is to assemble a set of model equations, usually partial differential equations (PDEs), that include all the relevant physics. *Modelling errors* are introduced when, for simplicity, terms are discarded because their effects are expected to be small in the problem under consideration. Found by balancing terms, the effects discarded can include viscosity, material strength, relativity etc. We consider model equations of the form:

$$\mathcal{E}(\boldsymbol{u}) \stackrel{\text{def}}{=} \frac{\partial \boldsymbol{u}}{\partial t} - \mathcal{L}(\boldsymbol{u}) = 0$$

where $\boldsymbol{u}$ is a scalar or vector of unknowns and $\mathcal{L}$ a spatial differential operator.

The next step is to set a length scale h and choose $\boldsymbol{u}_h$, a discrete (finite-dimensional) approximation to $\boldsymbol{u}$ which can be stored in computer memory. Finally a corresponding discrete approximation to $\mathcal{E}$, $\mathcal{E}_h$, must be defined and an algorithm developed to solve the resulting discretised system:

$$\mathcal{E}_h(\boldsymbol{u}_h) = 0$$

for $\boldsymbol{u}_h$. The *interpolation* or *discretisation error* is $\boldsymbol{u}-\boldsymbol{u}_h$, the difference between the true and approximate solutions.

Lastly, *round-off errors* are introduced by the finite accuracy of floating point numbers and arithmetic, but these are usually small and are not considered here.

There are currently four main domain-based methods of discretisation. In the *finite difference* method $\boldsymbol{u}_h$ approximates $\boldsymbol{u}$ at a grid of points, and $\mathcal{E}_h$ is formed by replacing derivatives with finite differences. The *truncation error* is $\mathcal{E}_h(\boldsymbol{u})-\mathcal{E}_h(\boldsymbol{u}_h)$ $(= \mathcal{E}_h(\boldsymbol{u}))$ which measures the error in the model equations.

In the *finite volume* and *finite element* methods the domain is decomposed into a mesh of polygonal or polyhedral cells. In each cell the finite volume approximation $\boldsymbol{u}_h$ of $\boldsymbol{u}$ is a single cell-averaged value, whereas the finite element approximation $\boldsymbol{u}_h$ is constructed from a set of basis functions (usually polynomial) defined throughout the cell. In both, the error in the model equations is measured by the *residual error* $\mathcal{E}(\boldsymbol{u}_h)$.

Lastly *mesh-free* or *particle* methods, e.g. Smoothed Particle Hydrodynamics (SPH) [63], are hybrids (and confusingly some construct temporary meshes at each timestep). Like the finite difference method they approximate $\boldsymbol{u}$ at a grid of points, but these can move round the domain. Like the finite element method they employ basis functions but these are associated with the points.

Away from discontinuities, the truncation or residual error can be analysed by expanding $\boldsymbol{u}$ in a Taylor series in the mesh size h and timestep δt. An algorithm of order p (in space) is one which agrees with the model equations for the first p terms i.e. the error is $O(h^p)$ (and similarly for δt). This implies the error at a point can be reduced either by increasing the order p or decreasing the mesh size h.

The constant factor multiplying h^p is both algorithm and solution dependent, making exact error estimation difficult, but in general it will be higher where the polynomial approximations for the variables are poor, e.g. at steep gradients. Typically however, such regions only cover a fraction of the total domain - for example a shock wave is a line or curve in 2D. Early algorithms employed a fixed, uniform ('Eulerian') mesh and fixed p, which is inefficient as the mesh size could be increased away from these regions without affecting the maximum error. In practice this inefficiency can be significant, with large amounts of computing memory and processing time being taken up by regions where very little is happening. The obvious solution is to adjust h and/or p locally. This is known as mesh *refinement* and there are three basic types:

h-refinement: Reduce h by subdividing a cell.

Also known as Adaptive Mesh Refinement or AMR [13], the new cells can either replace the old or be put in a separate mesh. Further subdivision is possible creating levels of successive refinement (Fig 1), allowing the error to be estimated via Richardson extrapolation. The first method creates 'hanging nodes' which can require special treatment, whereas with separate meshes the solution must be conservatively transferred between them, both of which can introduce perturbations at the boundaries between different levels of refinement.

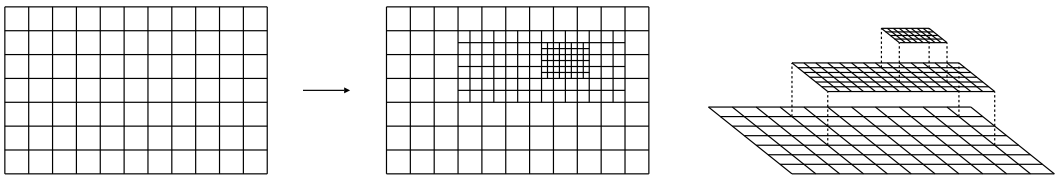


Figure 1: Adaptive Mesh Refinement

p-refinement: Increase the order p in a cell.

This requires higher-order derivatives of the variables, which can either be stored as additional data (as in the finite element method) or reconstructed at each timestep e.g. by polynomial interpolation. Higher order interpolation requires larger stencils and usually assumes smoothness, so special methods are needed to maintain monotonicity in the presence of discontinuities, e.g. ENO/WENO schemes [49] which employ solution-dependent stencils/coefficients.

r-refinement: Redistribute or move the mesh keeping the number of cells constant.

The simplest example is the Lagrangian method, where the mesh moves with the local fluid velocity (Fig 2). This often reduces h where needed but can result in the mesh becoming distorted or tangled in strong shear flows or vortices. In *Moving Mesh* methods [45] the movement is completely general, and can be used to control the mesh quality, truncation error or residual, or other user-supplied criteria. Many employ *monitor functions* as intermediaries to guide the mesh. Of these the Arbitrary Lagrangian-Eulerian (ALE) methods of [10] are closest to Lagrangian, relaxing the mesh at the end of each step to improve the mesh quality.

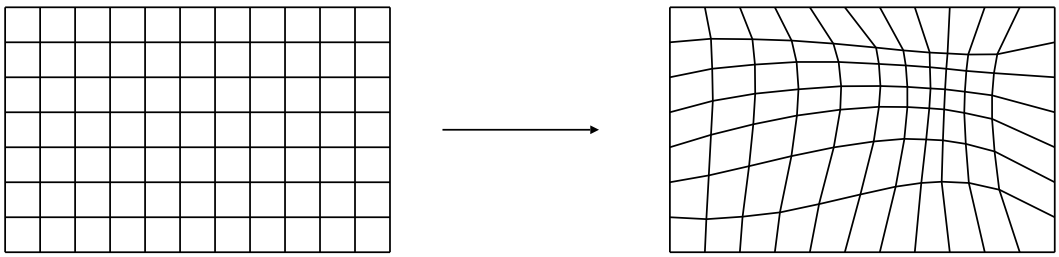


Figure 2: Moving Mesh methods

In addition there are hybrid schemes such as hp-refinement. Both h-refinement and p-refinement are fairly mature and have been used extensively. Experience and error analysis has led to effective refinement criteria and efficient methods for a wide range of application areas. However, unlike r-refinement the total number of cells varies with time necessitating more complex data structures and algorithms. In contrast, r-refinement is still the subject of much current research, mainly focussed on how to successfully use the generality of motion available in moving mesh methods.

The purpose of this thesis is to demonstrate a new moving mesh method based on a mesh construction algorithm originally developed for atmospheric modelling. The semi-geostrophic equations are a simplified weather model originally introduced by Hoskins in

1975 [41]. Analysis of them by Cullen and Purser [24] in the 1980s led to a novel solution algorithm called the Geometric Method that was further developed by Chynoweth [22] and later Purser (described in Cullen [23, Sec 5.3.2]). Meanwhile significant advances had been made in optimal transportation theory by Brenier [15] and others, and it was eventually realised that the semi-geostrophic equations were in fact a model optimal transportation problem, bringing a more complete understanding to the subject [23].

In discretising and solving the optimal transport problem, the Geometric Method splits the spatial domain up into an unstructured mesh of polyhedral cells having specified (positive) areas, and it was realised that this could be used for mesh generation. Furthermore, the generated mesh depends continuously on the areas specified, so if they are slowly modified then the mesh changes smoothly to accommodate this with no cells being created or destroyed suggesting it could also be used for mesh movement. One feature of the method is that the mesh connectivity is not fixed, so as the mesh changes new neighbours can be created or existing neighbours moved apart. Thus it provides a novel alternative to other methods of generating variable connectivity meshes, e.g. those based on the Voronoi mesh such as the Free Lagrange method [8], and could prove beneficial in problems where fixed-connectivity meshes encounter difficulties as described above. Another feature is that the geometric method is global - change the specified area in any one cell and the entire mesh adjusts. This means that the mesh can adapt to changing conditions much more quickly than other methods but as chapter 5 will demonstrate, this sensitivity can require very careful handling if the method of computing the specified areas is noisy.

An outline of the thesis is as follows. Chapter 1 sets the context by surveying current mesh movement techniques, with particular emphasis on harmonic methods and equidistribution. These are derived using variational methods, setting in place some mathematical machinery to be used later. Chapter 2 introduces basic optimal transportation theory, briefly describing its history and current status before sketching some solution methods and applications. Chapter 3 describes in detail the Geometric Method and Panel Beater algorithm and its implementation, before demonstrating it on some simple test problems and analysing the performance. The intended application here is the Euler equations, and chapter 4 describes in detail the discretization method for the unstructured mesh, including a least squares surface fitting procedure for gradient estimation, a new monotonic limiter and the choice of timestep. Then chapter 5 picks up from chapter 1, describing the

particular choice of monitor to be used, which is based on the solution gradient. However this soon runs into difficulties in the mesh adaption when used in problems with discontinuities, and is repeatedly modified to cope with successively more difficult problems, first by adapting the least squares surface fitting, then by using the least squares error itself for the monitor. In the process some new parameters have been introduced and their effects are investigated analytically. Chapter 6 demonstrates the complete algorithm on a selection of test problems, starting off with the simple Sod shock tube to help choose suitable parameter values, then moving on to some more challenging problems. Lastly, conclusions are made in chapter 7 along with suggested further work.

Note: all computations were carried out on a PC with an AMD Athlon 3800+ processor and 2GB of RAM.

Chapter 1

Moving Mesh methods

In this chapter we briefly survey current moving mesh methods. Broadly they are split into those that directly specify the location of the new mesh at each timestep and those that generate a mesh velocity which is integrated to advance the current mesh in time. The majority start by defining a functional measuring the quality of the mesh. The mesh motion is then found by maximising this functional, either by the Euler-Lagrange equations or other optimisation methods. We apply variational theory to derive the former for two popular functionals, the *harmonic method* and *equidistribution*.

1.1 Introduction

We start by introducing some notation to describe the mesh and its motion. The problem domain $\Omega \subset \mathbb{R}^n$ (boundary $\partial\Omega$) is called *physical space*. Each point $\boldsymbol{x} \in \Omega$ is assigned a fixed label $\boldsymbol{\xi}$, where $\boldsymbol{\xi}$ is a point in *computational space* $\Omega_c \subset \mathbb{R}^n$ (boundary $\partial\Omega_c$). It is assumed that Ω and Ω_c are compact and simply-connected, and that at time t the map $\boldsymbol{x}(\boldsymbol{\xi}, t) : \Omega_c \to \Omega$ is invertible and maps $\partial\Omega_c$ onto $\partial\Omega$, as illustrated in Fig 1.1, for example.

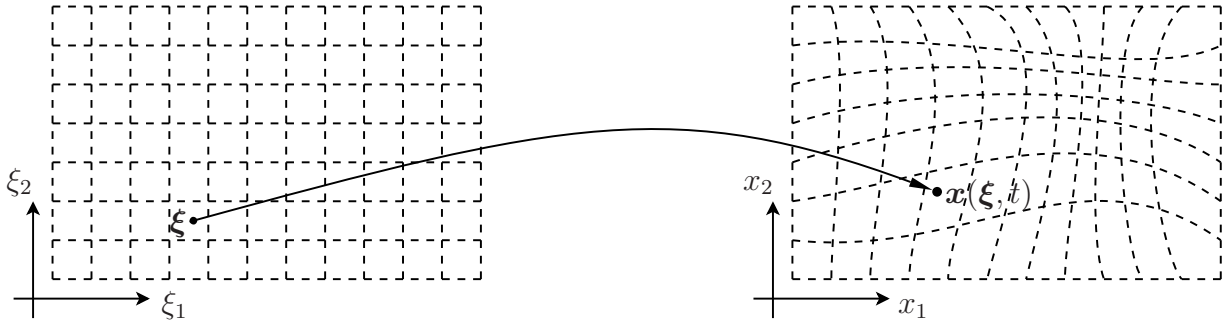


Figure 1.1: Computational space and physical space

The Jacobian of this map is the matrix $\boldsymbol{J}$ with components and determinant

$$J_{ij} = J_{ij}(\boldsymbol{\xi}) = \frac{\partial x^i}{\partial \xi^j}, \qquad J = det(\boldsymbol{J}) \tag{1.1}$$

where indices range from 1 to n. If Ω_c is subdivided into cells, then the map transforms these into cells in Ω. In Fig 1.1 the computational mesh is Cartesian resulting in a logically structured physical mesh but this approach works equally well for unstructured meshes.

A small area $|d\boldsymbol{\xi}|$ in computational space is mapped to $|d\boldsymbol{x}| = J(\boldsymbol{\xi})|d\boldsymbol{\xi}|$ in physical space, so $J(\boldsymbol{\xi})$ represents the area magnification factor at $\boldsymbol{\xi}$. If it becomes zero or negative then the physical mesh cells are contracted to a point or become folded over and $\boldsymbol{x}$ is no longer invertible, thus it is required that $J > 0$ to avoid this, i.e. $\boldsymbol{J}$ be positive definite. Some but not all schemes have this property. For later convenience we define the inverse Jacobian separately:

$$K_{ij} = K_{ij}(\boldsymbol{x}) = \frac{\partial \xi^i}{\partial x^j} \quad \Rightarrow \quad J_{ij}K_{jk} = \delta_{ik}, \quad K = \det(\boldsymbol{K}) = \frac{1}{J}$$

where δ_{ij} is the Kronecker delta and summation over repeated indices is implied. The inverse $\boldsymbol{\xi}(\boldsymbol{x}, t)$ satisfies

$$\boldsymbol{x}(\boldsymbol{\xi}(\boldsymbol{y}, t), t) = \boldsymbol{y} \quad \text{for all } t, \boldsymbol{y} \in \Omega.$$

Differentiating this with respect to time gives $J_{ij}\frac{\partial \xi^j}{\partial t} + \frac{\partial x^i}{\partial t} = 0 \ \forall i$. For a finite timestep δt, this is approximated by

$$J_{ij}\delta\xi^j + \delta x^i = 0 \tag{1.2}$$

where $\delta\boldsymbol{x} = \delta t\frac{\partial \boldsymbol{x}}{\partial t}$ and $\delta\boldsymbol{\xi} = \delta t\frac{\partial \boldsymbol{\xi}}{\partial t}$.

1.2 Location-based methods

A number of mesh movement schemes are based on minimizing some functional $I[\boldsymbol{\xi}]$. For example *harmonic methods* [18] minimize the total (elastic distortional) energy of the mapping:

$$I_{hrm}[\boldsymbol{\xi}] = \int_\Omega \sum_{i=1}^{n} (\nabla\xi^i)^T \boldsymbol{G}_i^{-1} (\nabla\xi^i) \mathrm{d}\boldsymbol{x} = \int_\Omega K_{ij}(G_i^{-1})_{jk}K_{ik}\mathrm{d}\boldsymbol{x} \tag{1.3}$$

where $\boldsymbol{G}_i^{-1}(\boldsymbol{x})$ are symmetric matrices specifying the 'coefficients of elasticity' at $\boldsymbol{x}$, called *monitor functions*. The existence and uniqueness of minimizers - which are harmonic maps (hence the name) - is guaranteed provided $\partial\Omega$ is convex [38, 79]. In 2D the mesh cannot fold but this is not true in 3D [62]. This general framework includes the functionals of:

- **Winslow** [95] $\boldsymbol{G}_i^{-1} = \boldsymbol{G}^{-1} = \frac{1}{w}\boldsymbol{I}$ where $\boldsymbol{I}$ is the identity matrix and w a scalar function controlling the relative size of mesh cells;
- **Dvinsky** [28] $\boldsymbol{G}_i^{-1} = \boldsymbol{G}^{-1} = \boldsymbol{I} + f(F)(\nabla F)(\nabla F)^T/|\nabla F|^2$ which concentrates mesh over the curve $F = 0$ in accordance with a distance function f;

- **Brackbill** [14] $\boldsymbol{G}_i^{-1} = \frac{1}{w}[(\boldsymbol{V}_i \cdot \boldsymbol{V}_i)\boldsymbol{I} - \boldsymbol{V}_i \boldsymbol{V}_i^T]$ for scalar w which aligns the mesh with the vectors $\boldsymbol{V}_i$;

amongst others [6] (some definitions have an explicit tensor weighting factor $\sqrt{\det \boldsymbol{G}}$ but here it is incorporated into $\boldsymbol{G}_i^{-1}$).

A number of schemes are based on the *equidistribution principle* [45]. For example Cao *et al* [18] define an equidistribution functional

$$I_{eq}[\boldsymbol{\xi}] = \int_\Omega \frac{m}{(Jm)^\gamma} \mathrm{d}\boldsymbol{x} \tag{1.4}$$

where $m(\boldsymbol{x}) > 0$ is a scalar monitor function and $\gamma > 1$. Some other functional approaches are those of Knupp [52]:

$$I[\boldsymbol{\xi}] = \int_\Omega \|\boldsymbol{K} - \boldsymbol{G}^{-1}\|_F^2 \mathrm{d}\boldsymbol{x}$$

where $\|\cdot\|_F$ is the Frobenius norm, and Thompson *et al* [87]:

$$I[\boldsymbol{\xi}] = \int_\Omega (\|\boldsymbol{K}\|_F^2 - P_i \xi^i) \mathrm{d}\boldsymbol{x}$$

where P_i are control functions.

In general

$$I[\boldsymbol{\xi}] = \int_\Omega c[\boldsymbol{\xi}(\boldsymbol{x})] \mathrm{d}\boldsymbol{x}$$

for some cost functional $c[\boldsymbol{\xi}(\boldsymbol{x})]$. Often I is minimized by the steepest descent method. Under the variation $\boldsymbol{\xi} \to \boldsymbol{\xi} + \delta\boldsymbol{\xi}$ (which must be *natural* to preserve $\partial\Omega$, i.e. $\delta\boldsymbol{\xi}$ and all its derivatives vanish on $\partial\Omega$) the (first) variation in I is

$$\begin{aligned} \delta I &= \int_\Omega \left[\frac{\partial c}{\partial \xi^i} \delta\xi^i + \frac{\partial c}{\partial \xi^i_{,j}} \delta\xi^i_{,j} + \frac{\partial c}{\partial \xi^i_{,jk}} \delta\xi^i_{,jk} + \ldots \right] \mathrm{d}\boldsymbol{x} \\ &= \int_\Omega \left[\frac{\partial c}{\partial \xi^i} - \frac{\partial}{\partial x^j}\left(\frac{\partial c}{\partial \xi^i_{,j}}\right) + \frac{\partial^2}{\partial x^j \partial x^k}\left(\frac{\partial c}{\partial \xi^i_{,jk}}\right) + \ldots \right] \delta\xi^i \mathrm{d}\boldsymbol{x} \end{aligned}$$

on integration by parts (commas denoting differentiation), assuming the relevant derivatives of the cost functional exist. This is written

$$\delta I = \int_\Omega \frac{\delta I}{\delta \xi^i} \delta\xi^i \mathrm{d}\boldsymbol{x} \tag{1.5}$$

where the *variational* or *Fréchet* derivative of I is

$$\frac{\delta I[\boldsymbol{\xi}]}{\delta \xi^i(\boldsymbol{x})} = \frac{\partial c}{\partial \xi^i} - \frac{\partial}{\partial x^j}\left(\frac{\partial c}{\partial \xi^i_{,j}}\right) + \frac{\partial^2}{\partial x^j \partial x^k}\left(\frac{\partial c}{\partial \xi^i_{,jk}}\right) + \ldots \quad . \tag{1.6}$$

The steepest descent direction (in computational space) is then

$$\delta\xi^i_{\Omega_c}(\boldsymbol{x}) = -\frac{\delta I[\boldsymbol{\xi}]}{\delta \xi^i(\boldsymbol{x})}. \tag{1.7}$$

At stationary points $\delta\boldsymbol{\xi}_{\Omega_c} = 0$ - the Euler-Lagrange equation for I.

For the harmonic functional (1.3), $c = \sum_k K_{kl}(G_k^{-1})_{lm}(\boldsymbol{x})K_{km}$ and (1.6) gives

$$\begin{aligned}
\frac{\delta I_{hrm}[\boldsymbol{\xi}]}{\delta\xi^i(\boldsymbol{x})} &= 0 - \sum_k \frac{\partial}{\partial x^j}\left(\frac{\partial}{\partial \xi^i_{,j}}\left[K_{kl}(G_k^{-1})_{lm}(\boldsymbol{x})K_{km}\right]\right) + 0... \\
&= -\sum_k \frac{\partial}{\partial x^j}\left((G_k^{-1})_{lm}(\boldsymbol{x})\frac{\partial}{\partial K_{ij}}\left[K_{kl}K_{km}\right]\right) \\
&= -\sum_k \frac{\partial}{\partial x^j}\left((G_k^{-1})_{lm}(\boldsymbol{x})\left[\delta_{ik}\delta_{jl}K_{km} + K_{kl}\delta_{ik}\delta_{jm}\right]\right) \\
&= -\sum_k 2\delta_{ik}\frac{\partial}{\partial x^j}\left((G_k^{-1})_{jm}K_{km}\right) \\
&= -2\nabla\cdot(\boldsymbol{G}_i^{-1}\nabla\xi^i) \quad \text{(no sum on } i\text{)}
\end{aligned}$$

where the symmetry of $\boldsymbol{G}_i^{-1}$ has been used. At the stationary point (a minimum if the $\boldsymbol{G}_i^{-1}$ are set up to be positive definite)

$$\nabla\cdot(\boldsymbol{G}_i^{-1}\nabla\xi^i) = 0, \quad i = 1,\ldots,n, \tag{1.8}$$

i.e. each ξ^i satisfies a generalized Laplace equation. In practice some discrete methods of solution can lead to mesh folding, despite this being prohibited analytically (in 2D). Ivanenko's variational barrier method [6] introduces a clever discretization to avoid this.

Before proceeding we derive a useful identity. From the cofactor expansion of K,

$$\frac{\partial K}{\partial K_{ij}} = KK^{-1}_{ji} \tag{1.9}$$

where K^{-1}_{ij} denotes $(K^{-1})_{ij}$. Now set

$$I[\boldsymbol{\xi}] = \int_\Omega K\mathrm{d}\boldsymbol{x} = \int_{\Omega_c}\mathrm{d}\boldsymbol{\xi} = |\Omega_c|.$$

A natural variation $\delta\boldsymbol{\xi}$ will leave $\partial\Omega_c$ and hence $|\Omega_c|$ unchanged, so taking the variational derivative

$$\begin{aligned}
0 = \frac{\delta I[\boldsymbol{\xi}]}{\delta\xi^i(\boldsymbol{x})} &= \int_\Omega \frac{\delta I}{\delta\xi^i}\delta\xi^i\mathrm{d}\boldsymbol{x} = \int_\Omega\left[\frac{\partial K}{\partial\xi^i} - \frac{\partial}{\partial x^j}\left(\frac{\partial K}{\partial\xi^i_{,j}}\right) + \ldots\right]\delta\xi^i\mathrm{d}\boldsymbol{x} \\
&= -\int_\Omega \frac{\partial}{\partial x^j}(KK^{-1}_{ji})\delta\xi^i\mathrm{d}\boldsymbol{x}.
\end{aligned}$$

But $\delta\boldsymbol{\xi}$ is arbitrary, so [93]

$$\frac{\partial}{\partial x^j}(KK^{-1}_{ji}) = 0. \tag{1.10}$$

Following the same argument but with $\boldsymbol{x}$ and $\boldsymbol{\xi}$ switched,

$$\frac{\partial}{\partial\xi^j}(JJ^{-1}_{ji}) = 0. \tag{1.11}$$

We now proceed to the equidistribution functional (1.4), for which $c = m/(Jm)^\gamma$:

$$\begin{aligned}\frac{\delta I_{eq}[\boldsymbol{\xi}]}{\delta \xi^i(\boldsymbol{x})} &= -\frac{\partial}{\partial x^j}\left(m^{1-\gamma}\frac{\partial}{\partial \xi^i_{,j}}(K^\gamma)\right)\\ &= -\frac{\partial}{\partial x^j}(m^{1-\gamma}\gamma K^{\gamma-1}KK^{-1}_{ji})\\ &= -\gamma K K^{-1}_{ji}\frac{\partial}{\partial x^j}\left[(m/K)^{1-\gamma}\right]\end{aligned}$$

using (1.9) and (1.10). A stationary point occurs when m/K is constant i.e.

$$Jm = \sigma \tag{1.12}$$

for a positive normalisation constant σ determined by integration:

$$\int_\Omega m\mathrm{d}\boldsymbol{x} = \int_\Omega \frac{\sigma}{J}\mathrm{d}\boldsymbol{x} = \sigma\int_{\Omega_c}\mathrm{d}\boldsymbol{\xi} = \sigma|\Omega_c|. \tag{1.13}$$

In fact from Hölders inequality:

$$\left(\int_\Omega f^p\mathrm{d}\boldsymbol{x}\right)^{1/p}\left(\int_\Omega g^q\mathrm{d}\boldsymbol{x}\right)^{1/q} \geqslant \int_\Omega fg\mathrm{d}\boldsymbol{x}, \qquad \frac{1}{p}+\frac{1}{q}=1, \quad p,q\geqslant 1$$

with $f^p = m/(Jm)^\gamma$, $p=\gamma$, $g^q = m$, $q = \gamma/(\gamma-1)$, we have $fg = 1/J$ so

$$I_{eq}[\boldsymbol{\xi}] \geqslant \left(\int_\Omega \frac{\mathrm{d}\boldsymbol{x}}{J}\right)^\gamma \Big/ \left(\int_\Omega m\mathrm{d}\boldsymbol{x}\right)^{\gamma-1} = \frac{|\Omega_c|^\gamma}{(\sigma|\Omega_c|)^{\gamma-1}} = \frac{|\Omega_c|}{\sigma^{\gamma-1}} \tag{1.14}$$

with equality when $Jm = \sigma$ so the stationary point is a minimum, at which m has been equidistributed (see also [44]). Also $J = \sigma/m > 0$ as required for a valid mesh. In 1D this determines $\boldsymbol{\xi}(\boldsymbol{x})$ uniquely, but not in higher dimensions where additional constraints will be needed.

To convert $\boldsymbol{\xi}(\boldsymbol{x})$ into mesh coordinates $\boldsymbol{x}(\boldsymbol{\xi})$, the steepest descent direction $\delta\boldsymbol{\xi}_{\Omega_c}(\boldsymbol{x})$ is transformed using (1.2) and

$$\boldsymbol{x} = \boldsymbol{x}(\boldsymbol{\xi}), \quad \frac{\partial}{\partial \xi^i} = \frac{\partial x^j}{\partial \xi^i}\frac{\partial}{\partial x^j} = K^{-1}_{ji}\frac{\partial}{\partial x^j} \quad \Rightarrow \quad \frac{\partial}{\partial x^i} = J^{-1}_{ji}\frac{\partial}{\partial \xi^j}. \tag{1.15}$$

For the harmonic functional

$$\begin{aligned}\delta x^i_{\Omega_c}(\boldsymbol{\xi}) &= -J_{ij}\delta\xi^j_{\Omega_c} = -J_{ij}\left(-\frac{\delta I_{hrm}}{\delta\xi^j}\right)\\ &= -2J_{ij}\frac{\partial}{\partial x^l}\left((G^{-1}_j)_{lm}K_{jm}\right)\\ &= -2J_{ij}J^{-1}_{kl}\frac{\partial}{\partial\xi^k}((G^{-1}_j)_{lm}J^{-1}_{jm}).\end{aligned} \tag{1.16}$$

When $\boldsymbol{G}^{-1}_j = \boldsymbol{G}^{-1}$, and setting $(\boldsymbol{a}_i)_j \equiv J_{ji}$, $(\boldsymbol{a}^i)_j \equiv J^{-1}_{ij}$, this can be rewritten

$$\begin{aligned}\delta x^k_{\Omega_c}(\boldsymbol{\xi}) &= -2J_{ki}J^{-1}_{jl}\frac{\partial}{\partial\xi^j}(G^{-1}_{lm}J^{-1}_{im})\\ &= -\frac{2}{J}J_{ki}\frac{\partial}{\partial\xi^j}\left(JJ^{-1}_{jl}G^{-1}_{lm}J^{-1}_{im}\right)\\ \text{hence}\quad \delta\boldsymbol{x}_{\Omega_c}(\boldsymbol{\xi}) &= -\frac{2}{J}\boldsymbol{a}_i\frac{\partial}{\partial\xi^j}\left(J\boldsymbol{a}^j\cdot\boldsymbol{G}^{-1}\boldsymbol{a}^i\right)\end{aligned}$$

agreeing with [42, (6)] (with $p = 2$). For the equidistribution functional

$$\begin{aligned}\delta x^i_{\Omega_c}(\boldsymbol{\xi}) &= -J_{ij}\left[-\gamma J^{-1}\frac{\partial}{\partial \xi^j}\left[(Jm)^{1-\gamma}\right]\right] \\ &= \frac{\gamma}{J}J_{ij}\frac{\partial}{\partial \xi^j}\left[(Jm)^{1-\gamma}\right].\end{aligned}$$

An obvious question is why iterate $\boldsymbol{\xi}(\boldsymbol{x})$ then convert to $\boldsymbol{x}(\boldsymbol{\xi})$ - why not just iterate $\boldsymbol{x}(\boldsymbol{\xi})$ in the first place? Well, $\delta\boldsymbol{x}_{\Omega_c}$ is not the steepest descent direction in physical space. To find that we rewrite (1.5) for δI using (1.7) and transform via (1.2):

$$\begin{aligned}\delta I &= -\int_\Omega \delta\xi^i_{\Omega_c}\delta\xi^i \mathrm{d}\boldsymbol{x} \\ &= -\int_{\Omega_c}(-J^{-1}_{ij}\delta\boldsymbol{x}^j_{\Omega_c})(-J^{-1}_{ik}\delta\boldsymbol{x}^k)J\mathrm{d}\boldsymbol{\xi} \\ &= -\int_{\Omega_c}(JJ^{-1}_{ij}J^{-1}_{ik}\delta\boldsymbol{x}^j_{\Omega_c})\delta\boldsymbol{x}^k\mathrm{d}\boldsymbol{\xi}\end{aligned}$$

Thus the steepest descent direction in physical space ($\delta\boldsymbol{x}_\Omega$) and the steepest descent in computational space *transformed* into physical space ($\delta\boldsymbol{x}_{\Omega_c}$) are related via

$$\delta\boldsymbol{x}_\Omega = J(\boldsymbol{J}^{-T}\boldsymbol{J}^{-1})\delta\boldsymbol{x}_{\Omega_c} \tag{1.17}$$

(presumably a standard result but the author is unaware of it being stated in this context). This can be checked for particular cases by transforming the functional before applying the variation, e.g. for the equidistribution functional:

$$\begin{aligned}I_{eq}[\boldsymbol{\xi}] &= \int_\Omega m^{1-\gamma}(\boldsymbol{x})J^{-\gamma}\mathrm{d}\boldsymbol{x} \\ \Rightarrow I_{eq}[\boldsymbol{x}] &= \int_{\Omega_c} m^{1-\gamma}[\boldsymbol{x}(\boldsymbol{\xi})]J^{-\gamma}J\mathrm{d}\boldsymbol{\xi}.\end{aligned}$$

Then the cost $c[\boldsymbol{x}(\boldsymbol{\xi})] = m^{1-\gamma}[\boldsymbol{x}(\boldsymbol{\xi})]J^{1-\gamma}$ and

$$\begin{aligned}\frac{\delta I_{eq}[\boldsymbol{x}]}{\delta x^i(\boldsymbol{\xi})} &= \frac{\partial c}{\partial x^i} - \frac{\partial}{\partial\xi^j}\left(\frac{\partial c}{\partial x^i_{,j}}\right) + \frac{\partial^2}{\partial\xi^j\partial\xi^k}\left(\frac{\partial c}{\partial x^i_{,jk}}\right) + \dots \\ &= J^{1-\gamma}\frac{\partial}{\partial x^i}\left(m^{1-\gamma}\right) - \frac{\partial}{\partial\xi^j}\left(m^{1-\gamma}\frac{\partial}{\partial x^i_{,j}}\left(J^{1-\gamma}\right)\right) + 0\dots \\ &= J^{1-\gamma}J^{-1}_{ji}\frac{\partial}{\partial\xi^j}\left(m^{1-\gamma}\right) - \frac{\partial}{\partial\xi^j}\left(m^{1-\gamma}(1-\gamma)J^{-\gamma}JJ^{-1}_{ji}\right) \\ &= J^{-1}_{ji}\left[J^{1-\gamma}\frac{\partial}{\partial\xi^j}\left(m^{1-\gamma}\right) - (1-\gamma)J\frac{\partial}{\partial\xi^j}\left(m^{1-\gamma}J^{-\gamma}\right)\right] \\ &= J^{-1}_{ji}\left[J^{1-\gamma}\frac{\partial}{\partial\xi^j}\left(m^{1-\gamma}\right) - (1-\gamma)J^{1-\gamma}\frac{\partial}{\partial\xi^j}\left(m^{1-\gamma}\right) - (1-\gamma)m^{1-\gamma}J\frac{\partial}{\partial\xi^j}\left(J^{-\gamma}\right)\right] \\ &= J^{-1}_{ji}\left[\gamma J^{1-\gamma}\frac{\partial}{\partial\xi^j}\left(m^{1-\gamma}\right) + \gamma m^{1-\gamma}\frac{\partial}{\partial\xi^j}\left(J^{1-\gamma}\right)\right] \\ &= \gamma J^{-1}_{ji}\frac{\partial}{\partial\xi^j}\left[(Jm)^{1-\gamma}\right] \\ &= JJ^{-1}_{ji}J^{-1}_{jk}\delta x^k_{\Omega_c}(\boldsymbol{\xi})\end{aligned}$$

as required. So what's the difference? In steepest descent methods the rate of convergence is governed by the condition number of the Hessian of I [33]. Convergence is slow if the condition number is high which corresponds to the level sets of I being highly ellipsoidal in shape near a minimum. In this situation the steepest descent direction will not point directly to the minimum at the centre of the ellipsoid and so the path taken will meander back and forth along valley floors. Whereas if the condition number is close to one then the Hessian is close to the identity, the level sets are close to spherical and few steps will be required. In practice the condition number can be reduced by *pre-conditioning* - linearly transforming the coordinates at each step in order to make the transformed Hessian close to the identity, then finding the steepest descent direction in the transformed coordinates. For example *quasi-Newton* (or *variable metric*) methods maintain and update an explicit approximation to the inverse Hessian. Here well-adapted coordinates will have a similar effect, with high gradients in physical space being smoothed out in computational space, reducing the condition number, so convergence in computational space should be quicker.

In [43] Huang sets $\boldsymbol{G}_i = \boldsymbol{M}/\sqrt{\det \boldsymbol{M}}$ ($\Rightarrow \det \boldsymbol{G}_i = 1$ in $n = 2$ dimensions) and $(m \equiv)\rho = \sqrt{\det \boldsymbol{M}}$ so that the same problem can be solved by both the equidistribution and harmonic methods. Setting $\boldsymbol{M}_c = \boldsymbol{J}^T \boldsymbol{M} \boldsymbol{J}$, then with his definitions

$$I_{eq}[\boldsymbol{\xi}] = \int_\Omega [n^n \det(\boldsymbol{M}_c^{-1})]^{\frac{\gamma}{2}} m(\boldsymbol{x}) \mathrm{d}\boldsymbol{x}, \quad I_{hrm}[\boldsymbol{\xi}] = \int_\Omega [\mathrm{tr}(\boldsymbol{M}_c^{-1})]^{\frac{n\gamma}{2}} m(\boldsymbol{x}) \mathrm{d}\boldsymbol{x}.$$

Applying the arithmetic-geometric mean inequality to the eigenvalues of $\boldsymbol{M}_c^{-1}$ gives

$$n^n \det(\boldsymbol{M}_c^{-1}) \leqslant \mathrm{tr}(\boldsymbol{M}_c^{-1})^n$$

from which it follows that $I_{eq}[\boldsymbol{\xi}] \leqslant I_{hrm}[\boldsymbol{\xi}]$. This makes sense because the equidistribution condition (1.12) is less restrictive than the harmonic (1.8). Furthermore equality occurs when $\boldsymbol{M}_c = \theta(\boldsymbol{x})\boldsymbol{I}$ for some scalar function $\theta(\boldsymbol{x})$, in which case the optimal mesh is isotropic and equidistributes m, so accessible to both methods. Also $\boldsymbol{M} = \theta(\boldsymbol{x})\boldsymbol{J}^{-T}\boldsymbol{J}^{-1}$ in which the change of variable factor (1.17) appears again.

1.3 Velocity-based methods

These schemes generate a mesh velocity $\boldsymbol{x}_t$ which is used to advance the current mesh to its new position.

The *Lagrangian* method sets $\boldsymbol{x}_t = \boldsymbol{v}$, the local fluid velocity. This has the advantage that convection terms are eliminated from the governing equations, but strong shear flows

and vortices can distort or even fold the mesh. An ALE method avoids this by adding a mesh relaxation phase at the end of each step to improve the mesh quality, a popular choice being a few iterations of the steepest descent method with Winslow's harmonic functional (1.16).

The *Moving Finite Element* (MFE) method [7] determines both $\boldsymbol{x}_t$ and $\frac{\partial \boldsymbol{u}_h}{\partial t}$ simultaneously by minimising the weighted L^2 norm of the residual:

$$I_{MFE}[\frac{\partial \boldsymbol{u}_h}{\partial t}, \boldsymbol{x}_t] = \int_\Omega [\mathcal{E}(\boldsymbol{u}_h)]^2 w(\boldsymbol{x}) \mathrm{d}\boldsymbol{x}.$$

In the original MFE the weight $w = 1$ whereas in the Gradient Weighted MFE (GWMFE) $w = 1/(1 + |\nabla \boldsymbol{u}|^2)$. In 1D, when $\mathcal{L} = \mathcal{L}(x, u, u_x)$, the MFE reduces to the Lagrangian method: $x_t = -\partial \mathcal{L}/\partial u_x = v$. As with static finite elements the solution is found by inverting a mass matrix, but unlike static elements the matrix is not always positive definite and can be singular in certain situations. This can be fixed with the addition of penalty functions or by choosing a different timestep when such situations arise.

The *Geometric Conservation Law* (GCL) [86] states that a constant $\boldsymbol{u}$ should be an exact solution of the numerical method. It can be expressed as a statement of the conservation of area:

$$\nabla \cdot \boldsymbol{x}_t = \frac{1}{J} \frac{DJ}{Dt}$$

where $\frac{D}{Dt} \equiv \frac{\partial}{\partial t} + \boldsymbol{x}_t \cdot \nabla$ is the usual Lagrangian derivative. Cao *et al* [18] choose $\boldsymbol{x}_t$ to equidistribute a monitor function ρ (1.12) $\Rightarrow \frac{D(\rho J)}{Dt} = 0$ which when combined with the above yields

$$\nabla \cdot (\rho \boldsymbol{x}_t) + \frac{\partial \rho}{\partial t} = 0.$$

However this is insufficient to uniquely determine $\boldsymbol{x}_t$. Motivated by the Helmholtz decomposition theorem, which states that a continuous, differentiable vector field can be uniquely resolved into the sum of the gradient of a scalar field and the curl of a vector field, it is sufficient to specify in addition the curl of $\boldsymbol{x}_t$, for example via $\nabla \times w(\boldsymbol{x}_t - \boldsymbol{V}) = 0$ for some scalar weight w and vector field $\boldsymbol{V}$. Typically $\boldsymbol{V} = 0$ and $w = 1$ (in which case $\nabla \times \boldsymbol{x}_t = 0$ i.e. the mesh is *irrotational*) or $w = \rho$. Together with suitable boundary conditions, these form an elliptic system which is solved for $\boldsymbol{x}_t$.

This completes the summary of the main methods currently in use. The analysis of the variational approach helped put the different methods into context and sets the scene for the next chapter, which introduces and explores an alternative functional from *optimal transportation theory*.

Chapter 2

Optimal Transportation

Optimal transportation theory considers the following problem: we are given an initial distribution μ of some material, for example a heap of sand, and we wish to rearrange this into some target distribution ν (Fig 2.1) of the same volume. The catch is that there is a cost associated with doing so - transporting a unit of sand from point $\boldsymbol{x}$ to point $\boldsymbol{y}$ costs $c(\boldsymbol{x}, \boldsymbol{y})$. The problem then is to find the rearrangement which minimizes the total cost or in other words the *optimal transport plan*. In this chapter we first summarise the main results and a little useful background theory for the case of continuous ν. Then we introduce some discontinuous variations and describe some current applications.

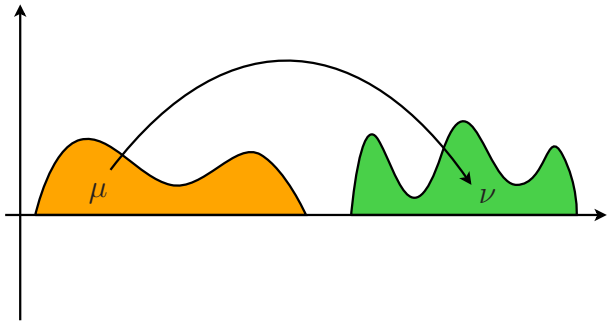


Figure 2.1: The Monge problem - move the heap of sand from μ to ν cheaply

The problem has a long history, being originally formulated by Gaspard Monge [64] in 1781 for $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ and cost equal to the distance moved: $c(\boldsymbol{x}, \boldsymbol{y}) = |\boldsymbol{x} - \boldsymbol{y}|$. However it is not until fairly recently that significant progress has been made, in the process bringing together work in fields as diverse as geometry, probability, functional analysis, partial differential equations, fluid dynamics and mathematical economics. It transpires that Monge's original problem turns out to be the most difficult, in part because for this cost function the solutions are highly degenerate. For example, suppose you wish to move a row of books one place to the right - you can either move all the books along one or move the leftmost book to the other end leaving the rest untouched - both are equally valid solutions.

Another subtlety introduced by Kantorovich [50, 51] (who shared the 1975 Nobel

prize in economics for 'contributions to the theory of optimal allocation of resources'), is whether grains of sand are considered divisible or not i.e. whether sand at a point can be split up and parts sent to different destinations. In the Monge problem (MP) this is not allowed and so any transport plan can be written as a function $\boldsymbol{y} = \boldsymbol{t}(\boldsymbol{x})$ (for $\boldsymbol{x}, \boldsymbol{y}$ in general measure spaces X, Y), whereas in the Monge-Kantorovich problem (MKP) this restriction is relaxed, and the transport plan must be recast as a joint probability measure $\pi(\boldsymbol{x}, \boldsymbol{y})$ - the probability that material at $\boldsymbol{x}$ is transported to $\boldsymbol{y}$. Any solution to the MP is automatically a solution to the associated MKP but the reverse doesn't hold and often there is a solution to the MKP but not the MP (e.g. when μ is a point mass but ν isn't then mass *must* be split). In 1987 Brenier [15] solved the MKP (for $X, Y \subset \mathbb{R}^n$, quadratic cost $c(\boldsymbol{x}, \boldsymbol{y}) = |\boldsymbol{x} - \boldsymbol{y}|^2$ and μ, ν absolutely continuous with respect to the Lebesgue measure $\mathcal{L}$) with the introduction of *polar factorization*. This stimulated a renewal of interest in the subject, and work by Evans, Gangbo, Knott and Smith, and McCann amongst many others, has led to a broad understanding of the problem for general cost functions and spaces X, Y, see e.g. [30, 31, 75, 91].

Qualitatively, the character of the solution depends on the 'regularity' of μ, ν, and whether the cost function is convex, linear or concave in the distance $\mathrm{d}(\boldsymbol{x}, \boldsymbol{y})$. For $X, Y \subset \mathbb{R}^n$, $\mathrm{d}(\boldsymbol{x}, \boldsymbol{y}) = |\boldsymbol{x} - \boldsymbol{y}|$ and $c(\boldsymbol{x}, \boldsymbol{y}) = |\boldsymbol{x} - \boldsymbol{y}|^p$ the cost is (strictly) concave when $0 < p < 1$, linear when $p = 1$ and (strictly) convex when $p > 1$. An optimal plan always exists for the MKP, but not necessarily for the MP. For strictly convex cost functions (and μ absolutely continuous) the MP and MKP share a unique solution (furthermore for quadratic cost the solution takes the simple form of the gradient of a convex function of a scalar quantity). When the cost is linear there can be more than one optimal plan for the MP (as seen above), and when concave there is in general no solution to the MP (except when μ and ν are concentrated on disjoint sets) and solutions display a much richer structure - locally reversing the orientation for example. The bookshelf example (with N books of width w) illustrates these other cases - for the convex cost it is now cheaper to move all the books along one (total cost $N \times w^p$) than the first book to the other end (total cost $1 \times (Nw)^p$), but for the concave cost the reverse is true (and no other permutations improve on these for either cost). A more relevant application is where the material represents some goods that need to be shipped from factories μ to shops ν. A concave cost function favours one long trip over two short trips which is often true in practice due to economies of scale.

2.1 Main results

To properly define the problem μ and ν are measures with supports X and $Y \subset \mathbb{R}^n$. The equal volume condition becomes $\mu(X) = \nu(Y) < \infty$. For the MP we consider the class of measurable mappings $\boldsymbol{t} : X \to Y$ which transport or *rearrange* μ into ν i.e.

$$\nu[B] = \mu[\boldsymbol{t}^{-1}(B)] \tag{2.1}$$

for any ν-measurable set $B \subset Y$, or equivalently

$$\int_X f(\boldsymbol{t}(\boldsymbol{x}))\mathrm{d}\mu(\boldsymbol{x}) = \int_Y f(\boldsymbol{y})\mathrm{d}\nu(\boldsymbol{y}) \tag{2.2}$$

for any ν-integrable function f. The mapping $\boldsymbol{t}$ is said to be *measure preserving* or *push-forward* μ onto ν, written $\boldsymbol{t}\#\mu = \nu$. The cost function $c : X \times Y \to \mathbb{R}_+$ and the total cost to be minimized is $I[\boldsymbol{t}] = \int_X c(\boldsymbol{x}, \boldsymbol{t}(\boldsymbol{x}))\mathrm{d}\mu(\boldsymbol{x})$. Define densities α, β by

$$\mathrm{d}\mu(\boldsymbol{x}) = \alpha(\boldsymbol{x})\mathrm{d}\boldsymbol{x}, \qquad \mathrm{d}\nu(\boldsymbol{y}) = \beta(\boldsymbol{y})\mathrm{d}\boldsymbol{y} \tag{2.3}$$

When μ and ν are absolutely continuous with respect to Lebesgue, α and β are bounded nonnegative functions, and the rearrangement condition becomes a *Monge-Ampere* equation for $\boldsymbol{t}$:

$$\alpha(\boldsymbol{x}) = \beta(\boldsymbol{t}(\boldsymbol{x}))\det\left(\frac{\partial \boldsymbol{t}(\boldsymbol{x})}{\partial \boldsymbol{x}}\right). \tag{2.4}$$

For the MKP, the transport plan π is a nonnegative measure defined on the product space $X \times Y$, and the rearrangement condition becomes

$$\pi[A \times Y] = \mu[A], \quad \pi[X \times B] = \nu[B] \tag{2.5}$$

for all measurable subsets $A \subset X$ and $B \subset Y$ (in other words μ and ν are the *marginals* of π), or equivalently $\int_Y \mathrm{d}\pi(\boldsymbol{x}, \boldsymbol{y}) = \mathrm{d}\mu(\boldsymbol{x})$ and $\int_X \mathrm{d}\pi(\boldsymbol{x}, \boldsymbol{y}) = \mathrm{d}\nu(\boldsymbol{y})$. Let $\Pi(\mu, \nu)$ denote the set of π satisfying the rearrangement condition. The MP is now a special case for which π takes the form $\pi(\boldsymbol{x}, \boldsymbol{y}) = \delta(\boldsymbol{x})\delta(\boldsymbol{y} - \boldsymbol{t}(\boldsymbol{x}))$. The total cost is $I[\pi] = \int_{X\times Y} c(\boldsymbol{x}, \boldsymbol{y})\mathrm{d}\pi(\boldsymbol{x}, \boldsymbol{y})$. Kantorovich also introduced the *dual problem*, which is to maximize

$$\begin{aligned} J[\widetilde{\varphi}, \widetilde{\psi}] &= \int_X \widetilde{\varphi}\mathrm{d}\mu + \int_Y \widetilde{\psi}\mathrm{d}\nu \\ &= \int_{X\times Y} [\widetilde{\varphi}(\boldsymbol{x}) + \widetilde{\psi}(\boldsymbol{y})]\mathrm{d}\pi(\boldsymbol{x}, \boldsymbol{y}) \quad \text{if and only if } \pi \in \Pi(\mu, \nu) \end{aligned} \tag{2.6}$$

over the set Φ_c of measurable functions $(\widetilde{\varphi}, \widetilde{\psi})$ such that $\widetilde{\varphi}(\boldsymbol{x}) + \widetilde{\psi}(\boldsymbol{y}) \leq c(\boldsymbol{x}, \boldsymbol{y})$. He showed that indeed $\inf_{\Pi(\mu,\nu)} I[\pi] = \sup_{\Phi_c} J[\widetilde{\varphi}, \widetilde{\psi}]$, and this minimum cost is known as

the Wasserstein distance between μ and ν. Unlike the Monge-Ampere equation which is highly nonlinear in $\boldsymbol{t}$, the dual problem is linear in $\widetilde{\varphi}, \widetilde{\psi}$ so numerical methods can use the techniques of linear programming.

For the quadratic cost $c(\boldsymbol{x}, \boldsymbol{y}) = |\boldsymbol{x} - \boldsymbol{y}|^2/2$, a useful change of variables is

$$\varphi(\boldsymbol{x}) = \frac{1}{2}|\boldsymbol{x}|^2 - \widetilde{\varphi}(\boldsymbol{x}), \quad \psi(\boldsymbol{y}) = \frac{1}{2}|\boldsymbol{y}|^2 - \widetilde{\psi}(\boldsymbol{y}). \tag{2.7}$$

In terms of these the dual condition is

$$\varphi(\boldsymbol{x}) + \psi(\boldsymbol{y}) \leq \boldsymbol{x} \cdot \boldsymbol{y} \tag{2.8}$$

which is automatically satisfied if $\psi = \varphi^*(\boldsymbol{y})$, where $\varphi^*(\boldsymbol{y})$ is the *Legendre-Fenchel* (LF) transform (or *convex conjugate* function) of $\varphi(\boldsymbol{x})$:

$$\varphi^*(\boldsymbol{y}) = \sup_{\boldsymbol{x} \in X} \{\boldsymbol{y} \cdot \boldsymbol{x} - \varphi(\boldsymbol{x})\} \tag{2.9}$$

(see section 2.3 for further properties). When φ is convex and differentiable this reduces to the familiar Legendre transform:

$$\varphi(\boldsymbol{x}) + \varphi^*(\boldsymbol{y}) = \boldsymbol{x} \cdot \boldsymbol{y} \tag{2.10}$$

where $\boldsymbol{x}$ and $\boldsymbol{y}$ are related by

$$\boldsymbol{y} = \nabla\varphi(\boldsymbol{x}), \qquad \boldsymbol{x} = \nabla\varphi^*(\boldsymbol{y}). \tag{2.11}$$

Brenier proved that if μ is absolutely continuous (with respect to Lebesgue) then there is a unique optimal plan and it is of the form $\boldsymbol{t} = \nabla\varphi(\boldsymbol{x})$ for some convex function φ (in fact the theorem required some additional technical assumptions which have since been proved unnecessary). Obviously this only defines $\boldsymbol{t}$ uniquely where φ is differentiable but this will be the case $\mathrm{d}\mu$-almost everywhere. Furthermore if ν is absolutely continuous (with respect to Lebesgue) then $\boldsymbol{x} = \nabla\psi(\boldsymbol{y})$ $\mathrm{d}\nu$-almost everywhere, where $\psi^* = \varphi$, $\varphi^* = \psi$, and is the solution of the MP for transporting ν back to μ.

Brenier also provided the equivalent result that any absolutely continuous vector field $\boldsymbol{f}$ has a unique *polar factorisation* $\boldsymbol{f}(\boldsymbol{x}) = \nabla\varphi(\boldsymbol{r}(\boldsymbol{x}))$ where φ is convex and $\boldsymbol{r}$ is a rearrangement of X.

2.2 Example: the semi-geostrophic system

The semi-geostrophic system is a set of equations used in atmospheric modeling first introduced by Hoskins [41]. Starting from the compressible Navier-Stokes equations in a

3D rotating frame of reference, together with the first law of thermodynamics for a medium containing different phases of water and the ideal gas equation of state, approximations and transformations are made [23, sec 2.3] which result in the *semi-geostrophic equations*:

$$\begin{aligned} \tfrac{\mathrm{D}}{\mathrm{D}t}(X,Y,Z) &= f(y-Y, X-x, 0) \\ \nabla\cdot\mathbf{u} &= 0 \\ (X,Y,Z) &= \nabla P \\ P &= \tfrac{1}{2}(x^2+y^2)+f^{-2}\varphi_g \end{aligned} \tag{2.12}$$

for $\boldsymbol{x}=(x,y,z)$ in some domain Γ, where f is the fixed Coriolis parameter, φ_g is the geopotential and $\mathrm{D}/\mathrm{D}t = \partial/\partial\mathrm{t}+\mathbf{u}\cdot\nabla$ is the usual Lagrangian (convective) derivative. This constitutes a system of seven equations for the seven unknowns $(X,Y,Z,P,\mathbf{u})$. The energy of the system is $E=\int_\Gamma c(\boldsymbol{x})\mathrm{d}\boldsymbol{x}$ where

$$c(\boldsymbol{x}) = f^2[\frac{1}{2}(x-X)^2+\frac{1}{2}(y-Y)^2 - zZ].$$

Setting $\boldsymbol{X}=\boldsymbol{X}(\boldsymbol{x})=(X,Y,Z)$ in some domain Σ, the potential vorticity q is defined to be

$$q = \det\left(\frac{\partial \boldsymbol{X}}{\partial \boldsymbol{x}}\right) = \det(D^2P) \tag{2.13}$$

where D^2P is the Hessian of P. This is a Monge-Ampere equation (2.4) for P with $\alpha=q$ and $\beta=1$.

Cullen's [23, sec 3.1] stability principle asserts that physical states correspond to minima of the energy with respect to variations preserving q. This is equivalent to saying they are solutions to an MKP with cost $c(\boldsymbol{x})$ from which we can deduce the existence and uniqueness of a physical state for a given q, characterised by a convex potential $P(\boldsymbol{x})$. We define $R(\boldsymbol{X})$ to be its Legendre transform:

$$R(\boldsymbol{X}) + P(\boldsymbol{x}) = \boldsymbol{x}\cdot\boldsymbol{X} \tag{2.14}$$

and then $\boldsymbol{x}=\nabla R$. Also

$$\begin{aligned} c(\boldsymbol{x}) &= f^2[\tfrac{1}{2}(x^2+y^2)+\tfrac{1}{2}(X^2+Y^2)-\boldsymbol{x}\cdot\boldsymbol{X}] \\ &= f^2[\tfrac{1}{2}(x^2+y^2)-P(\boldsymbol{x})+\tfrac{1}{2}(X^2+Y^2)-R(\boldsymbol{X})] \\ &= \widetilde{\varphi}(\boldsymbol{x})+\widetilde{\psi}(\boldsymbol{X}) \end{aligned}$$

where $\widetilde{\varphi}(\boldsymbol{x}) = -\varphi_g(\boldsymbol{x})$ and $\widetilde{\psi}(\mathbf{X}) = f^2[\frac{1}{2}(X^2+Y^2)-R(\mathbf{X})]$, which with $\boldsymbol{X}\to\boldsymbol{y}$ recovers the original MKP.

The remaining content of (2.12) is $\mathrm{D}q/\mathrm{D}t=0$ - thus the semi-geostrophic equations split into two parts: a fixed-time MKP coupled to a Lagrangian time-evolution equation for q.

2.3 The Legendre-Fenchel transform

Some properties of the LF transform (equation 2.9) are (see e.g. Rockafeller [77]):

(a) φ^* is always convex (and lower semi-continuous).

(b) φ^{**} is the *convex hull* of φ. If φ is convex (and lower semi-continuous) then $\varphi^{**} = \varphi$.

(c) As φ^* is always convex (and lower semi-continuous), $\varphi^{***} = \varphi^*$. In words the transform of a function is identical to the transform of its convex hull, which means that any non-convex parts of a function are eliminated by the transform.

(d) Linear functions (hyperplanes) are transformed into points. Convex piecewise linear functions are transformed into convex piecewise linear functions.

(e) Where functions are convex and differentiable the transform reduces to the Legendre transform - equations (2.10) and (2.11).

The following examples illustrate these properties ($\psi = \varphi^*$).

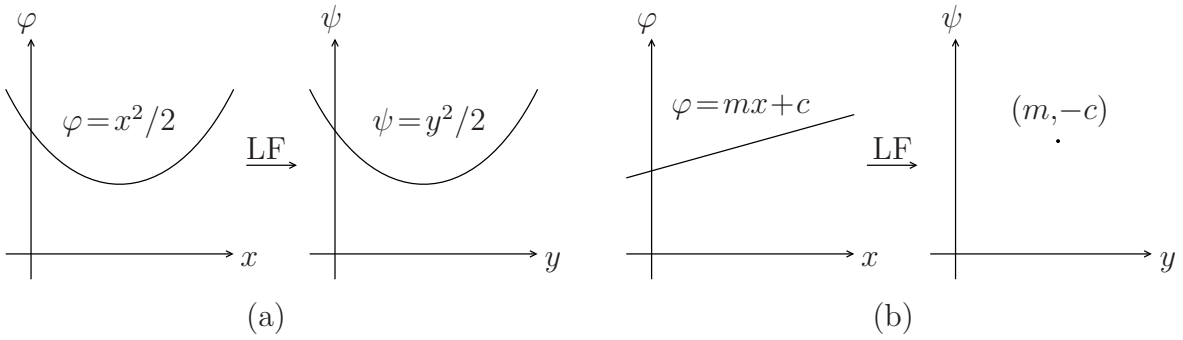


Figure 2.2: LF transform of convex differentiable, linear functions

1) Convex, differentiable φ (Fig 2.2a). If $\varphi = x^2/2$, (2.11) $\Rightarrow y = \varphi'(x) = x$, then (2.10) $\Rightarrow \psi(y) = xy - \varphi(x) = y^2/2$.

2) Linear φ (Fig 2.2b). If $\varphi(x) = mx + c$, (2.11) $\Rightarrow y = \varphi'(x) = m$. When $y = m$, (2.10) $\Rightarrow \psi(m) = xy - \varphi(x) = -c$, but this is not applicable for other values of y. In fact (2.9) implies $\psi(y)$ is not finite elsewhere. Thus the line $mx + c$ is transformed into the point $(m, -c)$.

3) Linear φ through a point (Fig 2.3a). If $\varphi(x) = \varphi_0 + m(x - x_0)$ then φ passes through the point v at (x_0, φ_0). Four different values for m are shown. Again (2.11) $\Rightarrow y = \varphi'(x) = m$ but now $\psi(m) = xy - \varphi(x) = mx_0 - \varphi_0$. The points $(m, mx_0 - \varphi_0)$ all lie on a line of slope x_0, and this line transforms back into the point v.

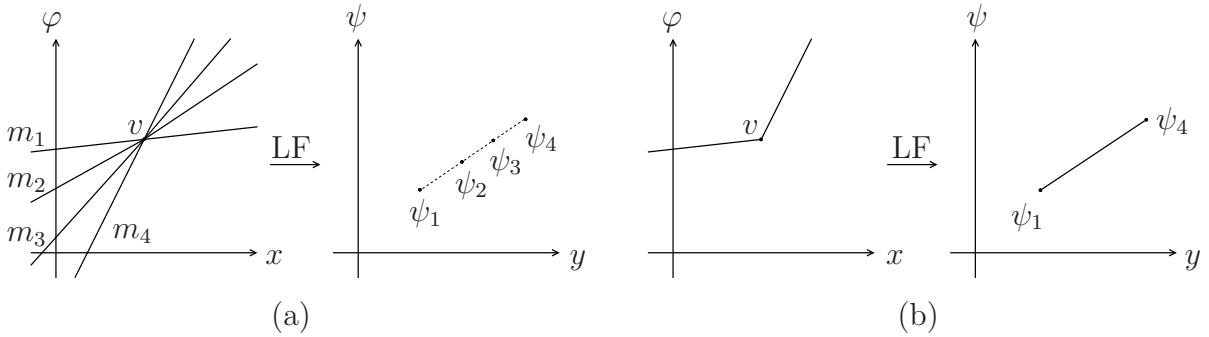


Figure 2.3: LF transform of linear functions, points

4) Piecewise linear φ with a vertex v (Fig 2.3b). Applying (2.9) directly only yields finite values for ψ when $m_1 \leqslant y \leqslant m_4$, and then $\psi(y) = yx_0 - \varphi_0$; elsewhere $\psi(y) = +\infty$.

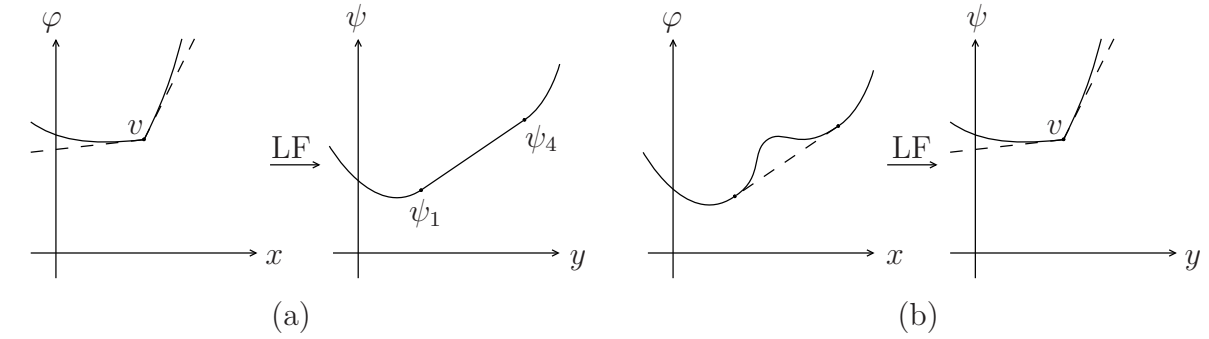
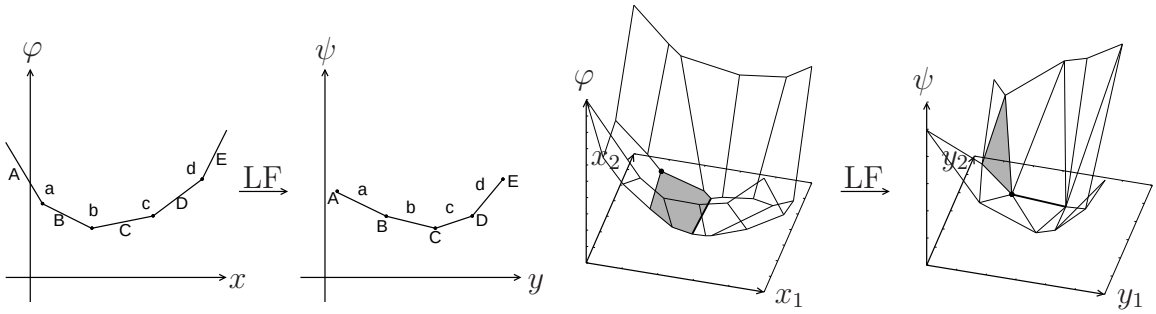


Figure 2.4: LF transform of convex non-differentiable, non-convex differentiable functions

5) When φ is convex and piecewise differentiable (Fig 2.4a), its transform combines the transform in the previous example with the transform of the differentiable sections. Because φ is convex, $\psi^* = \varphi^{**} = \varphi$ i.e. ψ transforms back into φ. Compare this with the case when the function contains non-convex sections (Fig 2.4b) - the supremum in (2.9) results in these sections being lost.

6) A convex piecewise linear function φ (Fig 2.5a) is transformed into a convex piecewise linear function ψ. The linear pieces (faces) are labelled A to E, and φ can be defined as the supremum of the lines created by extending each face out to infinity at both ends. Taken individually, these lines are transformed into correspondingly labelled vertices in ψ. Example (4) showed that the supremum of the two lines A and B is transformed into the face a in ψ. Similarly the suprema of the pairs of lines B/C, C/D and D/E are transformed into the faces b, c and d respectively. This establishes a duality between vertices and faces under the transform.



(a) (b)

Figure 2.5: LF transform of piecewise linear, piecewise planar functions

7) This duality extends naturally to $\boldsymbol{x} \in \mathbb{R}^2$, as shown in (Fig 2.5b). Each face of φ (e.g. the one shaded) can be uniquely identified with a vertex in ψ (the marked vertex), and each vertex in φ (e.g. the one marked) can be identified with a face in ψ (shaded). Also each edge between adjacent faces of φ (e.g. the thick edge) can be identified with an edge in ψ (also thick).

2.4 Discrete versions of the mass transport problem

When the cost is quadratic but μ, ν are not absolutely continuous then α, β (2.3) are distributions not functions so Brenier's result is not applicable. However an optimal plan can still exist, and these cases are of practical importance so are dealt with separately. For these we adopt the semi-geostrophic notation, $\boldsymbol{x} \in \Gamma$ and $\boldsymbol{y} \to \boldsymbol{X} \in \Sigma$.

2.4.1 Point distributions

Here α consists of m point masses α_i (>0) at points $\boldsymbol{x}_i$ ($1 \leqslant i \leqslant m$) i.e. $\alpha(\boldsymbol{x}) = \sum_{i=1}^m \alpha_i \delta(\boldsymbol{x} - \boldsymbol{x}_i)$, and β consists of n point masses β_j (>0) at points $\boldsymbol{X}_j$ ($1 \leqslant j \leqslant n$) i.e. $\beta(\boldsymbol{X}) = \sum_{j=1}^n \beta_j \delta(\boldsymbol{X} - \boldsymbol{X}_j)$, and the equal volume criterion is $\sum_i \alpha_i = \sum_j \beta_j$. The transport plan is a matrix where π_{ij} (>0) is the amount of mass to be moved from $\boldsymbol{x}_i$ to $\boldsymbol{X}_j$. If the cost incurred moving a unit of mass from $\boldsymbol{x}_i$ to $\boldsymbol{X}_j$ is c_{ij}, then the total cost to be minimised is $I[\pi] = \sum_{ij} c_{ij}\pi_{ij}$ and the rearrangement criterion becomes $\sum_j \pi_{ij} = \alpha_i$, $1 \leqslant i \leqslant m$, and $\sum_i \pi_{ij} = \beta_j$, $1 \leqslant j \leqslant n$.

For example if $m = n$ and $\alpha_i = \beta_j = 1$, $1 \leqslant i, j \leqslant n$, π is bi-stochastic matrix and the minimiser is a permutation [91], that is $\pi_{ij} = \delta_{i,\sigma(j)}$ for some permutation σ of $\{1, \cdots, n\}$ which corresponds to finding an optimal matching between the points $\boldsymbol{x}_i$ and $\boldsymbol{X}_j$ e.g. as in Fig 2.6 for $\mathbb{R}^2$.

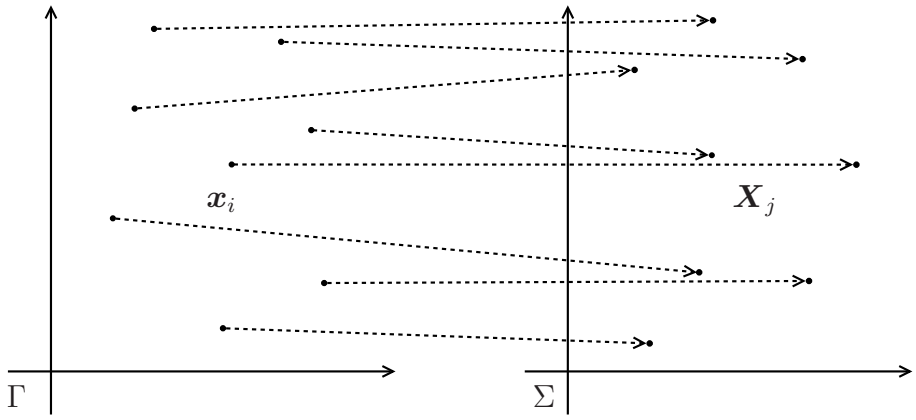


Figure 2.6: Monge problem with discrete unit masses

The dual problem is to maximise $J[\varphi, \psi] = \sum_i \varphi_i \alpha_i + \sum_j \psi_j \beta_j$ subject to the constraints $\varphi_i + \psi_j \leqslant c_{ij}$. This is a linear programming problem and so can be solved by e.g. the simplex method [73].

2.4.2 Constant distribution to point distribution

Let β be as before, but Γ now be a convex polyhedron in which $\alpha = 1$ (Fig 2.7). In other words the sand is initially spread evenly throughout Γ and is to be gathered into a set of containers of capacity β_i at points $\boldsymbol{X}_i$ $(1 \leqslant i \leqslant n)$, the equal volume condition requiring $\sum_i \beta_i = |\Gamma|$. This will be called the *Semi-discrete Monge-Kantorovich Problem* or SMKP.

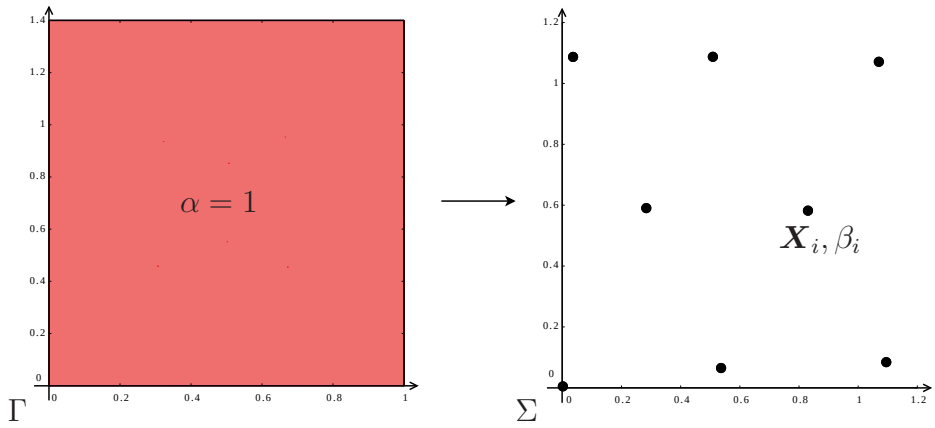


Figure 2.7: Monge problem gathering uniform distribution to a set of points

As $n \to \infty$ the SMKP can approximate ever more closely a continuum MKP, which suggests the solution to the SMKP will also be of the form $\boldsymbol{t} = \nabla P$ for convex P. Suppose so. As every point in Γ must be transported to one of the $\boldsymbol{X}_i$ then for any

$\boldsymbol{x} \in \Gamma$, $\boldsymbol{t}(\boldsymbol{x}) = \nabla P(\boldsymbol{x}) = \boldsymbol{X}_i$ for some i, so P must be piecewise linear e.g. as Fig 2.8a. From example 7 in the previous section, R is also piecewise linear (Fig 2.8b).

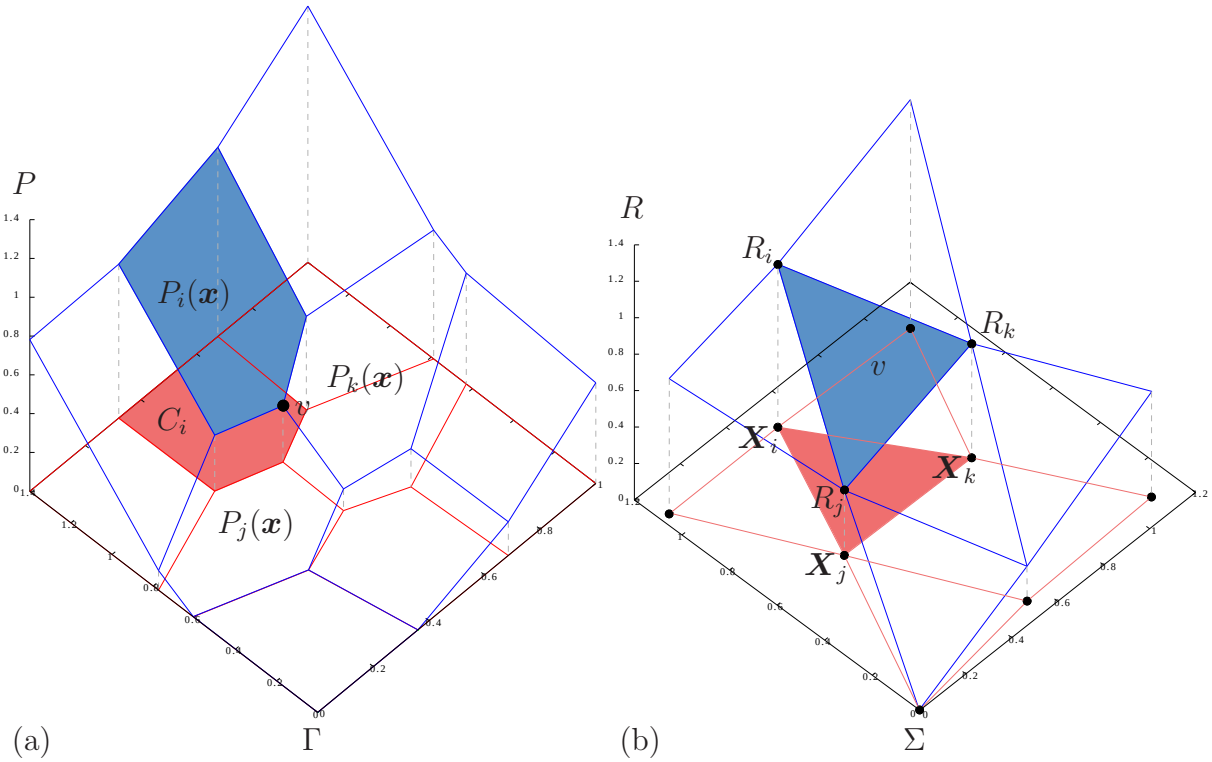


Figure 2.8: Piecewise linear $P(\boldsymbol{x})$ (blue), cells (red), and dual surface $R(\boldsymbol{X})$

Integrating $\boldsymbol{t}(\boldsymbol{x})$ gives the equation of the plane forming face i of P (blue in Fig 2.8a):

$$P_i(\boldsymbol{x}) = \boldsymbol{x} \cdot \boldsymbol{X}_i - R_i, \tag{2.15}$$

where R_i is a constant. As P is convex, it can be defined as the supremum over planes i:

$$P(\boldsymbol{x}) = \sup_{i=1}^{n} \{P_i(\boldsymbol{x})\} \tag{2.16}$$

and the projection of face i onto Γ defines the cell C_i (red in Fig 2.8a):

$$C_i = \{\boldsymbol{x} : P_i(\boldsymbol{x}) > P_j(\boldsymbol{x}), j \neq i\}, \tag{2.17}$$

with area $A_i = |C_i|$. Within C_i, P is differentiable so LF-transform property (e) applies:

$$R(\boldsymbol{X}) + P(\boldsymbol{x}) = \boldsymbol{x} \cdot \boldsymbol{X}, \qquad \boldsymbol{x} \in C_i. \tag{2.18}$$

Comparing this with (2.15) we see $R(\boldsymbol{X}_i) = R_i$ i.e. C_i is transported to the point $(\boldsymbol{X}_i, R_i)$ in the dual space Σ (Fig 2.8b). To establish that faces in Σ are indeed transformed back into vertices in Γ consider the vertex v in Fig 2.8a where the three faces $P_i(\boldsymbol{x}), P_j(\boldsymbol{x}), P_k(\boldsymbol{x})$ meet. If the vertex has coordinates $(\boldsymbol{x}_v, P_v)$ then it satisfies (2.15) for each of the three

planes:

$$P(\boldsymbol{x}_v) = P_v = \begin{cases} \boldsymbol{X}_i \cdot \boldsymbol{x}_v - R_i \\ \boldsymbol{X}_j \cdot \boldsymbol{x}_v - R_j \\ \boldsymbol{X}_k \cdot \boldsymbol{x}_v - R_k \end{cases} \text{ or } \begin{pmatrix} X_i & Y_i & -1 \\ X_j & Y_j & -1 \\ X_k & Y_k & -1 \end{pmatrix} \begin{pmatrix} x_v \\ y_v \\ P_v \end{pmatrix} = \begin{pmatrix} R_i \\ R_j \\ R_k \end{pmatrix}. \tag{2.19}$$

where $\boldsymbol{x} = (x, y), \boldsymbol{X} = (X, Y)$. The determinant of the matrix is $2A_{ijk}$, where A_{ijk} is the area of the triangle with vertices $\boldsymbol{X}_i$,$\boldsymbol{X}_j$, and $\boldsymbol{X}_k$ in Σ (coloured red in Fig 2.8b) :

$$A_{ijk} \stackrel{\text{def}}{=} \frac{1}{2}\left[(X_j - X_i)(Y_k - Y_i) - (Y_j - Y_i)(X_k - X_i)\right]. \tag{2.20}$$

Provided $A_{ijk} \neq 0$, (2.19) can be inverted to obtain $(\boldsymbol{x}_v, P_v)$. Alternatively the equation of a plane in dual space is of the form $R(\boldsymbol{X}) = \boldsymbol{x}_v \cdot \boldsymbol{X} - P_v$ for constants $\boldsymbol{x}_v, P_v$. The plane which passes through the points $(\boldsymbol{X}_i, R_i), (\boldsymbol{X}_j, R_j), (\boldsymbol{X}_k, R_k)$ must satisfy the same equations as (2.19) and so this $\boldsymbol{x}_v, P_v$ must be the same as before i.e. the vertex v is LF-transformed into the face v in Fig 2.8b as expected.

Lastly for P to solve the SMKP, it must satisfy the rearrangement criteria (2.1), (2.2). If B is some neighbourhood of $\boldsymbol{X}_i$ sufficiently small to contain no other $\boldsymbol{X}_j$ then (2.1) becomes

$$A_i = \beta_i \qquad i = 1, \ldots, n. \tag{2.21}$$

Thus Γ is decomposed into a set of convex polyhedral cells $\{C_i\}$, where C_i has area β_i and is transported to the point $\boldsymbol{X}_i$ (Fig 2.9). In the process the set $\{\boldsymbol{X}_i\}$ is also triangulated.

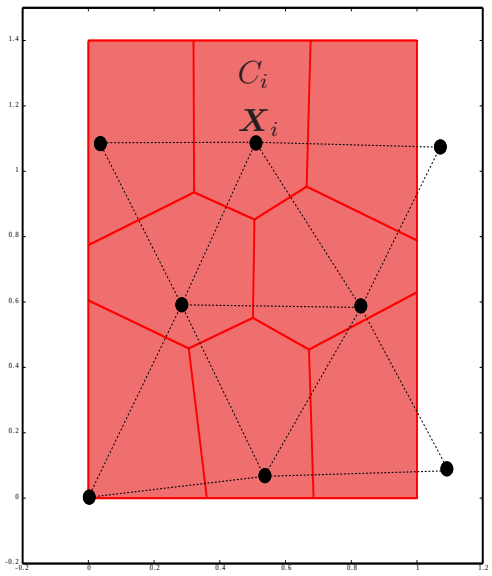


Figure 2.9: Convex polygons and triangulation created by the transport solution

To summarize, the solution of the SMKP, if it exists, is a convex polyhedral function P whose faces have gradients $\boldsymbol{X}_i$ and projected areas β_i, for $i = 1, \ldots, n$. Aleksandrov (described in Pogorelov [71]) first proved a solution exists and is unique, as independently did Cullen and Purser [24] when developing a numerical method for the solution of the semi-geostrophic equations.

2.5 Other applications

The Monge problem is so basic that it should be no surprise to find it cropping up elsewhere. The following sections describe applications of the Monge problem for which numerical solutions have been developed.

2.5.1 Optical reflector design

A *far field optical reflector* [67] (Fig 2.10) is a surface R in $\mathbb{R}^n$ that transforms a given spherical light source at the origin O which has intensity $I(\boldsymbol{m})$ in the direction $\boldsymbol{m}$ into a given intensity $L(\boldsymbol{y})$ in the direction $\boldsymbol{y}$ far away.

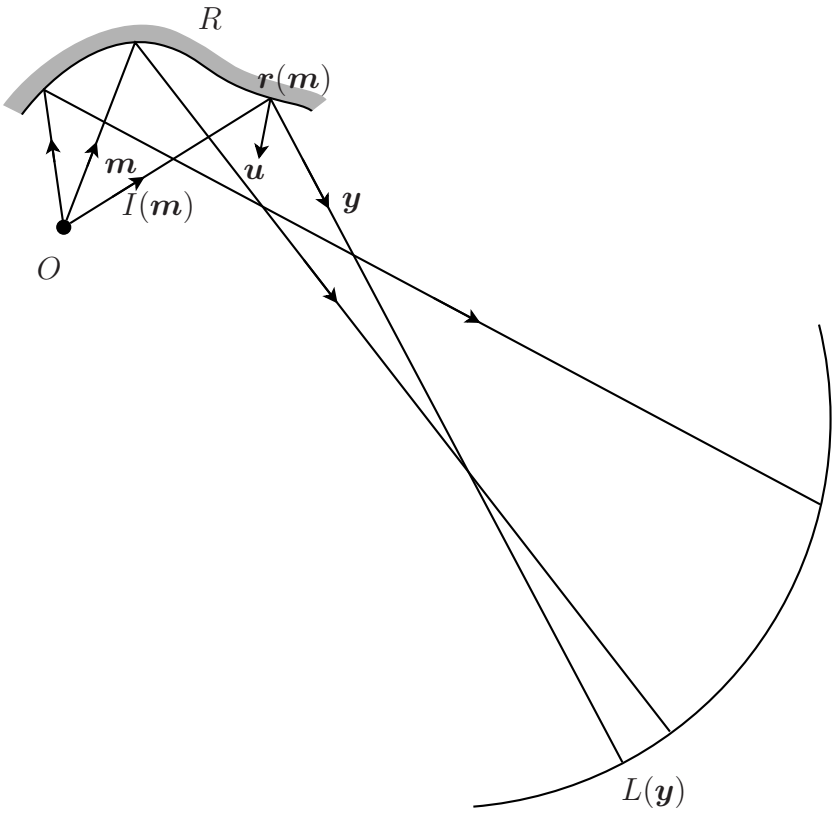


Figure 2.10: A far field optical reflector R

A ray $\boldsymbol{m}$ hits the surface R at $\boldsymbol{r}(\boldsymbol{m})$ where the normal is $\boldsymbol{u}$ and is reflected into the

direction $\boldsymbol{y}$. The map from $\boldsymbol{m}$ to $\boldsymbol{y}$ is given by Snell's law $\gamma : \boldsymbol{m} \mapsto \boldsymbol{y} = \boldsymbol{m} - 2\langle \boldsymbol{m}, \boldsymbol{u} \rangle \boldsymbol{u}$, where $\langle \cdot, \cdot \rangle$ denotes the inner product. When γ is a diffeomorphism, the intensity $L(\boldsymbol{y})$ must satisfy the Monge-Ampere equation $L(\gamma(\boldsymbol{m})) \det(D^2\gamma(\boldsymbol{m}))) = I(\boldsymbol{m})$. Solutions to this pointwise equation are called strong solutions, whereas solutions to the corresponding MKP are called weak solutions. Oliker and Kochengin [69] developed an algorithm, the *method of supporting paraboloids*, to find numerical solutions by approximating R by the envelope of a finite number of paraboloids, each of which has its focus at O and so reflects all the rays hitting it in the same direction. This is a SMKP where $\boldsymbol{m} \leftrightarrow \boldsymbol{x}, X = Y = S^2, I \leftrightarrow \alpha, L \leftrightarrow \beta, \gamma \leftrightarrow \boldsymbol{t}, \log(\rho) \leftrightarrow \widetilde{\varphi}, -\log(1 - \langle \boldsymbol{m}, \boldsymbol{y} \rangle) \leftrightarrow c(\boldsymbol{x}, \boldsymbol{y})$ [68], and the construction ensures that γ is the gradient of a convex function. The N paraboloids are iteratively adjusted until the required intensities are satisfied to within tolerance i.e. the rearrangement criterion (2.1) is met. The original method performed poorly for large N, but was improved by switching to the downhill simplex method [66] after an initial number of iterations.

Some related problems can also be formulated in terms of optimal transportation. A *near field* reflector illuminates a given hypersurface with a specified intensity field and can be solved numerically by decomposing R into sections of ellipsoids with one focus at O and the other on the hypersurface. Rearranging the intensity of a set of parallel rays, or generating parallel rays of given intensity from a point source are not possible with a single reflector as it does not have enough degrees of freedom but both are achievable using two successive reflectors (Fig 2.11). These cases also have parallels in antenna design.

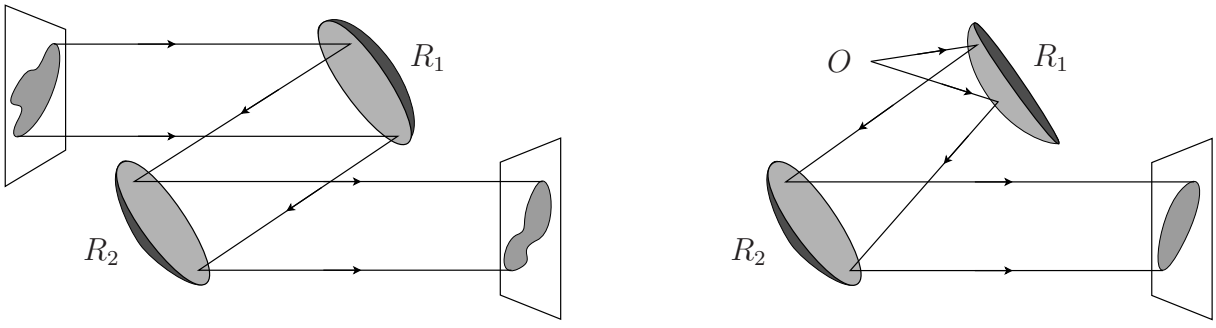


Figure 2.11: Some two-reflector systems

2.5.2 Image registration and warping

Image registration is the process of establishing a common geometric reference frame between two or more image data sets possibly taken at different times and warping is the

process of finding an explicit map from one image to another. Images here are rectangular arrays of pixels, each of which has a positive real intensity (and often colour information, but that is not considered here). An example application is in medicine, where different datasets e.g. X-rays and MRI scans need to be overlayed for diagnosis or pre-operative planning. Another example is comparing a series of scans of a tumour during the course of therapy. There are numerous different approaches to this task, ranging from methods using optical flow, statistics, computational fluid dynamics and warping methodologies. Typically they start by defining a function that quantifies the difference between two images which might include the differences between intensity values or the distance between specific features (e.g. surface contours or other landmarks) once located on each image. Then the transformation is found that minimises this e.g. by solving some optimisation problem. Haker *et al* [37] suggested using the Wasserstein distance (section 2.1) with quadratic cost as the difference function and then the registration problem becomes a MKP.

Unlike Oliker's algorithm and the geometric method (chapter 3), which vary a convex function until the rearrangement criteria is met, their algorithm varies rearrangements until the map is a gradient of a convex function. Starting from an initial rearrangement u^0 of image μ_0 in domain $\Omega_0 \subset \mathbb{R}^n$ onto image μ_1 in Ω_1, new rearrangements u are generated by $u = u^0 \circ s^{-1}$, where s is a rearrangement of μ_0. Initially s is set to the identity map but is iteratively updated via the steepest descent direction of the Wasserstein distance $M = \int_{\Omega_0} \|u(x) - x\|^2 \mu_0(x) \mathrm{d}x$ until the unique global minimum is found.

If s and u are parameterized by t, they show that rearrangements are generated by $s_t = \left(\frac{1}{\mu_0}\zeta\right) \circ s$, $u_t = -\frac{1}{\mu_0} \det(Du)\zeta$, where ζ is any vector field such that $\nabla \cdot \zeta = 0$, and $\langle \zeta, \vec{n} \rangle = 0$ on $\partial\Omega_0$ (to ensure that $s(\Omega_0) = \Omega_0$). Also $M_t = -2 \int_{\Omega_0} \langle \zeta, \chi \rangle \mathrm{d}x$ where $u = \nabla w + \chi$ is the unique Helmholtz decomposition of u ($\nabla \cdot \chi = 0$). Thus the steepest descent direction is $\zeta = \chi$. To effect the decomposition requires solving the Poisson equation $\Delta w = \nabla \cdot u$ for w, with Neumann-type boundary condition $\langle \nabla w, \vec{n} \rangle = \langle u, \vec{n} \rangle$ on $\partial\Omega_0$, from which $\chi = u - \nabla w$. This can be done efficiently by e.g. the multigrid method [76] so the algorithm is expected to be of comparable computational efficiency with Oliker's algorithm and the geometric method.

Chartrand *et al* [21] have developed an algorithm for the quadratic cost which computes the optimal transport plan by maximising the dual cost $J[\widetilde{\varphi}, \widetilde{\psi}]$ (2.6) by steepest

descent. With the change of variables (2.7), this is equivalent to minimising

$$L[\varphi,\psi]=\int_X \varphi(\boldsymbol{x})\mathrm{d}\mu(\boldsymbol{x})+\int_Y \psi(\boldsymbol{y})\mathrm{d}\nu(\boldsymbol{y}) \tag{2.22}$$

under the condition (2.8). They impose $\psi=\varphi^*$ by defining

$$M[\varphi]=\int_X \varphi(\boldsymbol{x})\alpha(\boldsymbol{x})\mathrm{d}\boldsymbol{x}+\int_Y \varphi^*(\boldsymbol{y})\beta(\boldsymbol{y})\mathrm{d}\boldsymbol{y},$$

and prove the variational derivative exists, being

$$M'[\varphi]=\frac{\delta M[\varphi]}{\delta\varphi}=\alpha-\beta(\nabla\varphi^{**})\det(D^2\varphi^{**}) \tag{2.23}$$

where $D^2\varphi^{**}$ is the absolutely continuous part of the Hessian of φ^{**}. Assuming φ is convex ($\varphi^{**}=\varphi$), at the minimum $M'[\varphi]=0$ which recovers the Monge-Ampere equation (2.4). The steepest descent method in some pseudo-time parameter t is

$$\varphi_t=-M'[\varphi]=\beta(\nabla\varphi)\det(D^2\varphi)-\alpha. \tag{2.24}$$

For the image warping problem, α and β are approximated by arrays of greyscale pixel intensities and the discretised descent step for pseudo-timestep $\triangle t_n$ is

$$\varphi_{n+1}=\varphi_n+\triangle t_n[\beta(\nabla\varphi_n)\det(D^2\varphi_n)-\alpha]. \tag{2.25}$$

They use a Lax-type numerical scheme, evaluating φ at pixel vertices and the derivatives at pixel centres by centred differencing. The second term in (2.25) is computed at pixel centres and φ_{n+1} updated from φ_n by averaging over pixel centres. This was found to produce numerical artifacts which worsened on further iteration, so instead they Gaussian-smoothed α and β first and iterated φ, then smoothed α and β to a lesser degree and continued iterating, and so on, eventually using the original α and β. In practice they found just one change of smoothing produced good results.

2.5.3 The Parabolic Monge-Ampere method

For mesh movement we identify X with computational space Ω_c, Y with physical space Ω and solve the Monge problem for $\alpha=$ *constant*. The Monge-Ampere equation (2.4) now expresses the equidistribution of β ((1.12) with $J\equiv\det(D^2\varphi)$, $\rho\equiv\beta$ and $\sigma\equiv\alpha$). This is also the continuous analogue of the SMKP. The additional requirement of optimal transportation singles out a unique equidistribution regardless of the number of dimensions. Furthermore the form of the optimal transport plan ($\boldsymbol{y}=\nabla\varphi$) also means that $\nabla\times\boldsymbol{y}=0$ in $\mathbb{R}^2$ and $\mathbb{R}^3$ i.e. the mapping is *irrotational*.

To eliminate α from (2.24) we set $Q = \varphi + \alpha t$ so that the steepest descent is

$$Q_t = \rho(\nabla Q, t)\det(D^2 Q).$$

Budd and Williams [16] observe that this is a parabolic Monge-Ampere equation for Q [59], and prove that it converges to the solution of the MP when the monitor is constant in time. In practical applications, it is usually necessary to smooth Q [42], typically by local averaging. The *Parabolic Monge-Ampere method* uses the Laplacian to smooth:

$$\epsilon(I - \gamma\Delta)Q_t = [\rho(\nabla Q, t)\det(D^2 Q)]^{1/d} \tag{2.26}$$

for $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$, where ϵ and γ are constants controlling the relaxation rate and degree of spatial smoothing and the power $1/d$ is introduced for scale invariance.

2.5.4 Power diagrams and interface reconstruction

Power diagrams are a variant of the Voronoi construction which has been used in a recent interface reconstruction algorithm.

Given a set of *sites* $\{\boldsymbol{X}_i\} \in \mathbb{R}^d$, the *Voronoi region* [5] associated with site $\boldsymbol{X}_i$ is the set of points closest to $\boldsymbol{X}_i$: $VR(\boldsymbol{X}_i) = \{\boldsymbol{x} : \mathrm{d}(\boldsymbol{x}, \boldsymbol{X}_i) < \mathrm{d}(\boldsymbol{x}, \boldsymbol{X}_j), \forall j \neq i\}$ where the metric $\mathrm{d}(\boldsymbol{x}, \boldsymbol{y}) = |\boldsymbol{x} - \boldsymbol{y}|$ is the Euclidian distance. The Voronoi regions are polyhedral and their boundaries together form the Voronoi diagram. They can be generalized by choosing different metrics - for *power diagrams* the metric is

$$\mathrm{d}(\boldsymbol{x}, \boldsymbol{y}) = |\boldsymbol{x} - \boldsymbol{y}|^2 - w(\boldsymbol{y})$$

where w is a weight function, which is only evaluated at the sites so can be replaced by the set $\{w_i\}$ where $w_i = w(\boldsymbol{X}_i)$. Power regions are also polyhedral, and reduce to the Voronoi regions when the w_i are identical. As w_i increases with respect to its neighbours, the associated power region $PR(\boldsymbol{X}_i)$ increases in size. For a positive weight the metric can be interpreted as the tangential distance (squared) to a ball of radius $\sqrt{w_i}$.

If we associate with each point $\boldsymbol{X}_i$ the plane $\varphi_i(\boldsymbol{x}) = \boldsymbol{x} \cdot \boldsymbol{X}_i - \psi_i$ in $\mathbb{R}^{d+1}$, where $\psi_i = (|\boldsymbol{X}_i|^2 - w_i)/2$ then the power region of $\boldsymbol{X}_i$ becomes simply $PR(\boldsymbol{X}_i) = \{\boldsymbol{x} : \varphi_i(\boldsymbol{x}) > \varphi_j(\boldsymbol{x}), \forall j \neq i\}$. In other words the power diagram is the projection of the convex hull of the planes, $\varphi(\boldsymbol{x}) = \sup_i\{\varphi_i(\boldsymbol{x})\}$, onto $\varphi = 0$. Alternatively any convex polyhedral function when projected down yields a power diagram. The *polar point* [5] associated with the plane φ_i is $(\boldsymbol{X}_i, \psi_i)$ so the polarity transform is actually a Legendre transform

[82]. Furthermore the solution to the SMKP is thus a power diagram which also satisfies the rearrangement criteria (2.21), providing an existence and uniqueness proof for power diagrams with specified areas in a domain with a polyhedral boundary.

In multi-material simulations of fluids some mechanism is required to keep track of the locations of the different materials. A popular choice is the *Volume of Fluid* (VOF) method [40] in which each cell maintains information on which materials are present and what fraction of the volume each takes up. Each timestep the interfaces between the materials in each cell are reconstructed using the volume fractions in the cell and surrounding cells. Typically these interfaces are linear (planar in 3D). More recently the *Moment of Fluid* (MOF) method [2] maintains centroid information in addition to the volume fractions to reconstruct the interfaces without using information from surrounding cells, but the interfaces are still piecewise linear/planar. Schofield *et al.* [80] instead construct a power diagram using the centroids as sites and varying the weights with Newton's method until the volume fractions match.

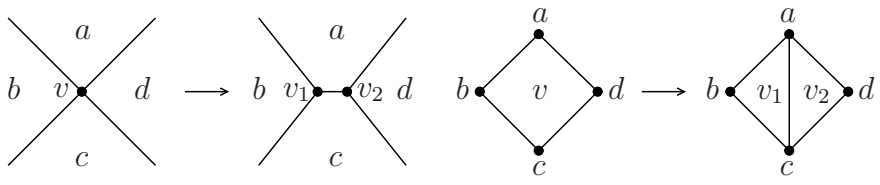
This concludes the summary of the basic theory and existing numerical techniques, although the list is likely to grow as new areas of application are found.

Chapter 3

The Geometric Method and Panel Beater algorithm

In this chapter we describe in detail some algorithms for the numerical solution of the optimal transportation problem, and investigate their performance in some relevant test problems. The *geometric method* was developed to solve the optimal transport component of the semi-geostrophic equations (2.13), and approximates the MP by a SMKP (which has the additional benefit that the remaining time evolution component becomes particularly simple - the geopotential q is conserved in each cell). Chynoweth [22] solves this with the multi-dimensional Newton-Raphson technique - each iteration constructing the dual convex polyhedral potentials $P(\boldsymbol{x})$ and $R(\boldsymbol{X})$ (2.12), (2.14) from scratch. The method has been implemented in two and three dimensions but is quite general, taking $O(N^2)$ operations for N cells. Purser has improved the geometric method by using a convex hull algorithm to construct the initial convex potentials in just $O(N \log N)$ operations, adjusting the potentials with the faster conjugate gradient method, and repairing the convexity if needed each iteration with the *panel beater* algorithm in on average $O(N)$ operations. This has been implemented in two dimensions and could in principle be extended to higher dimensions.

We identify Γ with physical space Ω and Σ with computational space Ω_c. As P and R are dual, one can be completely determined from the other, and we choose R to be the independent variable from which P is derived. Now R is the convex hull of N vertices $\{(\boldsymbol{X}_i, R_i)\}$ in $\mathbb{R}^{n+1}$ and an arbitrary set of points in $\mathbb{R}^{n+1}$ is said to be *in general position* if no n+2 points lie on an n-dimensional hyperplane, otherwise they are *in special position*. Point sets in special position form a measure zero subset of all point sets, in other words a random set of points is *almost always* in general position. In this case (n=2) all faces of R will be triangular (furthermore as R is convex the triangulation is *regular* [78]) and by duality exactly three edges meet at every vertex of P. A general position can always be recovered by moving offending vertices an infinitesimal distance off the hyperplane (illustrated by Fig 3.1) so this is assumed from here on. The domain boundary can be



Physical space Dual space

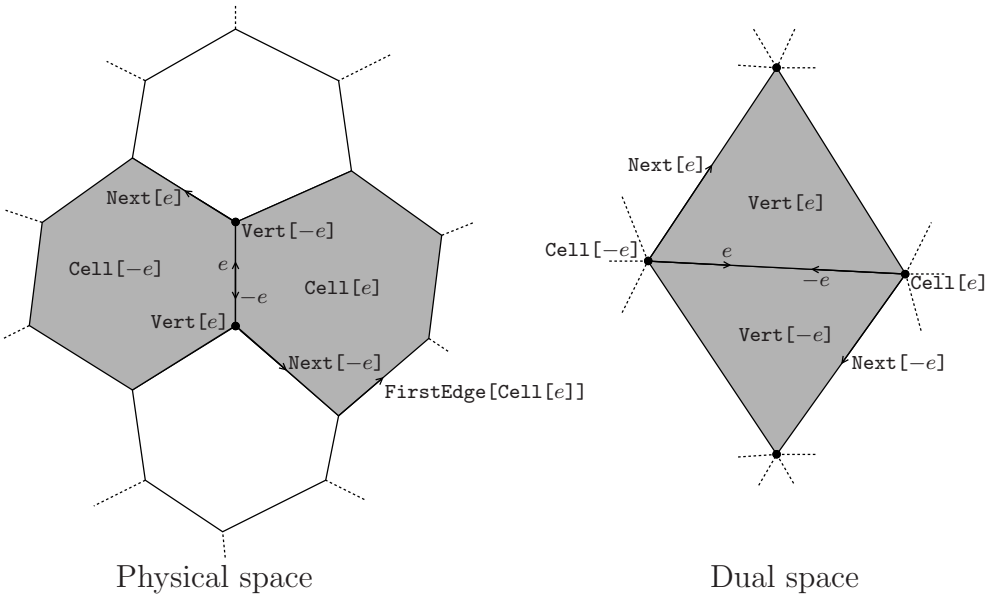
Figure 3.1: Recovering a general position

included quite naturally in this representation as cells $N+1, \ldots, N_C$ having dual space vertices at infinity (in practice a finite but large distance from the origin).

3.1 Description of data structures

Before describing their algorithms in detail, we introduce the data structures used to represent R and P, which are based on the *doubly-connected edge list* (DCEL) [11] for simply-connected polyhedra and consist of (Fig 3.2):

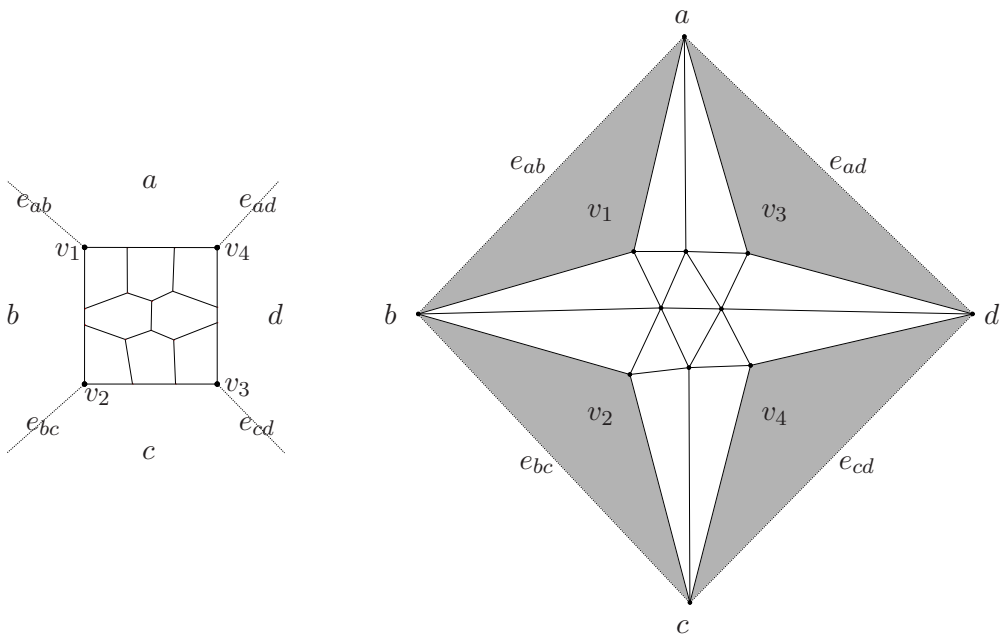
- N_V vertices in physical space at coordinates (`x[`v`]`,`y[`v`]`,`p[`v`]`), $1 \leqslant v \leqslant N_V$.
- $2N_E$ directed edges numbered -N_E to -1, 1 to N_E.
  Edge e starts at vertex `Vert[`e`]`, has cell number `Cell[`e`]` on its right hand side, and the next edge travelling anti-clockwise around the cell perimeter is `Next[`e`]`. Edge $-e$ overlays e but travels in the opposite direction.
- N_C cells with dual space coordinates (`X[`c`]`,`Y[`c`]`,`R[`c`]`), $1 \leqslant c \leqslant N_C$. `FirstEdge[`$c$`]` is some edge on the perimeter of cell c.



Physical space Dual space

Figure 3.2: Data representation for mesh

For each boundary vertex an extra edge is required - in Fig 3.3 the example of Fig 2.9 is extended by edges e_{ab}, e_{bc}, e_{cd} and e_{ad} which complete the representation of vertices v_1 to v_4 respectively (shaded), where boundary cells a, b, c and d are a large distance from the interior cells in dual space.



Physical space Dual space

Figure 3.3: Data representation for boundary cells

In dual space there are in total N_C vertices, N_E edges and N_V triangular faces, of which (N_C-N+1) vertices and (N_C-N+1) edges lie on the boundary. Summing three edges for every triangle counts the internal edges twice and the external edges once so $3N_V=2N_E+(N_C-N+1)$. Euler's theorem for a plane graph $(V-E+F=1)$ here reads $N_C-N_E+N_V=1$ and combining these two expressions yields

$$N_V = 3N + 4(N_C - N) - 2, \quad N_E = 2N + 3(N_C - N) - 1. \tag{3.1}$$

3.2 The Geometric Method

3.2.1 Construction of the potential P

The overall strategy is to find a cell on the domain boundary and trace round its edges to find its vertices and neighbouring cells. Then by tracing round these neighbours more vertices and neighbours are found, and so on spreading out until all the cells have been constructed and the domain X covered.

First some definitions $(1 \leqslant i, j, k \leqslant N_C)$: $\boldsymbol{x}_{ijk}$ is the point of intersection of three planes i, j, k (2.19), L_{ij} is the line of intersection of two planes i and j, $\overrightarrow{L_{ij}}$ a vector along

it, and $e_{ij} \subset L_{ij}$ is the edge between cells i and j if it exists in P and R. Points $(\boldsymbol{x}, p)$ on L_{ij} satisfy (2.15)

$$p = \boldsymbol{x} \cdot \boldsymbol{X}_i - R_i = \boldsymbol{x} \cdot \boldsymbol{X}_j - R_j$$
$$\Rightarrow \boldsymbol{x} \cdot \hat{\boldsymbol{n}}_{ij} = d_{ij}, \quad \hat{\boldsymbol{n}}_{ij} = \frac{\boldsymbol{X}_j - \boldsymbol{X}_i}{|\boldsymbol{X}_j - \boldsymbol{X}_i|}, \quad d_{ij} = \frac{R_j - R_i}{|\boldsymbol{X}_j - \boldsymbol{X}_i|}. \tag{3.2}$$

As $\hat{\boldsymbol{n}}_{ij}$ is normal to the line L_{ij}, $\hat{\boldsymbol{n}}_{ij} \cdot \overrightarrow{L_{ij}} = 0$ and so the line $\boldsymbol{X}_i \to \boldsymbol{X}_j$ in dual space is perpendicular to the corresponding line L_{ij} in physical space. If $|\boldsymbol{X}_i| \to \infty$ then the normal $\hat{\boldsymbol{n}}_{ij} \to \hat{\boldsymbol{n}}_i = -\boldsymbol{X}_i/|\boldsymbol{X}_i|$ and $d_{ij} \to d_i = -R_i/|\boldsymbol{X}_i|$ so that as long as $R_i = -d_i|\boldsymbol{X}_i|$ the line is still well defined but insensitive to j which is exactly the property required of a boundary segment.

A key step in the method is determining the vertices at each end of an edge e_{ij} known to be part of P. Chynoweth [22] finds them by testing all the possible vertices along L_{ij}: $\{\boldsymbol{x}_{ijk}, k = 1, \ldots, N_C, k \neq i, j\}$. Fig 3.4 shows the cross section of P along L_{ij}. The shaded region, $P \geqslant P(\boldsymbol{x})$, is the intersection of the half-planes $P \geqslant P_k(\boldsymbol{x})$, for $k = 1, \ldots, N_C$. Travelling along the line, the intersection points fall into two categories depending on whether the associated half-plane is entered (black circles) or exited (white) i.e. whether $\boldsymbol{X}_k \cdot \overrightarrow{L_{ij}}$ is positive or negative. The vertices sought then correspond to the last half-plane entered and first half-plane exited, with the order along the line being determined by the scalar parameter $\boldsymbol{x} \cdot \overrightarrow{L_{ij}}$.

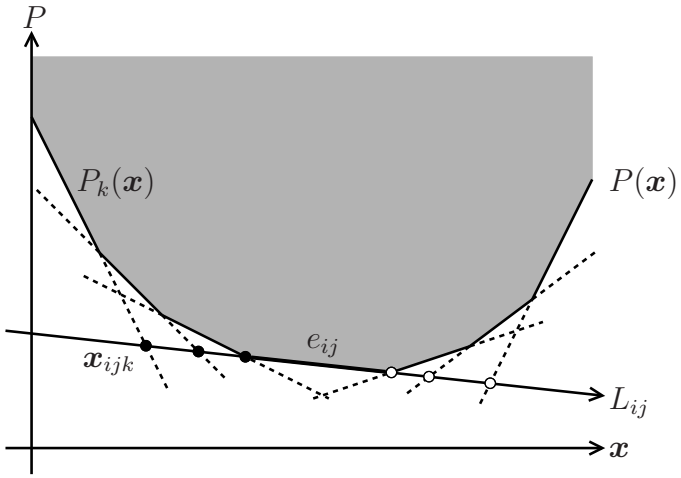


Figure 3.4: Determining the end vertices of edge e_{ij}

Initially the data arrays `x,y,p,X,Y,R,FirstEdge,Next,Cell` are empty, and new vertices, cells and edges are numbered sequentially as they are found and corresponding array elements filled in (except boundary cells are numbered from $N+1$ to N_C). The method is:

1. Initialise the boundaries.

Each boundary plane i (line in n=2) is defined by a normal $\hat{\boldsymbol{n}}_i$ and perpendicular distance d_i. Set $|\boldsymbol{X}_i|$ to some large number and $\boldsymbol{X}_i = -|\boldsymbol{X}_i|\hat{\boldsymbol{n}}_i$, $R_i = -d_i|\boldsymbol{X}_i|$ and store in boundary cell $N+i$, $N+1 \leqslant N+i \leqslant N_C$.

2. Find a cell a on the boundary.

An arbitrary point $\boldsymbol{x}_b$ is picked on the boundary plane b (Fig 3.5). From (2.16) $\boldsymbol{x}_b$ lies on the plane that maximises $\{P_a(\boldsymbol{x}_b), a = 1, \ldots, N, a \neq b\}$. If two planes a, c share this maximum value then they are neighbouring cells and $\boldsymbol{x}_b = \boldsymbol{x}_{abc}$ is a vertex of $P(\boldsymbol{x})$. The next step can then be skipped.

3. Find vertex $\boldsymbol{x}_{abc}$ on the boundary using the method above with line L_{ab}.

4. Starting at $\boldsymbol{x}_{abc}$ find the vertex $\boldsymbol{x}_{acd}$ at the other end of edge e_{ac}.

5. Repeat, starting from known vertices, until there are no cells left.

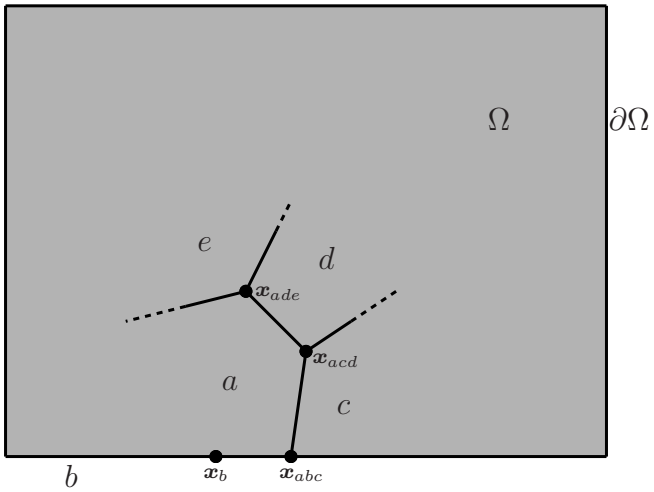


Figure 3.5: The Geometric Method

To find a new vertex $O(N)$ intersection points are evaluated, and this is repeated for every edge totalling $O(N^2)$ operations per reconstruction.

3.2.2 The iteration procedure

To reduce the number of degrees of freedom the $\boldsymbol{X}_i$ are fixed at the start and throughout, leaving only $\boldsymbol{R} = \{R_i\}$, $1 \leqslant i \leqslant N$ variable (from here on unless otherwise stated this range is always assumed). To solve the SMKP the rearrangement criteria (2.21) must be satisfied which is rewritten as the single vector equation:

$$\boldsymbol{F}(\boldsymbol{R}) \stackrel{\text{def}}{=} \boldsymbol{\beta} - \boldsymbol{A}(\boldsymbol{R}) = \boldsymbol{0}, \tag{3.3}$$

where $\boldsymbol{\beta} = \{\beta_i\}$, $\boldsymbol{A}(\boldsymbol{R}) = \{A_i\}$. We wish to solve this by Newton-Raphson so need to compute the Jacobian matrix $J_{ij} = \frac{\partial A_i}{\partial R_j}\Big|_{\boldsymbol{R}}$ which represents the change in area of cell i on varying R_j. As adjusting R_j will only change the areas of cell j and its immediate neighbours the matrix is sparse. If $R_i \to R_i + \delta R_i$, then by (3.2) if $\delta R_i < 0$ the edge e_{ij} will move sideways into neighbour cell j by $\delta d_{ij} = -\delta R_i/|\boldsymbol{X}_j - \boldsymbol{X}_i|$ (Fig 3.6) so cell j loses the shaded region δC_{ij} which has area $\delta A_{ij} \approx -|e_{ij}|\delta R_i/|\boldsymbol{X}_j - \boldsymbol{X}_i|$ from which the off-diagonal elements are

$$J_{ji} = -\frac{\delta A_{ij}}{\delta R_i} = |e_{ij}|/|\boldsymbol{X}_j - \boldsymbol{X}_i| = J_{ij} > 0, \quad j \neq i. \tag{3.4}$$

The total area $\sum_j A_j = |\Omega|$, a constant, and differentiating this with respect to R_i gives

$$\sum_j J_{ji} = 0 \quad \Rightarrow \quad J_{ii} = -\sum_{j \neq i} J_{ij} < 0, \quad 1 \leqslant i \leqslant N. \tag{3.5}$$

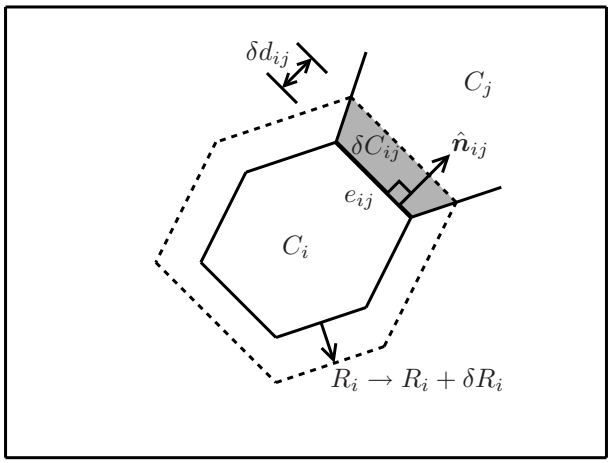


Figure 3.6: Computing elements of the Jacobian

Equation (3.5) can be written as $\boldsymbol{J}\boldsymbol{v} = \boldsymbol{0}$ where $\boldsymbol{v} = (1, 1, \ldots, 1)$, so that $\boldsymbol{v}$ is an eigenvector of $\boldsymbol{J}$ with eigenvalue zero. The interpretation of this is that if all the R_is are changed by the same amount then the areas remain unchanged. It also means $\boldsymbol{J}$ is singular and cannot be inverted. Fortunately this is the only zero eigenvalue because

$$\begin{aligned} \boldsymbol{R}^T\boldsymbol{J}\boldsymbol{R} &= \sum R_i J_{ij} R_j \\ &= 2\sum_{i>j} R_i J_{ij} R_j + \sum_i R_i^2 J_{ii} \\ &= 2\sum_{i>j} R_i J_{ij} R_j + \sum_i R_i^2 (-\sum_{j \neq i} J_{ij}) \\ &= -\sum_{i>j} J_{ij}(R_i - R_j)^2 \leqslant 0. \end{aligned} \tag{3.6}$$

Equality occurs if $R_i = R_j$ whenever cells i,j are neighbours ($J_{ij} \neq 0$). But if the neighbours of cell i must have the same value of R then so also must *their* neighbours and so on spreading out until all cells must have the same value of R i.e. $\boldsymbol{R}^T\boldsymbol{J}\boldsymbol{R} = 0 \Leftrightarrow \boldsymbol{R}$ is a multiple of $\boldsymbol{v}$. A simple way to remove this eigenvector is to fix one of the R_i's, say that of cell a. The reduced Jacobian $\boldsymbol{J}^a$ is formed by removing row and column a from $\boldsymbol{J}$ (but retaining the index numbering) and $J_{ii} = -\sum_{j\neq i} J_{ij} = -\sum_{j\neq i,a} J_{ij} - J_{ia} \Rightarrow J^a_{ii} \geqslant -\sum_{j\neq i,a} J^a_{ij}$ with strict inequality when cell i is a neighbour of cell a ($J_{ia} \neq 0$). Thus $\boldsymbol{J}^a$ is diagonally dominant, strictly for some rows, and so invertible.

We can now apply Newton-Raphson to the reduced system (dropping the superscript a):

$$\boldsymbol{R}^{n+1} = \boldsymbol{R}^n - \boldsymbol{J}^{-1}(\boldsymbol{\beta} - \boldsymbol{A}(\boldsymbol{R}^n)) \tag{3.7}$$

Chynoweth applies Euler's theorem for a plane graph to show that the number of non-zero elements of $\boldsymbol{J}$ is $O(N)$ allowing the use of fast sparse matrix inversion methods e.g. Lanczos. It doesn't matter that cell a has been removed from $\boldsymbol{R}$ because when all the other cells have the correct area $\sum_j A_j = \sum_j \beta_j = |\Gamma|$ ensures that so too will cell a. However it is possible for one of the R_i to be raised so high that the vertex is no longer part of the convex hull of $R(\boldsymbol{X})$, which corresponds to the cell's area becoming zero (Fig 3.7).

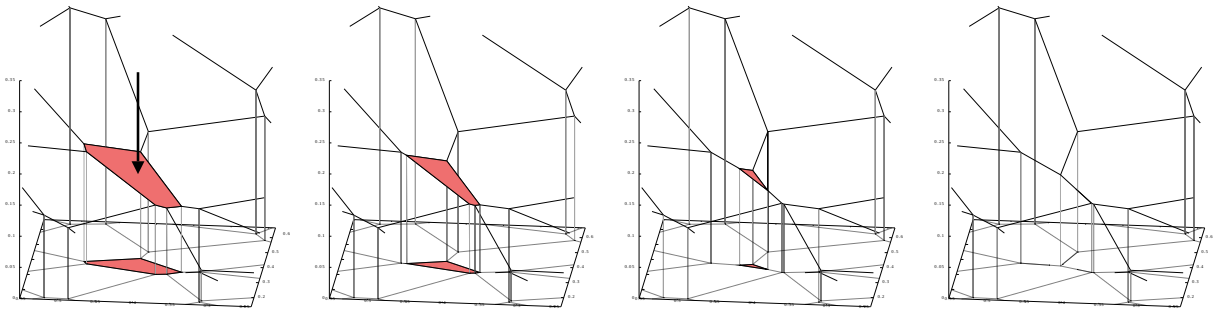


Figure 3.7: Disappearing cell (red) as R_{red} increased

When this happens the iteration is stopped and, as long as they are isolated, any offending cells are re-inserted by lowering their R_i until the number of vertices where $P_i(\boldsymbol{x}) > P(\boldsymbol{x})$ is either one or two (this ensures no other cells can be accidentally removed as every cell has at least three vertices). With this capability the initial guess $\boldsymbol{R}^0$ can consist of just a few cells with non-zero values, with the rest being inserted in successive iterations.

3.3 Purser's implementation

Purser's implementation of the geometric method [74] differs from Chynoweth's [22] in four areas.

Firstly we still expand $\boldsymbol{F}(\boldsymbol{R})$ (3.3) to first order but do not multiply through by $\boldsymbol{J}^{-1}$:

$$\boldsymbol{J}(\boldsymbol{R}^{n+1} - \boldsymbol{R}^n) = \boldsymbol{\beta} - \boldsymbol{A}(\boldsymbol{R}^n) \tag{3.8}$$

This is solved for $\boldsymbol{R}^{n+1}$ by the (linear preconditioned) conjugate gradient method [33]. Unlike the original geometric method no cells need be removed from $\boldsymbol{R}$, but the possibility arises that it could acquire a large component in the direction of $\boldsymbol{v}$ which must be subtracted to keep the numbers sensible. A simple choice is to reset R_1 to zero after each iteration i.e. $\boldsymbol{R} \to \boldsymbol{R} - R_1\boldsymbol{v}$.

Secondly, instead of reinserting cells that have been 'cast adrift' of the convex hull (Fig 3.7), the solution of (3.8) is split up into a number of smaller steps, enough so that this doesn't happen in the first place. Each step the target areas $\boldsymbol{\beta}$ are slowly modified, being initially set equal to the current areas (so no change in $\boldsymbol{R}$ is required) and are adjusted linearly with step number until after N_{gm} steps they are the intended areas. So starting from an initial guess $\boldsymbol{R}^0$ the target areas on step n are

$$\boldsymbol{\beta}^{(n)} = \left(1 - \frac{n}{N_{gm}}\right)\boldsymbol{A}(\boldsymbol{R}^0) + \frac{n}{N_{gm}}\beta. \tag{3.9}$$

The parameter N_{gm} is determined experimentally and for all the test problems in this thesis $N_{gm} = 3$ was found sufficient for each timestep, but sometimes needed to be more during the initial mesh convergence phase.

Thirdly the potentials P and R are constructed by finding the convex hull of the vertices $\{(\boldsymbol{X}_i, R_i)\}$ using the method of Preparata and Hong [72] which is described next, followed by some details of the implementation. The fourth and last difference is that instead of constructing new potentials R and P from scratch each iteration, the current potentials are retained and modified if necessary to maintain convexity with the panel beater algorithm, described in section 3.3.3.

3.3.1 The convex hull algorithm of Preparata and Hong

The basic strategy is to divide-and-conquer: the set of vertices (excluding vertices representing boundary cells) is split into two spatially-disjoint subsets (i.e. a dividing plane exists between the two), the convex hulls of each of these are found separately

and then they are merged together. The algorithm is recursively applied to each subset, breaking them up in turn into smaller and smaller pieces, until each piece consists of just one or two vertices. Then the pieces are repeatedly merged together, back up the binary tree structure, until there is just one piece - the complete convex hull.

The sets can be split efficiently if the vertices are initially renumbered in order along an axis, say the x-axis. For $N = 2^M$ vertices this takes $O(N \log N)$ operations but then a subset $\{\boldsymbol{x}_a, \ldots, \boldsymbol{x}_b\}$ can be immediately split, by a plane normal to the axis just to one or other side of the median vertex, into $\{\boldsymbol{x}_a, \ldots, \boldsymbol{x}_{\lfloor (a+b)/2 \rfloor}\}$ and $\{\boldsymbol{x}_{\lfloor (a+b)/2 \rfloor + 1}, \ldots, \boldsymbol{x}_b\}$. After i iterations there are 2^i subsets of $N/2^i = 2^{M-i}$ vertices, and so if two subsets of size p and q can be merged in $O(p+q)$ operations, the total computational cost will be $O(N \log N) + \sum_{i=1}^{M} 2^i O(2^{M-i}) = O(N \log N)$ operations.

In two dimensions a convex hull is a convex polygon and the merge routine amounts to finding mutual tangents (green in Fig 3.8) of the two polygons A and B, and eliminating the points now inside the new hull.

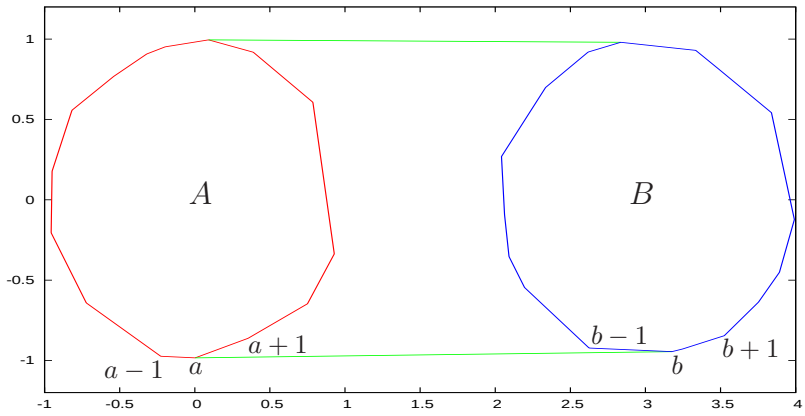


Figure 3.8: Merging two convex hulls in 2D with the mutual tangents (blue)

Let the vertices of A and B be numbered anticlockwise from 1 to p and 1 to q respectively, and for any three vertices i, j, k the area of the triangle they form is

$$A(i,j,k) = \frac{1}{2}[(x_j - x_i)(y_k - y_i) - (y_j - y_i)(x_k - x_i)] \tag{3.10}$$

which is positive(negative) if k is on the left(right) side of the line from i to j. Then the conditions for ab ($a \in A$, $b \in B$) to be the lower(upper) mutual tangent are

$$A(a,b,a-1), A(a,b,b+1) \geqslant (<)0 \quad \text{and} \quad A(a,b,a+1), A(a,b,b-1) > (\leqslant)0,$$

where arithmetic is modulo p, q for A, B respectively. The algorithm for finding the lower mutual tangent $a_L b_L$ iteratively moves either a or b until the conditions are met:

1. *Set a to be the leftmost vertex of A and b to be the rightmost vertex of B.*
2. *If $A(a, b, a+1) \leqslant 0$ then set $a \leftarrow a+1$ and goto 2.*
3. *Elseif $A(a, b, b-1) \leqslant 0$ then set $b \leftarrow b-1$ and goto 2.*
4. *Else ab is the required tangent. Set $a_L = a$, $b_L = b$ and exit.*

The first step takes $O(p) + O(q)$ operations and then on each iteration the convexity of A and B ensure that either $A(a, b, a+1)$ is increased or $A(a, b, b-1)$ is decreased, so the algorithm is guaranteed to stop in $O(p+q)$ steps. The algorithm for the upper mutual tangent $a_U b_U$ is similar, also taking $O(p+q)$ steps, and the merged convex hull is $\{a_L, b_L, b_L+1, \ldots, b_U, a_U, a_U+1, \ldots, a_L-1\}$, evaluated in $O(p+q)$ operations as required.

The three dimensional case is more complex but follows the same basic idea of constructing the tube-shaped surface T (green in Fig 3.9) bridging the convex hulls A and B, then removing any vertices that are now internal.

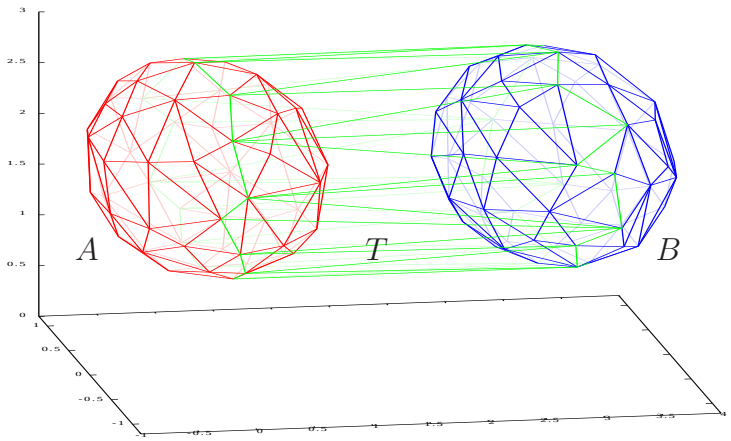


Figure 3.9: Merging two convex hulls in 3D

Starting with some edge E of the tube the *gift-wrapping principle* of Chand and Kapur [20] is used to find an adjoining face, then a face adjoining that, and so on round the tube until it is complete. The first step then is to find an edge on the tube and this is easily done with the help of the two dimensional algorithm. Hulls A and B are projected onto the plane beneath and the algorithm above is used to find the lower mutual tangent $a_L b_L$. This must be the projection of an edge E of T (Fig 3.10), so set a_1=a_L, b_1=b_L and E=$a_1 b_1$.

Given a face f of a polyhedron the gift-wrapping principle states that the face adjoining f along an edge e is contained in the plane through e and another vertex, the one

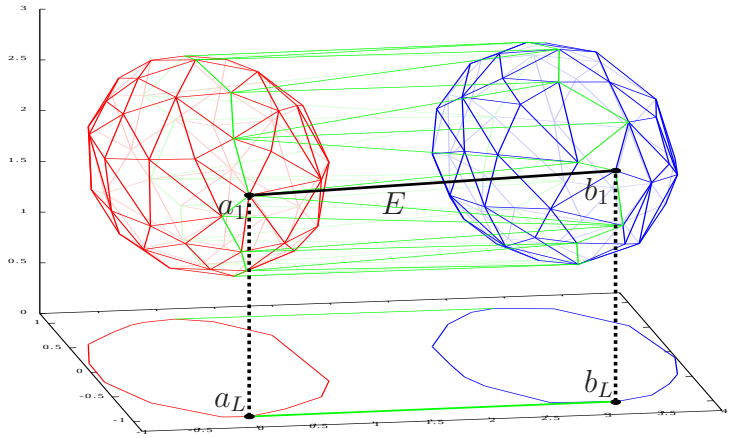


Figure 3.10: Finding an edge on T

whose plane forms the minimum angle with f. Chand and Kapur's convex hull algorithm tests every other vertex to find this minimum each edge so their algorithm takes $O(N^2)$ operations in total but the convex hulls A and B can be used to reduce the number of tests significantly. In fact it is only necessary to test the vertices connected by an existing edge to either a_1 or b_1 (marked black in Fig 3.11). The first face f is constructed by joining E to its projection on the plane below.

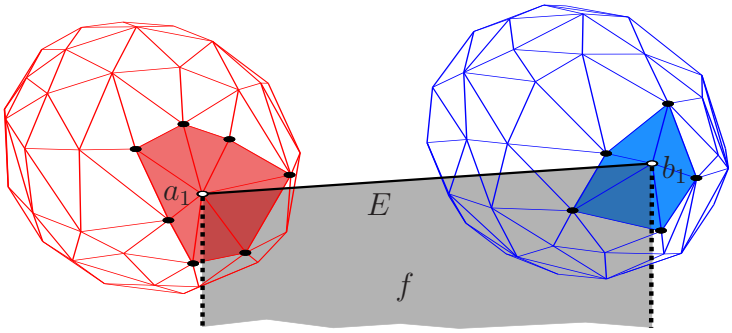


Figure 3.11: Applying the gift wrapping principle

Applying the gift wrapping principle to (f,E) yields a vertex from either A or B, in this case $b_2 \in B$, and the first face of T, $a_1b_1b_2$ (green in Fig 3.12).

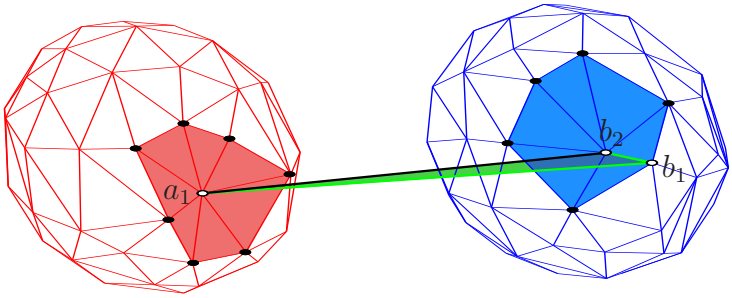


Figure 3.12: The first face of T

Now we can set $E{=}a_1b_2$, $f{=}a_1b_1b_2$ and apply the gift wrapping principle again, testing vertices connected to a_1 or b_2 (marked black in Fig 3.12), to get the next face of T (Fig 3.13), and so on, with the final result in Fig 3.9. No special treatment is required for degeneracies (vertices on A and B both sharing the same minimum angle).

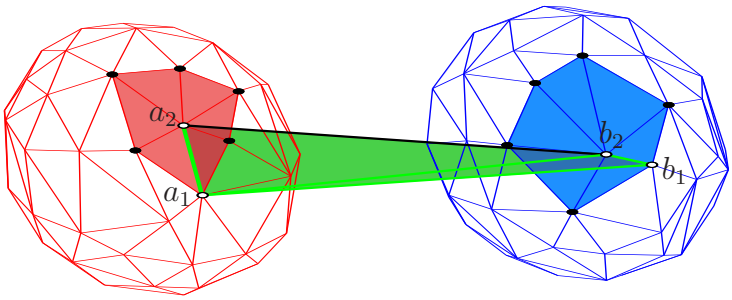


Figure 3.13: The next face of T

At each stage at least two vertices are tested. The successful vertex is added to T which removes it from further consideration, and leaves at least one unsuccessful edge that can therefore never be part of T so can also be removed from further consideration. From Euler's formula for convex polyhedra $V - E + F = 2$ (for V vertices, E edges and F faces), the number of edges of A and B can be at most $3p - 6$ and $3q - 6$ respectively, so the total number of tests carried out before all the vertices and edges are exhausted is bounded by $O(p + q)$.

In the process the ends of E trace out two loops of vertices $a_1a_2\ldots$ and $b_1b_2\ldots$ (marked yellow and light blue in Fig 3.9) which divide A and B into two disjoint pieces, one of which is internal to the new hull and must be removed. It is possible to determine which vertices these are in $O(p + q)$ operations, for example by 'shrinking' the loops (to the right side of a_1a_2 in A and the left side of b_1b_2 in B) down to a point. In total, including finding the two-dimensional hull, the merge takes $O(p + q)$ operations as required.

3.3.2 Pertinent details of the implementation

Ordering all the vertices along an axis to simplify the splitting of sets of vertices into disjoint subsets is elegant theoretically but results in increasingly thin subsets which could give rise to difficulties numerically (Fig 3.14). Instead Purser [74] splits the sets alternatively in the x and y directions (forming a *balanced* k*d-tree* [73]), which results in more evenly proportioned subsets. To partition a set of n vertices by the median is an example of *selection* and is achievable in $O(n)$ operations [73, sec 8.5]. To partition all the

subsets of $N = 2^M$ vertices that are created takes $O(2^M) + 2O(2^{M-1}) + 4O(2^{M-2}) + \ldots + 2^M O(1) = O(M2^M) = O(N \log N)$ operations so the overall complexity is unchanged.

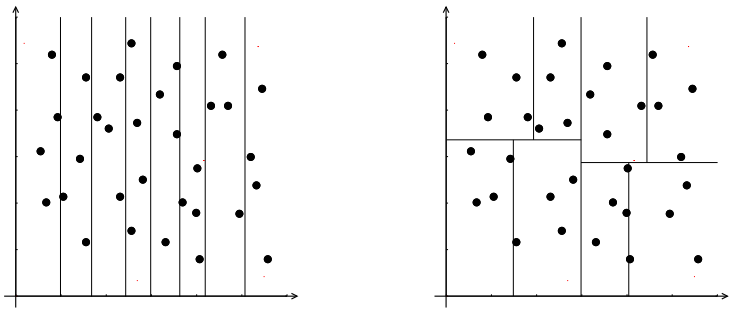


Figure 3.14: Dividing vertices by the x coordinate (left) or by both coordinates alternatively (right)

The division process is stopped when a piece has three or four vertices and the convex hull of these explicitly constructed. For three vertices there are just two cases (the triangle they form when projected onto the plane beneath is either clockwise or anti-clockwise), and twenty different cases for four vertices. If $\boldsymbol{V}_i = (X_i, Y_i, R_i)$ then the volume of the tetrahedron formed by vertices i, j, k, l is

$$\begin{aligned} V_{ijkl} &= \tfrac{1}{6}(\boldsymbol{V}_j - \boldsymbol{V}_i) \cdot (\boldsymbol{V}_k - \boldsymbol{V}_i) \times (\boldsymbol{V}_l - \boldsymbol{V}_i) \\ &= \tfrac{1}{3!} \det \begin{pmatrix} X_{ij} & Y_{ij} & R_{ij} \\ X_{ik} & Y_{ik} & R_{ik} \\ X_{il} & Y_{il} & R_{il} \end{pmatrix} \end{aligned} \quad . \tag{3.11}$$

where $X_{ij} = X_j - X_i$ etc. There are just two basic patterns for four vertices, types A and B:

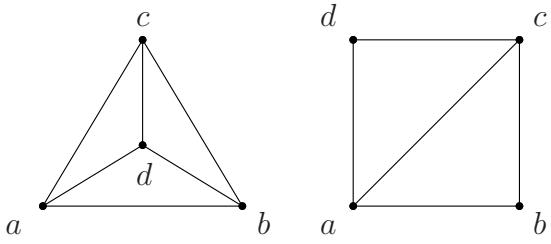


Figure 3.15: Type A (left) and B (right) four-vertex convex hulls

Using the binary digits 1 and 0 to represent truth and falsehood, table (3.1) lists all the combinations of area and volume by sign and, for each that is possible, its type and the vertices a, b, c, d. The case number is just the binary number formed from the first five rows. For vertices on the perimeter `FirstEdge` is initialised to point to the next vertex round the perimeter clockwise e.g. for type A `FirstEdge[`a`]`$=e_{ac}$ etc.

case	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
$A_{123}>0$	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
$A_{134}>0$	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
$A_{142}>0$	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
$A_{234}>0$	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
$V_{1234}>0$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
type	A	B	B	-	B	-	-	-	-	A	A	B	A	B	B	-	-	B	B	A	B	A	A	-	-	-	-	B	-	B	B	A
a	2	1	1	-	1	-	-	-	-	1	1	4	1	3	2	-	-	2	3	1	4	1	1	-	-	-	-	1	-	1	1	2
b	4	2	3	-	4	-	-	-	-	2	3	1	4	1	1	-	-	4	2	2	3	4	3	-	-	-	-	2	-	4	3	3
c	3	4	2	-	3	-	-	-	-	3	4	2	2	4	3	-	-	3	4	4	2	3	2	-	-	-	-	3	-	2	4	4
d	1	3	4	-	2	-	-	-	-	4	2	3	3	2	4	-	-	1	1	3	1	2	4	-	-	-	-	4	-	3	2	1

Table 3.1: Cases of four vertex convex hulls

The algorithm as described is applicable to a general set of points scattered throughout $\mathbb{R}^3$, but the set of dual space vertices $\{R_i\}$ define a single-valued function $R(\boldsymbol{X})$, which makes certain simplifications possible. The three dimensional hull is bowl-shaped, and the corresponding two dimensional hull is then formed by the projection of the rim (the thick lines in Fig 3.16). The data structures described earlier make navigation anticlockwise

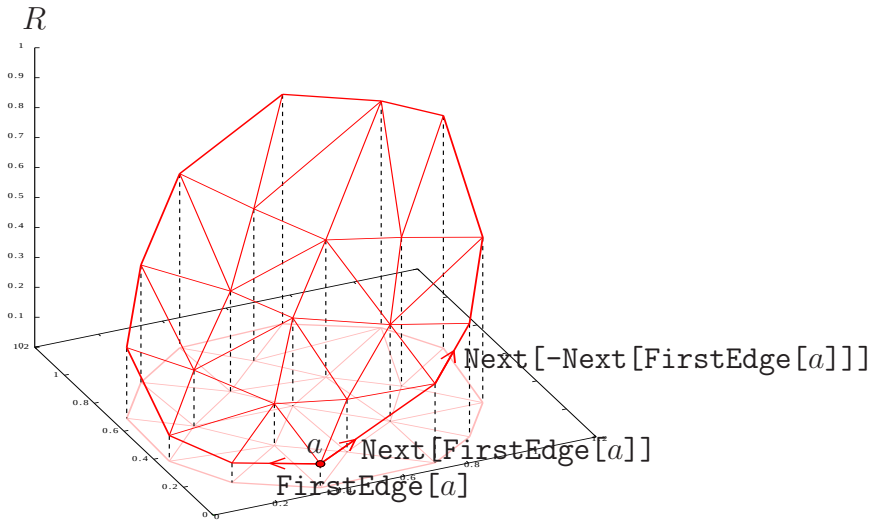


Figure 3.16: The convex hull of single-valued R

around the rim very simple - if e is an edge anticlockwise round the rim then the next is `Next[-`e`]`. Also as `FirstEdge` can be any edge starting from the same vertex, for vertices on the rim it is set to the clockwise edge so that travel is now simple in both directions. Lastly the rim can be quickly located by storing any vertex on it along with each hull.

For example the vertex stored with the merged hull is a_1.

Instead of creating new data arrays to represent the merged hull, the arrays for A and B are adjusted in place as T is constructed, making sure that the anticlockwise ordering of edges at each is maintained. A typical stage in the construction is shown, looking down from above, in Fig 3.17a, where f and E=ab are the current face and edge of T, and it is assumed that `FirstEdge[`a`]`=E and `FirstEdge[`b`]`=$-E$.

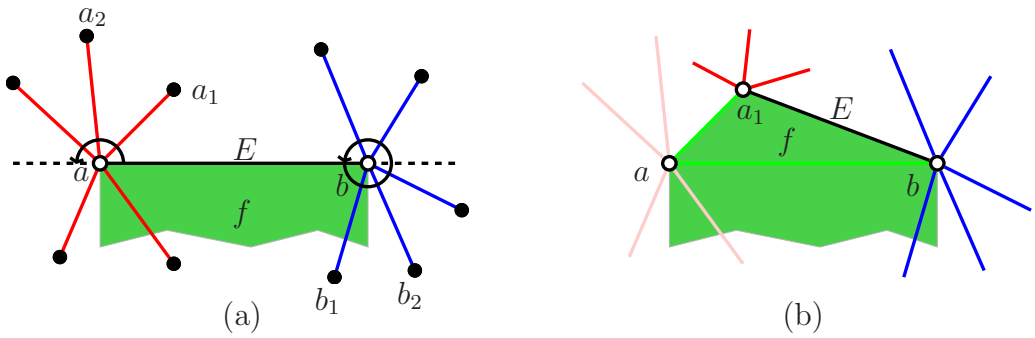


Figure 3.17: Advancing the edge E

The procedure to advance E is as follows.

1. Find the vertex a_i connected to a whose plane has the minimum angle to f.
Set e_0=`FirstEdge[`a`]`$(=E)$ and e_i=`Next[`e_{i-1}`]`, a_i=`Cell[`e_i`]`, $i = 1, \ldots$ which generates vertices in the order indicated by the arrow. If the normals to the planes f and $f_i = aba_i$ are $\boldsymbol{f}$ and $\boldsymbol{f}_i$ then the angle θ_i between them satisfies $\tan\theta_i = |\boldsymbol{f} \times \boldsymbol{f}_i|/(\boldsymbol{f} \cdot \boldsymbol{f}_i)$ (a little computational effort can be spared in determining the minimum by comparing $\tan\theta$ not θ). The iteration stops when either $e_i = e_0$ (have gone right round a), $\tan\theta_i > \tan\theta_{i-1}$ or $A(a, b, a_i) < 0$ (a_i is behind the dashed line formed by E in Fig 3.17a).
2. Find the vertex b_i connected to b whose plane has the minimum angle to f.
Set e_0=`FirstEdge[`b`]`$(=-E)$ and e_i=`Next[`e_{i-1}`]`, b_i=`Cell[`e_i`]`, $i = 1, \ldots$. Vertices are not counted until $A(a, b, b_i) > 0$ (they are ahead of the dashed line), and again the iteration stops when either $e_i = e_0$ or $\tan\theta_i > \tan\theta_{i-1}$.
3. Compare the 'winning' vertices from A and B, if they exist, and move forward.
In this example (Fig 3.17b) vertex a_1 has the least angle overall so a new edge $E = a_1b$ is created, f becomes aba_1, `FirstEdge[`a_1`]`=E, `FirstEdge[`b`]`=$-E$ and the data arrays for a_1 and b are adjusted to accomodate the new edge E.

To remove vertices internal to the merged hull, they are found with a fill routine seeded by the vertices just inside the loop. It uses three auxilary variables - a list of vertices to be

removed (initially empty), a logical flag that indicates whether a given vertex is in either the loop or the list (initially true for vertices in the loop, false otherwise), and an integer counter (intially zero). The routine starts with $i = 1$ and proceeds as follows (for A):

1. Find an edge from a_i to an unflagged vertex inside the loop. If none goto step 5.
First find the edge to a_{i-1}: set $e =$`FirstEdge[`a_i`]` and repeat $e \leftarrow$`Next[`e`]` until `Cell[`e`]` $= a_{i-1}$. Then repeat $e \leftarrow$`Next[`e`]`, $v =$`Cell[`e`]` until either v is unflagged or $v = a_{i+1}$.
2. Flag v, add it to the list and remove edge e from a_i and v.
3. Flag any unflagged vertices adjacent to v and add them to the list.
4. If the counter is not at the end of the list then increment it, set v to be the vertex in this position in the list and return to step 3.
5. If a_i is the last vertex in the loop then exit otherwise set $i \leftarrow i+1$ and return to step 1.

If the vertices all lie on a single convex surface then there will be no internal vertices and the above routine is unnecessary, e.g. as in Fig 3.18.

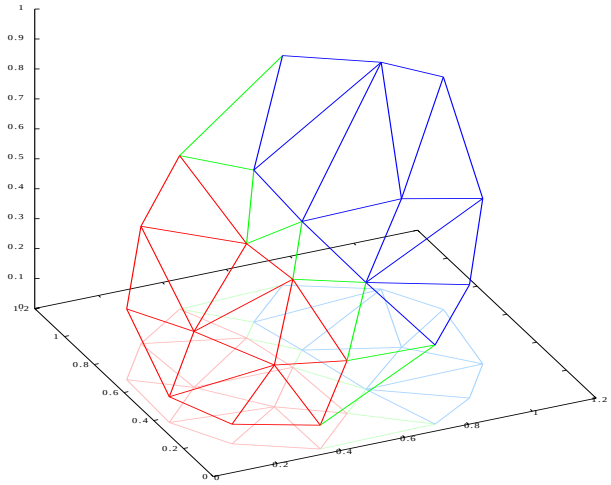


Figure 3.18: Merging two hulls (red and blue) with no internal vertices

It is possible to arrange this as the convex hull is only computed once at the start and we are free to choose the initial set $\{R_i\}$. A suitable convex surface is the paraboloid: $R(X,Y) = aX^2/2 + bX + cY^2/2 + dY$ - the resulting hull, when projected onto $R = 0$, is just the Delauney triangulation of $\{\boldsymbol{X}_i\}$ [5]. From (2.14) $x = aX + b$, $y = cY + d$, $P = (x-b)^2/2a + (y-d)^2/2c$ and $\det(D^2P) = 1/ac$, a constant. If $\{\boldsymbol{X}_i\}$ are evenly spaced throughout Σ then this has the additional benefit that cells in physical space all have the same area and a rectangular domain in physical space is transformed into a rectangular domain in dual space e.g. $[b, b+a] \times [d, d+c]$ is transformed to $[0,1] \times [0,1]$, and so

this is fixed from now on. Finally to minimise the possibility of the vertices $\{(\boldsymbol{X}_i, R_i)\}$ initially falling into special position (resulting in four or more edges meeting at a vertex in physical space violating the assumption of three currently hard-coded) the $\{\boldsymbol{X}_i\}$ are slightly randomised.

3.3.3 Panel beating

The behaviour of the conjugate gradient method (3.8) is governed by the Jacobian $\frac{\partial \boldsymbol{A}}{\partial \boldsymbol{R}}$. Equations (3.4), (3.5) evaluate the Jacobian at the point $\boldsymbol{R}$, providing only first derivative information, so for higher order terms a more accurate expression is required for the area of C_i. If b, c, d are three neighbours in anticlockwise order around cell a then

$$A_a = \frac{1}{2}\sum_b (x_{abc}y_{acd} - x_{acd}y_{abc})$$

and inverting (2.19):

$$\begin{pmatrix} x_{abc} \\ y_{abc} \\ P_{abc} \end{pmatrix} = \frac{1}{2A_{abc}} \begin{pmatrix} Y_b - Y_c & Y_c - Y_a & Y_a - Y_b \\ -X_b + X_c & -X_c + X_a & -X_a + X_b \\ X_cY_b - X_bY_c & X_aY_c - X_cY_a & X_bY_a - X_aY_b \end{pmatrix} \begin{pmatrix} R_a \\ R_b \\ R_c \end{pmatrix} \tag{3.12}$$

and similarly for x_{acd}, y_{acd}. These are all linear functions of $\boldsymbol{R}$, and so the area is quadratic. Vary $\boldsymbol{R}$ sufficiently and the convex hull will change discretely so the area is in general piecewise quadratic, and the Jacobian piecewise linear. We can therefore expect successive iterations $\boldsymbol{R}^n$ to be fairly similar which suggests it would be more efficient to modify the existing convex hull from one iteration to the next rather than generate a new one from scratch each time. The simplest possible change in connectivity that can occur between iterations is an *edge flip* (Fig 3.19), in which a dual space edge flips to the alternate diagonal of the quadrilateral which surrounds it (here e_{ac} flips to e_{bd} in $abcd$), which in physical space corresponds to neighbours a, c moving apart and cells b, d becoming neighbours.

Figures 3.20 and 3.21 show how the potentials R and P cause this. The left hand panels show the initial configuration. The surface R is convex, so the grey line which has been added to join cells b and d lies above R, as does the tetrahedron it forms with the edge e_{ac}, and V_{abcd} (3.11) is positive. If R_b is lowered (middle panels) then at some point the line db in dual space will drop below the plane acd and V_{abcd} becomes negative which indicates that R has become non-convex.

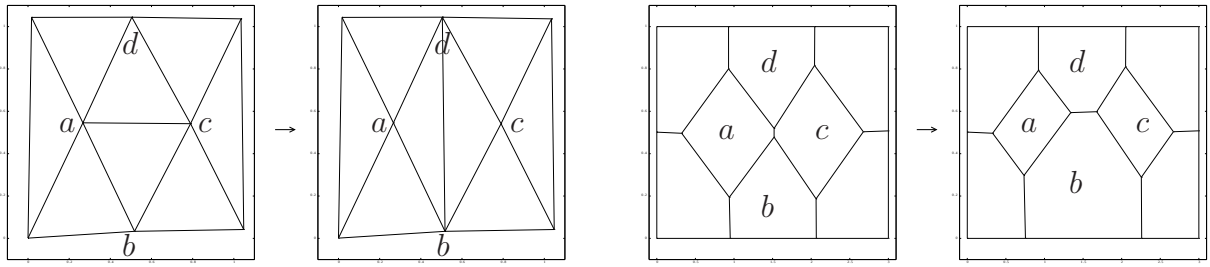


Figure 3.19: An edge flip

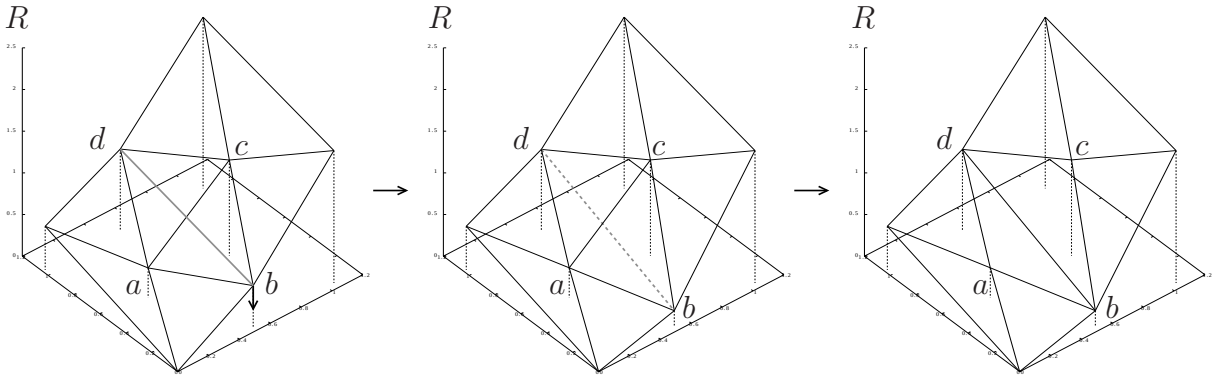


Figure 3.20: Edge-flip in dual space

In physical space this corresponds to the plane of cell b being raised until the surface P becomes involuted, with a tetrahedral 'pocket' underneath it. The right hand panels show the corrected convex hull in which the faces abc and acd of R have been replaced by the faces abd and bcd, in effect beating the non-convex 'bump' out of the surface R in a manner somewhat reminiscent of panel beating sheet metal, hence the name. This suggests an alternative interpretation - that the volume above $R(\boldsymbol{X})$ is being maximised.

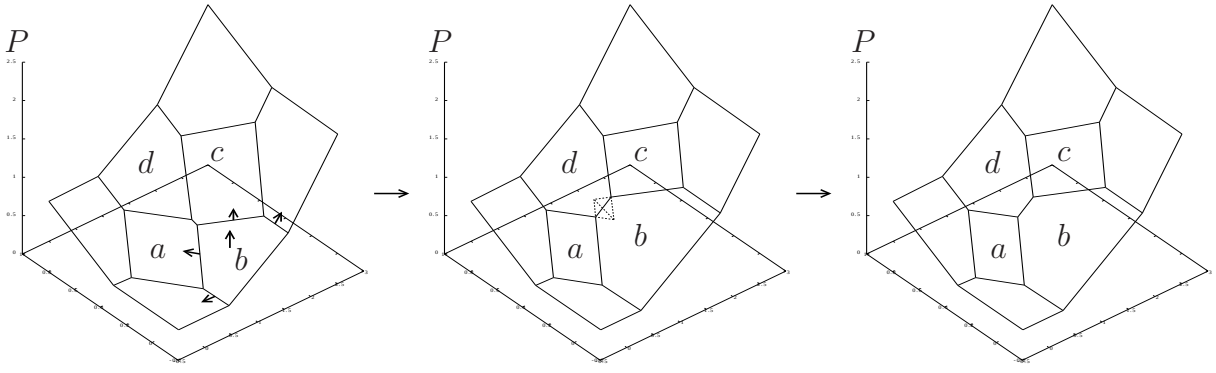


Figure 3.21: Edge-flip in physical space

A naive panel beating algorithm assumes that the only changes between one iteration of R and the next are isolated edge flips which can be detected by tetrahedra V_{abcd} with negative volume:

1. Loop over edges $e = 1, \ldots, N_V$.

Set $c =$`Cell[`e`]`, $a =$`Cell[`$-e$`]`, $d =$`Cell[Next[`e`]]`, $b =$`Cell[Next[`$-e$`]]`.

2. Compute volume V_{abcd} *and if negative flip* e *from* ac *to* bd *and adjust data arrays.*

Set $e_{ad} =$`Next[`e`]`, $e_{da} = -e_{IL}$, $e_{dc} =$`Next[`e_{da}`]`, $e_{cd} = -e_{dc}$, $e_{cb} =$`Next[`$-e$`]`, $e_{bc} = e_{cb}$, $e_{ba} =$`Next[`e_{bc}`]`, $e_{ab} = -e_{ba}$ (Fig 3.19). Then remove e from a with `Next[`e_{ab}`]`$= e_{ad}$ and insert it at b with `Next[`e_{bc}`]`$= e$, `Next[`e`]`$= e_{ba}$, `Vert[`e`]`=`Vert[`e_{bc}`]`. Also remove $-e$ from c with `Next[`e_{cd}`]` $= e_{cb}$ and insert it at d with `Next[`e_{da}`]`$= -e$, `Next[`$-e$`]`$= e_{dc}$ and `Vert[`$-e$`]`=`Vert[`e_{da}`]`. Lastly set `Cell[`$-e$`]`$= b$, `Cell[`e`]`$= d$.

Evidently this takes $O(N_V)$ operations, which by (3.1) is $O(N)$.

If (3.8) casts a cell a adrift of the convex hull then in physical space its area shrinks to zero (Fig 3.22). In dual space the vertex a rises above the plane bcd to form a tetrahedral bump in R (Fig 3.23a). The above repair procedure applied to edge ac will identify the non-convexity from the sign of V_{abcd} and replace faces abc and acd by the faces abd and bcd. Face bcd is fine but face abd forms with adb an opposing pair (same vertices but opposite orientation) visible as a 'flap' of zero volume sticking out from R (Fig 3.23b). This situation can be detected from $A_{abd} < 0$ or $A_{bcd} < 0$ (3.11), and rectified either by eliminating the opposing faces abd, adb (Fig 3.23c) and optionally reinserting cell a at the end of this iteration (page 37), or by aborting the iteration procedure and restarting with a higher value of N_{gm}.

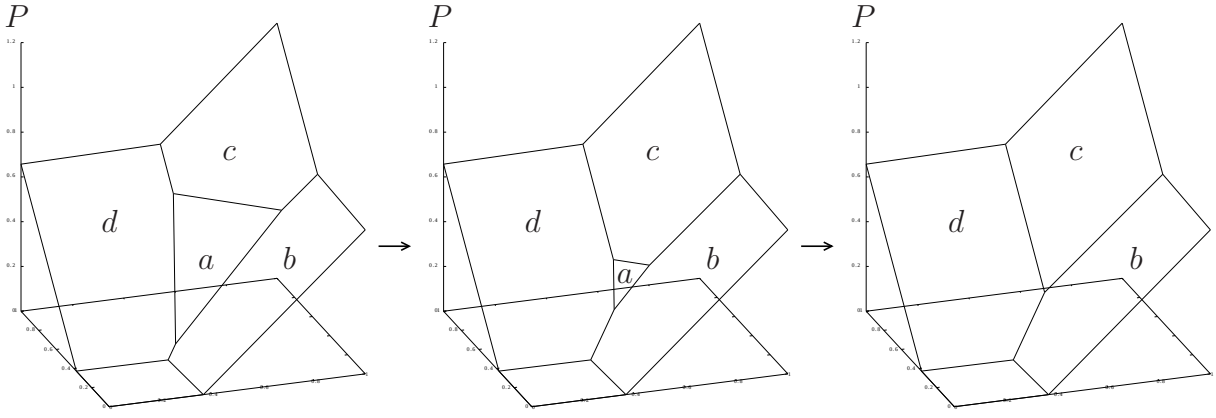


Figure 3.22: Cell a being cast adrift (physical space)

The edge flip, insertion and deletion operations together comprise the set of planar *geometric bi-stellar flips* [78] and form a basis of operations for modifying point set triangulations. In the notation the edge flip is called a (2,2)-flip, the insertion a (1,3)-flip

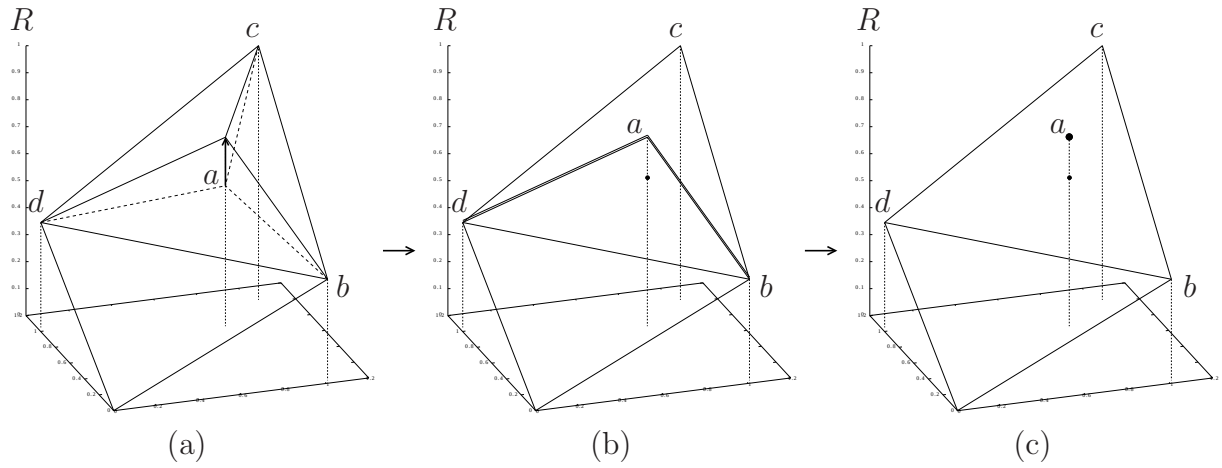


Figure 3.23: The cast adrift cell a undergoing repair in dual space

and the deletion a (3,1)-flip. In two dimensions, the set of all possible triangulations is known to be connected which means that any change to the connectivity can always be built up by repeated edge flips. The simplest way to extend the previous algorithm to allow these is to test each edge in turn and flip if needed as before, but then also recheck any adjacent edges in case they have been made non-convex as a result:

1. Initialise a priority queue to be empty and set edge $e = 0$.
2. Take the next edge from priority queue or if empty increment e *and take that edge.*
3. Check edge as above and if repair required then do so and add any of the adjacent edges $|e_{ab}|$, $|e_{bc}|$, $|e_{cd}|$ *and* $|e_{da}|$ *that are less than* e *to the priority queue (any above will be checked anyway in due course).*
4.If $e = N_E$ *and the priority queue is empty then exit else goto step 2.*

As it stands the algorithm looks like it could cycle indefinitely but this cannot happen because each edge flip increases the volume above the surface R by some tetrahedron (the tetrahedron *abcd* in Fig 3.20) and the total number of tetrahedra is $\binom{N}{4}$ which is finite. However many of these tetrahedra overlap so an edge-flip can add pieces of a number of tetrahedra - which raises the question: how many edge-flips are required in the worst-case scenario? In two dimensions, despite the fact that the total number of triangulations rises exponentially with N [3], it is known that $O(N^2)$ edge-flips are always sufficient [39] and sometimes necessary [48]. So there exist cases where modifying an existing hull will be slower than the $O(N \log N)$ required to construct the convex hull from scratch, and this is before the expense of finding flippable edges is considered. However, because here the difference between successive iterates $\boldsymbol{R}^n$ is expected to be small, so too is the expected

number of edge flips which suggests a reasonable strategy would be to start with the panel beater algorithm but if the number of edges flipped reaches some cutoff e.g. $N \log N$ then switch to the full reconstruction.

Two possible extensions of the panel beater scheme are briefly mentioned. Firstly if not just $\{R_i\}$ but $\{\boldsymbol{X}_i, R_i\}$ are allowed to vary, then Purser [74] shows that it is possible for the surface R to become non-convex without any of the volumes of the tetrahdra formed by adjacent triangular faces of R becoming negative. His example demonstrates how it is possible to repair the surface R by using the full range of bi-stellar flips.

Fig 3.24a shows an initial configuration in which the convex surface $R(\boldsymbol{X})$ contains vertices a to f, of which d and e move as indicated resulting in a self-intersecting or 'snagged' surface of (3.24b). Despite the tetrahedra all having positive volume, at least one triangle's normal now points downwards (ebd and edc), and R becomes multi-valued there (d' and d'' below d and e' above e). Assuming a downward normal is sufficient to detect all such snags, we then have the choice of which vertex on the common edge e_{ed} to treat first. Suppose we delete vertex d from R by replacing faces dbe, dec and dcb by ecb (3.24c). Since $R_d > R_{d''}$ the cell d has been cast adrift. However R is still not convex across edge e_{bc} so it must be flipped to e_{ae}, replacing faces abc and ecb with abe and aec. The final R is shown in Fig 3.24d.

The alternate strategy is to delete cell e first, by removing faces edb, ebf, efc and ecd and refilling the tetrahedral gap $dbfc$ with either dbf and dfc or bfc and bcd. The former choice results in a situation like that in Fig 3.23 and cell d will be cast adrift. Fig 3.24e shows the latter choice, and again a pair of opposing faces, bcd, bdc are created, which when eliminated cast cell d adrift. Now however $R_e < R_{e'}$, so cell e must be reinstated. As e' lies in face abc, this face is replaced by the faces ebc, eca and eab as shown in (3.24f). Again R is still not convex, this time across edge e_{bc}, which when flipped to e_{ef} results in the same final configuration as before in (3.24d). Although successful in repairing R, there is a hidden cost here - computing $R_{d'}$ and $R_{e'}$ requires first locating the faces containing vertices d and e. For general configurations this search could be expensive but the panel beater method is only expected to be used when the connectivity isn't changing too much in which case the overall cost will not be significantly affected. The same techniques could be used to resolve more complicated snags, but at the expense of a more complicated algorithm.

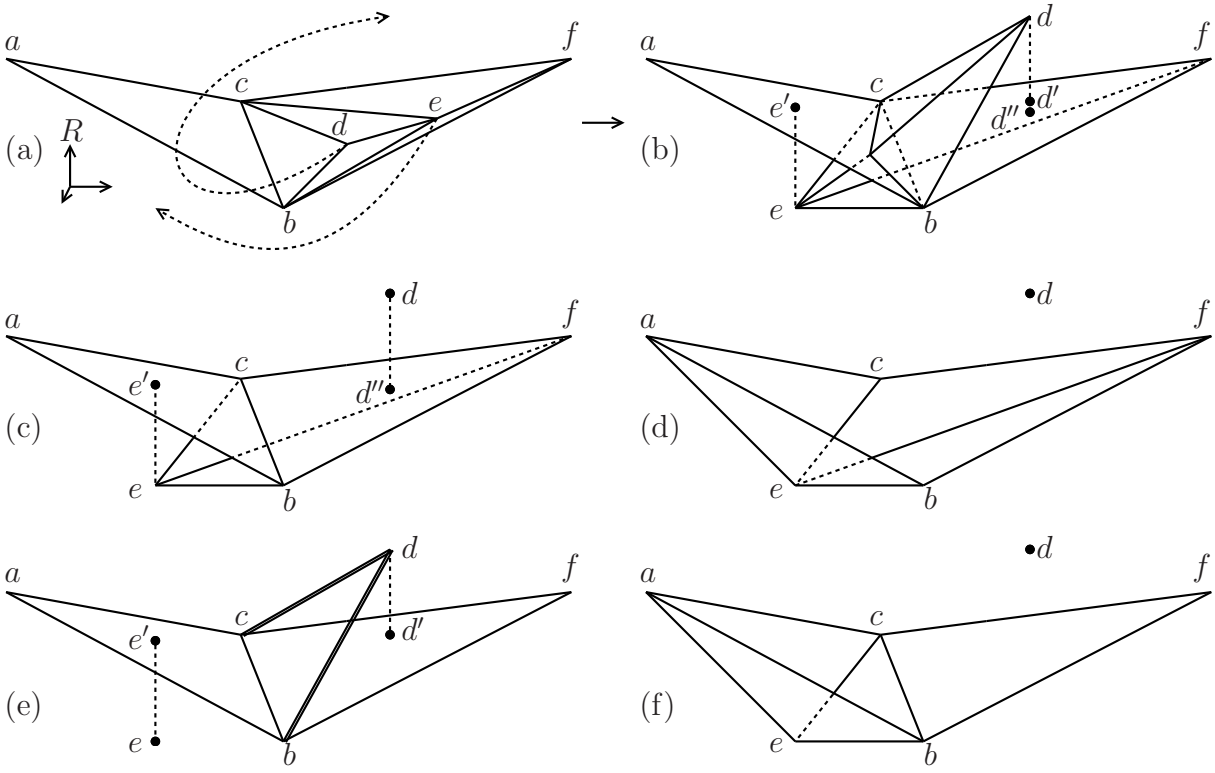


Figure 3.24: A 'snag' and its resolution

The second extension is into three dimensions. For $\boldsymbol{x} \in \mathbb{R}^3$ polyhedral cells, polygonal faces, edges and vertices in physical space are now LF-transformed into vertices, edges, triangular faces and tetrahedra in dual space. There are again two bistellar flips, illustrated in dual space in Fig 3.25. The (1,4)-flip decomposes a tetrahedron into four sub-tetrahedra and the (2,3)-flip rearranges the two tetrahedra separated by the gray triangle (*abcd* and *abec*) into the three tetrahedra shown (*abed*, *bced* and *caed*).

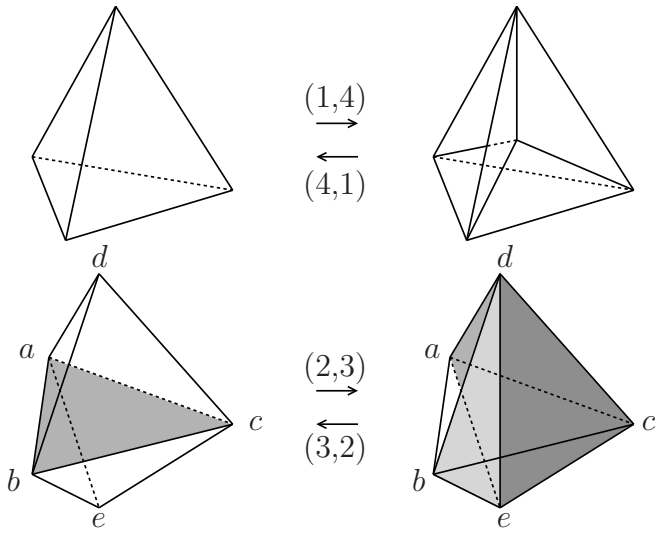


Figure 3.25: Three dimensional geometric bistellar flips (dual space)

Fig 3.26 shows the bistellar flips in physical space.

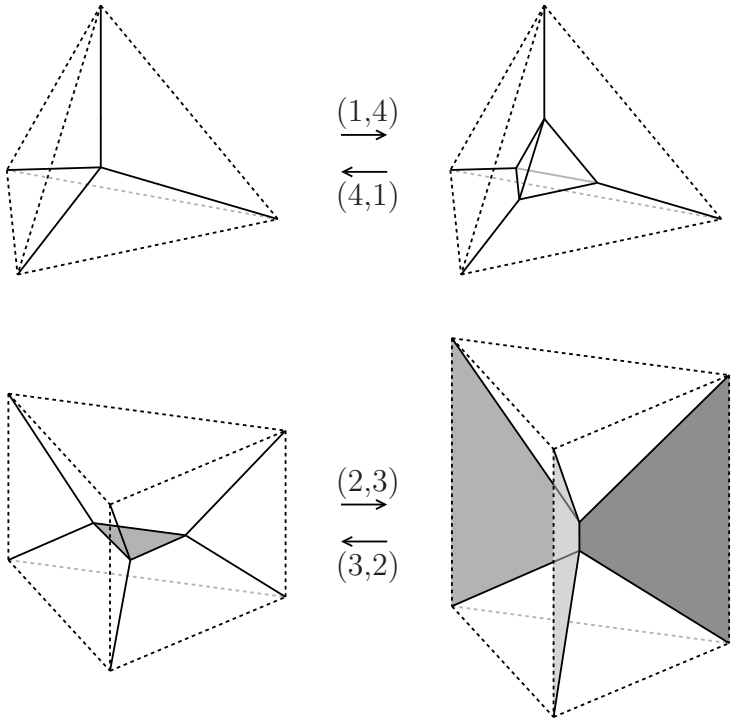


Figure 3.26: Three dimensional geometric bistellar flips (physical space)

Purser's panel beater algorithm in three dimensions [74] loops over the triangular faces in dual space, combines the tetrahedra each side into a four-simplex and computes its volume via the generalisation of (3.11):

$$V_{ijklm} = \frac{1}{4!} \det \begin{pmatrix} X_{ij} & Y_{ij} & Z_{ij} & R_{ij} \\ X_{ik} & Y_{ik} & Z_{ik} & R_{ik} \\ X_{il} & Y_{il} & Z_{il} & R_{il} \\ X_{im} & Y_{im} & Z_{im} & R_{im} \end{pmatrix}. \tag{3.13}$$

If negative it performs a (2,3)-flip, eliminating any opposing pairs of tetrahedra that may result (tetrahedra with the same vertices but opposite sign volume). Again tetrahedra adjacent to a flip will need to be rechecked as well. Snags can also be treated using the same method as before. Reinstating a vertex (c.f. reinstating vertex a in Fig 3.24f) is achieved easily enough by a (1,4)-flip but refilling the arbitrary polyhedral gap left by removing a vertex (c.f. refilling the gap $dbfc$ in Fig 3.24e) with new tetrahedra is slightly more complex. One method would be to pick a vertex on the boundary and join that up to each triangular face on the inside of the gap. However in three dimensions the set of all possible triangulations is not always connected (see [4] for a counterexample) so the panel beater scheme is not guaranteed to succeed, although again as the changes in $\boldsymbol{R}^n$ are expected to be small this would be unlikely to occur in practice.

3.4 Steepest descent methods

The SMKP can also be solved by the steepest descent method. Rewriting (2.22) in terms of ψ instead of φ:

$$M[\psi] = \int_X \psi^*(\boldsymbol{x})\alpha(\boldsymbol{x})\mathrm{d}\boldsymbol{x} + \int_Y \psi(\boldsymbol{y})\beta(\boldsymbol{y})\mathrm{d}\boldsymbol{y}. \tag{3.14}$$

Switching to semi-geostrophic notation, and assuming R is convex ($R^{**} = R$), the variational derivative is

$$M'[R] = \beta - \alpha(\nabla R)\det(D^2 R). \tag{3.15}$$

For the SMKP $\alpha(\boldsymbol{x}) = 1$, $\beta(\boldsymbol{X}) = \sum_i \beta_i \delta(\boldsymbol{X} - \boldsymbol{X}_i)$ and $P(\boldsymbol{x}) = \boldsymbol{x}\cdot\boldsymbol{X}_i - R_i$ for $\boldsymbol{x} \in C_i$ (unless qualified indices range from 1 to N). Plugging this into (3.14) gives

$$\begin{aligned} M[R] &= \sum_i \left[\int_{C_i} (\boldsymbol{x}\cdot\boldsymbol{X}_i - R_i)\mathrm{d}\boldsymbol{x} + R_i\beta_i\right] \\ &= \sum_i \left[\boldsymbol{X}_i \cdot \int_{C_i} \boldsymbol{x}\mathrm{d}\boldsymbol{x} + R_i(\beta_i - A_i)\right]. \end{aligned}$$

To compute the integrals they are decomposed into integrals over the triangles formed by connecting the vertices to the origin O in physical space (Fig 3.27).

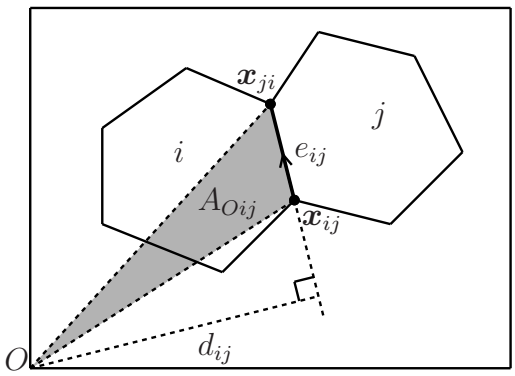


Figure 3.27: Computation of dual cost for the SMKP

Let e_{ij} be the edge having cell i on the left and j on the right and $\boldsymbol{x}_{ij}$ its starting vertex. Then e_{ji} has j on the left and i on the right, so is the same as e_{ij} but travelling in the opposite direction, and starts from $\boldsymbol{x}_{ji}$. With this convention the perpendicular distance to e_{ij} is d_{ij} (3.2) and the area of the shaded triangle is $A_{Oij} = d_{ij}|e_{ij}|/2 = -A_{Oji}$. This definition is also valid for boundary edges e.g. if $i \leqslant N < j \leqslant N_C$ then $d_{ij} = d_j = R_j/|\boldsymbol{X}_j|$. Lastly set $A_{Oij} = 0$ if $N < i, j \leqslant N_C$ or e_{ij} doesn't exist. By using signed areas, cancellation ensures that $A_i = \sum_{j \leqslant N_C} A_{Oij}$ for all $i \leqslant N$. The centroid of the shaded triangle is $(\boldsymbol{x}_{ij} + \boldsymbol{x}_{ji})/3$, so

$$\begin{aligned} \int_{C_i} \boldsymbol{x}\mathrm{d}\boldsymbol{x} &= \sum_{j \leqslant N_C} A_{Oij}(\boldsymbol{x}_{ij} + \boldsymbol{x}_{ji})/3, \\ \sum_i \boldsymbol{X}_i \cdot \int_{C_i} \boldsymbol{x}\mathrm{d}\boldsymbol{x} &= \sum_{i \leqslant N, j \leqslant N_C} A_{Oij}\boldsymbol{X}_i \cdot (\boldsymbol{x}_{ij} + \boldsymbol{x}_{ji})/3 \end{aligned}$$

We split the sum $\sum_{i\leqslant N, j\leqslant N_C} = \sum_{i<j\leqslant N} + \sum_{j<i\leqslant N} + \sum_{i\leqslant N<j\leqslant N_C}$. For the first two parts

$$\begin{aligned}
&\textstyle\sum_{i<j} A_{Oij}\boldsymbol{X}_i\cdot(\boldsymbol{x}_{ij}+\boldsymbol{x}_{ji})/3 + \sum_{j<i} A_{Oij}\boldsymbol{X}_i\cdot(\boldsymbol{x}_{ij}+\boldsymbol{x}_{ji})/3\\
=\ &\textstyle\sum_{i<j} A_{Oij}\boldsymbol{X}_i\cdot(\boldsymbol{x}_{ij}+\boldsymbol{x}_{ji})/3 + \sum_{i<j} A_{Oji}\boldsymbol{X}_j\cdot(\boldsymbol{x}_{ji}+\boldsymbol{x}_{ij})/3\\
=\ &\textstyle\sum_{i<j} A_{Oij}(\boldsymbol{X}_i-\boldsymbol{X}_j)\cdot(\boldsymbol{x}_{ij}+\boldsymbol{x}_{ji})/3\\
=\ &\textstyle\sum_{i<j} A_{Oij}2(R_i-R_j)/3\\
=\ &\textstyle\frac{2}{3}\sum_{i<j} A_{Oij}R_i - \frac{2}{3}\sum_{i<j} A_{Oij}R_j\\
=\ &\textstyle\frac{2}{3}\sum_{i<j} A_{Oij}R_i - \frac{2}{3}\sum_{j<i} A_{Oji}R_i\\
=\ &\textstyle\frac{2}{3}\sum_{i,j\leqslant N} A_{Oij}R_i\\
=\ &\textstyle\frac{2}{3}\sum_{i\leqslant N, j\leqslant N_C} A_{Oij}R_i - \frac{2}{3}\sum_{i\leqslant N<j\leqslant N_C} A_{Oij}R_i
\end{aligned}$$

where the first and fifth steps just swap i and j in the second term. Plugging this in:

$$\begin{aligned}
\textstyle\sum_i \boldsymbol{X}_i\cdot\int_{C_i}\boldsymbol{x}\mathrm{d}\boldsymbol{x} &= \textstyle\frac{2}{3}\sum_{i\leqslant N} R_i \sum_{j\leqslant N_C} A_{Oij} + \sum_{i\leqslant N<j\leqslant N_C} A_{Oij}[\boldsymbol{X}_i\cdot(\boldsymbol{x}_{ij}+\boldsymbol{x}_{ji}) - 2R_i]/3\\
&= \textstyle\frac{2}{3}\sum_i A_iR_i \qquad + \qquad B\\
M[R] &= B + \textstyle\frac{2}{3}\sum_i A_iR_i + \sum_i R_i(\beta_i - A_i) = B + \sum_i R_i(\beta_i - \frac{1}{3}A_i)
\end{aligned}$$

where the boundary term B depends only on the cells just inside the boundary. Away from these, i.e. for $i,j\leqslant N$, (3.4) $\Rightarrow A_{Oij} = (R_j - R_i)J_{ij}/2$ so $A_i = \frac{1}{2}\sum_j R_jJ_{ij}$ (using $\sum_j J_{ij} = 0$) and

$$\begin{aligned}
\frac{\partial M}{\partial R_i} &= \frac{\partial}{\partial R_i}\sum_j R_j(\beta_j - \frac{1}{3}A_j) = \beta_i - \frac{1}{3}A_i - \frac{1}{3}\sum_j R_jJ_{ji}\\
&= \beta_i - \frac{1}{3}A_i - \frac{1}{3}(2A_i) = \beta_i - A_i
\end{aligned}$$

which is the discrete analogue of (3.15). Differentiating again $\partial^2 M/\partial R_i\partial R_j = -J_{ij}$ and from (3.6) $-\boldsymbol{R}^T\boldsymbol{J}\boldsymbol{R}\geqslant 0$, as expected for a minimum. Furthermore $\partial M/\partial\boldsymbol{R} = \boldsymbol{F}$ (3.3) so minimising M is equivalent to solving $\boldsymbol{F} = 0$. The steepest descent step is

$$\boldsymbol{R}^{n+1} = \boldsymbol{R}^n - \tau_n(\boldsymbol{\beta} - \boldsymbol{A}(\boldsymbol{R}^n)), \tag{3.16}$$

the SMKP version of (2.25). The steepest descent method is slow, converging linearly with iteration near a minimum, because it doesn't make use of any derivative information (e.g. the Jacobian $\boldsymbol{J}$). Also τ_n must be chosen somehow, typically either from a line search along $\boldsymbol{F}$ (expensive) or set to a constant (very slow convergence, but the algorithm is then parallelizable). A simple improvement is to pre-condition by approximating the full Jacobian by its diagonal which can be inverted explicitly:

$$R_i^{n+1} = R_i^n - \tau(\beta_i - A_i(\boldsymbol{R}^n))/J_{ii}. \tag{3.17}$$

For the SMKP the dual cost $J[\widetilde{\varphi}, \widetilde{\psi}]$ becomes

$$\begin{aligned} J[R] &= \int_\Omega \tfrac{1}{2}|\boldsymbol{x}|^2\alpha(\boldsymbol{x})\mathrm{d}\boldsymbol{x} + \int_{\Omega_c} \tfrac{1}{2}|\boldsymbol{X}|^2\beta(\boldsymbol{y})\mathrm{d}\boldsymbol{y} - M[R] \\ &= \int_\Omega \tfrac{1}{2}|\boldsymbol{x}|^2\mathrm{d}\boldsymbol{x} + \textstyle\sum_i \tfrac{1}{2}|\boldsymbol{X}_i|^2\beta_i - M[R]. \end{aligned}$$

The first two terms are independent of R so $J'[R] = -M'[R]$. The cost is

$$\begin{aligned} I[R] &= \int_\Omega \tfrac{1}{2}|\boldsymbol{x} - \boldsymbol{X}(\boldsymbol{x})|^2\alpha(\boldsymbol{x})\mathrm{d}\boldsymbol{x} \\ &= \textstyle\sum_i \int_{C_i} \tfrac{1}{2}|\boldsymbol{x} - \boldsymbol{X}_i|^2\mathrm{d}\boldsymbol{x} \\ &= J[R] + \textstyle\sum_i (\tfrac{1}{2}|\boldsymbol{X}_i|^2 - R_i)(A_i - \beta_i). \end{aligned}$$

Recall that for the MP, the unique minimum occurs when ψ and φ are Legendre transforms and the rearrangement criteria is met. At this minimum the cost and dual cost coincide, to the Wasserstein distance. Here R and P are Legendre transforms by construction and when in addition the rearrangement criteria is met ($\boldsymbol{A} = \boldsymbol{\beta}$) we see that $I[R] = J[R]$ as expected. Differentiating,

$$\frac{\partial I}{\partial R_i} = \sum_j (\frac{1}{2}|\boldsymbol{X}_j|^2 - R_j)J_{ji}.$$

When $R_i = \frac{1}{2}|\boldsymbol{X}_i|^2 \ \forall i$ then

$$d_{ij} = \frac{R_j - R_i}{|\boldsymbol{X}_j - \boldsymbol{X}_i|} = \frac{\frac{1}{2}|\boldsymbol{X}_j|^2 - \frac{1}{2}|\boldsymbol{X}_i|^2}{|\boldsymbol{X}_j - \boldsymbol{X}_i|} = \frac{(\boldsymbol{X}_j + \boldsymbol{X}_i)\cdot(\boldsymbol{X}_j - \boldsymbol{X}_i)}{2|\boldsymbol{X}_j - \boldsymbol{X}_i|} = \left(\frac{\boldsymbol{X}_i + \boldsymbol{X}_j}{2}\right)\cdot\hat{\boldsymbol{n}}_{ij}$$

so points $\boldsymbol{x}$ on the line L_{ij} satisfy $(\boldsymbol{x} - \frac{\boldsymbol{X}_i + \boldsymbol{X}_j}{2})\cdot\hat{\boldsymbol{n}}_{ij} = 0$. Thus L_{ij} is perpendicular to the line $\boldsymbol{X}_i \to \boldsymbol{X}_j$ and also passes through its midpoint so is the perpendicular bisector. This is true for all edges with the result that the physical mesh is Voronoi and the corresponding dual space triangulation is Delauney. The power diagram representation (section 2.5.4) provides a more direct derivation - when $R_i = \frac{1}{2}|\boldsymbol{X}_i|^2$ the weights w_i are all constant. Note that the Voronoi mesh is only a solution of the SMKP if the rearrangement condition is also met - Fig 2.9 displays an example in which the solution is not Voronoi.

3.5 Test problems

In this section simple test problems are used to compare the performance of the various schemes for the solution of the SMKP, and further investigate their properties.

3.5.1 Static smooth β

The first test problem is set up as follows. Both domains are square: $\Omega = [0,2] \times [0,2]$ and recall $\Omega_c = [0,1] \times [0,1]$ is fixed. The dual space vertices $\boldsymbol{X}_i$ are located at $(j/N_x, k/N_y)$, $0 \leqslant j \leqslant N_x$, $0 \leqslant k \leqslant N_y$, and $((j+\frac{1}{2})/N_x, (k+\frac{1}{2})/N_y)$, $0 \leqslant j \leqslant N_x - 1$, $0 \leqslant k \leqslant N_y - 1$, where $N_x = N_y = 200$, forming a lattice of equilateral triangles. Each X_i, Y_i is then displaced by a uniform random number in the range $[0, 0.001]$ and a Voronoi mesh set up by $R_i = X_i^2 + Y_i^2$. The convex hull routine is used to construct the mesh, the cells of which will have slightly randomised areas so the geometric method is applied with $\beta_i = |\Omega|/N$ to obtain equal areas. When both domains are square and $N_x = 2N_y$ or $2N_x = N_y$ the resulting physical mesh is (randomised) hexagonal with distance between the centroids (a convenient length scale for later)

$$R_0 = \sqrt{\frac{2|\Omega|}{N_C\sqrt{3}}} \tag{3.18}$$

The resulting physical and dual space meshes are displayed in Figs 3.28, 3.29 and 3.30.

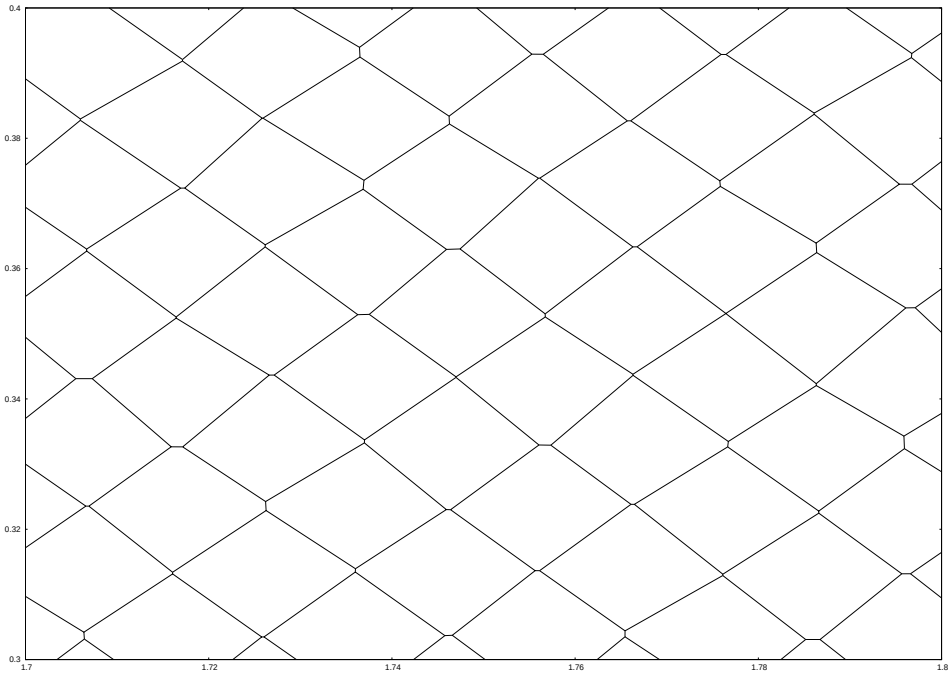


Figure 3.28: Close-up of the initial mesh

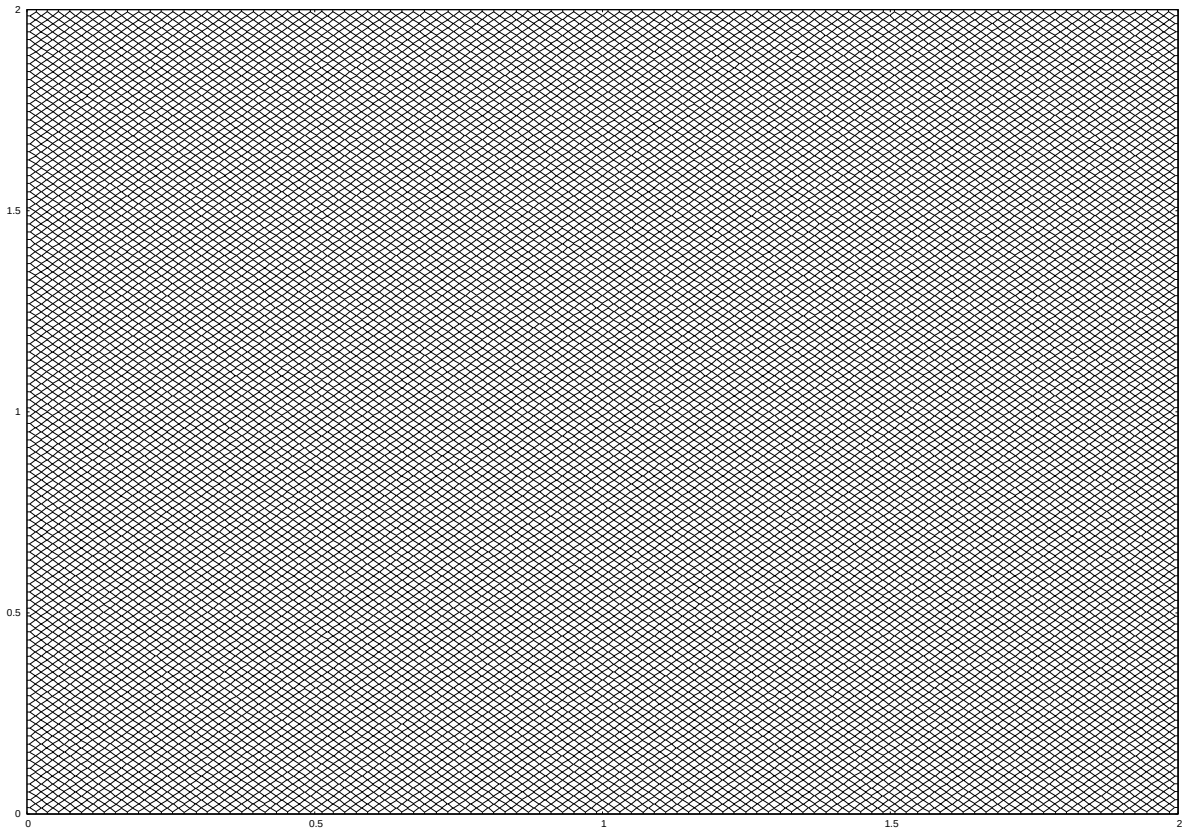


Figure 3.29: Initial mesh for static smooth $\boldsymbol{\beta}$ test

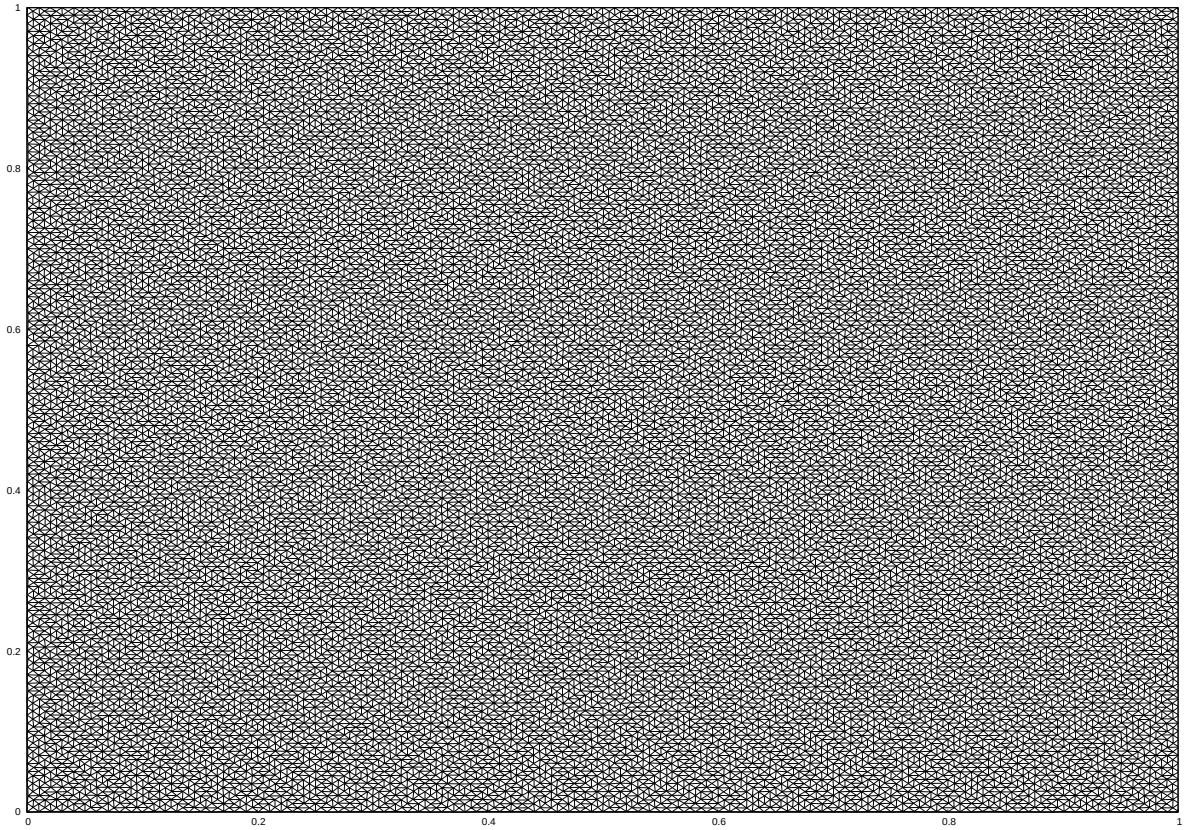


Figure 3.30: Initial dual space mesh for static smooth $\boldsymbol{\beta}$ test

A target area is set for each cell:

$$\beta_i = 1.1 + \sin(5\pi\bar{x}_i/2)\sin(5\pi\bar{y}_i/2),$$

where $\overline{\boldsymbol{x}}_i = (\bar{x}_i, \bar{y}_i) = \int_{C_i} \boldsymbol{x}\mathrm{d}\boldsymbol{x}/A_i$ is the centroid of cell C_i, and the areas are then normalised to fill the domain:

$$\beta_i \leftarrow \beta_i \left(\frac{|\Omega|}{\sum_i \beta_i} \right). \tag{3.19}$$

Either these target areas remain fixed ('static') throughout the test or are recalculated each time the cells move and $\overline{\boldsymbol{x}}_i$ changes ('dynamic'). We compare the steepest descent method (3.16) with $\tau_n = 1/\max_i(J_{ii})$, the pre-conditioned steepest descent method (3.17) with $\tau = 1/3$, and the conjugate gradient method (3.8) with $N_{gm} = 3$ and stopping criteria $\|\boldsymbol{\beta} - \boldsymbol{A}(\boldsymbol{R}^n)\| < \mathrm{tol}\|\boldsymbol{\beta}\|$. These values were found by trial and error as the largest possible that do not set any cells adrift. The solution $\boldsymbol{R}^*$ (such that $\boldsymbol{A}(\boldsymbol{R}^*) = \boldsymbol{\beta}$) is approximated by applying the conjugate gradient method with a tolerance close to machine accuracy ($\mathrm{tol} = 10^{-14}$), and is shown in Fig 3.31. Fig 3.32 shows the number of edge-flips that

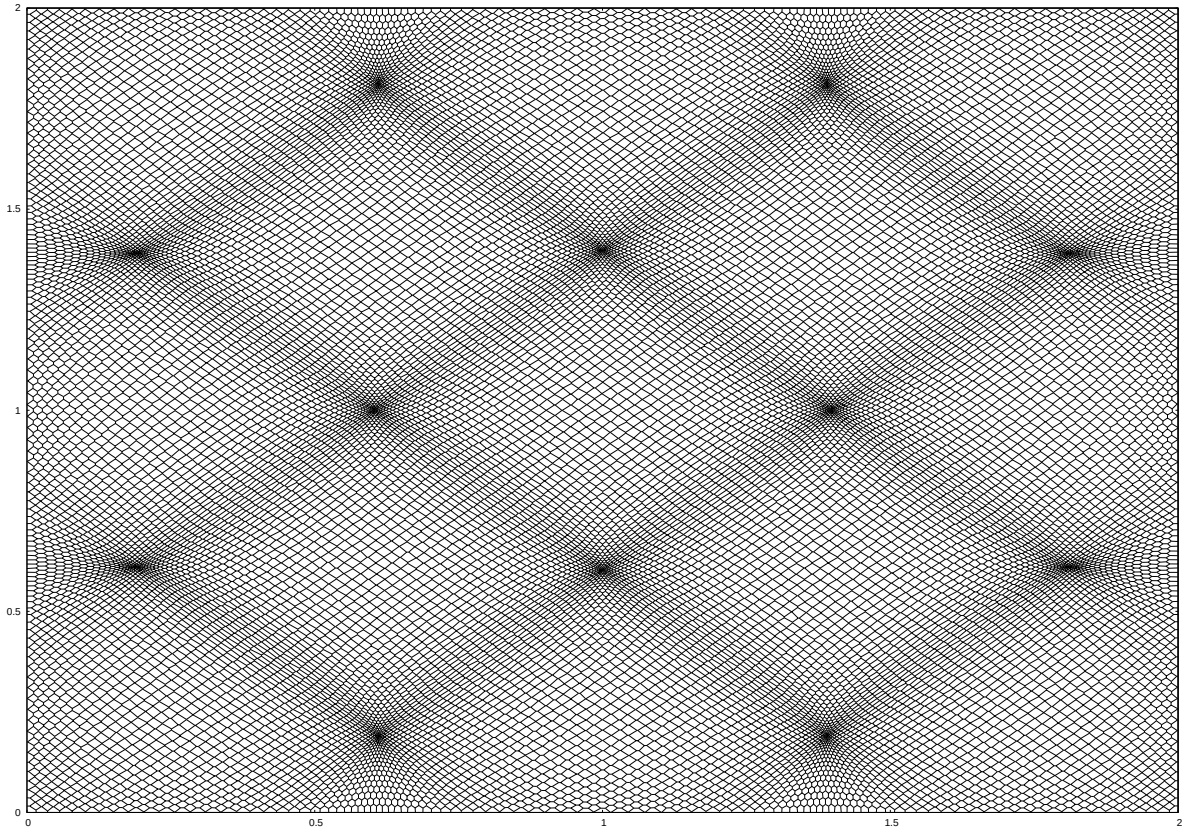


Figure 3.31: Converged mesh for static smooth $\boldsymbol{\beta}$ test

occurred on the boundary of each cell by this point. It ranges from zero to eight, tending to be lower where cells have changed size less but is otherwise fairly random, as might be expected.

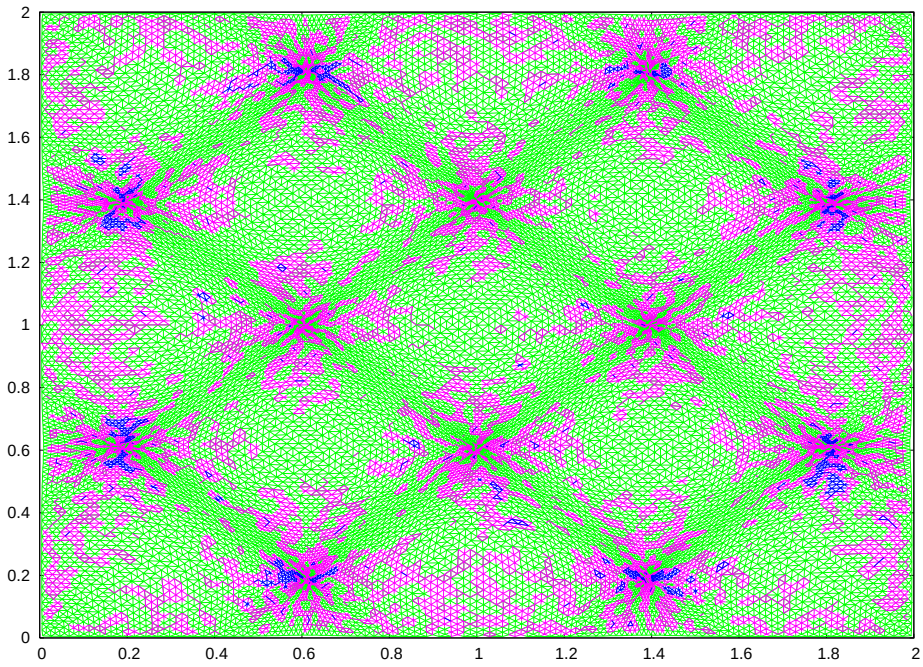
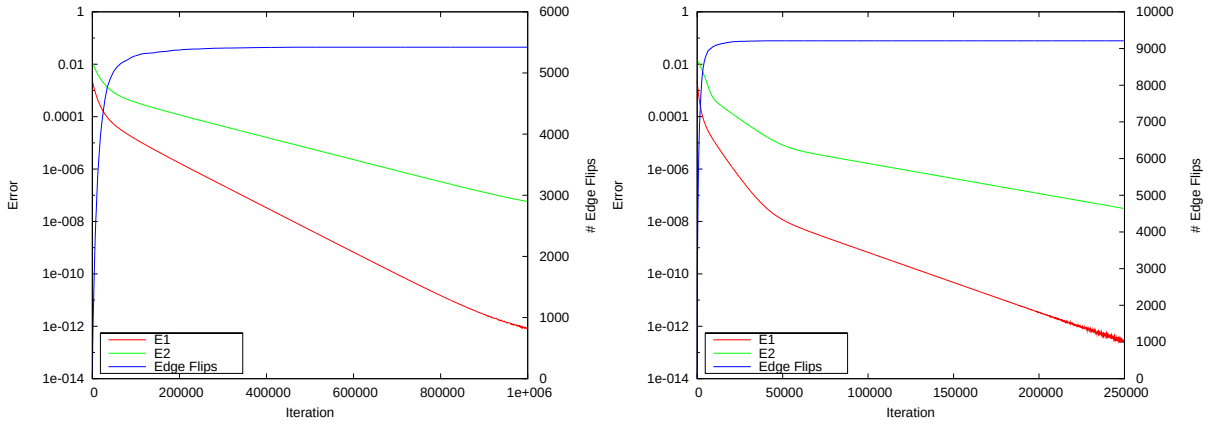


Figure 3.32: Plot of edge-flips per cell for static smooth $\boldsymbol{\beta}$ test (green=0-2, pink=2-4, blue=4-8)

Two metrics are used to compare the schemes for static $\boldsymbol{\beta}$:

$$E_1 = M[\boldsymbol{R}^n] - M[\boldsymbol{R}^*], \quad E_2 = \|\boldsymbol{F}^n\| = \|\boldsymbol{\beta} - \boldsymbol{A}(\boldsymbol{R}^n)\|$$

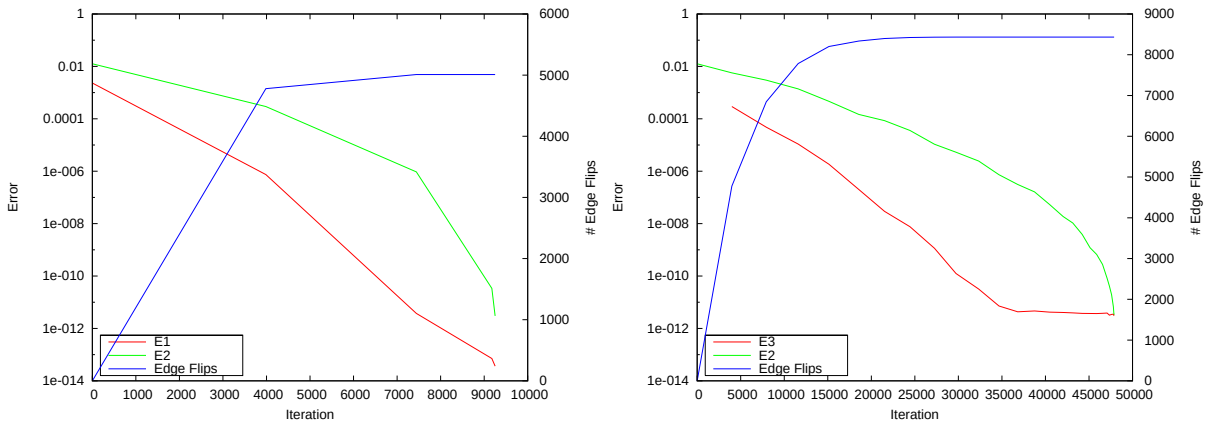
As expected the steepest descent and pre-conditioned steepest descent (Fig 3.33) converge linearly with iteration (after all edge flips have occurred), with the latter about four times faster.



steepest descent pre-conditioned steepest descent

Figure 3.33: Convergence histories for descent-based methods with static smooth $\boldsymbol{\beta}$ test

Fig 3.34 shows the corresponding results for the conjugate gradient method. It is applied repeatedly, each application or *cycle* comprising N_{gm} steps with tol $= 10^{-10}$ throughout.



static $\boldsymbol{\beta}$ dynamic $\boldsymbol{\beta}$

Figure 3.34: Convergence histories for the conjugate gradient method

The method is much faster, with E_2 displaying quadratic convergence with cumulative iteration. The paucity of data points indicates another benefit of the method - it is not necessary to recompute the Jacobian and apply the panel beater algorithm to update the mesh connectivity every iteration because close to convergence slight errors in the mesh/Jacobian are unlikely to set cells adrift or significantly affect the convergence rate. For the descent-based methods the mesh/Jacobian could have been updated less frequently, for example at some fixed frequency of iterations, but this would introduce another *ad hoc* parameter to the method. However with the conjugate gradient method there is a natural choice - update after each cycle. Here convergence occurred after just four cycles hence four data points and a running time of seconds compared with hours for the descent-based methods. For this reason only the conjugate gradient method is used from here on.

3.5.2 Dynamic smooth β

Fig 3.34 shows the convergence history for the same problem but with $\boldsymbol{\beta}$ now continually updated. Now we are chasing a moving target and in this case $M[R^n]$ does not converge monotonically to $M[R^*]$ so E_1 is discarded in favour of

$$E_2 = \|\boldsymbol{\beta}^n - \boldsymbol{A}(\boldsymbol{R}^n)\|, \quad E_3 = \|\boldsymbol{\beta}^n - \boldsymbol{A}(\boldsymbol{R}^{n+1})\|.$$

E_2 now describes how far away from the target we are at the start of each cycle (i.e. how far we want to go) and E_3 how far away at the end (i.e. how close we got). By about 35,000 iterations E_3 has dropped close to machine accuracy indicating we reached the target in that cycle, and again E_2 converges quadratically. Overall 27 cycles were required for this problem, compared with 4 for the static $\boldsymbol{\beta}$.

3.5.3 Static nonsmooth β

For this problem the domain is $\Omega = [0,1] \times [0,1]$ and the dual space mesh has $N_x = 10$, $N_y = 20$ with random offsets in the range $[0, 0.001]$ as before. At the top of Fig 3.35 is the initial mesh in which the cells have equal areas. Then the target areas of the cells whose centroid lies inside the circle are reduced:

$$\beta_i = \begin{cases} 1 & \text{if } (\bar{x}_i - 0.25)^2 + (\bar{y}_i - 0.6)^2 \leqslant 0.15^2 \\ 3 & \text{otherwise} \end{cases}$$

resulting in the middle mesh (taking 2 cycles to converge). As the cells in the circle shrink, the normalisation (3.19) causes cells across the entire domain to expand to fill the space. This is visible at the bottom of the figure, in which the arrows indicate the overall displacement of the cell centroids. This can be viewed as a positive property of the scheme - cells can respond immediately to changing conditions throughout the entire domain and move automatically to where they are needed - but can also result in limit cycles, as the final example demonstrates.

3.5.4 Dynamic nonsmooth β

Using the same initial mesh as before, the target areas of any cells whose centroids lie in a narrow vertical band (shaded gray in Fig 3.36) are reduced:

$$\beta_i = \begin{cases} 0.03 & \text{if } 0.4 \leqslant \bar{x}_i \leqslant 0.4256 \\ 1 & \text{otherwise} \end{cases}$$

This example is representative of problems containing line features such as discontinuities over which the mesh is to be concentrated. After a few cycles the mesh begins to oscillate between the two positions shown. What is happening is that a few of the small cells in the upper figure have just moved so their centroids are now to the left of the gray band, in the process pulling some of the centroids of the larger cells to their right into the band. Next iteration (lower figure) those smaller cells have expanded so their centroids are back in the band, and vice-versa for the larger cells. Their target areas then return to their former values, so the mesh is then adjusted back to its former position, and the loop continues.

In this example the limit cycle has period two, but if the problem is modified slightly e.g. by widening the gray band, limit cycles are soon encountered with other periods such

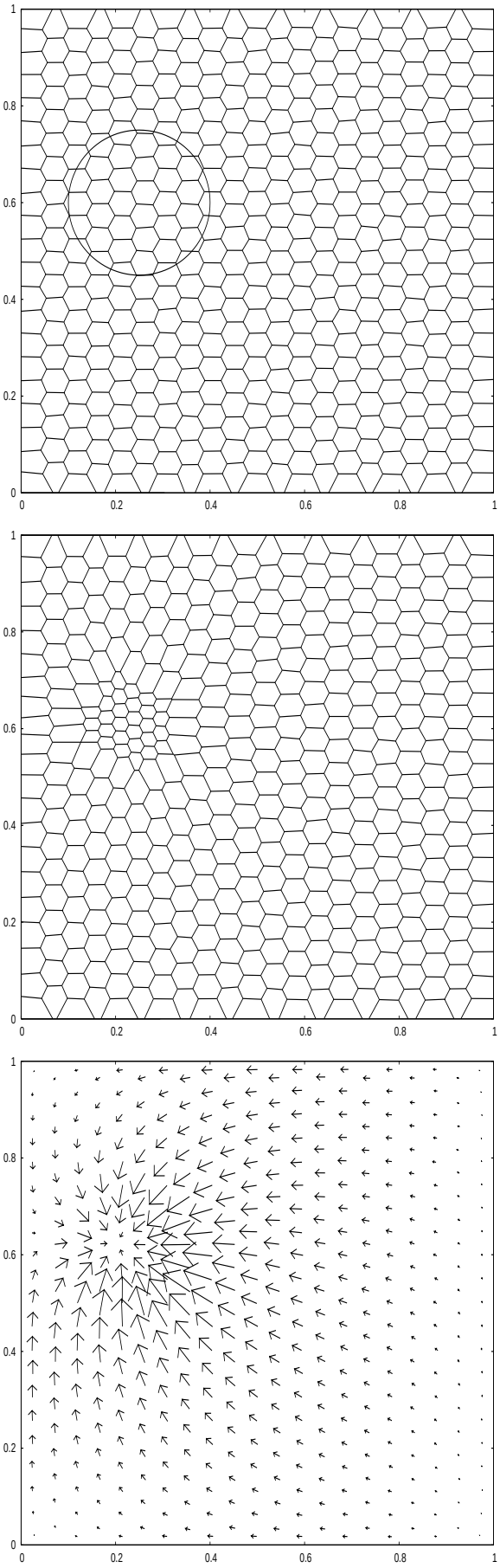


Figure 3.35: Initial mesh (top), final mesh (middle) and centroid displacement (bottom)

as 3,4,6,8,9,10,11 etc. This is potentially very serious because any unintended movement of the mesh could affect the quality or even stability of the solution to the model equations. It could be argued that this example is quite tough because often the model will contain dissipative processes such as viscosity that tend to stabilise the mesh movement. However, when setting up a problem by adapting the mesh to the initial conditions the model equations cannot help because there is no time evolution.

It should be noted that this problem is not specific to the geometric method and a standard remedy in moving mesh methods is to smooth the monitor function (here target areas) in space or time [42]. The latter is easier to apply, with:

$$\boldsymbol{\beta}^n \leftarrow \boldsymbol{\beta}^{n-1} + \lambda_a(\boldsymbol{\beta}^n - \boldsymbol{\beta}^{n-1}) \tag{3.20}$$

for some constant $0 < \lambda_a \leqslant 1$ which determines what fraction of the change in target area to accept each cycle. In this example dropping λ_a to 0.98 dampens the mesh movement sufficient for convergence, in 13 cycles.

To summarize: we have described in detail Purser's algorithm and demonstrated that it solves the SMKP efficiently for fixed target areas $\boldsymbol{\beta}$, but when the target areas are variable, straightforward iteration can, in certain circumstances, lead to undesirable behaviour such as limit cycles.

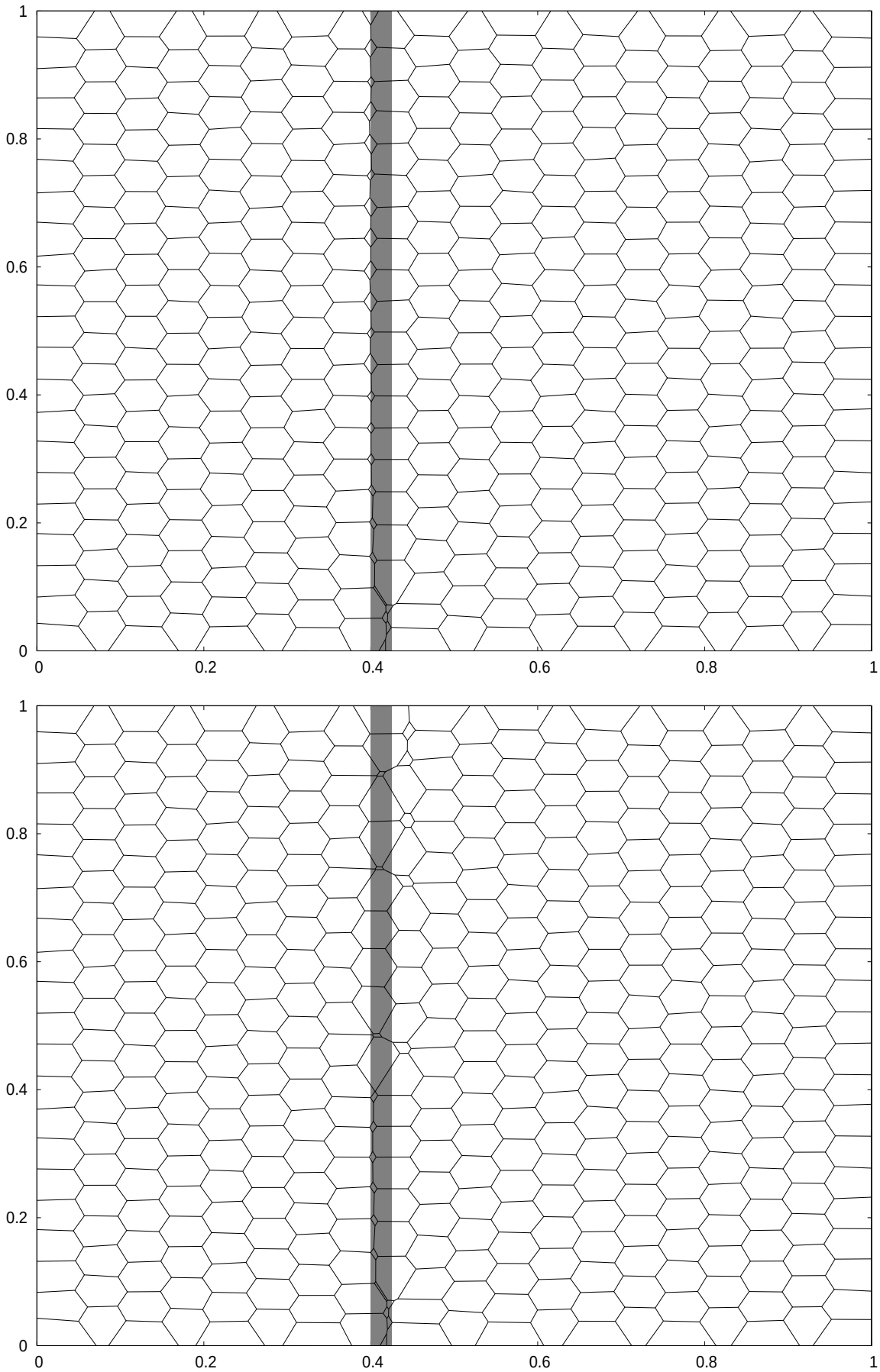


Figure 3.36: A limit cycle of period two

Chapter 4

Application to the Euler equations

The preceding chapters detailed a method that generates an unstructured polygonal mesh from prescribed cell areas. This purpose of this chapter is to describe a discretization scheme for the Euler equations that can advance the solution from one such mesh at the start of a timestep to another at the end.

The choice of discretization is determined primarily by the fact that the mesh connectivity can change during the timestep. To accommodate this would significantly complicate the set of finite element basis functions, so the finite volume method is chosen instead. As vertices and edges jump around when the connectivity changes, all variables are cell-centred. In this regard it is similar to mesh-free schemes such as SPH [63] or Free-Lagrange [8], however they do not explicitly model the solution through the connectivity changes as here. The scheme has been adapted from the Godunov Linear Flux Correction (GLFC) scheme of Azarenok *et al* [6]. Godunov's scheme is only first order accurate due to the piecewise constant reconstruction of the variables used in the Riemann solver. One method of achieving higher order accuracy is to use piecewise polynomial reconstructions and solve a generalized Riemann problem (GRP) at cell boundaries e.g. the ADER approach [88]. In contrast the GLFC is a MUSCL-Hancock two-step method in which the predictor provides half-timestep values used as initial data for a piecewise constant Riemann problem in the corrector. Furthermore it solves the Riemann problem on the moving mesh where possible to minimise the diffusive effects of interpolation or remapping.

Since the development of this scheme Shashkov [81] has extended the ALE method to handle changing mesh connectivity in a similar manner by introducing a single intermediate mesh and computing fluxes via swept regions over each phase.

In the following we reserve the vector $\boldsymbol{x}$ for coordinates in space i.e. $\boldsymbol{x} = (x, y)$, and $\boldsymbol{r}$ for coordinates in space-time, i.e. $\boldsymbol{r} = (x, y, t)$.

4.1 The Euler equations

The Euler equations for an ideal gas in planar or axisymmetric geometry are

$$\frac{\partial \boldsymbol{U}}{\partial t} + \frac{\partial \boldsymbol{F}}{\partial x} + \frac{\partial \boldsymbol{G}}{\partial y} = \boldsymbol{S} \tag{4.1}$$

where

$$\boldsymbol{U} = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ E \end{pmatrix}, \ \boldsymbol{F} = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ u(E+p) \end{pmatrix}, \ \boldsymbol{G} = \begin{pmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ v(E+p) \end{pmatrix}, \ \boldsymbol{S} = \frac{\nu v}{y} \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ E+p \end{pmatrix}, \ \boldsymbol{W} = \begin{pmatrix} \rho \\ u \\ v \\ p \end{pmatrix} \tag{4.2}$$

and ρ, u, v, e are the density, components of velocity and specific internal energy respectively. The total energy $E = \rho\left[e + \frac{1}{2}(u^2+v^2)\right]$, the pressure p is given by the ideal gas equation of state $p = (\gamma - 1)\rho e$ where γ is the heat capacity ratio ($\gamma = 5/3$ for a monatomic gas). The vector $\boldsymbol{U}$ contains the conserved variables (mass, momentum and total energy), $\boldsymbol{F}$ and $\boldsymbol{G}$ their fluxes in the x and y directions respectively, $\boldsymbol{S}$ is a pseudo-source term in which $\nu = 0/-1$ for planar/axisymmetric problems, and $\boldsymbol{W}$ collects the primitive or flow variables.

4.2 Construction of the finite volume

To solve (4.1) numerically it is integrated over a finite (control) volume $\mathcal{V}$. To the two dimensions of space Azarenok adds time as a third to construct a three dimensional *space-time* finite volume for each cell by joining up the vertices at the start of a timestep ($t = t^n$) to the corresponding vertices at the end ($t = t^{n+1}$), as shown in Fig 4.1. In general the sides are ruled surfaces not planes.

This method leaves a gap whenever a connectivity change occurs between the two timesteps, as shown in Fig 4.2a. Here the cell c becomes a new neighbour of cell a during the timestep as cells b and d move apart, and the maroon tetrahedron in between is not a part of any cell. We fill the gap by decomposing the tetrahedron into four smaller tetrahedra, formed by joining the centroid to each of the vertices (Fig 4.2b), and assigning each sub-tetrahedron to the adjacent cell. In this example the maroon sub-tetrahedron is assigned to cell a whose resulting space-time volume $\mathcal{V}$ is shown in Fig 4.2c.

If adjacent connectivity changes occur then the tetrahedral gaps will join up to form more general polyhedral gaps. Fig 4.3 shows an extreme example in which there are a

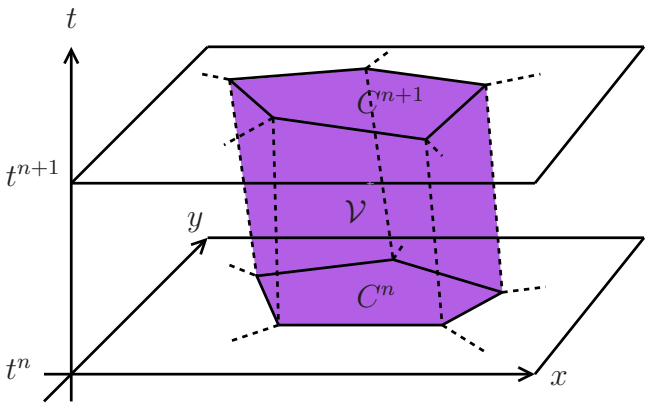


Figure 4.1: Construction of the space-time finite volume $\mathcal{V}$ for a cell

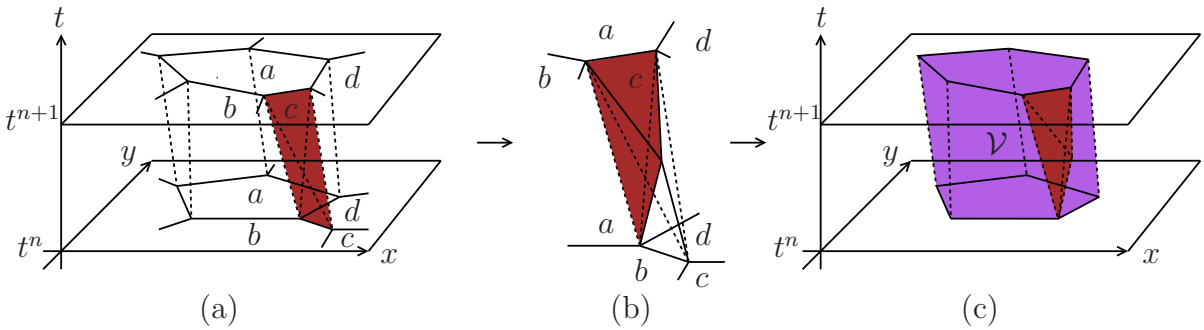


Figure 4.2: Construction of the space-time finite volume $\mathcal{V}$ for a cell with single connectivity change

number of very short edges on the right hand side of the red cell. If R_{red} is reduced so that the cell expands out to the red line, a number of connectivity changes occur. Two views of the subsequent gap are shown in Fig 4.4.

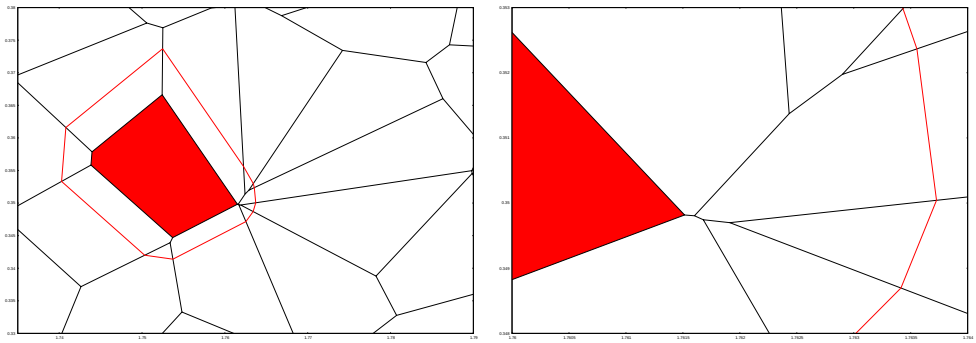


Figure 4.3: An example of adjacent connectivity changes

The above procedure - join all the vertices to the centroid of the gap - can still be applied and three different views of the result are shown in the top row and bottom left of Fig 4.5. The decomposition is successful in that no gaps remain, and furthermore other than the single point at the centroid, cells do not come into contact with any cells they were not already in contact with. However some cells are poorly shaped, such as the one

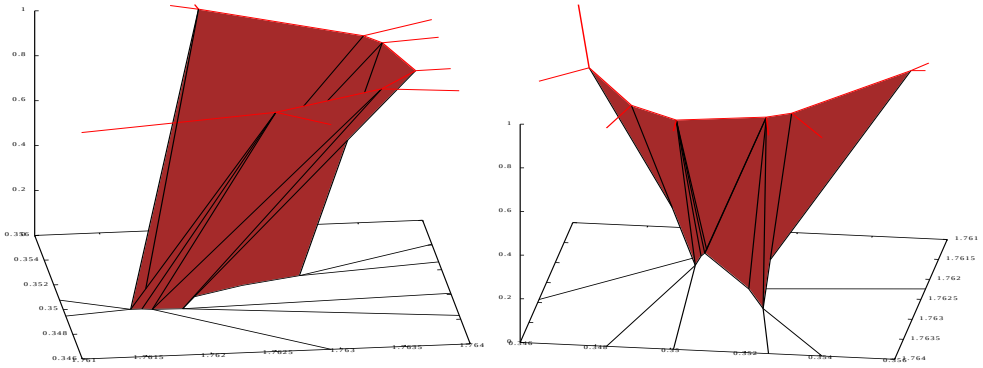


Figure 4.4: The polyhedral gap in space-time

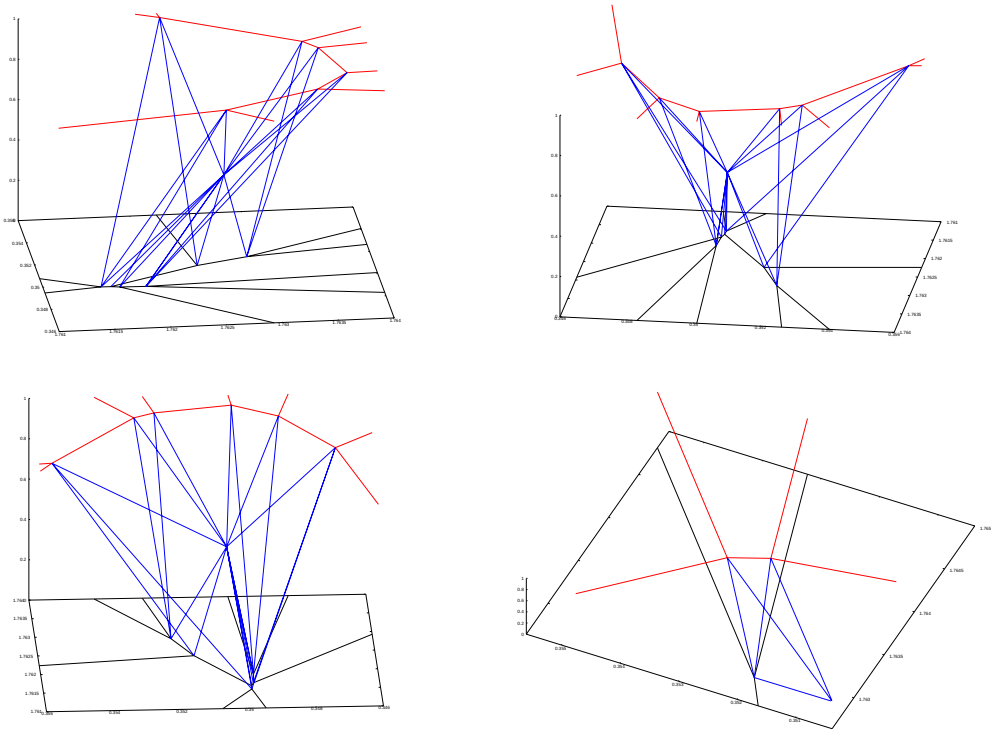


Figure 4.5: Example decomposition for adjacent connectivity changes

shown in the bottom right of the figure, which could affect the quality of the solution. More generally, if the gap is not star-shaped then self-intersecting cells will be generated. To rectify this more points would need to be inserted into the gap necessitating a more sophisticated decomposition algorithm.

4.3 Discretization Scheme

Assuming a valid space-time decomposition, the Euler equations are integrated over each space-time cell $\mathcal{V}_i$, $i = 1, \ldots, N$. The application of Gauss' law to (4.1) for cell i yields

$$\int_{\mathcal{V}_i} \left[\frac{\partial \boldsymbol{U}}{\partial t} + \frac{\partial \boldsymbol{F}}{\partial x} + \frac{\partial \boldsymbol{G}}{\partial y} - \boldsymbol{S} \right] \mathrm{d}V = \oint_{\partial \mathcal{V}_i} (\boldsymbol{U}, \boldsymbol{F}, \boldsymbol{G}) \cdot \mathrm{d}\boldsymbol{n} - \int_{\mathcal{V}_i} \boldsymbol{S} \mathrm{d}V = \boldsymbol{0} \qquad (4.3)$$

which is valid for discontinuous as well as continuous solutions. Equation (4.3) is discretized thus:

$$A_i^{n+1} \boldsymbol{U}_i^{n+1} - A_i^n \boldsymbol{U}_i^n + \sum_j (\boldsymbol{U}_{ij}^{n+1/2}, \boldsymbol{F}_{ij}^{n+1/2}, \boldsymbol{G}_{ij}^{n+1/2}) \cdot \boldsymbol{n}_{ij} = V_i \boldsymbol{S}_i^{n+1/2} \qquad (4.4)$$

where $\boldsymbol{U}_i^n$ is the cell average over C_i^n, centered at the cell centroid $\overline{\boldsymbol{r}}_i^n = (\overline{\boldsymbol{x}}_i^n, t^n)$ (Fig 4.6), and similarly for $\boldsymbol{U}_i^{n+1}$; $\boldsymbol{U}_{ij}^{n+1/2}$, $\boldsymbol{F}_{ij}^{n+1/2}$ and $\boldsymbol{G}_{ij}^{n+1/2}$ are averages over the face $\mathcal{F}_{ij}$ between $\mathcal{V}_i$ and neighbouring $\mathcal{V}_j$, centered at the face centroid $\overline{\boldsymbol{r}}_{ij} = (\overline{\boldsymbol{x}}_{ij}, t_{ij})$; and $\boldsymbol{S}_i^{n+1/2}$ is the average over the space-time cell $\mathcal{V}_i$, centered at the space-time cell centroid $\overline{\boldsymbol{r}}_i$.

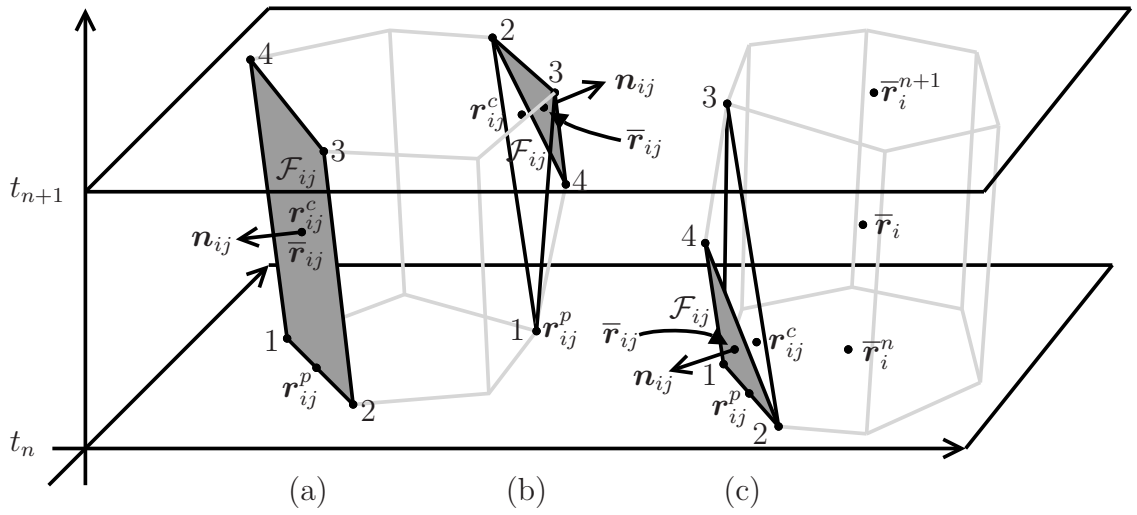


Figure 4.6: Variable centering

The geometric factors are V_i, the volume of $\mathcal{V}_i$, and $\boldsymbol{n}_{ij} = \int_{\mathcal{F}_{ij}} \mathrm{d}\boldsymbol{n}$, the integrated face normal over $\mathcal{F}_{ij}$ (outward from $\mathcal{V}_i$). For the quadrilateral face (Fig 4.6a) the vertices $\boldsymbol{r}_k$ are numbered 1 to 4 anticlockwise round the perimeter and $\overline{\boldsymbol{r}}_{ij} = \frac{1}{4} \sum_{k=1}^4 \boldsymbol{r}_k$. From the divergence theorem $\oint_S \mathrm{d}\boldsymbol{n} = \boldsymbol{0}$ for any closed surface S so by setting $S = \mathcal{F}_{ij} - \mathcal{F}'_{ij}$ the integral over $\mathcal{F}_{ij}$ is equal to the integral over any surface $\mathcal{F}'_{ij}$ spanning $\partial \mathcal{F}_{ij}$. Here we set $\mathcal{F}'_{ij}$ to be the two triangular faces $(\boldsymbol{r}_1, \boldsymbol{r}_2, \boldsymbol{r}_3)$ and $(\boldsymbol{r}_1, \boldsymbol{r}_3, \boldsymbol{r}_4)$, finding

$$\boldsymbol{n}_{ij} = (\boldsymbol{r}_2 - \boldsymbol{r}_1) \times (\boldsymbol{r}_3 - \boldsymbol{r}_1) + (\boldsymbol{r}_3 - \boldsymbol{r}_1) \times (\boldsymbol{r}_4 - \boldsymbol{r}_1) = (\boldsymbol{r}_3 - \boldsymbol{r}_1) \times (\boldsymbol{r}_4 - \boldsymbol{r}_2).$$

For the triangular faces (Fig 4.6b,c) we number the vertices $\boldsymbol{r}_k$ of each tetrahedron 1 to 4, with 4 the newly created vertex and 1 to 3 anticlockwise around it. This generates three triangular faces $(\boldsymbol{r}_1, \boldsymbol{r}_2, \boldsymbol{r}_4)$, $(\boldsymbol{r}_2, \boldsymbol{r}_3, \boldsymbol{r}_4)$ and $(\boldsymbol{r}_3, \boldsymbol{r}_1, \boldsymbol{r}_4)$, which have centroids, normals $\frac{1}{3}(\boldsymbol{r}_1 + \boldsymbol{r}_2 + \boldsymbol{r}_4)$, $(\boldsymbol{r}_2 - \boldsymbol{r}_1) \times (\boldsymbol{r}_4 - \boldsymbol{r}_1)$ etc. (b) shows an example of a new neighbour $\mathcal{V}_j$ joining $\mathcal{V}_i$ and (c) the current neighbour $\mathcal{V}_j$ leaving.

To compute V_i, lines are added from the vertices to the origin forming a 3D region for each face, with the signed volumes summing to V_i. The contribution from the bottom and top faces is $-A_i^n t^n/3$, $A_i^{n+1} t^{n+1}/3$ respectively, triangular faces contribute e.g. $V_{124} = \frac{1}{6}\boldsymbol{r}_1 \cdot \boldsymbol{r}_2 \times \boldsymbol{r}_4$, and Dukowicz [26] has shown that each quadrilateral face contributes

$$V_{1234} = \frac{1}{12}(\boldsymbol{r}_1 + \boldsymbol{r}_2) \cdot (\boldsymbol{r}_1 + \boldsymbol{r}_4) \times (\boldsymbol{r}_2 + \boldsymbol{r}_3).$$

At the start of the predictor phase only the values of the flow variables at $t = t^n$ are known. So predictor values for $\mathcal{F}_{ij}$ are taken from a nearby point $\boldsymbol{r}_i^p = (\boldsymbol{x}_i^p, t^n)$ on the boundary of C_i^n - either the midpoint of an edge if present (as in (a),(c)) or a vertex (b). Similarly corrector values for $\mathcal{F}_{ij}$ are interpolated at a point $\boldsymbol{r}_i^c = (\boldsymbol{x}_i^c, t_i^c)$. For a scheme to be second order (in space) it must be exact when the variables are linear in which case the correct values for the face $\mathcal{F}_{ij}$ are located at $\overline{\boldsymbol{r}}_{ij}$. Unfortunately setting $\boldsymbol{r}_{ij}^c = \overline{\boldsymbol{r}}_{ij}$ can create a problem because for triangular faces $\overline{\boldsymbol{r}}_{ij}$ cannot be written as a convex combination (interpolation) between a point on C_i^n and a point on C^{n+1} (it lies outside the convex hull in space-time between C_i^n and C_i^{n+1}) but only as an extrapolation, so the values of the flow variables at $\overline{\boldsymbol{r}}_{ij}$ will similarly be an extrapolation not an interpolation. This means that even if the flow variables are monotonically limited both in C_i^n and C_i^{n+1}, their values at $\overline{\boldsymbol{r}}_{ij}$ might not be. To guarantee monotonicity on triangular faces the point $\boldsymbol{r}_{ij}^c$ must be moved back into the space-time convex hull e.g. put on the face $(\boldsymbol{r}_1, \boldsymbol{r}_2, \boldsymbol{r}_3)$. For (b) $\boldsymbol{r}_{ij}^c = \frac{1}{3}[\boldsymbol{r}_2 + \boldsymbol{r}_3 + \frac{1}{3}(\boldsymbol{r}_1 + \boldsymbol{r}_2 + \boldsymbol{r}_3)]$ and for (c) $\boldsymbol{r}_{ij}^c = \frac{1}{3}[\boldsymbol{r}_1 + \boldsymbol{r}_2 + \frac{1}{3}(\boldsymbol{r}_1 + \boldsymbol{r}_2 + \boldsymbol{r}_3)]$. Currently both options (second order or monotone) are implemented.

Lastly in the evaluation of $\boldsymbol{S}_i^{n+1/2}$, $\overline{\boldsymbol{r}}_i$ is approximated by $\frac{1}{2}(\overline{\boldsymbol{r}}_i^n + \overline{\boldsymbol{r}}_i^{n+1})$ and $\boldsymbol{W}_i^{n+1/2}$ approximated by $\boldsymbol{W}_i^n$ in the predictor and $\frac{1}{2}(\boldsymbol{W}_i^n + \boldsymbol{W}_i^{n+1})$ in the corrector.

The predictor-corrector scheme for $\mathcal{V}_i$ is:

Predictor

1. Compute $\nabla \boldsymbol{W}_i^n$, the gradients of the flow variables at t^n (section 4.4).
2. Monotonically limit the gradients (section 4.5).
3. For each neighbour j extrapolate the flow variables to $\boldsymbol{r}_{ij}^p$ via $\boldsymbol{W}_{ij}^{n+1/2} = \boldsymbol{W}^n + (\boldsymbol{x}_{ij}^p - \overline{\boldsymbol{x}}_i^n) \cdot \nabla \boldsymbol{W}_i^n$ and derive $\boldsymbol{U}_{ij}^{n+1/2}$, $\boldsymbol{F}_{ij}^{n+1/2}$ and $\boldsymbol{G}_{ij}^{n+1/2}$.
4. Evaluate $\boldsymbol{S}_i^{n+1/2}$ and solve (4.3) for the predictor values $\boldsymbol{U}_i^{n+1}$ and $\boldsymbol{W}_i^{n+1}$.

Corrector

1. Compute $\nabla \boldsymbol{W}_i^{n+1}$, the gradients of the predictor variables at t^{n+1} (section 4.4).
2. Monotonically limit the gradients (section 4.5).
3. For each neighbour j compute the flow variables at $\boldsymbol{r}_{ij}^c$ via $\boldsymbol{W}_{ij}^{n+1/2}=(1-\lambda)\left[\boldsymbol{W}_i^n+(\boldsymbol{x}_{ij}^c-\overline{\boldsymbol{x}}_i^n)\cdot\nabla\boldsymbol{W}_i^n\right]+\lambda\left[\boldsymbol{W}_i^{n+1}+(\boldsymbol{x}_{ij}^c-\overline{\boldsymbol{x}}_i^{n+1})\cdot\nabla\boldsymbol{W}_i^{n+1}\right]$ where $\lambda=(t_{ij}^c-t^n)/\triangle t^n$.
4. Similarly find $\boldsymbol{W}_{ji}^{n+1/2}$ and solve the 1-D Riemann problem normal to $\mathcal{F}_{ij}$ (section 4.6) to get the post-wave $\boldsymbol{W}_{ij}^{n+1/2}$.
5. Compute the fluxes across $\mathcal{F}_{ij}$ for the moving mesh and update the allowed timestep $\triangle t^n$ from the wave-speeds and cell geometries (section 4.7).
6. Repeat for all neighbours, evaluate $\boldsymbol{S}_i^{n+1/2}$ and solve (4.3) for the full-timestep values $\boldsymbol{U}_i^{n+1}$ and $\boldsymbol{W}_i^{n+1}$.

The main algorithm embeds the predictor-corrector scheme into a mesh adaption cycle:

Main algorithm

1. Create a new mesh for time t^{n+1} by copying the mesh at t^n and set the timestep by $\triangle t^n = C_{cfl}\min\{\triangle t^{n-1}\}$ where $C_{cfl}\leqslant 1$ is the CFL coefficient.
2. Advance the flow variables to t^{n+1} using the predictor-corrector scheme above for each cell. If adjacent connectivity changes have occurred or the CFL condition is violated then if present retrieve $\boldsymbol{W}^{n+1}$ from the previous cycle and goto step 6 otherwise remap $\boldsymbol{W}^n$ onto the t^{n+1} mesh (section 4.8) and goto step 1.
3. Compute the monitor and set new cell areas $\boldsymbol{\beta}$ (chapter 5).
4. Apply the geometric method to adjust the mesh at t^{n+1} (chapter 3).
5. Repeat steps 2-4 until the movement of the mesh drops below some tolerance or a maximum number of cycles N_{cycle} (here $N_{cycle}=5$) is reached.
6. If problem endtime reached then exit otherwise set $n\leftarrow n+1$ and goto step 1.

It is possible for cells to be cast adrift in the geometric method if their prescribed areas change too rapidly but this is eliminated by a combination of smoothing the monitor function (chapter 5) and setting N_{gm} sufficiently high (section 3.3).

However adjacent connectivity changes can still arise naturally (e.g. as in Fig 4.3). As the primary focus of this thesis is upon the optimal-transport-based mesh movement, it was decided to minimise any side effects from the novel discretization by avoiding adjacent connectivity changes. Four options were considered: reduce the timestep or restrict the

change in target areas until the connectivity changes occur in separate timesteps, remap the solution from one mesh to the other, or just retrieve the previous cycle of the mesh adaption loop and continue from that.

The first option is inefficient when vertices are close together e.g. as in Fig 4.3 because the allowed timestep can be very small. This is exacerbated by the global nature of the conjugate gradient scheme in the optimal transport problem - any slight change to the target areas in one part of the mesh affects the mesh everywhere else (i.e. the problem is stiff). This also makes computation of the allowed timestep a global problem.

The second option does not work because if the mesh is held back in one timestep it will just try to compensate by moving further the next compounding the problem. The last option is simplest but could run into similar problems, and also requires there to be at least one successful cycle of the mesh adaption loop. Thus in some cases remapping is the only option.

Remapping is an essential part of many algorithms - each timestep in an Eulerian or ALE scheme can be interpreted as a Lagrange step followed by a remap step - so is perfectly acceptable and as long as it is conservative, positivity preserving and second-order accurate the overall order of the scheme will not be affected. As no time evolution will have occurred, for consistency the timestep must be set to zero, so the computational efficiency of the scheme is slightly reduced however. To avoid this the previous cycle of the mesh adaption loop is retrieved when available, otherwise the solution is remapped onto the new mesh. This turns out to be satisfactory in practice.

The procedure is the same if the CFL condition is violated. This would appear to cause unnecessary remapping as the timestep could be reduced instead but in some circumstances this turns out not to work. If the mesh is still moving when the maximum number of cycles is reached, then on the next timestep as long as the flow variables do not change too much the mesh will just carry on moving. Reduce the timestep to zero and the flow variables (and hence monitor and target areas) will be unchanged but the mesh still moves the same amount guaranteeing the CFL condition will be violated at some point. Remapping guarantees progress can always be made.

A final point to make is that if any cells were cast adrift in the geometric method then reinserting them is likely to result in their positions being shifted causing adjacent connectivity changes or CFL violations. Reapplying the geometric method with a higher N_{gm} is therefore the safer option.

4.4 Gradient Estimation by Least-Squares Surface Fitting

The first step in the predictor and corrector is to estimate the gradient of the primitive variables at cell i. To minimise any mesh bias we use the moving least squares (MLS) [55] approach in which a quantity u is approximated in cell i by a polynomial determined by minimising an interpolation error functional over its *support* - a set of local cells. Here we set the support $Sup^n(i)$ of cell i at time t^n to be itself and its immediate neighbours. Dropping the time superscript for clarity, the approximation in cell i is

$$u_i(\boldsymbol{x}) = \sum_I c_I P_I(\boldsymbol{x} - \overline{\boldsymbol{x}}_i) = \boldsymbol{c}^T \boldsymbol{P}(\boldsymbol{x} - \overline{\boldsymbol{x}}_i) \tag{4.5}$$

for a vector of coefficients $\boldsymbol{c}$ and polynomial basis functions

$$\boldsymbol{P}(\boldsymbol{x}) = (1, x, y, x^2, xy, y^2, \ldots).$$

As the data is in the form of cell-averaged values U_i, Garimella *et al* [32] propose an error E which compares the cell-averages of the approximation to the data:

$$E = \sum_{j \in Sup(i)} \left[\frac{1}{A_j} \int_{C_j} u_i(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} - U_j \right]^2$$

but on substituting for $u_i(\boldsymbol{x})$ this will involve integrals of the basis functions over the cells which are expensive to compute for an unstructured mesh so we approximate the integrals with centroid values (or alternatively reinterpret the data as point values located at the cell centroids) to get the cheaper error

$$E = \sum_{j \in Sup(i)} \left[u_i(\overline{\boldsymbol{x}}_j) - U_j \right]^2 .$$

The two error functionals coincide as the cells shrink to zero size and the sum becomes an integral. They also agree when U is linear but differ more as the curvature of U increases, although where the curvature is high e.g. near discontinuities, the gradient is likely to be limited anyway so the difference is not considered significant.

Substituting in for $u_i(\boldsymbol{x})$ and expanding the brackets,

$$E = \boldsymbol{c}^T M \boldsymbol{c} - 2\boldsymbol{e}^T \boldsymbol{c} + E_0$$

where the moment matrix M, $\boldsymbol{e}$ and E_0 are given by

$$M_{IJ} = \sum_{j \in Sup(i)} P_I(\overline{\boldsymbol{x}}_j - \overline{\boldsymbol{x}}_i) P_J(\overline{\boldsymbol{x}}_j - \overline{\boldsymbol{x}}_i)$$

$$e_I = \sum_{j \in Sup(i)} U_j P_I(\overline{\boldsymbol{x}}_j - \overline{\boldsymbol{x}}_i), \quad E_0 = \sum_{j \in Sup(i)} U_j^2.$$

Differentiating with respect to the coefficients c_I, $\nabla E = 2(M\boldsymbol{c} - \boldsymbol{e})$ so that the minimum occurs at $\boldsymbol{c} = M^{-1}\boldsymbol{e}$, when M is invertible. As M is a *Gram* matrix and the $P_I(\boldsymbol{x})$ are linearly independent (over a general region in $\mathbb{R}^2$), it is invertible when the number of linearly independent centroid displacements $\{\overline{\boldsymbol{x}}_j - \overline{\boldsymbol{x}}_i\}_{j \in Sup(i)}$ is greater than the number of coefficients. For example a typical cell in a slightly randomized hexagonal mesh such as Fig 3.35 has six neighbours so six centroid displacements which is sufficient to fit up to quadratic terms in $\boldsymbol{P}$ but no more. A number of options are available for inverting M - because the moment matrix is quite small little difference was found in accuracy or timing between the LU, Cholesky or Singular Value Decomposition (SVD) inversion routines in [73] so the latter is used for safety.

The gradient at cell i is then the coefficients of the basis functions $P_2(\boldsymbol{x}) = x$ and $P_3(\boldsymbol{x}) = y$. The reason for including the constant function ($P_1(\boldsymbol{x}) = 1$) in the basis set is that the extra degree of freedom results in smoother, less mesh-dependent gradients (the fact that the cell average of $u_i(\boldsymbol{x})$ might not now match U_i is irrelevant as only the gradient information is used). No special treatment is required at boundaries.

4.5 Limiting

Discontinuities such as shocks can give rise to spurious oscillations in the physical variables without some form of limiting. This can be applied either in the variable reconstructions (slope limiting) or to the resulting fluxes (flux limiting), although for linear reconstructions the two approaches are equivalent [83]. Various limiting strategies have been proposed, the most popular being Total Variation Diminishing (TVD) schemes [58], MUSCL schemes [90] and Positivity Preserving (PP) (or positive definite, monotone) schemes [83]. In 1D the total variation of U is $TV(U) = \sum_i |U_{i+1} - U_i|$ and a TVD scheme is one for which $TV(U^{n+1}) \leqslant TV(U^n)$. Hyperbolic systems have bounded total variation in 1D but not in higher dimensions where focusing can occur [61], and so TVD schemes become at most first order accurate [35]. PP schemes give positive updates for positive data, or equivalently the cell averages at t^{n+1} are bounded by the cell averages at t^n [84], and in MUSCL schemes the reconstructed variable is bounded by the neighbouring cell averages. PP schemes are most general, including the rest as special cases in 1D [94]. In higher dimensions, simple schemes can be constructed from 1D limiters, but these can

be more diffusive than genuinely multidimensional schemes. Another issue is that often the limiter is used to generate the slopes/fluxes rather than to limit values found by another method e.g. averaging or least-squares fitting. This can lead to artifacts where the scheme switches between different regimes or other ill-effects such as 'staircasing' caused by over-compression in the Superbee scheme [12].

The limiter described here takes as its starting point Hubbard's [47] MUSCL scheme for triangular cells. First the linear reconstruction of a quantity u in cell i is generated from its cell-averaged value U_i and the least-squares gradient (section 4.4). Hubbard then limits the reconstruction at the middle of each edge to lie in the range of the minimum/maximum cell averages of the two cells either side. We sample the reconstruction at other points so instead limit the reconstruction at each vertex $v \in C_i$ to lie in the range of the minimum/maximum cell averages of the set $\mathcal{N}(v)$ of neighbouring cells at v. Note that this is a slight variation on the standard multidimensional Van Leer limiter, due to Barth and Jesperson [9] or Dukowicz and Kodis [27], which limit the reconstruction at v to the range of the minimum/maximum cell averages over the larger set $\mathcal{N}(i) = \bigcup_{v \in C_i} \mathcal{N}(v)$. The reason for this is that in certain cases the standard method can increase the variation locally (but not the total variation). Fig 4.7 shows an example with a hexagonal mesh in which u is unity everywhere except two adjacent cells h and l where it is respectively higher and lower. These cells cause the least-squares fit at neighbouring cells to have non-zero gradient, and allows the two cells adjacent to both h and l (m and n say) to retain this after limiting, resulting in undershoots or overshoots.

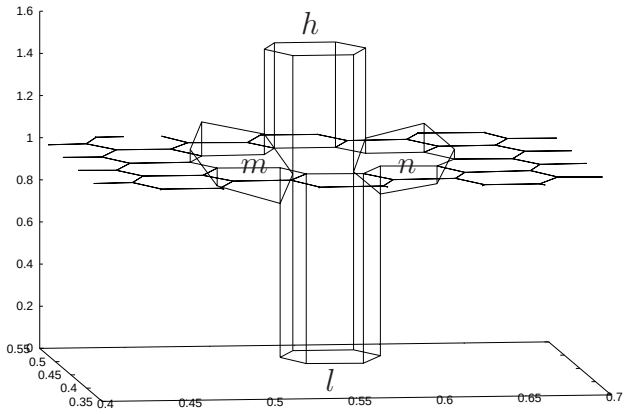


Figure 4.7: Least-squares gradients limited to neighbourhood minimum/maximum

For the standard method Swartz [85] has shown that a necessary and sufficient con-

dition for the set of neighbours $\mathcal{N}(i)$ to limit correctly is that the convex hull formed by their centroids completely contain C_i. For vertices the equivalent condition is that the convex hull formed by the centroids of $\mathcal{N}(v)$ must contain v, for all $v \in C_i$. Usually the three adjacent cells suffice, but not always, as illustrated by Fig 4.8.

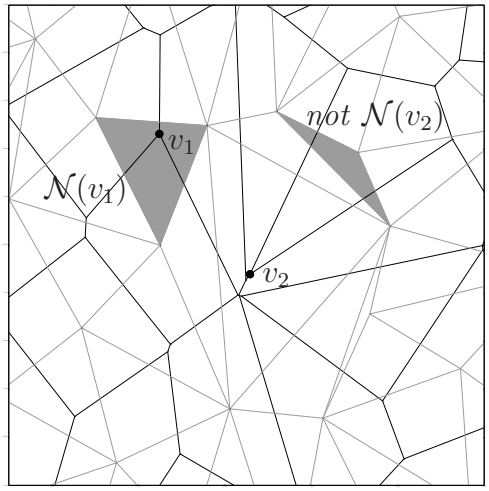


Figure 4.8: Dual space triangulation of centroids

The grey lines connect the cell centroids of Fig 4.3 with the triangulation provided by the dual space potential R. In this example vertex v_1 lies inside the triangle formed by the centroids of the three surrounding cells (shaded) whereas v_2 does not. Finding which triangle in a triangulation contains a given point is a basic problem in computational geometry and can be achieved in many ways, the simplest being to walk from triangle to adjacent triangle towards the point until it is covered [25]. For v_2 two steps are required:

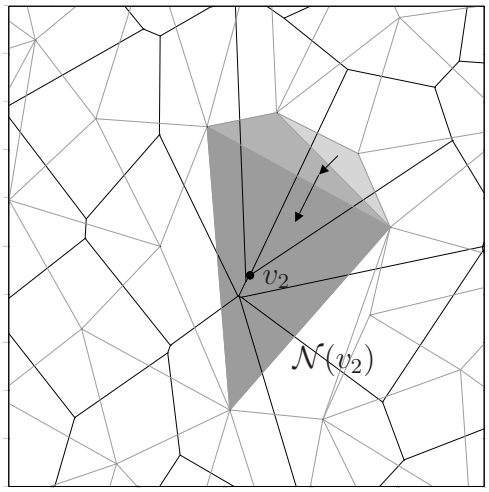


Figure 4.9: Walking through the triangulation to find the triangle covering vertex v_2

At each step the edges of the triangle are checked to see if the point lies on the opposite

side to the triangle. If this is true for just one edge then the next step is to the triangle adjacent at this edge, and if true for two edges a random choice is made between them. If not true for any edges then we are done. Termination is guaranteed and rarely takes more than a step or two. If $\mathcal{N}(v)$ no longer includes cell i it is added back in: $\mathcal{N}(v) \leftarrow \{\mathcal{N}(v), i\}$.

The linear reconstruction of u in cell i is

$$u_i(\boldsymbol{x}) = U_i + (\boldsymbol{x} - \overline{\boldsymbol{x}}_i) \cdot \nabla u, \quad \boldsymbol{x} \in C_i$$

where U_i is the cell average, $\overline{\boldsymbol{x}}_i$ the cell centroid and $\nabla u = (u_x, u_y)$ the gradient. The limiting conditions are

$$U_v^{min} = \min_{j \in \mathcal{N}(v)} U_j \quad \leqslant \quad u_i(\boldsymbol{x}_v) \quad \leqslant \quad U_v^{max} = \max_{j \in \mathcal{N}(v)} U_j, \quad \text{for all } v \in C_i.$$

Each inequality is linear in u_x, u_y so describes a half-plane in (u_x, u_y) space. Their intersection - the *monotonicity region* M - is convex and describes the set of feasible gradients (the shaded region in Fig 4.10a). As $i \in \mathcal{N}(v)$ $\forall v$ the origin is always feasible so no special treatment is required for maxima or minima.

The least square surface fit provides an initial estimate of the gradient $\nabla u^{LS} = (u_x^{LS}, u_y^{LS})$. If this lies outside M it must be moved inside. Fig 4.10b contrasts three different methods of generating a limited gradient ∇u^L.

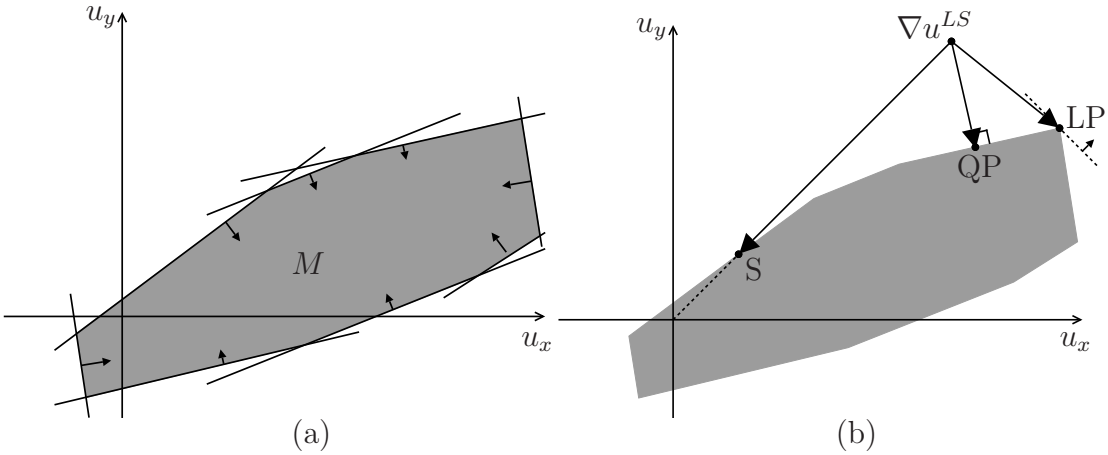


Figure 4.10: The Monotonicity Region and various limiters

The scalar limiter (labelled S) of Barth & Jesperson [9] maximises λ such that $\lambda \leqslant 1$ and $\lambda \nabla u^{LS} \in M$ by applying each constraint in turn. The linear programming (LP) method of Berger *et al* [12] maximises the functional $\nabla u \cdot \nabla u^{LS}$ i.e. it finds the vertex on M furthest in the direction ∇u^{LS} e.g. by the simplex method [73]. Lastly the quadratic programming (QP) method finds the point in M closest to ∇u^{LS} by minimising the quadratic functional $|\nabla u - \nabla u^{LS}|^2$. For all three methods there is always a unique

optimum and in general the QP method alters the least-squares gradient the least so is the method adopted here. The implementation is in the form of an *active set* method, in which at any one time a number of constraints (the active set) are in force and constraints are swapped in and out of this set as the minimisation progresses (Fig 4.11).

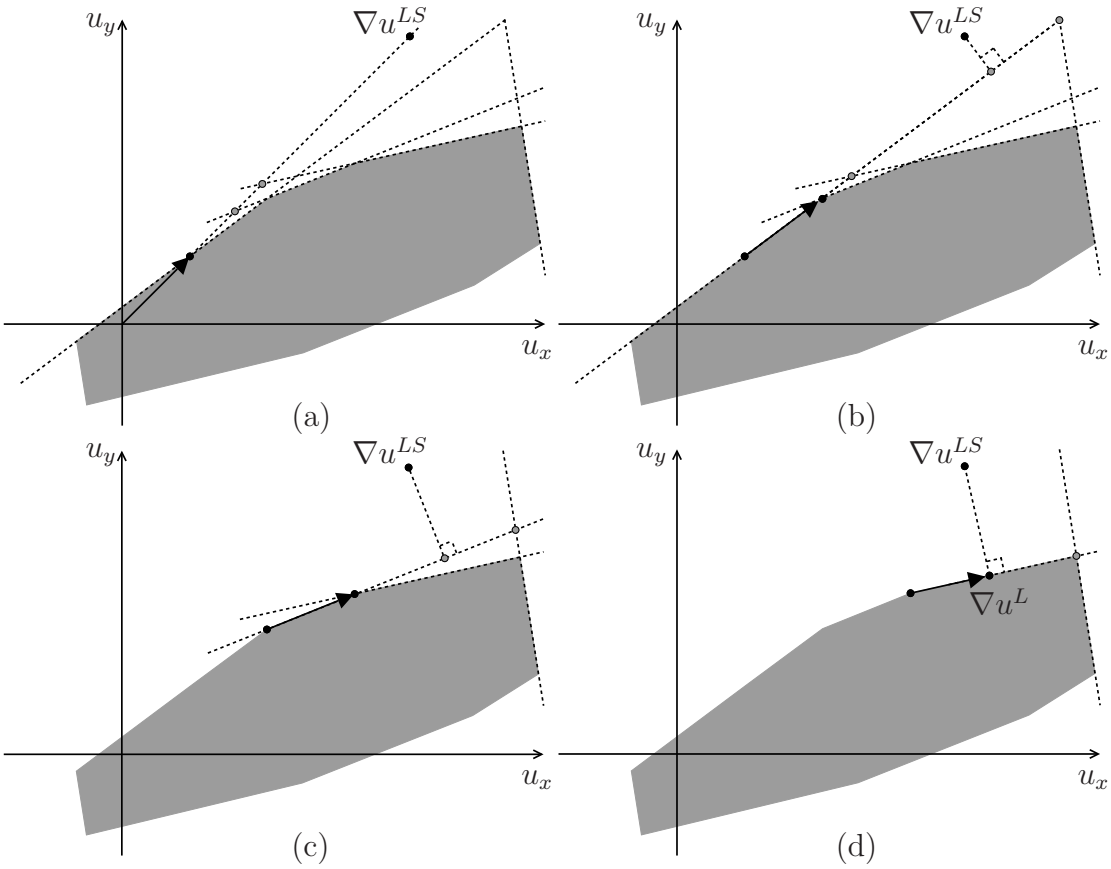


Figure 4.11: The Quadratic Programming (QP) limiter

First, in (a), the scalar limiter is applied to find a point on the boundary of M, then in (b)-(d) we move from vertex to vertex round the boundary checking each time to see if the perpendicular bisector through ∇u^{LS} has been passed. The dotted lines indicate the constraints relevant at each stage. Termination is guaranteed, either at the bisector if it is passed, or by reaching a vertex from which no further improvement is possible. The method has the same order of complexity as the simplex method.

Vertices on the domain boundary do not contribute to the limiting process.

Two examples demonstrate the behaviour of the limiter - (5.3) and (5.18). The first is continuous everywhere and smooth inside the ellipse, whereas the second is discontinuous on the ellipse. Fig 4.12 displays the unlimited least squares gradients on the left and the limited gradients on the right - in both examples all under/overshoots have been eliminated. In the first example there is one cell in which the constraints conflict reducing

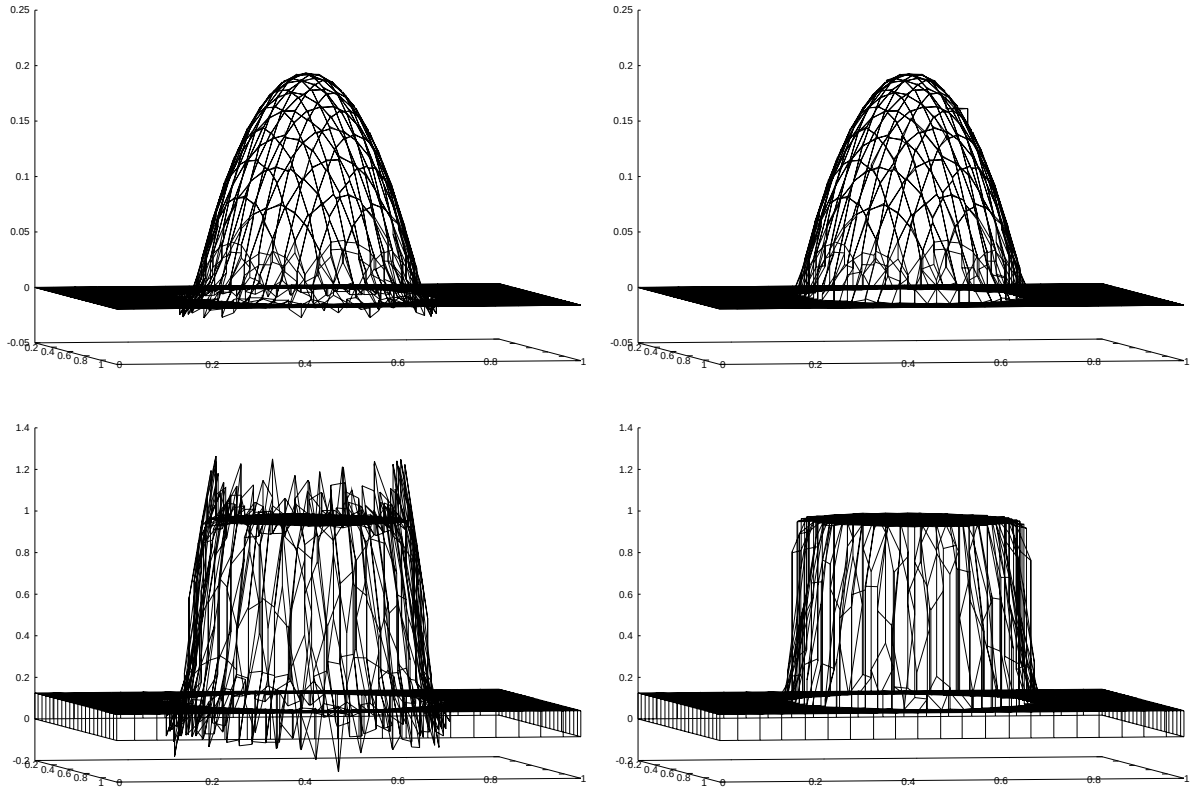


Figure 4.12: Limiter examples for continuous (top) and discontinuous (bottom) u

M to the origin (as $i \in \mathcal{N}(v)$) and resulting in the flat gradient clearly visible on the right side. This is correct but unsatisfying - one could instead remove i from $\mathcal{N}(v)$ and minimise the piecewise linear error functional $E(u_x, u_y)$ that sums the violation of each constraint. Then if M exists, the functional is zero inside it so minimisation would still result in a nearby point in M, but if it does not exist the functional will still have a non-trivial minimum which will look better when drawn but could increase the total variation. In the second example there are many faces with vertices on $u = 0$ or $u = 1$ indicating the over/undershoot has only just been removed, i.e. the gradient has been adjusted by the minimum amount as intended.

4.6 1-D Riemann problem on the moving mesh

In step 4 of the corrector scheme the flow variables each side of a face $\mathcal{F}_{LR}$, $\boldsymbol{W}_L^{n+1/2}$ and $\boldsymbol{W}_R^{n+1/2}$, are used as input for a 1-D Riemann problem (and for convenience we drop the superscript). First the velocities are decomposed into components normal and tangential to the face. From the face normal $\boldsymbol{n}_{LR} = (n_x, n_y, n_z)$ we set a unit 2D normal

$\hat{\boldsymbol{n}} = (n_x, n_y)/\sqrt{n_x^2 + n_y^2}$ from which the components of velocity are

$$\begin{aligned} v_i^{\perp} &= \hat{n}_x u_i + \hat{n}_y v_i \\ v_i^{\parallel} &= \hat{n}_y u_i - \hat{n}_x v_i, \quad i = L, R. \end{aligned}$$

and the speed of the face is $V_{LR}^{face} = -n_z/\sqrt{n_x^2 + n_y^2}$ (positive if moving $L \rightarrow R$). For cells on the domain boundary (e.g. $L \leqslant N < R \leqslant N_C$), boundary conditions are imposed by defining

$$(\rho_R, p_R, v_R^{\perp}) = \begin{cases} (\rho_L, p_L, -v_L^{\perp}) & \text{on reflective boundaries} \\ (\rho_L, p_L, v_L^{\perp} = 0) & \text{on transmissive boundaries.} \end{cases}$$

Then $(\rho_L, p_L, v_L^{\perp})$ and $(\rho_R, p_R, v_R^{\perp})$ form the initial left and right states for the exact ideal gas Riemann solver of Toro [89]. It returns the solution $(\rho_{LR}, p_{LR}, v_{LR}^{\perp})$ at similarity coordinate $\xi \equiv x/t = V_{LR}^{face}$ i.e. the values on the moving face, together with the speeds of the contact discontinuity V_{LR}^{cont} and fastest waves V_{LR} and V_{RL} travelling from $L \rightarrow R$ and $R \rightarrow L$ respectively. Upwinding is completed by setting the tangential component of velocity on the face to be

$$v_{LR}^{\parallel} = \begin{cases} v_L^{\parallel} & \text{if } V_{LR}^{face} < V_{LR}^{cont} \\ v_R^{\parallel} & \text{otherwise.} \end{cases}$$

Finally the Cartesian components of the velocity on the face are recovered with

$$\begin{aligned} u_{LR} &= \hat{n}_x v_{LR}^{\perp} + \hat{n}_y v_{LR}^{\parallel} \\ v_{LR} &= \hat{n}_y v_{LR}^{\perp} - \hat{n}_x v_{LR}^{\parallel} \end{aligned}$$

and the post-wave flow variables are $\boldsymbol{W}_{LR}^{n+1/2} = (\rho_{LR}, u_{LR}, v_{LR}, p_{LR})$.

4.7 Timestepping

For the scheme to be stable the timestep must satisfy certain conditions. The CFL condition states that for hyperbolic systems the numerical domain of dependence of each cell must include the physical domain of dependence, or equivalently characteristics cannot completely traverse any cell during the timestep. For a fixed, rectangular mesh, this requires the timestep $\triangle t$ to satisfy

$$\triangle t \leqslant \min(\triangle t_x, \triangle t_y),$$

where $\triangle t_x$, $\triangle t_y$ are the times taken by the fastest waves in the x and y directions to traverse a cell. The CFL condition is not always sufficient - linearised (von Neumann)

stability analysis or positivity can impose stricter constraints. For a scalar hyperbolic system, Liu [60] proves positivity of Runge-Kutta schemes on a *fixed* triangular mesh if

$$\triangle t \leqslant \frac{1}{3} \min_i(\triangle t_i),$$

where $\triangle t_i = \{Area(\triangle_i)/Perimeter(\triangle_i)\}/v$ and v is the fastest wavespeed in the domain.

For a *moving* quadrilateral mesh, Godunov [34] shows that a suitable energy norm is non-increasing if

$$\triangle t \leqslant \left(\frac{1}{\triangle t_x} + \frac{1}{\triangle t_y} \right)^{-1}.$$

As $\frac{1}{2}\min(\triangle t_x, \triangle t_y) \leqslant (\frac{1}{\triangle t_x} + \frac{1}{\triangle t_y})^{-1}$ this can be replaced by

$$\triangle t \leqslant \frac{1}{2} \min(\triangle t_x, \triangle t_y).$$

The author is unaware of any results for a moving arbitrary mesh with changing connectivity but the above suggest the form

$$\triangle t = C_{cfl} \min_{\mathcal{F}_{ij}}(\triangle t_{ij})$$

where $\triangle t_{ij}$ is the time taken for the fastest wave emanating from face $\mathcal{F}_{ij}$ to reach the other side of cell i and C_{cfl} is the CFL coefficient, applied in step 1 of the main loop. The computation can be simplified by observing that the time it takes for a wave to traverse a moving cell will be roughly twice the time it takes to reach the middle (the space-time centroid $\overline{\boldsymbol{r}}_i$), so long as the cell is not changing shape or size too much. If $V_{ji} < V_{ji}^{face}$ (see section 4.6 for definitions) the wave is travelling towards the centroid so $\triangle t_{ij} \approx 2d_{ij}/(-V_{ji})$, where $d_{ij} = (\overline{\boldsymbol{x}}_i - \boldsymbol{x}_{ij}^p) \cdot (-\hat{\boldsymbol{n}}_{ij})$ is the perpendicular distance from the face (unit 2D normal $\hat{\boldsymbol{n}}_{ij}$) to the centroid. This turns out to be bad at connectivity changes as the three triangular faces can vary significantly in direction, resulting occasionally in very small timesteps which can lead to difficulties if the mesh movement is large. Setting $d_{ij} = |\overline{\boldsymbol{x}}_i - \boldsymbol{x}_{ij}^p|$ (as if the wave travels directly towards the centroid) was found to produce sufficiently smooth timestepping (for comparison Ball [8] sets $V_{ij}^{face} = 0$, $d_{ij} = |\overline{\boldsymbol{r}}_i^{n+1} - \overline{\boldsymbol{r}}_j^{n+1}|$ and $C_{cfl} = \frac{1}{2}$).

In any case this is not the only source of error - Hubbard [47] notes that not using a Riemann solver in the predictor can lead to small overshoots and undershoots. For these reasons a suitable C_{cfl} is found for each problem by trial and error.

4.8 Remapping

There are two main approaches to remapping, based either on computing intersections of cells between two arbitrary meshes (e.g. [36]) or computing the volumes swept out by the faces as one mesh is deformed into the other (e.g. [32]). Although the latter is cheaper it assumes the deformation is sufficiently small, and connectivity changes require special treatment [81], so the former method is used here.

In it the solution is remapped from the t^n mesh to the t^{n+1} mesh by forming the limited linear reconstruction $\boldsymbol{u}_j^n(\boldsymbol{x})$ of the conserved variables $\boldsymbol{U}_j^n$ in each cell j on the t^n mesh and integrating these over each cell i in the t^{n+1} mesh:

$$\begin{aligned} A_i^{n+1}\boldsymbol{U}_i^{n+1} &= \sum_j \int_{C_{ij}} \boldsymbol{u}_j^n(\boldsymbol{x})\mathrm{d}\boldsymbol{x} \\ &= \sum_j \int_{C_{ij}} \left[\boldsymbol{U}_j^n + \nabla\boldsymbol{u}_j^n \cdot (\boldsymbol{x} - \overline{\boldsymbol{x}}_j^n)\right] \mathrm{d}\boldsymbol{x} \\ &= \sum_j A_{ij} \left[\boldsymbol{U}_j^n + \nabla\boldsymbol{u}_j^n \cdot (\overline{\boldsymbol{x}}_{ij} - \overline{\boldsymbol{x}}_j^n))\right] \end{aligned}$$

where $C_{ij} = C_i^{n+1} \bigcap C_j^n$ - the intersection of t^{n+1} cell i with t^n cell j - has area $A_{ij} = \int_{C_{ij}} \mathrm{d}\boldsymbol{x}$ and centroid $\overline{\boldsymbol{x}}_{ij} = \int_{C_{ij}} \boldsymbol{x}\mathrm{d}\boldsymbol{x}/A_{ij}$. These satisfy $\sum_i A_{ij} = A_j^n$, $\sum_i A_{ij}\overline{\boldsymbol{x}}_{ij} = A_j^n\overline{\boldsymbol{x}}_j^n$ so that

$$\begin{aligned} \sum_i A_i^{n+1}\boldsymbol{U}_i^{n+1} &= \sum_{ij} A_{ij} \left[\boldsymbol{U}_j^n + \nabla\boldsymbol{u}_j^n \cdot (\overline{\boldsymbol{x}}_{ij} - \overline{\boldsymbol{x}}_j^n))\right] \\ &= \sum_j A_j \left[\boldsymbol{U}_j^n + \nabla\boldsymbol{u}_j^n \cdot (\overline{\boldsymbol{x}}_j^n - \overline{\boldsymbol{x}}_j^n))\right] \\ &= \sum_j A_j^n\boldsymbol{U}_j^n \end{aligned}$$

ensuring conservation, so it just remains to find the polygonal intersections C_{ij}. Numerous convex polygon intersection algorithms exist having optimal computational complexity $O(m+n)$ for polygons with m and n vertices respectively, and here we employ the particularly simple algorithm of O'Rourke [70].

As there are $\binom{N}{2}$ possible intersections C_{ij} between arbitrary meshes, computing every one would be expensive, but as the deformation is expected to be fairly small this number can be significantly reduced. In the computation of the monitor function for C_i^n, the first step is to find the support $Sup^n(i,R)$ - the set of t^n cells intersecting a disc of radius R centered on $\overline{\boldsymbol{x}}_i^n$ (shaded in Fig 4.13a) and we reuse that subroutine here. It employs a Huygens-type construction, spreading out from cell C_i^n in an expanding front (Fig 4.13b). Each iteration we loop over the cells in the current front (medium grey), testing for intersection any untested neighbouring cells, and if so adding them to the next

front (dark grey) and the support. If the next front is empty we are done otherwise that becomes the current front and another iteration begins. Auxiliary variables are used to keep track of whether a cell has been tested, the current/next fronts and the support.

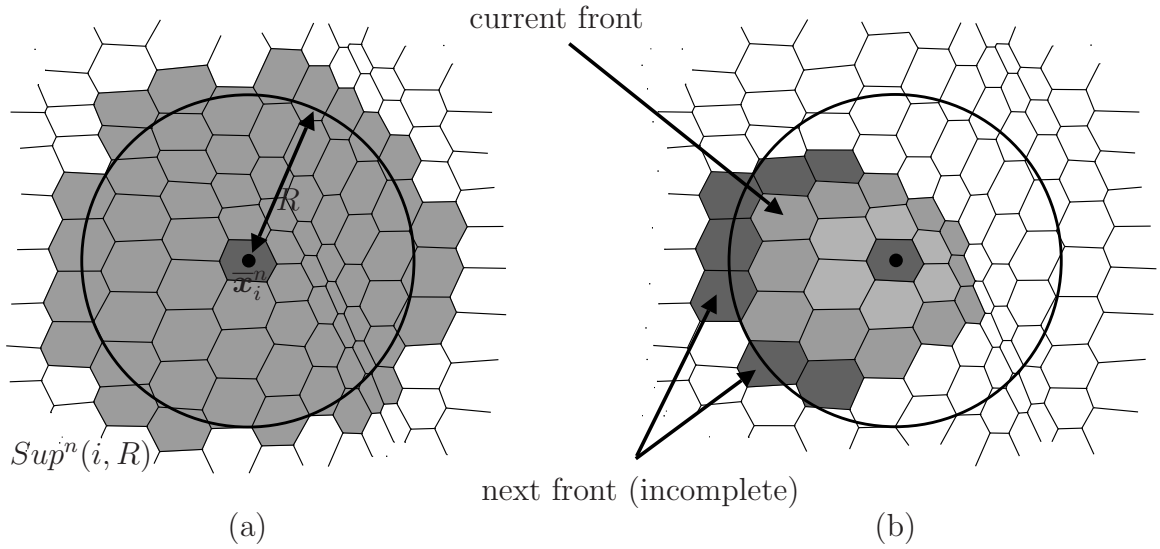


Figure 4.13: The expanding front method for filling the support

We set the radius R to be just large enough so the disc completely covers C_i^{n+1} (Fig 4.14a) via $R = \max_{v \in C_i^{n+1}} |\boldsymbol{x}_v^{n+1} - \overline{\boldsymbol{x}}_i^n|$, then the support is guaranteed to contain all the t^n cells that could intersect C_i^{n+1} so we need only compute C_{ij} for $j \in Sup^n(i,R)$ (Fig 4.14b).

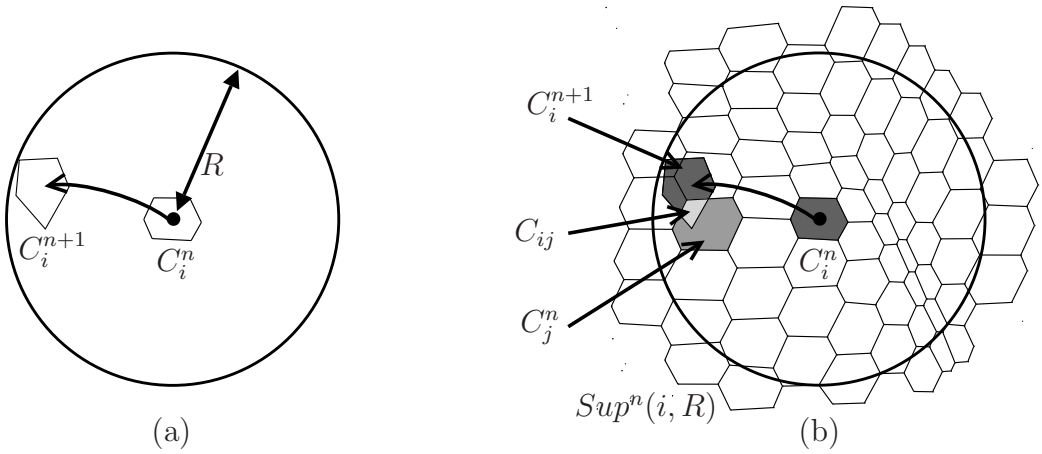


Figure 4.14: Using the support to reduce the number of polygon intersections

To test a cell for intersection with the disc we compute the area of intersection, and check if that is positive. The details are postponed to section 5.2 where more general integrals are considered.

This concludes the description of the discretisation scheme to advance the solution in time from one mesh to another.

Chapter 5

Monitor functions

To complete the discretisation it just remains to specify the end-of-timestep mesh, which is determined here by setting the final cell areas to equidistribute a monitor function and applying the geometric method.

The plan for this chapter is a little different from preceding chapters because a novel monitor is introduced which requires some justification. After briefly mentioning existing monitors, a common gradient based monitor is chosen and applied, on our unstructured mesh, to simple test problems containing discontinuities. The mesh adaption keeps failing, and each time the monitor is modified to cope, first by changing the support in the least squares surface fitting, then by introducing weighting, making the support adaptive, and finally by using the least squares error itself instead of the gradient. After looking at some properties of the new monitor, one further test problem admitting some theoretical analysis is used to shed some light on why previous monitors didn't work and how to set the parameters that have been introduced.

Monitor functions were introduced in chapter 1 equations (1.3) and (1.4) but were left unspecified. They set desired properties of the mesh such as the size, shape and orientation of the cells through the Jacobian [44], and can be used for a variety of purposes. These can be problem-specific, e.g. to focus on a particular region or feature of interest or align the mesh with an external vector field, but more generally they are used to try to maintain the mesh or solution quality. The latter can be measured by the interpolation or truncation error, which in general will be higher where polynomial approximations for the variables are poor, e.g. at steep gradients. This leads to the popular *arc-length* monitor [42]:

$$m = \sqrt{1 + \frac{1}{\alpha}|u_x|^2}$$

in 1D which when applied will equidistribute the arc-length of u over the cells, making them smaller where the gradient of u is high. The *adaption intensity parameter* α governs how much the mesh reacts to the monitor - strongly for small α but less as $\alpha \to \infty$ at which point the mesh becomes regular. In $n = 2$ dimensions, scalar and isotropic/anisotropic

matrix variants in use include [42]:

$$m = \sqrt{1 + \frac{1}{\alpha}|\nabla u|^2}, \quad \boldsymbol{G} = m\boldsymbol{I}, \quad \boldsymbol{G} = \boldsymbol{I} + \frac{1}{\alpha}\nabla u \nabla u^T.$$

Why not directly minimise the error instead of equidistributing it? Firstly from (1.14) the solution is the same in both cases and secondly because direct minimisation often leads to ill-conditioned problems [46].

In [44] Huang applies finite element error analysis to derive various estimates of the interpolation error. These depend on the differentiability of the function being approximated (assumed to be in the Sobolev space $W^{l,p}$ i.e. all partial derivatives up to total order l are p-integrable), the order of polynomial approximation (k) and error norm ($W^{m',q}$), where k, l, m' are integers such that $0 \leqslant m' \leqslant l \leqslant k+1$ and $p, q \in [1, \infty]$. We use [44, (3.20)], which is best suited to isotropic meshes ($\boldsymbol{J} \propto \boldsymbol{I}$):

$$|u - \Pi_k u|^q_{W^{m',q}(\Omega)} \leqslant C \int_\Omega J^{\frac{q(l-m')}{n}} \left(1 + \frac{1}{\alpha}\|D^l u\|_p\right)^q \mathrm{d}\boldsymbol{x},$$

where Π_k is the projection operator onto the space of polynomials of order k and C is a constant. In fact this choice of regularization excludes some of the previous monitors so we modify it to

$$|u - \Pi_k u|^q_{W^{m',q}(\Omega)} \leqslant C \int_\Omega J^{\frac{q(l-m')}{n}} \left(1 + \frac{1}{\alpha}\|D^l u\|^s_p\right)^{q/s} \mathrm{d}\boldsymbol{x}$$

for yet another number $s > 0$. Then for an (asymptotically) optimally interpolating isotropic mesh

$$m = \left(1 + \frac{1}{\alpha}\|D^l u\|^s_p\right)^{\gamma/s}, \qquad \gamma = \frac{nq}{n + q(l - m')}, \tag{5.1}$$

extending the 1D results of Carey and Dinh [19] and Baines [7]. Now we are able to recover $m = \sqrt{1 + |\nabla u|^2/\alpha}$ with $\gamma = 1, s = 2$ whereas if $s = 1$ then $m = 1 + |\nabla u|/\alpha$. Huang suggests setting α so that a fraction $\beta \in (0, 1)$ of the mesh cells are placed in regions where the monitor is high ($m \gg 1$) leading to

$$\alpha = \left[\frac{(1-\beta)}{\beta|\Omega|}\int_\Omega \|D^l u\|^\gamma_p \mathrm{d}\boldsymbol{x}\right]^{s/\gamma}. \tag{5.2}$$

He also derives matrix monitors for an optimally interpolating anisotropic mesh. For all the constraints on l, m', p, q require u to be at least continuous.

Unfortunately, none of the monitors are directly applicable here because in general the mesh solving the optimal transport problem will be different from either of the optimally interpolating meshes. It is true that the solution to the optimal transport problem is

an equidistribution (of $1/\beta$), but it is also irrotational ($\nabla \times \boldsymbol{\xi} = \nabla \times \nabla P = 0$) and not in general isotropic ($\boldsymbol{J} = \text{Hessian}(P) \not\propto \boldsymbol{I}$). So the mesh is not able to optimally interpolate the data, but we still require a scalar monitor and (5.1) is at least reasonable. The problem is that numerical implementation can be sufficiently noisy to cause the sort of behaviour seen in section 3.5, as illustrated in the following test problem.

5.1 Ellipsoid test problem

We set $\Omega = \Omega_c = [0,1] \times [0,1]$, $N_x = N_y = 20$ and the data U to be zero except where the ellipsoid with semi-major axes $(a,b,c) = (1.1, 0.8, 1)$, centred on $(x_0, y_0, U_0) = (0.5, 0.5, -0.8)$, rises above the plane $U = 0$ (Fig 5.1):

$$U(x,y) = \max\left(0, U_0 + c\sqrt{1 - \left(\frac{x-x_0}{a}\right)^2 - \left(\frac{y-y_0}{b}\right)^2}\right). \tag{5.3}$$

This means ∇U is continuous everywhere except the ellipse $\left(\frac{x-x_0}{a}\right)^2 + \left(\frac{y-y_0}{b}\right)^2 = 1 - \left(\frac{U_0}{c}\right)^2$ where the ellipsoid and plane meet (drawn in blue). Fig 5.2 compares the analytic values

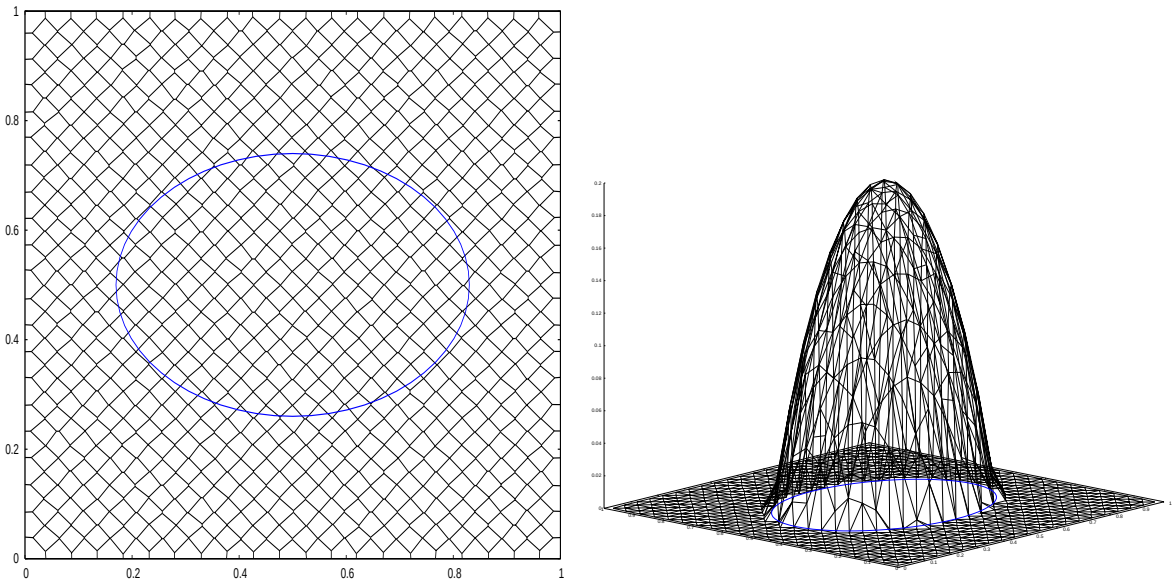


Figure 5.1: Uniform mesh and data U for ellipsoid test

of $|\nabla U|^2$ at the cell centroids (as the mesh is not structured we cannot use *superconvergent* points [96]) with the values obtained using the least squares surface fit (section 4.4) with $\boldsymbol{P} = (1, x, y, x^2, xy, y^2)$ and support $Sup(i)$ comprising cell i and its immediate neighbours. The agreement is excellent for those cells whose support does not cross the ellipse, but very poor for those that do.

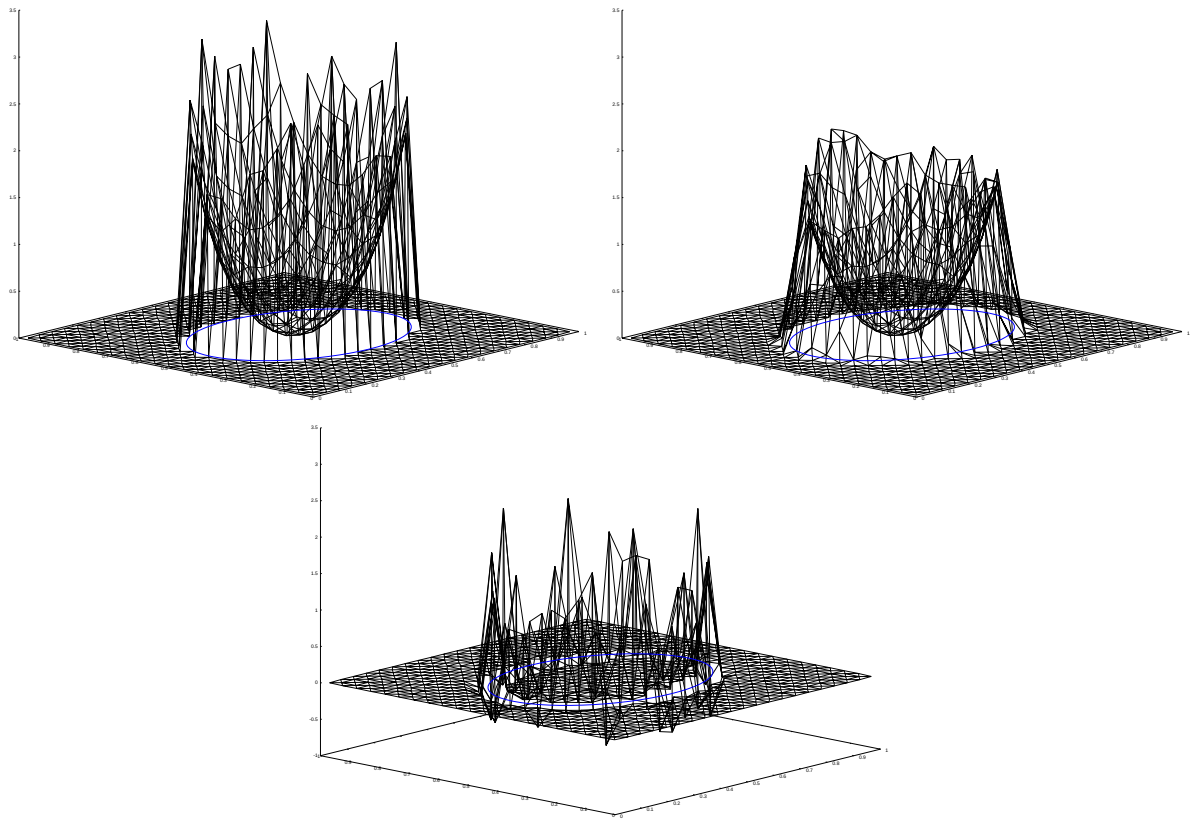


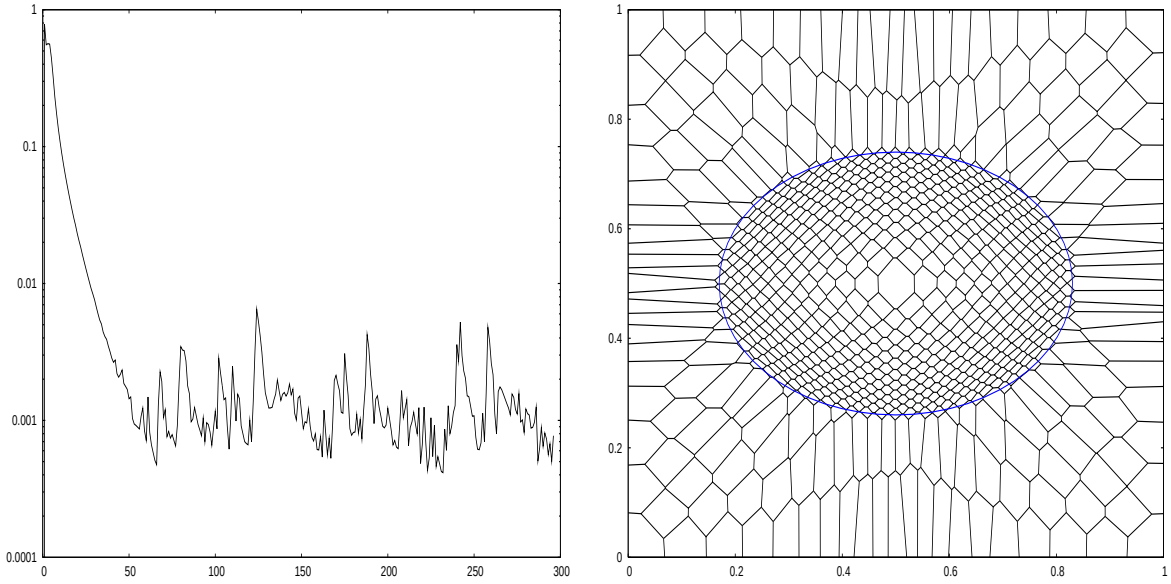
Figure 5.2: Comparison of $|\nabla U|^2$: analytic top left, computed top right, difference below

We now use the computed gradients in the mesh iteration. The monitor is set by (5.2), (5.1) with $\beta = 0.75$, $p = q = n = s = 2, l = 1, m' = 0$ i.e. $m = \sqrt{1 + |\nabla U|^2/\alpha}$, and then smoothed:

$$\boldsymbol{m}^k \leftarrow \boldsymbol{m}^{k-1} + \lambda_{mon}(\boldsymbol{m}^k - \boldsymbol{m}^{k-1}) \tag{5.4}$$

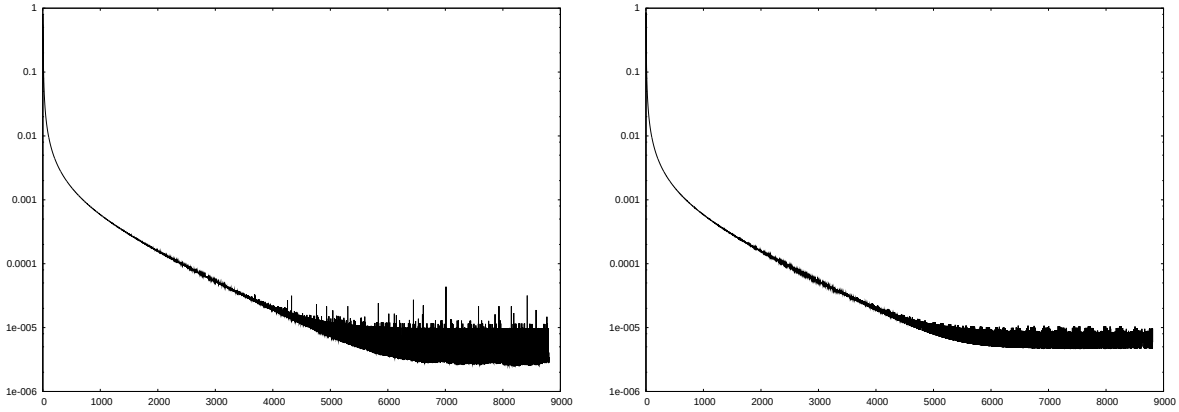
where $\lambda_{mon} \in (0, 1]$ and $\boldsymbol{m}^k$ is the vector of monitors for each cell on the k^{th} cycle. The cell areas are set by equidistributing the monitor i.e. $\beta_i m_i = 1$, and finally normalised via (3.19). Figure 5.3a plots the fractional change in monitor $\|\boldsymbol{m}^k - \boldsymbol{m}^{k-1}\|/\|\boldsymbol{m}^{k-1}\|$ per cycle for $\lambda_{mon} = 0.1$. The initially uniform mesh adapts satisfactorily for the first fifty or so cycles, reaching a position (Fig 5.3b) that doesn't substantially change from then on. However the mesh does not settle down, but continues to oscillate around this position indefinitely.

This is caused by the noisy, mesh-dependent gradients near the ellipse, and cannot be fixed by increasing the smoothing. Fig 5.4a shows the equivalent history for $\lambda_{mon} = 0.001$ - it takes longer to approach the equilibrium position, and the oscillations around it are smaller, but qualitatively it's the same. Enlarging $Sup(i)$ to include neighbours of neighbours as well does not help either (Fig 5.4b).



(a) (b)

Figure 5.3: Fractional monitor change per cycle for $\lambda_{mon} = 0.1$ and mesh at cycle 300



(a) (b)

Figure 5.4: Fractional monitor change per cycle for $\lambda_{mon} = 0.001$ for normal and enlarged support

Not only will mesh instabilities such as this be more diffusive, they also make it much more likely that the CFL condition will be violated or adjacent connectivity changes occur further degrading the solution. In the worst case they could trigger other (hydrodynamic) instabilities, so it is important they be eliminated if possible.

5.2 Weighted least squares

Meshfree methods suggest a possible remedy. In SPH a quantity $A(\boldsymbol{r})$ is spatially smoothed by the convolution $\widehat{A}(\boldsymbol{r}) = \int A(\boldsymbol{r}')W(\boldsymbol{r}-\boldsymbol{r}', h)\mathrm{d}\boldsymbol{r}'$ where W is a kernel function with characteristic lengthscale h [63]. Initially a Gaussian kernel was used ($W(\boldsymbol{r}, h) \propto$

e^{-r^2/h^2}, $r = |\boldsymbol{r}|$) but this is nonzero everywhere making computation inefficient, so various kernels with compact support ($W(\boldsymbol{r}, h) = 0, r > h$) are used instead. We implement this here by introducing a fixed radius R and radial weight function $w(r/R)$. For continuous data $U(\boldsymbol{x})$ the weighted least squares error of the approximation $u_i(\boldsymbol{x})$ centred at $\overline{\boldsymbol{x}}_i$ is

$$E = \frac{1}{\overline{w}} \int w(r/R) \left[u_i(\boldsymbol{x}) - U(\boldsymbol{x})\right]^2 \mathrm{d}\boldsymbol{x}, \qquad \overline{w} = \int w(r/R) \mathrm{d}\boldsymbol{x}, \quad r = |\boldsymbol{x} - \overline{\boldsymbol{x}}_i| \tag{5.5}$$

For a general weight the integral is over $\mathbb{R}^2$ but for a weight with compact support ($w = 0, r > R$) it can be reduced to the disc $\overline{D}_i = \{\boldsymbol{x} : r \leqslant R\}$. For discrete data U_j, we change the fixed support $Sup(i)$ to the geometrically based $Sup(i, R)$ (section 4.8) and approximate the integral by:

$$E = \frac{1}{\overline{w}_i} \sum_{j \in Sup(i,R)} w_{ij} \left[u_i(\overline{\boldsymbol{x}}_j) - U_j\right]^2, \qquad \overline{w}_i = \sum_{j \in Sup(i,R)} w_{ij} \tag{5.6}$$

$$\begin{aligned} w_{ij} &= \int_{C_j} w(r/R) \mathrm{d}\boldsymbol{x} = \int_{C_j} w(r/R) r \mathrm{d}r \mathrm{d}\theta, \quad \boldsymbol{x} = \overline{\boldsymbol{x}}_i + r(\cos\theta, \sin\theta) \\ &= R^2 \int_{C_j} w(\lambda) \lambda \mathrm{d}\lambda \mathrm{d}\theta \end{aligned} \tag{5.7}$$

where $\lambda = r/R$ is the normalised radius. Substituting in the approximation (4.5) for $u_i(\boldsymbol{x})$:

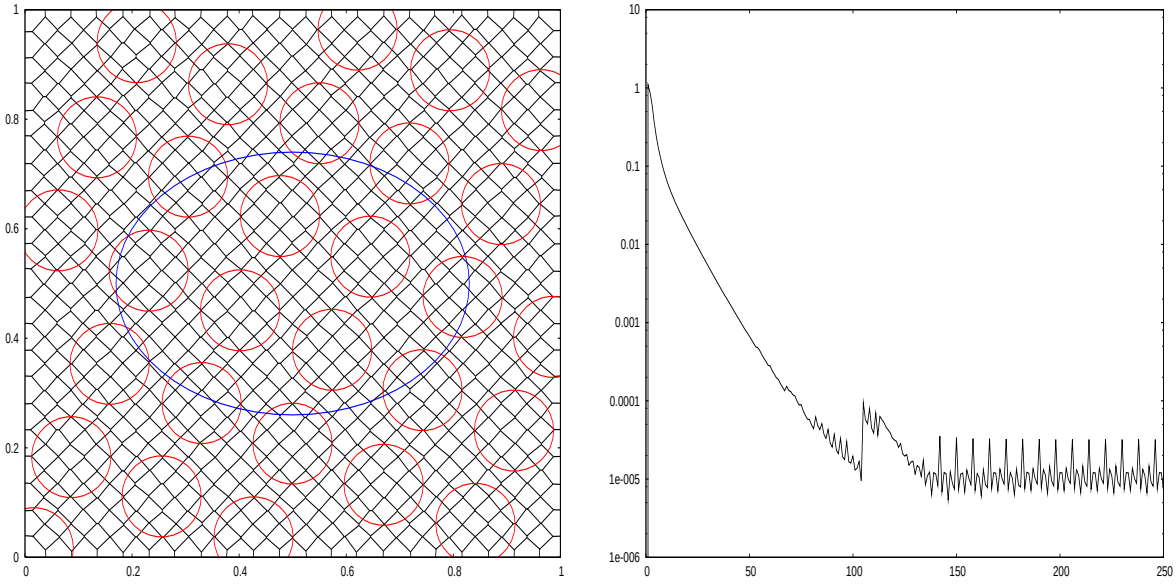
$$E = \boldsymbol{c}^T M \boldsymbol{c} - 2\boldsymbol{e}^T \boldsymbol{c} + E_0,$$

$$M_{IJ} = \frac{1}{\overline{w}_i} \sum_{j \in Sup(i,R)} w_{ij} P_I(\overline{\boldsymbol{x}}_j - \overline{\boldsymbol{x}}_i) P_J(\overline{\boldsymbol{x}}_j - \overline{\boldsymbol{x}}_i),$$

$$e_I = \frac{1}{\overline{w}_i} \sum_{j \in Sup(i,R)} w_{ij} U_j P_I(\overline{\boldsymbol{x}}_j - \overline{\boldsymbol{x}}_i), \quad E_0 = \frac{1}{\overline{w}_i} \sum_{j \in Sup(i,R)} w_{ij} U_j^2. \tag{5.8}$$

The optimal coefficients are again $\boldsymbol{c} = M^{-1}\boldsymbol{e}$. We set R to be twice the distance between centroids on a uniform hexagonal mesh ($R = 0.075$), so $Sup(i, R)$ contains on average two layers of cells around i - drawn in red for selected cells in Fig 5.5a.

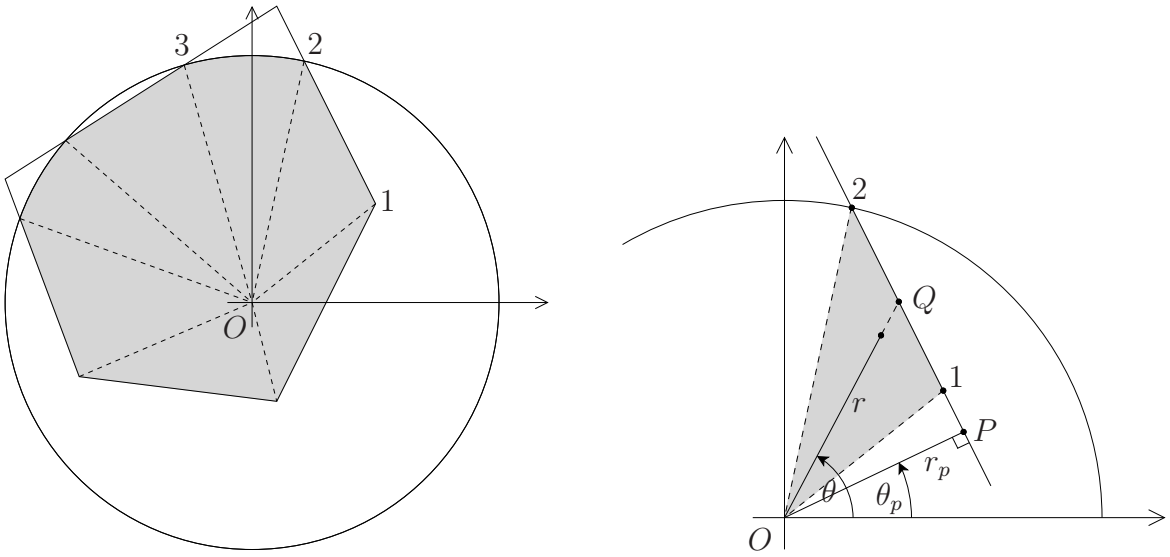
To verify the necessity of (5.7) we instead set $w_{ij} = 1$ in (5.8) and iterate with $\beta = 0.75, \lambda_{mon} = 0.1$, as before. The mesh still doesn't settle (Fig 5.5b) but the oscillations have dropped an order of magnitude. This suggests that the monitor is now too sensitive to cells entering or leaving the support, but we can use the weights to make this transition smoother - for example even with $w = 1$, w_{ij} is then the area of intersection between the disc and cell j which goes to zero as they separate. Smoother still would be a weight function $w(\lambda)$ that goes to zero as $\lambda \to 1$ e.g. $w(\lambda) = 1 - \lambda$ or $w(\lambda) = \cos^2(\lambda\pi/2)$.



(a) (b)

Figure 5.5: Support radius R for selected cells and convergence history for $w_{ij} = 1$

To compute the integral (5.7) we find any intersections between the edge of cell j and disc, and draw lines from these, and any vertices inside the disc, to the origin (Fig 5.6a). This splits the integral into integrals over triangles and integrals over sectors of the disc.



(a) (b)

Figure 5.6: Convex polygon/circle intersection

First the integral over the triangle $O12$ (shaded in Fig 5.6b):

$$w_{O12} = R^2 \int_{\theta_1}^{\theta_2} \int_0^{r_q/R} w(\lambda)\lambda \mathrm{d}\lambda \mathrm{d}\theta = R^2 \int_{\theta_1}^{\theta_2} \mathrm{d}\theta \; w_1(r_q/R)$$

where $r_q = |\overrightarrow{OQ}|$ and

$$w_n(\lambda) = \int_0^{\lambda} w(\lambda')\lambda'^n \mathrm{d}\lambda'. \tag{5.9}$$

From $\triangle OPQ$, $\cos(\theta-\theta_p)=r_p/r_q$, so

$$\begin{aligned} w_{O12} &= R^2\int_{\theta_1}^{\theta_2}\mathrm{d}\theta\ w_1(r_p\sec(\theta-\theta_p)/R) \\ &= R^2\int_{\theta_1-\theta_p}^{\theta_2-\theta_p}\mathrm{d}\theta\ w_1(\lambda_p\sec\theta),\quad \lambda_p=r_p/R \\ &= R^2[W(\lambda_p,\theta_2-\theta_p)-W(\lambda_p,\theta_1-\theta_p)] \end{aligned}$$

where

$$W(\lambda,\theta)=\int_0^\theta w_1(\lambda\sec\varphi)\mathrm{d}\varphi. \tag{5.10}$$

For the sector $O23$,

$$w_{O23}=R^2\int_{\theta_2}^{\theta_3}\mathrm{d}\theta w_1(1)=R^2(\theta_3-\theta_2)w_1(1).$$

Not that many weight functions are integrable in (5.10) e.g. $w=\cos^2(\lambda\pi/2)$ can be integrated to w_1 but not W. Some that are include ($\lambda\leqslant\lambda\sec\theta\leqslant 1$ to be inside the disc, $n\geqslant 0$):

$$w(\lambda)=(n+2)\lambda^n \ \Rightarrow\ w_1(\lambda)=\lambda^{n+2},\quad W(\lambda,\theta)=\lambda^{n+2}\int_0^\theta\sec^{n+2}\varphi\mathrm{d}\varphi \tag{5.11}$$

$$\begin{aligned} w(\lambda)=(n+2)(1-\lambda^2)^{n/2} \ &\Rightarrow\ w_1(\lambda)=1-(1-\lambda^2)^{n/2+1}, \\ &\quad W(\lambda,\theta)=\theta-\int_0^\theta(1-\lambda^2\sec^2\varphi)^{n/2+1}\mathrm{d}\varphi \end{aligned} \tag{5.12}$$

which are normalised so $w_1(1)=1$. Away from domain boundaries $\overline{w}_i=2\pi R^2$. Recurrence relations for W are found by integration by parts, for (5.11) and (5.12) respectively:

$$(n+1)W_n = \lambda^2 nW_{n-2}+\lambda^{n+2}\tan\theta\sec^n\theta, \tag{5.13}$$

$$(n+1)W_n = [n(2-\lambda^2)+1]W_{n-2}-n(1-\lambda^2)W_{n-4}+\lambda\tan\theta(1-\lambda^2\sec^2\theta)^{n/2}, \tag{5.14}$$

with $W_0=\lambda^2\tan\theta$ for both and $W_2=W_0[2-\lambda^2(3+\tan^2\theta)/3]$ for (5.12). For the intersection area required in the construction of $Sup(i,R)$, $w=1$ so $w_1(\lambda)=\frac{1}{2}\lambda^2, W(\lambda,\theta)=\frac{1}{2}\lambda^2\tan\theta$.

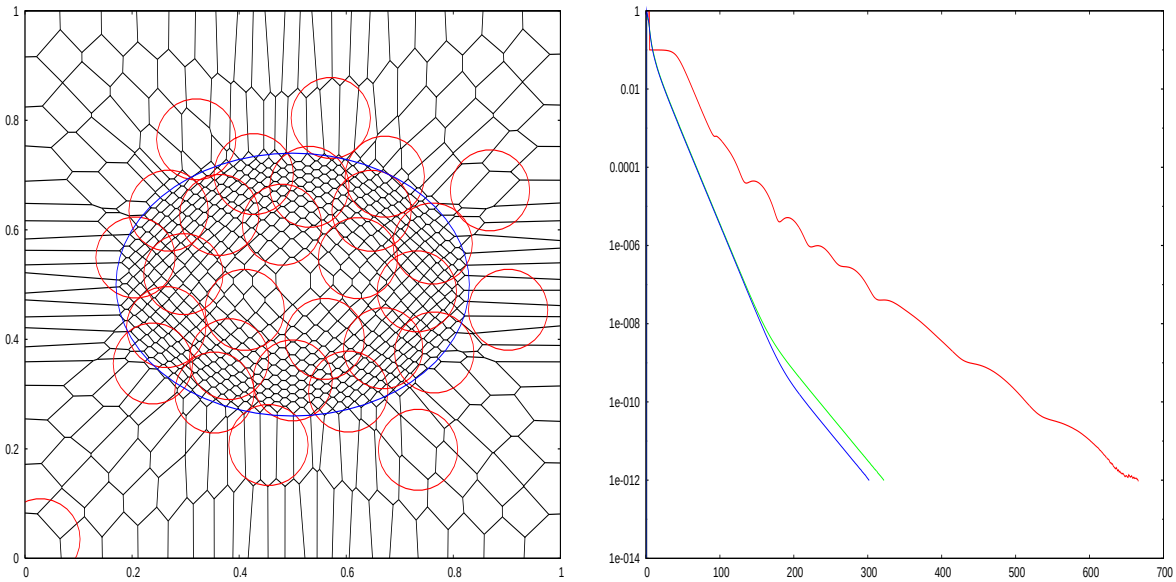
Of course evaluating these integrals exactly is expensive, but they can be approximated by quadrature e.g. $W(\lambda,\theta)=\int_0^\theta w_1(\lambda\sec\varphi)\mathrm{d}\varphi\approx\sum_i\rho_i w_1(\lambda\sec\varphi_i)$ for a set of sample angles $\varphi_i\in[0,\theta]$ and weights ρ_i. To check whether computing the intersection is actually necessary we include two approximations that don't - the first splits C_j into triangles and uses the weights at the vertices $\boldsymbol{x}_k\in C_j$:

$$w_{ij}=\sum_k Area(\overline{\boldsymbol{x}}_j,\boldsymbol{x}_k,\boldsymbol{x}_{k+1})\left[w(|\overline{\boldsymbol{x}}_j-\overline{\boldsymbol{x}}_i|/R)+w(|\boldsymbol{x}_k-\overline{\boldsymbol{x}}_i|/R)+w(|\boldsymbol{x}_{k+1}-\overline{\boldsymbol{x}}_i|/R)\right]/3. \tag{5.15}$$

and the second uses only the single weight at the cell centroid:

$$w_{ij} = A_j w(|\overline{\boldsymbol{x}}_j - \overline{\boldsymbol{x}}_i|/R). \tag{5.16}$$

We set $w = (1-\lambda^2)^2$ because it goes to zero smoothly as $\lambda \to 1$ and can be evaluated either approximately with (5.15) or (5.16) or exactly with (5.14). Returning to the test problem (with $\beta = 0.75, \lambda_{mon} = 0.1$ again), all three reach equilibrium (defined here to be when the fractional change in monitor drops below 10^{-12} - far stricter than required in practice but more illuminating). The final meshes (Fig 5.7a) are indistinguishable, but the convergence rates differ (Fig 5.7b). As expected, (5.14) in blue is slowest, taking 51 seconds, whereas (5.15) in green took roughly the same number of cycles but only 24 seconds. The cruder (5.16) in red shows signs of oscillatory behaviour and took double the number of cycles but in only 26 seconds.



(a) (b)

Figure 5.7: Equilibrium mesh and convergence history for $w = (1-\lambda^2)^2$

This confirms that the monitor was indeed too sensitive to cells entering or leaving the support, and that with a suitable weighting function the geometrically based support can work where the fixed support cannot. Furthermore the weight integrals do not need to be evaluated exactly, but the approximation used can make a difference.

However the support circles in Fig 5.7a indicate some problems still remain. For cells outside the ellipse their support is too small for sensible derivative estimates to be recovered (in the bottom left corner the support contains only one or two other cells), but as $U = 0$ outside the ellipse this is not so important here. Conversely cells in the

middle have the opposite problem - their supports are too large which is both inefficient and smoothes out fine details. A solution is again provided by SPH - make R variable.

5.3 Adaptive supports

In [63] it suggests varying the characteristic lengthscale so as to keep the size of the support approximately constant. Here this means $R \propto \sqrt{J}$ (equation 1.1). We add a minimum value R_a to R to stop any potential problems with cells getting very small, setting the support radius for cell i to be:

$$R_i = R_a + R_b\sqrt{A_i/\pi}. \tag{5.17}$$

The support therefore has area πR_i^2, so if the mesh is approximately uniform, the number of cells this contains, N, satisfies $NA_i = \pi R_i^2 \Rightarrow N = \pi R_i^2/A_i = (R_a\sqrt{\pi/A_i} + R_b)^2 \approx R_b^2$ when R_a small. So for example if $R_b = 4$ then $N \approx 16$ giving just over two layers of neighbours on a hexagonal mesh. In practice (5.17) seems to work very well, adding a stabilising influence to the adaption as well as improving efficiency. Fig (5.8) shows the equivalent results for $R_a = 0$, $R_b = 4$. The supports (Fig 5.8a) are clearly much better adapted, solving most of the problems mentioned earlier. The times are shorter for all three because convergence is much smoother, (5.14) in blue took 33 seconds, (5.15) in green took 15 seconds and (5.16) in red 20 seconds. The average size of the support over all the cells is 28 for (5.14) and (5.15) and 17 for (5.16). The reason this is higher than $R_b^2 = 16$ is clear from the figure - some cells outside the ellipse have a wide support which includes many small cells just inside. This is a consequence of the support being isotropic (circular) and could be reduced with an anisotropic (elliptical) support (as used in Adaptive SPH) but this is not explored here.

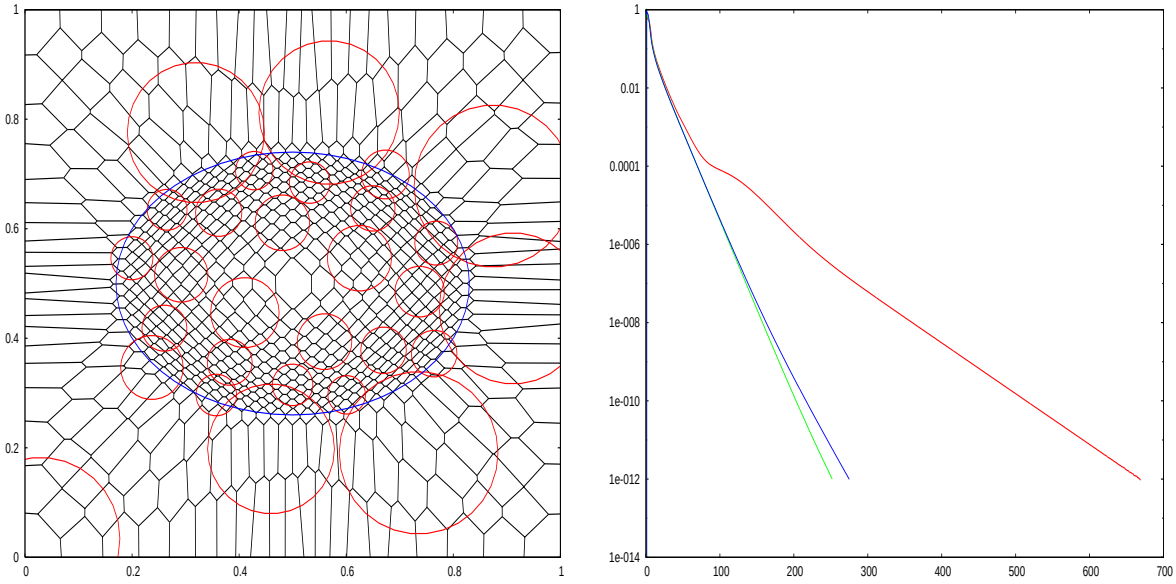
So far U has been continuous, with a discontinuity in ∇U. We now explore discontinuities in U itself.

5.4 Discontinuous data U

We keep the same geometry but change U to be constant inside the ellipse:

$$U(x,y) = \begin{cases} 1 & \text{if } \left(\frac{x-x_0}{a}\right)^2 + \left(\frac{y-y_0}{b}\right)^2 \leqslant 1 - \left(\frac{u_0}{c}\right)^2 \\ 0 & \text{otherwise} \end{cases} \tag{5.18}$$

Of course analytically ∇U is a delta function on the ellipse so numerically doesn't exist there but it will acquire finite values on the mesh which will be entirely mesh and algorithm



(a) (b)

Figure 5.8: Equilibrium mesh and monitor change history with adaptive support

dependent making this a more challenging problem. In fact with the same parameters as before ($\beta = 0.75, \lambda_{mon} = 0.1, R_a = 0, R_b = 4, w = (1 - \lambda^2)^2$), (5.14) and (5.15) are both successful (Fig 5.9b), in 230 and 57 seconds respectively, but (5.16) fails completely to settle. In the equilibrium mesh (Fig 5.9a) there is a band of concentrated cells around the ellipse with maximum width R as expected, although this narrows where it comes close to the domain boundary. Beyond this band the cell areas jump dramatically which could cause difficulties if the discontinuity moves with time because cells coming into the band will suddenly shrink.

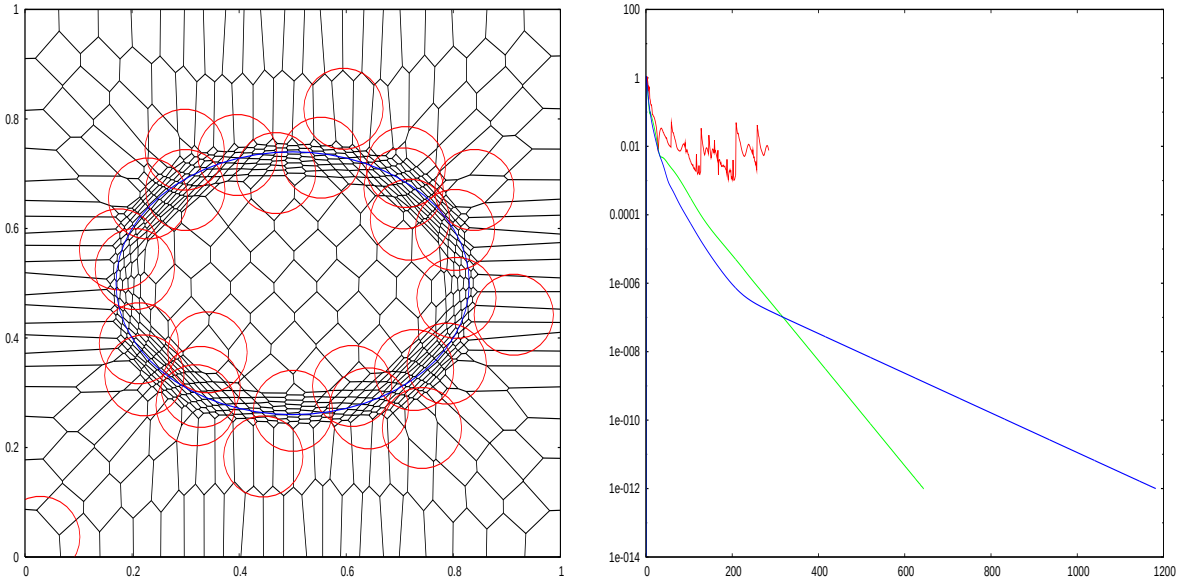
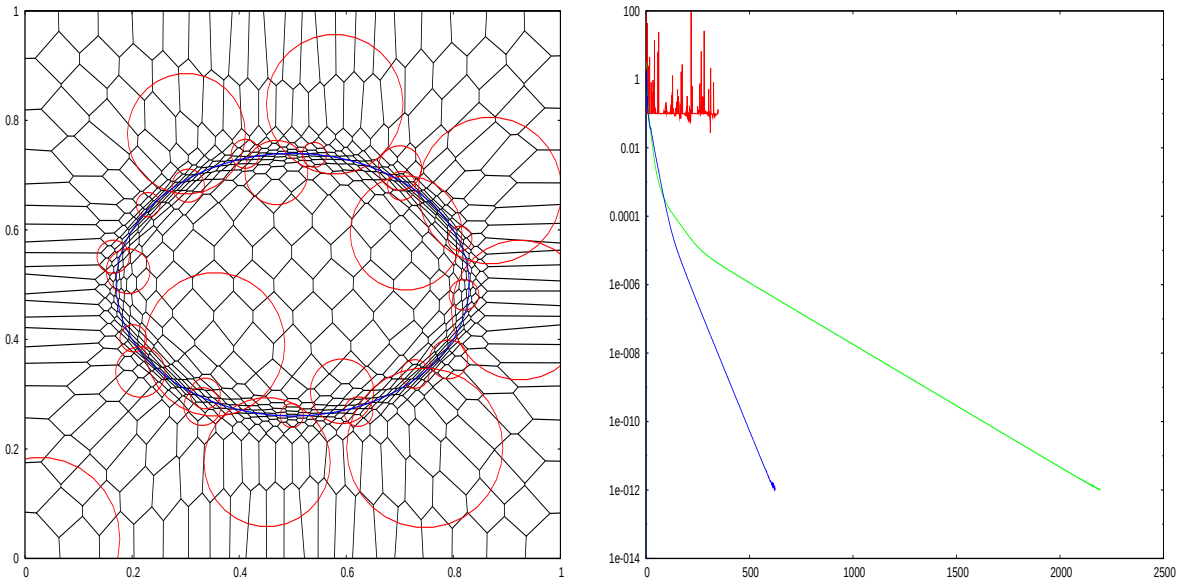


Figure 5.9: Equilibrium mesh and monitor change history for discontinuous U

The results are better for the adaptive support - (5.14) and (5.15) taking 87 and 140 seconds but (5.16) still failing. Around the ellipse the cell sizes change more evenly and are less affected by the boundary.



(a) (b)

Figure 5.10: Equilibrium mesh and monitor change history for discontinuous U

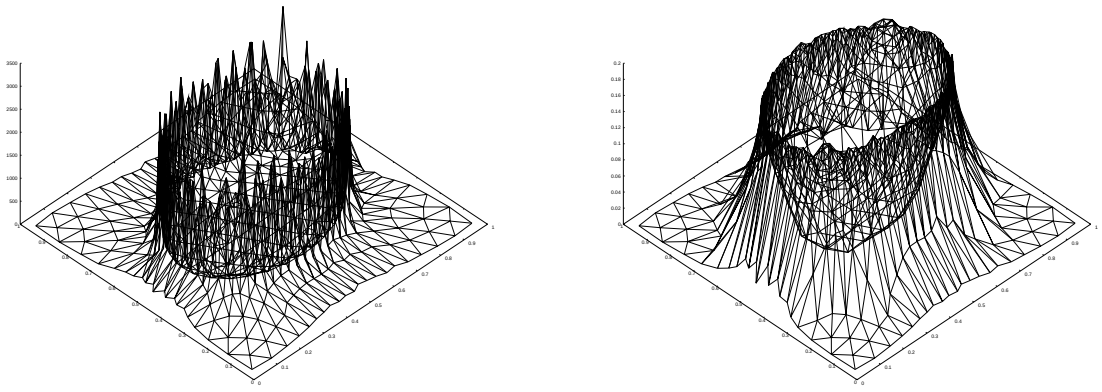
So far so good, but if we then change from $m = \sqrt{1 + |\nabla U|^2/\alpha}$ to $m = 1 + |\nabla U|^2/\alpha$ then all three weight computations fail. Reducing λ_{mon} has no effect but (5.14) succeeds if R_a is raised to 0.007, although this increases the average size of the support from 34 to 43. A plot of the computed values of $|\nabla U|^2$ on the equilibrium mesh (Fig 5.11a) shows that the mesh has settled despite quite noisy values and therefore the size of the smallest cells in the mesh will be uneven. This raises the question: are there any other less noisy quantities available that can be used for the monitor? The answer is yes - the least squares error itself (5.6).

5.5 The least-squares error monitor

Denote by E_N the least squares error when fitting polynomials up to order $N-1$, so E_1 is the error fitting a constant function - $\boldsymbol{P} = (1)$, E_2 a linear function - $\boldsymbol{P} = (1, x, y)$, E_3 a quadratic - $\boldsymbol{P} = (1, x, y, x^2, xy, y^2)$ etc. Some properties of E_N are:

- If the data is polynomial of order M then $E_N = 0$ for $N-1 \geqslant M$
- If the data is continuous then $E_N \to 0$ as $N \to \infty$
- In all cases if $U \to kU$, $k \in \mathbb{R}$, then $M \to M, \boldsymbol{e} \to k\boldsymbol{e}, E_0 \to k^2 E_0$ so $E_N \to k^2 E_N$.

Fig 5.11 compares $|\nabla U|^2$ with $E = E_1 - E_2$ on the equilibrium mesh - the latter is much more even over the discontinuity.

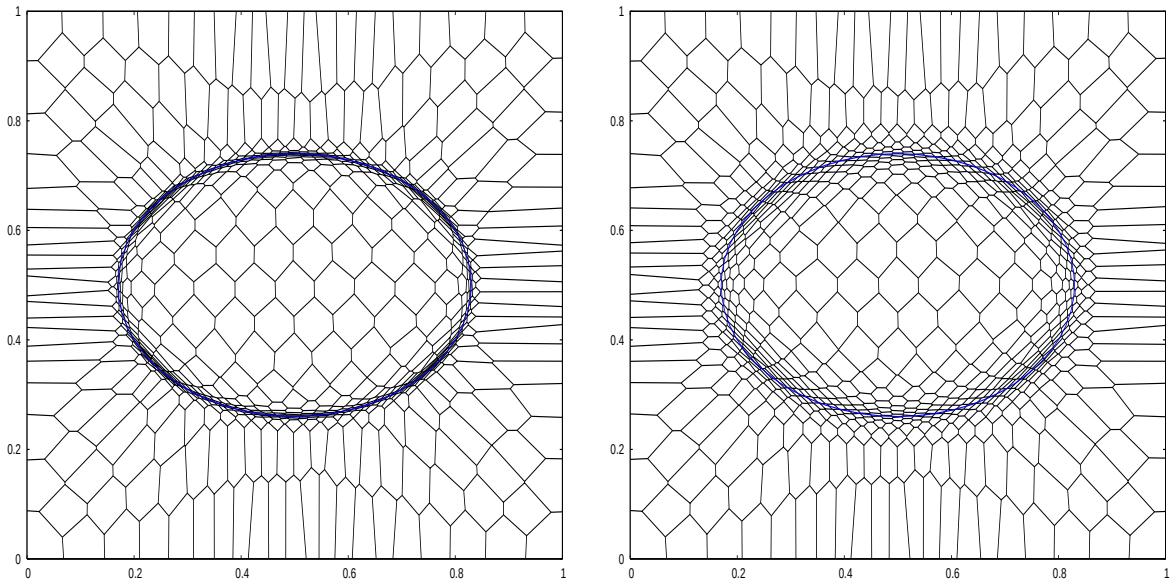


(a) (b)

Figure 5.11: $|\nabla U|^2$ and the least squares error $E = E_1 - E_2$ on the equilibrium mesh

Using the least squares error as a monitor is attractive theoretically because it is a direct measure of the interpolation error, naturally taking into account the local characteristics of the mesh. And Cao *et al* [17] have found the interpolation error to be superior to the gradient and comparable with the a posteriori error when used as monitors.

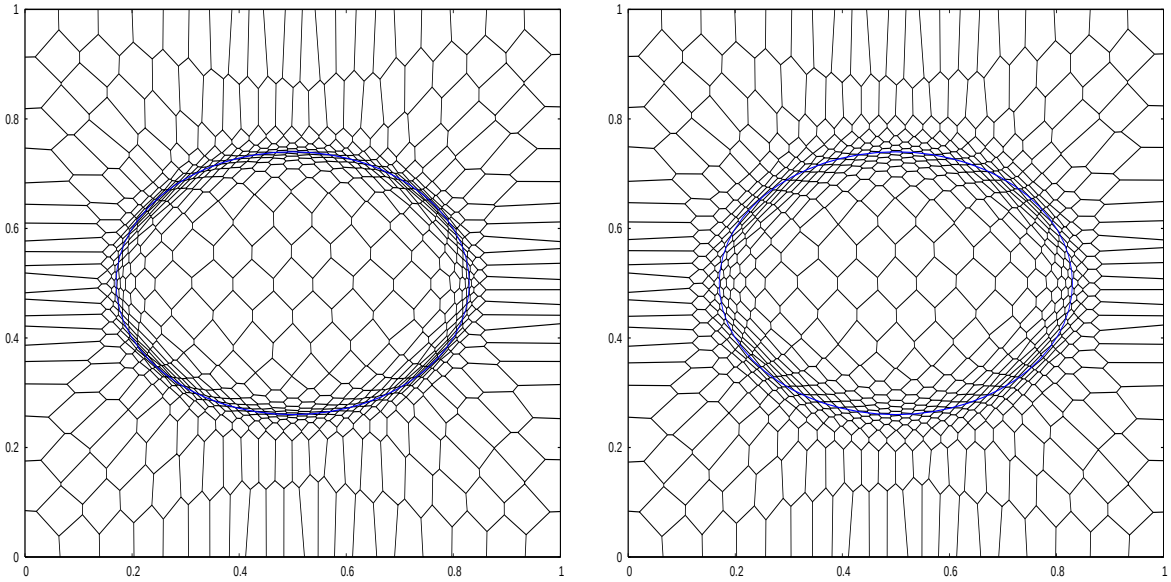
With $m = 1+E/\alpha$, all three of (5.14), (5.15) and (5.16) reach equilibrium very quickly even with $R_a = 0$, the latter with an average support of only 22 cells. As Fig 5.12 shows, there are differences between the equilibrium meshes for the two monitors, with that of the E monitor closer to Fig 5.10a.



(a) (b)

Figure 5.12: Equilibrium meshes for (a) $m = 1 + |\nabla U|^2/\alpha$ and (b) $m = 1 + E/\alpha$

The question naturally arises as to how this new monitor compares to the old in previous problems. Fig 5.13 compares the equilibrium meshes for $m = \sqrt{1+|\nabla U|^2/\alpha}$ and $m = \sqrt{1+E/\alpha}$. As before, the smoother E concentrates the mesh slightly less over the ellipse but the difference is not that great.



(a) (b)

Figure 5.13: Equilibrium meshes for (a) $m = \sqrt{1+|\nabla U|^2/\alpha}$ and (b) $m = \sqrt{1+E/\alpha}$

To gain some understanding of these errors we investigate them analytically. Away from domain boundaries, from (5.5)

$$\overline{w} = \int_{r\leqslant R} w(r/R)\mathrm{d}\boldsymbol{x} = 2\pi R^2 w_1(1). \tag{5.19}$$

Let $P_I(\boldsymbol{x}) = x^{i_1}y^{j_1}$, $P_J(\boldsymbol{x}) = x^{i_2}y^{j_2}$, then from (5.8)

$$\begin{aligned}
M_{IJ} &= \frac{1}{\overline{w}}\int_{r\leqslant R} w(r/R)P_I(\boldsymbol{x}-\overline{\boldsymbol{x}}_i)P_J(\boldsymbol{x}-\overline{\boldsymbol{x}}_i)\mathrm{d}\boldsymbol{x}, \qquad \boldsymbol{x} = \overline{\boldsymbol{x}}_i + \lambda R(\cos\theta,\sin\theta)\\
&= \frac{1}{\overline{w}}\int_0^{2\pi}\mathrm{d}\theta\int_0^1 R^2\lambda\mathrm{d}\lambda\ w(\lambda)(R\lambda\cos\theta)^{i_1+i_2}(R\lambda\sin\theta)^{j_1+j_2}\\
&= \frac{R^{i_1+i_2+j_1+j_2+2}}{2\pi R^2 w_1(1)}\int_0^1 \lambda^{i_1+i_2+j_1+j_2+1}w(\lambda)\mathrm{d}\lambda\int_0^{2\pi}\cos\theta^{i_1+i_2}\sin\theta^{j_1+j_2}\mathrm{d}\theta\\
&= \begin{cases} \frac{(2i)!(2j)!}{i!j!(i+j)!}\left(\frac{R}{2}\right)^{2i+2j}\frac{w_{2i+2j+1}(1)}{w_1(1)} & \text{if } i_1+i_2 = 2i, j_1+j_2 = 2j\\ 0 & \text{otherwise.}\end{cases}
\end{aligned} \tag{5.20}$$

The first few elements are:

$$M_{IJ} = \begin{pmatrix} M_{11} & 0 & 0 & M_{13} & 0 & M_{13} & \cdots \\ 0 & M_{13} & 0 & 0 & 0 & 0 & \\ 0 & 0 & M_{13} & 0 & 0 & 0 & \\ M_{13} & 0 & 0 & M_{15} & 0 & M_{33} & \\ 0 & 0 & 0 & 0 & M_{33} & 0 & \\ M_{13} & 0 & 0 & M_{33} & 0 & M_{15} & \\ \vdots & & & & & & \ddots \end{pmatrix} \tag{5.21}$$

For E_N, M has $\frac{N}{2}(N+1)$ rows and columns. At the minimum error, $\boldsymbol{c} = M^{-1}\boldsymbol{e}$ so $E_N = E_0 - \boldsymbol{e}^T M^{-1}\boldsymbol{e}$. Computing the first few explicitly

$$\begin{aligned} E_1 &= E_0 - \frac{e_1^2}{M_{11}} \\ E_2 &= E_0 - \frac{e_1^2}{M_{11}} - \frac{e_2^2+e_3^2}{M_{13}} \\ E_3 &= E_0 - \frac{e_1^2}{M_{11}} - \frac{e_2^2+e_3^2}{M_{13}} - \frac{(e_4-e_6)^2}{4M_{33}} - \frac{(2M_{13}e_1 - M_{11}e_4 - M_{11}e_6)^2}{4M_{11}(2M_{11}M_{33}-M_{13}^2)} - \frac{e_5^2}{M_{33}}. \end{aligned} \tag{5.22}$$

The denominator $(2M_{11}M_{33}-M_{13}^2)$ can be shown to be positive using the Cauchy-Schwarz inequality, and from (5.20) $M_{11}=1$, $M_{15}=3M_{33}$. As expected, as new degrees of freedom are included nonnegative quantities are subtracted indicating the fit approximates the data better. Obviously an orthonormal basis would be useful here, and one has recently been developed for general weight functions [92] but is quite cumbersome.

Next we Taylor expand $U(\boldsymbol{x})$ (assuming it is continuous) about $\overline{\boldsymbol{x}}_i$:

$$U(\boldsymbol{x}) = \sum_{J=1}^{\infty} U_J P_J(\boldsymbol{x}-\overline{\boldsymbol{x}}_i) = U_1 + U_2 x' + U_3 y' + U_4 x'^2 + U_5 x'y' + U_6 y'^2 + \ldots$$

in local coordinates $\boldsymbol{x}' = (x', y') = \boldsymbol{x} - \overline{\boldsymbol{x}}_i$. So at $\overline{\boldsymbol{x}}_i$, $\nabla U = (U_2, U_3)$, and from (5.8),

$$\begin{aligned} e_I &= \frac{1}{\overline{w}} \int_{r\leqslant R} w(r/R) U(\boldsymbol{x}) P_I(\boldsymbol{x}-\overline{\boldsymbol{x}}_i) \mathrm{d}\boldsymbol{x} \\ &= \sum_{J=1}^{\infty} U_J \frac{1}{\overline{w}} \int_{r<R} w(r/R) P_I(\boldsymbol{x}-\overline{\boldsymbol{x}}_i) P_J(\boldsymbol{x}-\overline{\boldsymbol{x}}_i) \mathrm{d}\boldsymbol{x} = \sum_{J=1}^{\infty} M_{IJ} U_J. \end{aligned} \tag{5.23}$$

Finally we can substitute this into (5.22):

$$\begin{aligned} E_0 - E_1 &= \frac{e_1^2}{M_{11}} = \frac{1}{M_{11}}(M_{11}U_1 + \ldots)^2 \approx M_{11}U_1^2 \\ E_1 - E_2 &= \frac{e_2^2+e_3^2}{M_{13}} = \frac{1}{M_{13}}\left[(M_{13}U_2 + \ldots)^2 + (M_{13}U_3 + \ldots)^2\right] \\ &\approx M_{13}(U_2^2+U_3^2) = M_{13}|\nabla U|^2. \end{aligned} \tag{5.24}$$

To see how good an approximation this is, Fig 5.14 overlays the computed values of $|\nabla U|^2$ (green) and $(E_1 - E_2)/M_{13}$ (red) for the ellipsoid problem (with $w = (1-\lambda^2)^2$ using (5.14) and $R_a = 0.075, R_b = 0$) - the approximation is very good here.

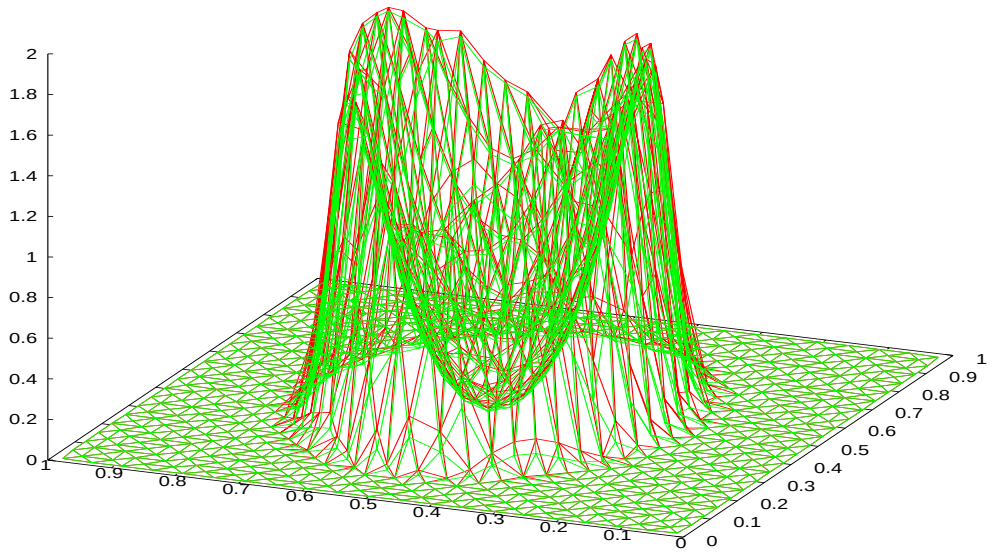


Figure 5.14: Comparison of $|\nabla U|^2$ (green) and $(E_1 - E_2)/M_{13}$ (red)

For E_3, substituting (5.23) into (5.22) yields

$$\frac{E_2 - E_3}{M_{33}} \approx \left(\frac{M_{15}M_{11} - M_{13}^2}{M_{11}M_{33}}\right)(U_4 + U_6)^2 - (4U_4U_6 - U_5^2). \tag{5.25}$$

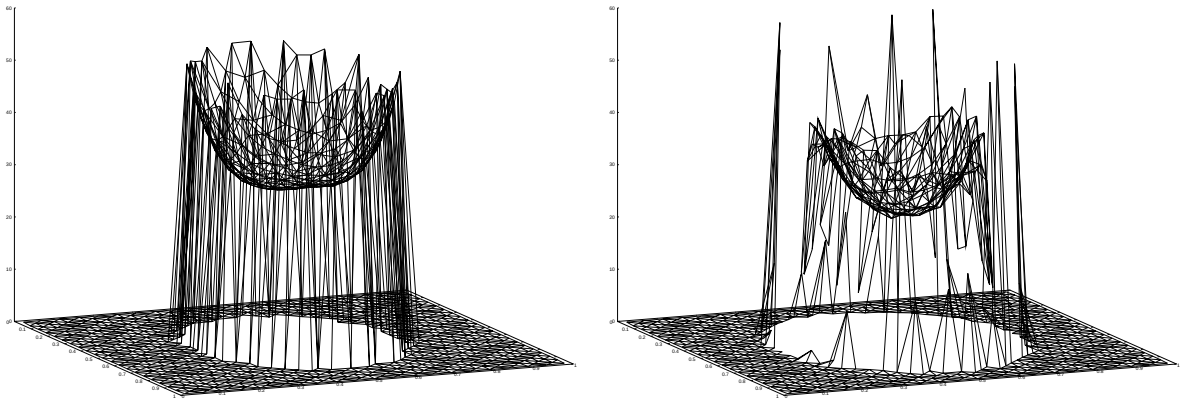
Now the Gaussian and mean curvatures at $\overline{\boldsymbol{x}}_i$ are:

$$\begin{aligned}
\kappa &= \frac{U_{xx}U_{yy} - U_{xy}^2}{(1+U_x^2+U_y^2)^2} = \frac{4U_4U_6 - U_5^2}{(1+U_2^2+U_3^2)^2} \\
H &= \frac{(1+U_x^2)U_{yy} - 2U_xU_yU_{xy} + (1+U_y^2)U_{xx}}{2(1+U_x^2+U_y^2)^{3/2}} = \frac{(1+U_2^2)U_6 - U_2U_3U_5 + (1+U_3^2)U_4}{(1+U_2^2+U_3^2)^{3/2}} \\
&\approx U_4 + U_6, \qquad U_2, U_3 \ll 1.
\end{aligned}$$

So that

$$\frac{E_2 - E_3}{M_{33}} \approx \left(\frac{M_{15}M_{11} - M_{13}^2}{M_{11}M_{33}}\right)H^2 - \kappa(1+U_x^2+U_y^2)^2. \tag{5.26}$$

Fig 5.15a shows the analytic values for the right hand side of (5.26) for the ellipsoid problem and Fig 5.15b $(E_2 - E_3)/M_{33}$. The errors became quite noisy near the discontinuity (as would be expected for higher order derivatives) so were clipped for the figure, but elsewhere the agreement is good given the crude approximation for H.



(a) (b)

Figure 5.15: Comparison of curvature term and $(E_2 - E_3)/M_{33}$ for the ellipsoid problem

So for smooth U the least squares errors contain information about the gradient and curvature. What do they look like for discontinuous U?

5.6 Line discontinuity - fixed radius support

Let

$$U(\boldsymbol{x}) = H(x) = \begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{otherwise.} \end{cases}$$

and the support radius R be constant. For the support to intersect the discontinuity $|\overline{x}_i| \leqslant R$ (Fig 5.16), so let $-R \leqslant \overline{x}_i \leqslant 0$. Put $-\overline{x}_i = R\lambda_0$ then $x \geqslant 0 \Rightarrow \lambda\cos\theta \geqslant \lambda_0$ where $\boldsymbol{x} = \overline{\boldsymbol{x}}_i + \lambda R(\cos\theta, \sin\theta)$.

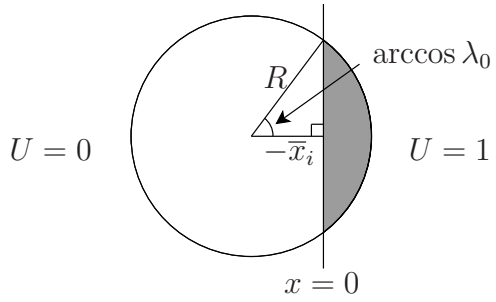


Figure 5.16: Computing E_N for $U = H(x)$

For $w = (n+2)\lambda^n$ and $P_I(\boldsymbol{x}) = x^j y^k$, $\overline{w} = 2\pi R^2$ and from (5.8),

$$\begin{aligned} e_{In} &= \frac{n+2}{2\pi R^2}\int_{r\leqslant R}(r/R)^n H(x) P_I(\boldsymbol{x} - \overline{\boldsymbol{x}}_i)\mathrm{d}\boldsymbol{x} \\ &= \frac{n+2}{2\pi R^2}\int_{\lambda\leqslant 1, \lambda\cos\theta\geqslant\lambda_0} \lambda^n (R\lambda\cos\theta)^j (R\lambda\sin\theta)^k R^2 \lambda \mathrm{d}\lambda\mathrm{d}\theta \\ &= \frac{n+2}{2\pi} R^{j+k} \int_{-\arccos\lambda_0}^{\arccos\lambda_0} \mathrm{d}\theta \cos^j\theta \sin^k\theta \int_{\lambda_0\sec\theta}^{1} \lambda^{j+k+n+1}\mathrm{d}\lambda \end{aligned}$$

$$
\begin{aligned}
&= \frac{n+2}{\pi} R^{j+k} \int_0^{\arccos\lambda_0} \mathrm{d}\theta \cos^j\theta \sin^k\theta \left[\frac{1-(\lambda_0\sec\theta)^{j+k+n+2}}{j+k+n+2}\right], \qquad k \text{ even} \\
&= \frac{n+2}{j+k+n+2} \frac{R^{j+k}}{2\pi} \left[B_{1-\lambda_0^2}\left(\frac{k+1}{2}, \frac{j+1}{2}\right) - \lambda_0^{j+k+n+2} B_{1-\lambda_0^2}\left(\frac{k+1}{2}, -\frac{k+n+1}{2}\right)\right]
\end{aligned}
$$

where $B_x(p,q)$ is the incomplete beta function. This is ungainly but computable, and can be used to construct general polynomial weights e.g. for $w = 6(1-\lambda^2)^2 = 3(2\lambda^0) - 3(4\lambda^2) + (6\lambda^4)$, $e_I = 3e_{I0} - 3e_{I2} + e_{I4}$. Values for $\lambda_0 < 0$ can be deduced from the symmetry $e_{In}(-\lambda_0) + e_{In}(\lambda_0) = 2e_{In}(0) = \frac{n+2}{j+k+n+2}\frac{R^{j+k}}{\pi} B\left(\frac{k+1}{2}, \frac{j+1}{2}\right)$. Thanks to the identity $H^2(x) \equiv H(x)$, $E_0 = e_{1n}$ $(P_1(\boldsymbol{x}) = 1)$. Now we can numerically evaluate M, $\boldsymbol{e}$ and E_0 for polynomial weights, we can compute E_N.

Fig 5.17 shows the results for $w = (n+2)(1-\lambda^2)^{n/2}$ with $n = 0, 2, 4$ as functions of λ_0, with the support just touching the line at $\lambda_0 = \pm 1$ and sitting directly on top at $\lambda_0 = 0$. The surprising thing about the errors is that they are not all singly peaked - sometimes being further away from the discontinuity makes it harder to approximate not easier, although this effect weakens as n increases. This is not good if the monitor is intended to focus the mesh over the discontinuity as the error will instead focus the mesh slightly more over bands either side. However a single peak can easily be restored with a linear combination of the errors:

$$
E = \sum_N \mu_N E_N \tag{5.27}
$$

for constants μ_N. The second point to notice is how the errors behave at $\lambda_0 = \pm 1$, which governs the smoothness of the transition as cells become aware of a discontinuity and so is important to the mesh iteration. As expected this transition becomes smoother as n increases, and there appears to be a jump in the gradient for $n = 0$ although this is not so. For general w and $P_I(\boldsymbol{x}) = x^j y^k$,

$$
\begin{aligned}
e_I &= \frac{1}{\overline{w}} \int_{r\leqslant R} w(r/R) H(x) P_I(\boldsymbol{x} - \overline{\boldsymbol{x}}_i) \mathrm{d}\boldsymbol{x} \\
&= \frac{R^{j+k}}{2\pi w_1(1)} \int_{-\arccos\lambda_0}^{\arccos\lambda_0} \mathrm{d}\theta \cos^j\theta \sin^k\theta \int_{\lambda_0\sec\theta}^1 \lambda^{j+k+1} w(\lambda) \mathrm{d}\lambda \\
&= \frac{R^{j+k}}{\pi w_1(1)} \int_0^{\arccos\lambda_0} \mathrm{d}\theta \cos^j\theta \sin^k\theta \int_{\lambda_0\sec\theta}^1 \lambda^{j+k+1} w(\lambda) \mathrm{d}\lambda, \qquad k \text{ even}.
\end{aligned}
$$

Differentiating,

$$
\begin{aligned}
e_I' = \frac{\mathrm{d}e_I}{\mathrm{d}\lambda_0} &= -\frac{R^{j+k}}{\pi w_1(1)} \int_0^{\arccos\lambda_0} \mathrm{d}\theta \cos^j\theta \sin^k\theta (\lambda_0\sec\theta)^{j+k+1} w(\lambda_0\sec\theta) \sec\theta \\
&= -\frac{R^{j+k}}{\pi w_1(1)} \lambda_0^{j+k+1} \int_0^{\arccos\lambda_0} \sec^2\theta \tan^k\theta w(\lambda_0\sec\theta) \mathrm{d}\theta
\end{aligned} \tag{5.28}
$$

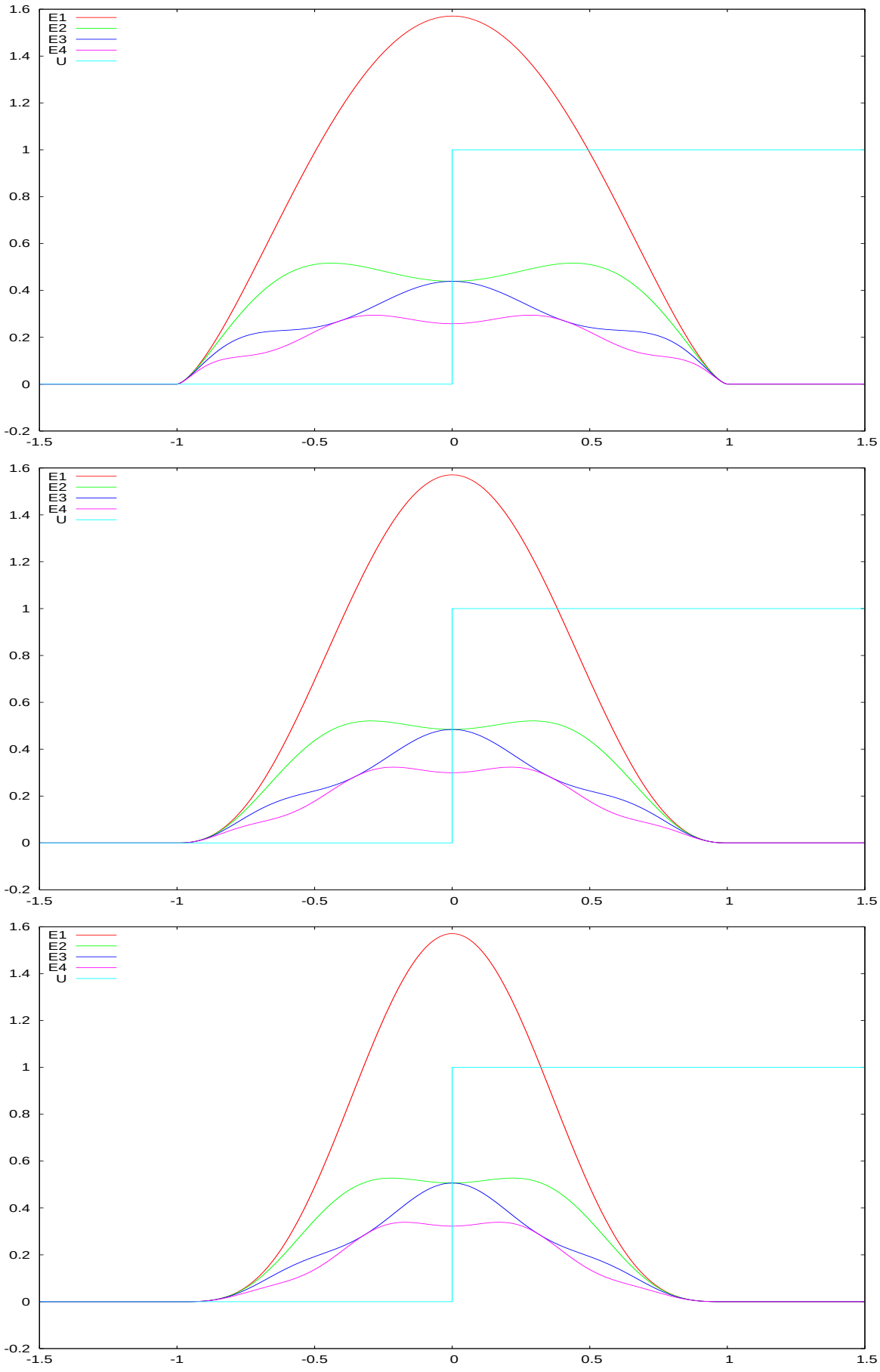


Figure 5.17: Least squares errors E_N for $U = H(x)$ and $w = 2$ (top), $w = 4(1-\lambda^2)$ (middle) and $w = 6(1-\lambda^2)^2$ (bottom)

$$\therefore |e'_I| \leqslant \frac{R^{j+k}}{\pi w_1(1)} \lambda_0^{j+k+1} \|w\|_\infty \int_0^{\arccos\lambda_0} \sec^2\theta \tan^k\theta \mathrm{d}\theta$$
$$= \frac{R^{j+k}}{\pi w_1(1)} \|w\|_\infty \frac{\lambda_0^j (1-\lambda_0^2)^{\frac{k+1}{2}}}{k+1} \qquad = 0 \text{ at } \lambda_0 = \pm 1 \text{ if } \|w\|_\infty < \infty.$$

So from $E_N = E_0 - \boldsymbol{e}^T M^{-1} \boldsymbol{e}$ the errors are always differentiable at $\lambda_0 = \pm 1$. The third observation is that $E_{2N+1}(0) = E_{2N}(0)$ which makes sense as $U - \frac{1}{2}$ is an odd function around $\lambda_0 = 0$, so enlarging the basis to include even functions cannot help reduce the error. Lastly there are points $\lambda_0 \neq 0$ where $E_{2N+1}(\lambda_0) = E_{2N}(\lambda_0)$ which presumably are related to the superconvergent points.

As an aside, for $I = 1$ ($j = k = 0$), changing the variable of integration in (5.28) to λ:

$$e'_1 = -\frac{\lambda_0}{\pi w_1(1)} \int_0^{\arccos\lambda_0} \sec^2\theta w(\lambda_0 \sec\theta) \mathrm{d}\theta$$
$$= -\frac{1}{\pi w_1(1)} \int_{\lambda_0}^1 \frac{w(\lambda)\lambda \mathrm{d}\lambda}{\sqrt{\lambda^2 - \lambda_0^2}} \tag{5.29}$$

which is the Abel transform of w, and can be inverted to give w in terms of e'_1 ($w_1(1)$ being a free parameter as e'_1 is unchanged if $w \to kw$). Also e'_1 is related to E_1 through $E_1 = e_1 - \frac{e_1^2}{M_{11}}$ (using (5.22) and again $E_0 = e_1$ as $H^2(x) \equiv H(x)$). And so if the monitor is chosen to be a function of E_1, e.g. $m = 1 + E_1/\alpha$, then w can be given directly in terms of m. In other words it is possible to decide exactly how you want the cell areas to change across a discontinuity and then derive the weight needed to achieve this. For example $E_1(\lambda_0) = \lambda_0^2(1-\lambda_0^2)$ if $e_1 = \lambda_0^2$ ($M_{11} = 1$), in which case $w(\lambda) = 4w_1(1)[\cosh^{-1}(1/\lambda) - 1/\sqrt{1-\lambda^2}]$, but this weight is not very practical as it diverges at $\lambda = 0, \pm 1$. Alternatively, inverting $e'_1 = -\frac{B\left(\frac{1}{2}, n+\frac{1}{2}\right)}{2\pi w_1(1)}(1-\lambda_0^2)^n$ yields $w = (1-\lambda_0^2)^{n-\frac{1}{2}}$ so for example if $w = \sqrt{1-\lambda^2}$ then $e_1 = \frac{1}{4}(2 - 3\lambda_0 + \lambda_0^3)$ (using the constant of integration to set $e_1(1) = 0$) and $E_1 = \frac{1}{4}(1-\lambda_0^2)^2(1-\lambda_0^2/4)$.

5.7 Line discontinuity - adaptive support

So far the support has been fixed in size but now we will relax that condition. The resulting system turns out to be soluble, enabling tangible qualities of the equilibrium mesh, such as the width of the band, to be related to the parameters R_a, R_b and β.

Suppose that a weight w has been chosen and a set of constants μ_N picked (5.27) and so on resulting in a final known error $E(\lambda)$ where $x = R\lambda$, but now $R = R(\boldsymbol{x})$. We set up a mesh iteration in a rectangular domain of size $\Omega_x \times \Omega_y$, with a large number of cells N, so that the cell area $a(\boldsymbol{x})$ and monitor $m(\boldsymbol{x})$ can be treated as continuous. In

computational space Ω_c the cells are spread evenly so each has area $|d\boldsymbol{\xi}| = |\Omega_c|/N$, and from $|d\boldsymbol{x}| = J|d\boldsymbol{\xi}|$ we have $a(\boldsymbol{x}) = J|\Omega_c|/N$. From the equidistribution criterion (1.12) $Jm = \sigma$ so $a(\boldsymbol{x}) = \frac{\sigma|\Omega_c|}{Nm(\boldsymbol{x})}$. As the problem is symmetric in y so is the equilibrium mesh, i.e. $a = a(x)$, $m = m(x)$ and $R = R(x)$. We use (5.1) for the monitor with E in place of $\|D^l u\|_p^2$ to match the scaling (under $U \to kU$, $E \to k^2 E$ and $\|D^l U\|_p^2 \to k^2 \|D^l U\|_p^2$), i.e.

$$m = \left(1 + \frac{E(\lambda)^{s/2}}{\alpha}\right)^{\gamma/s}, \tag{5.30}$$

set $s = 2$, and change $\gamma \to 2\gamma$ for ease of notation, so $a(x) = \frac{\sigma|\Omega_c|}{N}\left(1 + \frac{E(\lambda)}{\alpha}\right)^{-\gamma}$. The cells are largest away from the discontinuity ($|\lambda| \geqslant 1$), where $E(\lambda) = 0$ and $a(x) = a_{max} = \frac{\sigma|\Omega_c|}{N}$. To close the system we have the definitions of α (5.2) and the support size (5.17):

$$\begin{aligned}
\alpha &= \left[\frac{(1-\beta)}{\beta|\Omega|}\int_\Omega E(\lambda)^\gamma \mathrm{d}\boldsymbol{x}\right]^{1/\gamma} = \left[\frac{(1-\beta)}{\beta\Omega_x}\int_{\Omega_x} E(\lambda)^\gamma \mathrm{d}x\right]^{1/\gamma} \\
R(x) &= R_a + R_b\sqrt{a(x)/\pi}. \\
a(x) &= a_{max}\left(1 + \frac{E(\lambda)}{\alpha}\right)^{-\gamma} \\
\lambda &= x/R(x).
\end{aligned}$$

Some properties of the equilibrium mesh can be immediately inferred: $R(x)$ is largest far from the discontinuity, where $R = R_a + R_b\sqrt{a_{max}/\pi}$ and this sets the maximum (half) width of the band of concentrated cells - beyond this no cell's support is big enough for it to be aware of the discontinuity. By definition, α puts a fraction β of the mesh cells in the band, so the area outside is $N(1-\beta)a_{max} = \Omega_y(\Omega_x - 2R_{max}) \approx |\Omega|$ so

$$a_{max} \approx \frac{|\Omega|}{N(1-\beta)}, \qquad R_{max} = R_a + R_b\sqrt{\frac{a_{max}}{\pi}}. \tag{5.31}$$

Continuing on, we rearrange α and substitute $a(x)$ into $R(x)$, then $R(x)$ into λ:

$$\begin{aligned}
\frac{\beta\Omega_x\alpha^\gamma}{1-\beta} &= \int_{\Omega_x} E(\lambda)^\gamma \mathrm{d}x \\
x &= \lambda\left[R_a + R_b\sqrt{\frac{a_{max}}{\pi}\left(1 + \frac{E(\lambda)}{\alpha}\right)^{-\gamma}}\right] = \lambda S(\lambda), \quad \text{say.}
\end{aligned} \tag{5.32}$$

Finally, we substitute $\mathrm{d}x = \frac{\mathrm{d}x}{\mathrm{d}\lambda}\mathrm{d}\lambda$ and use $E(\lambda) = E(-\lambda)$, $E(\lambda) = 0, |\lambda| \geqslant 1$ to change the limits of integration:

$$\frac{\beta\Omega_x\alpha^\gamma}{1-\beta} = \int_{-1}^{1} E^\gamma \frac{\mathrm{d}}{\mathrm{d}\lambda}[\lambda S]\mathrm{d}\lambda = 2\int_0^1 E^\gamma \frac{\mathrm{d}}{\mathrm{d}\lambda}[\lambda S]\mathrm{d}\lambda. \tag{5.33}$$

Therefore

$$\begin{aligned}\frac{\beta\Omega_x\alpha^\gamma}{2(1-\beta)} &= [E^\gamma\lambda S]_0^1 - \int_0^1 \lambda S\frac{\mathrm{d}E^\gamma}{\mathrm{d}\lambda}\mathrm{d}\lambda \\ &= -\int_0^1 \lambda\left[R_a + R_b\sqrt{\frac{a_{max}}{\pi}\left(1+\frac{E}{\alpha}\right)^{-\gamma}}\right]\frac{\mathrm{d}E^\gamma}{\mathrm{d}\lambda}\mathrm{d}\lambda \\ &= -R_a\int_0^1 \lambda\frac{\mathrm{d}E^\gamma}{\mathrm{d}\lambda}\mathrm{d}\lambda - R_b\sqrt{\frac{a_{max}}{\pi}}\int_0^1 \lambda\left(1+\frac{E}{\alpha}\right)^{-\gamma/2}\frac{\mathrm{d}E^\gamma}{\mathrm{d}\lambda}\mathrm{d}\lambda.\end{aligned}$$

We arrive at an implicit equation for α in terms of the other parameters:

$$\frac{\beta\Omega_x\alpha^\gamma}{2(1-\beta)} = F(\infty)R_a + F(\alpha)R_b\sqrt{\frac{a_{max}}{\pi}}, \tag{5.34}$$

where

$$\begin{aligned}F(\alpha) &= -\int_0^1 \lambda\left(1+\frac{E}{\alpha}\right)^{-\gamma/2}\frac{\mathrm{d}E^\gamma}{\mathrm{d}\lambda}\mathrm{d}\lambda, \qquad \alpha \geqslant 0 \\ F(\infty) &= -\int_0^1 \lambda\frac{\mathrm{d}E^\gamma}{\mathrm{d}\lambda}\mathrm{d}\lambda = -[\lambda E^\gamma]_0^1 + \int_0^1 E^\gamma\mathrm{d}\lambda = \int_0^1 E^\gamma\mathrm{d}\lambda.\end{aligned}$$

We first characterize $F(\alpha)$. If $E(\lambda)$ is a decreasing function of λ for $\lambda > 0$, then $\frac{\mathrm{d}E^\gamma}{\mathrm{d}\lambda} < 0 \Rightarrow F(\alpha) > 0$. And if $\alpha_1 < \alpha_2$ then $\left(1+\frac{E}{\alpha_1}\right)^{-\gamma/2} < \left(1+\frac{E}{\alpha_2}\right)^{-\gamma/2}$, so $F(\alpha)$ is an increasing function of α, in particular $F(\alpha) < F(\infty)$, $\alpha < \infty$. Also, as $\alpha \to 0$, $\alpha^{-\gamma/2}\left(1+\frac{E}{\alpha}\right)^{-\gamma/2} = (\alpha+E)^{-\gamma/2} \to E^{-\gamma/2}$ so $\alpha^{-\gamma/2}F(\alpha) \to$ a constant, so $F(0) = 0$. Thus $0 \leqslant F(\alpha) < F(\infty)$.

Assuming R_a, $R_b \geqslant 0$, then as α increases from 0 to ∞, the right hand side of (5.34) increases from $F(\infty)R_a$ to $F(\infty)R_{max}$ whereas the left hand side increases from 0 to ∞ so by continuity they must cross, i.e. there is always a positive solution for α, and therefore an equilibrium position for the mesh. Once α is known, then for any $\lambda \in [0,1]$, (5.32) gives the corresponding x from which $a(x)$ and $R(x)$ can be derived, completely characterising the mesh.

To verify the accuracy of (5.34) we perform some numerical tests. For this we need a specific $E(\lambda)$ - one that is representative and can also be integrated. We use $w = \sqrt{1-\lambda^2}$ so $E_1(\lambda) = \frac{1}{4}(1-\lambda^2)^2(1-\lambda^2/4)$, and set $E = 4E_1$, $\gamma = 1$. Integrating by parts:

$$\begin{aligned}F(\alpha) &= -\int_0^1 \lambda\left(1+\frac{E}{\alpha}\right)^{-1/2}\frac{\mathrm{d}E}{\mathrm{d}\lambda}\mathrm{d}\lambda \\ &= -\left[\lambda 2\alpha\sqrt{1+\frac{E}{\alpha}}\right]_0^1 + \int_0^1 2\alpha\sqrt{1+\frac{E}{\alpha}}\mathrm{d}\lambda = 2\alpha\left[\int_0^1\sqrt{1+\frac{E}{\alpha}}\mathrm{d}\lambda - 1\right], \\ \int_0^1\sqrt{1+\frac{E}{\alpha}}\mathrm{d}\lambda &= \frac{1}{2\sqrt{\alpha}}\int_1^\infty\frac{\mathrm{d}t}{t^3}\sqrt{\alpha t^3 + (t-1)^2(t-1/4)}, \qquad t = 1/\lambda^2.\end{aligned}$$

Standard techniques [1] can be used to convert this integral into Jacobian elliptic integrals for numerical evaluation [73]. Figure 5.18 displays an example equilibrium mesh with some

supports in green and the band delineated in blue ($\Omega = [-2, 2] * [0, 1.732], N_x = 40, N_y = 10, R_a = 0.05, R_b = 2.5, \beta = 0.8$). The first test varies R_a and R_b with fixed $\beta = 0.6$,

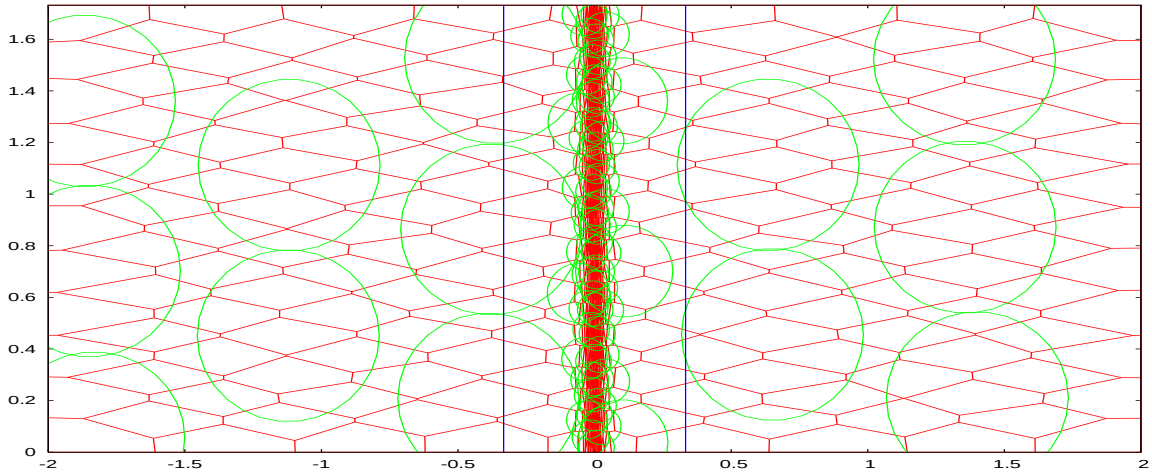


Figure 5.18: A typical equilibrium mesh with some supports (green) and the band (blue)

and the second varies β with $R_a = 0.18$, $R_b = 5.65$. Fig 5.19 compares the analytic and computed α for the equilibrium meshes - the agreement is very good.

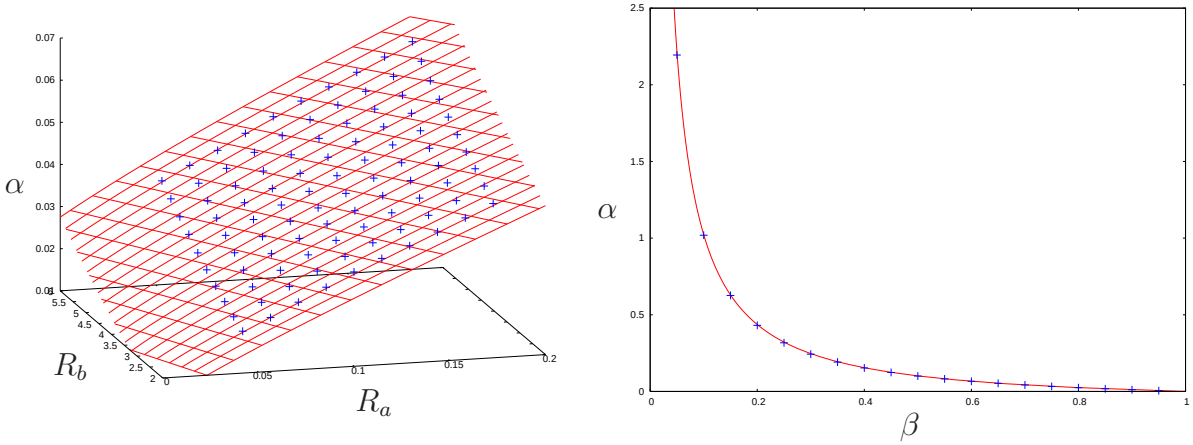


Figure 5.19: Comparison of analytic (red) and computed (blue) α

We are now in a position to ask how to set the parameters R_a, R_b and β to achieve tangible mesh qualities such as the width of the band, the minimum and maximum cell sizes and so on. We already have expressions for the width and maximum cell sizes (5.31), and the minimum cell size is

$$a_{min} = a_{max} \left(1 + \frac{E(0)}{\alpha}\right)^{-\gamma} \tag{5.35}$$

Suppose we wish to set all three - a_{min}, a_{max} and R_{max}. From (5.31) $\beta = 1 - |\Omega|/N a_{max}$, so that for $\beta > 0$, a_{max} must be greater than $|\Omega|/N$ which is reasonable because if cells in

the band are to shrink then cells outside must expand to compensate. Then from (5.35) $\alpha = E(0)/\left[\left(\frac{a_{max}}{a_{min}}\right)^{1/\gamma} - 1\right]$ and finally inverting (5.31), (5.34):

$$\begin{aligned} \frac{R_a}{\Omega_x} &= \frac{\alpha^\gamma/2}{F(\infty)-F(\alpha)}\left(\frac{F(\infty)}{\alpha^\gamma}\frac{R_{max}}{\Omega_x/2} - \frac{\beta}{1-\beta}\right) \\ \sqrt{\frac{a_{max}}{\pi}}\frac{R_b}{\Omega_x} &= \frac{\alpha^\gamma/2}{F(\infty)-F(\alpha)}\left(\frac{\beta}{1-\beta} - \frac{F(\alpha)}{\alpha^\gamma}\frac{R_{max}}{\Omega_x/2}\right). \end{aligned} \tag{5.36}$$

Now R_a, R_b cannot be negative, which restricts the values of the parameters:

$$\begin{aligned} R_a, R_b \geqslant 0 \quad &\Leftrightarrow \quad \frac{\beta}{1-\beta}\frac{\alpha^\gamma}{F(\infty)} \leqslant \frac{R_{max}}{\Omega_x/2} \leqslant \frac{\beta}{1-\beta}\frac{\alpha^\gamma}{F(\alpha)} \\ &\Leftrightarrow \quad \left(1 + \frac{\Omega_x/2}{R_{max}}\frac{\alpha^\gamma}{F(\alpha)}\right)^{-1} \leqslant \beta \leqslant \left(1 + \frac{\Omega_x/2}{R_{max}}\frac{\alpha^\gamma}{F(\infty)}\right)^{-1}. \end{aligned}$$

This is indicating that some combinations of mesh qualities are incompatible. Heuristically we are asking for a mesh with given cell sizes at the discontinuity and at the edge of the band a set distance away, plus they must expand between these smoothly in a defined profile - sometimes the cells just aren't being allowed to change quickly enough. Although this is for continuous cell areas $a(x)$, it is reminiscent of what we have seen already for finite cells, e.g. the cells in Fig 5.9a between the ellipse and the boundaries, despite R_a and R_b being manifestly nonnegative. It suggests that for finite cells the answer is similarly to increase R_a or R_b - which happened automatically by introducing R_b in the first place.

Figs 5.20 and 5.21 plot the surfaces $R_a = 0$ (red) and $R_b = 0$ (green) along with the previous sets of test data. To make the domain finite, the coordinates used are β, $\frac{a_{min}}{a_{max}}$ - which is one for a uniform mesh, and $\frac{R_{max}}{\Omega_x/2}$ - the fraction of the the domain covered by the band. The surface closest is $R_b = 0$ with $R_a = 0$ just behind, and between them is the thin region of achievable equilibrium meshes. The test data (in blue) all lie in the region as expected. The diagram confirms that there is a trade-off between the parameters - e.g. fix the width of the band (take a horizontal slice) and change the ratio of cell sizes away from uniformity and the cells outside the band must grow to compensate those shrinking inside. As the green surface is above the red it suggests that using R_b instead of R_a to gain the necessary stability in the mesh iteration comes at the expense of widening the band.

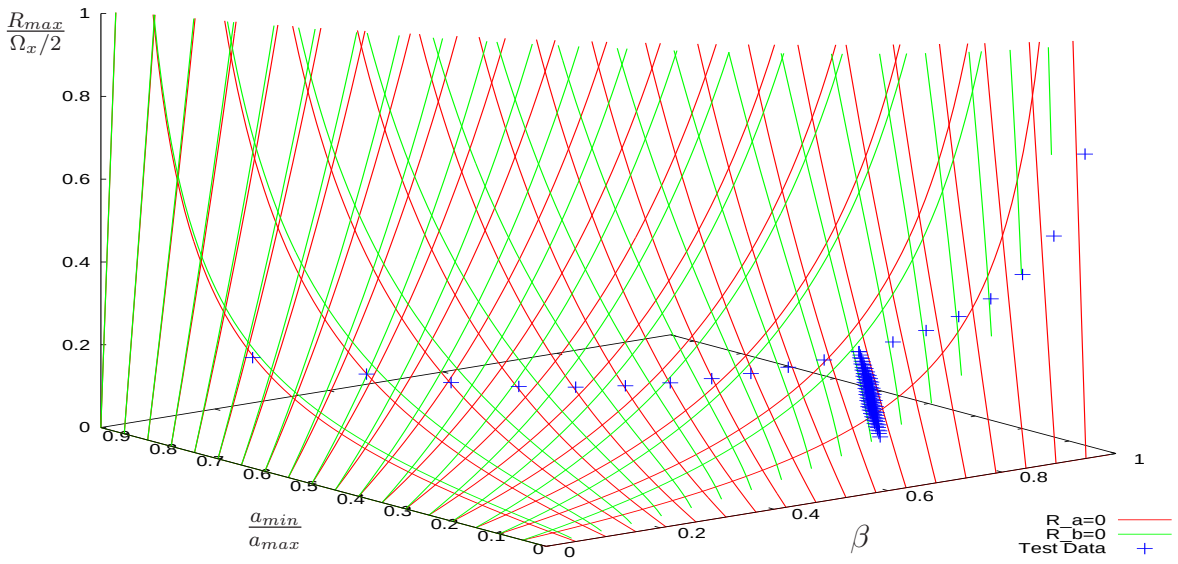


Figure 5.20: The achievable region in parameter space

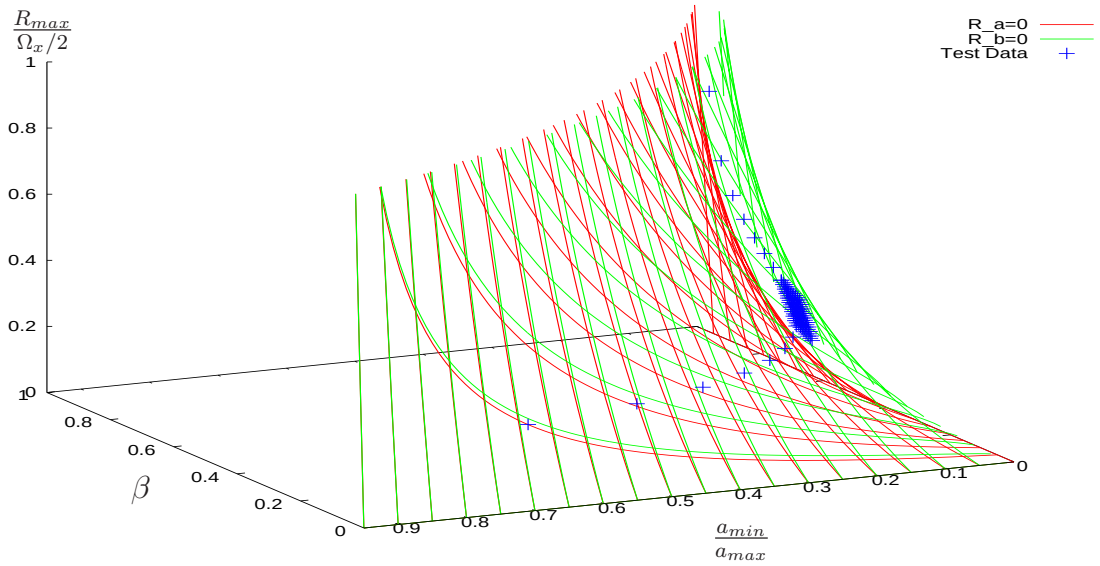


Figure 5.21: The achievable region in parameter space

It should be noted that the smoothing inherent in the weight w obviates the need for any further spatial smoothing of the monitor, unlike many moving mesh methods [42].

In conclusion we have developed an error-based monitor function that is sufficiently smooth for the mesh adaption cycle (Chapter 4, section 4.3) to converge to a static mesh for static data, despite the presence of discontinuities. Unfortunately, in the process some new parameters were introduced, but these have been related to tangible mesh qualities for both continuous and discontinuous data.

Chapter 6

Results

The algorithm is demonstrated in six test problems. The first, the Sod shock tube, is used mainly to explore the effects of the parameters and functions introduced by the novel monitor, and determine reasonable values where possible for the following problems. Some include comparisons with published solutions but others are more exploratory.

6.1 Sod shock tube

The problem definition is as follows: $\Omega = [-2, 2] \times [0, 0.5]$, $(\rho, p, u, v) = (0.125, 0.1, 0, 0)$ for $x < 0$ and $(1, 1, 0, 0)$ for $x > 0$ with reflective boundary conditions. $N_x = 40$, $N_y = 10$ (851 cells), $\beta = 0.8$ and $C_{cfl} = 0.9$. For the monitor we use (5.30) with $s = 2$, $\gamma = 1$ and $E = E_1$ (a simpler version of the gradient-like $E_1 - E_2$). The monitor quantity $U = \rho$ as it is discontinuous at both the shock and contact, $w = (1 - \lambda^2)^2$ with (5.16), and the support is set quite broad so cells are reasonably spaced for visualization ($R_a = 0.5R_0$, $R_b = 4$) where $R_0 \approx 0.052$ here (3.18). For the mesh to adapt to the initial conditions $\lambda_{mon} \lesssim 0.02$ was found necessary, but could be raised to 0.05 for $t > 0$, and to minimise the probability of remapping it is scaled by k/N_{cycle} on the k^{th} cycle.

Fig 6.1 shows the initial mesh with selected supports:

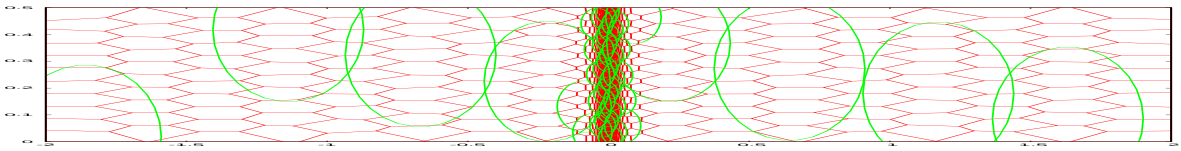


Figure 6.1: Initial mesh with selected supports for Sod shock tube problem

Fig 6.2 zooms in on the discontinuity to see the cells.

Fig 6.3 shows the mesh at 0.2 intervals (all time units are in seconds) until the end time $t = 0.8$ - the mesh has concentrated over the shock, contact and rarefaction fan as expected and has successfully remained one dimensional. No remapping was required.

Fig 6.4 shows the flow variables at end time. In each plot a cell is drawn as single point over the centroid, with connecting lines corresponding to shared edges (i.e. the dual space

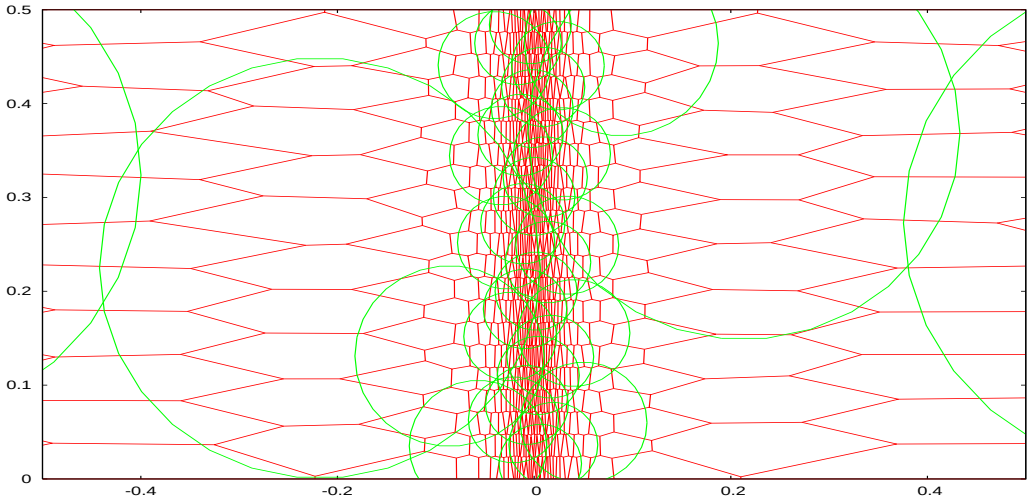


Figure 6.2: Close up of the cells over the discontinuity

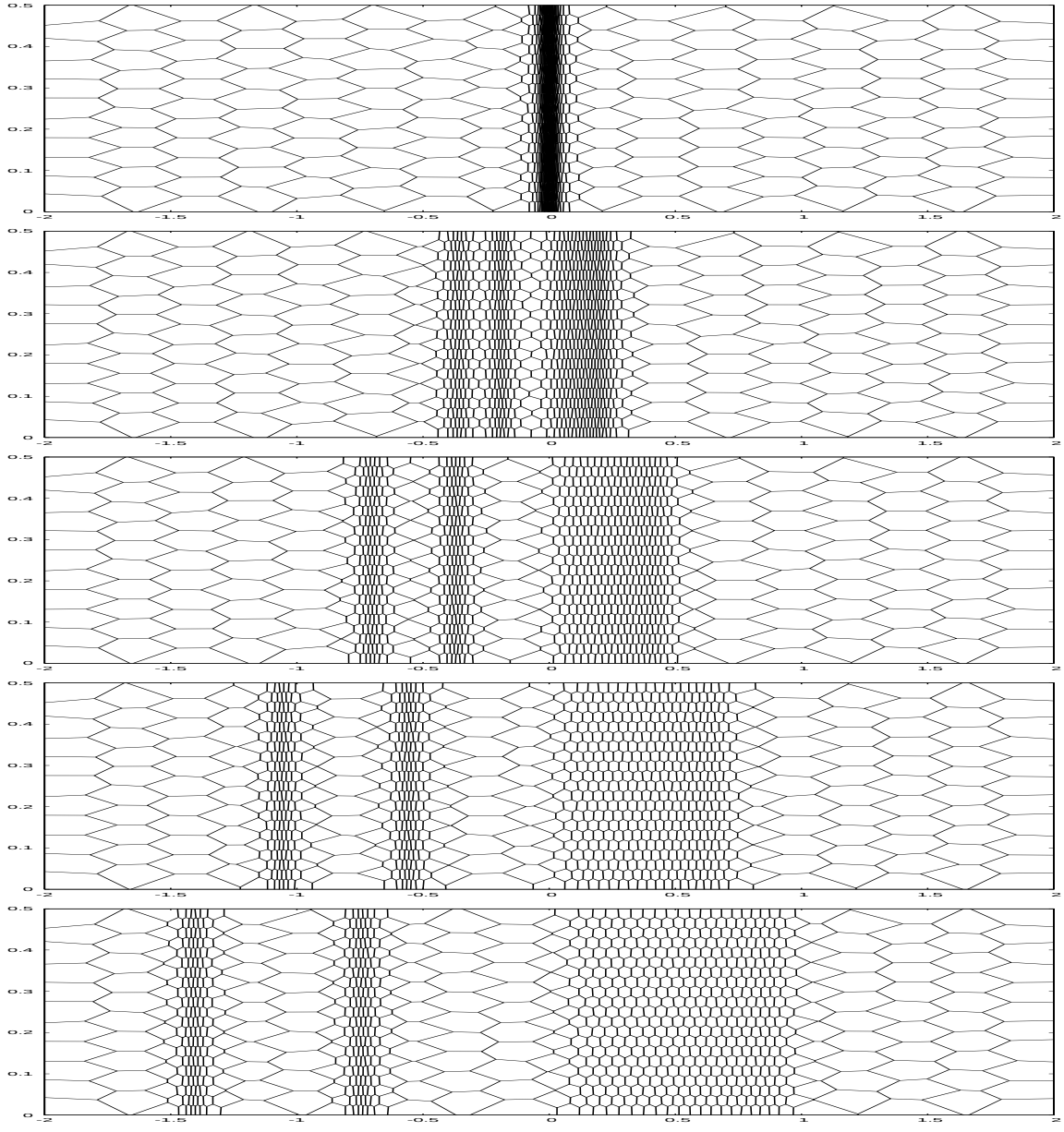


Figure 6.3: Mesh for Sod shock tube problem at (top to bottom) $t = 0, 0.2, 0.4, 0.6, 0.8$

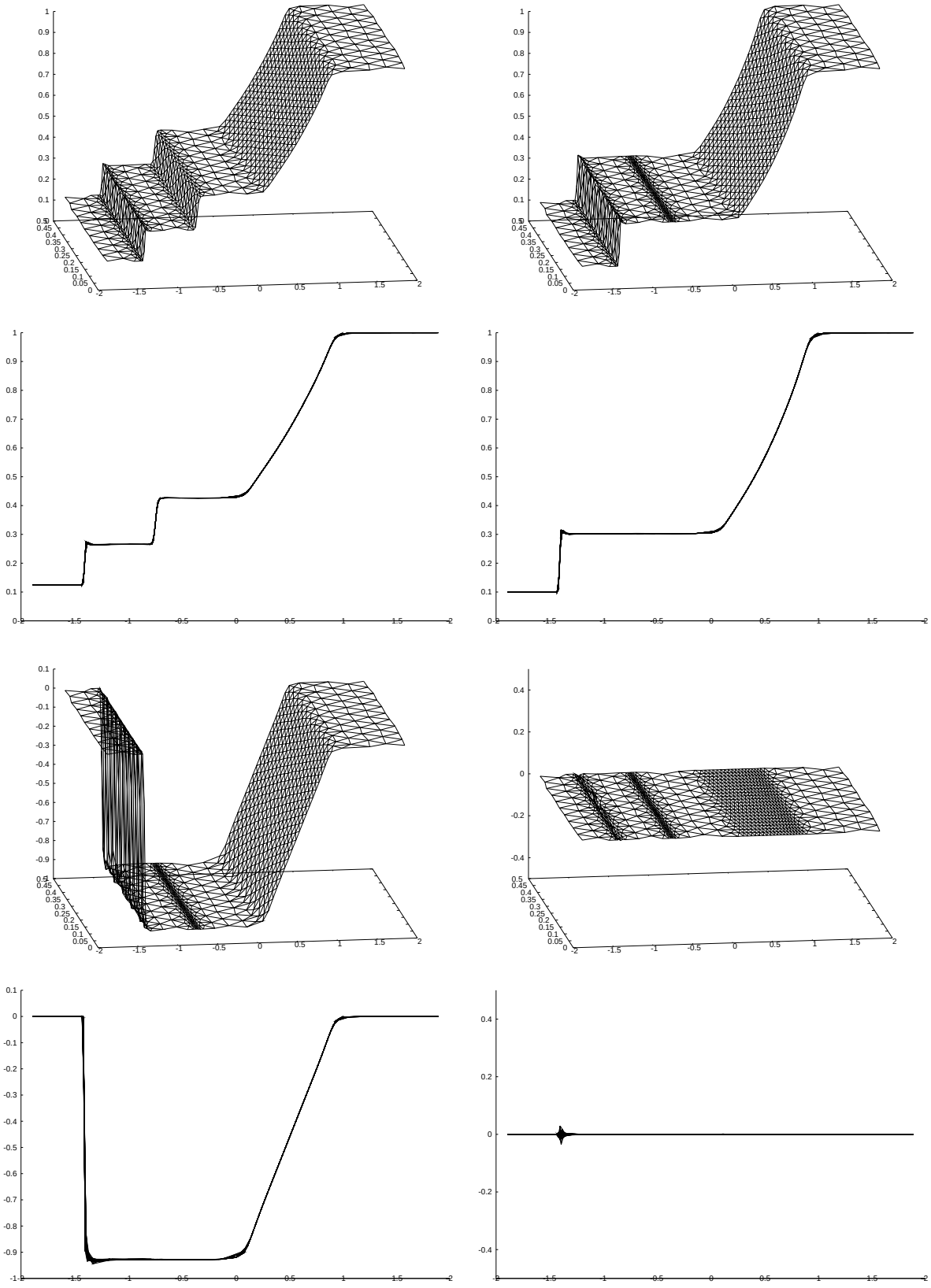


Figure 6.4: Density (top left), pressure (top right), u (bottom left) and v (bottom right) at end time viewed both from an angle and side-on

triangulation). The shock is smeared over 2 cells and the contact about 4 cells, and the variables have remained one dimensional, but there are noticeable overshoots at the shock despite this using the monotone corrector (section 4.3). The results for the second-order corrector are indistinguishable, so this is attributed to the fact that the predictor does not use any information from neighbouring cells.

Fig 6.5 compares the effect on the end time mesh of different λ_{mon} for $t > 0$. The mesh is identical for 0.1 and 0.05, validating the earlier choice of 0.05, but there are slight differences with 0.02 - the cells between the discontinuities have not been allowed to fully expand back - and with 0.01 the cells cannot keep up with the discontinuities in the first place.

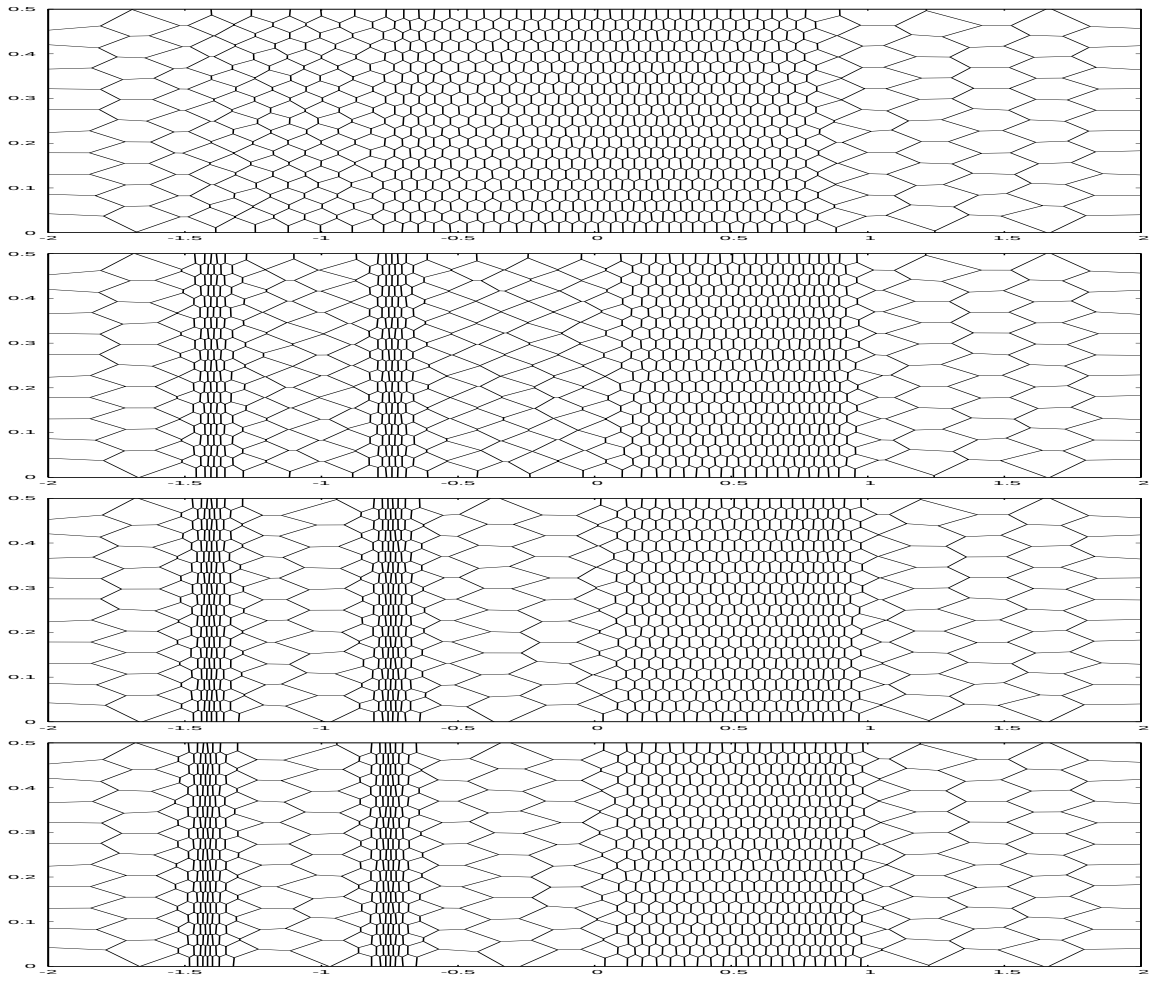


Figure 6.5: End time mesh for (top to bottom) $\lambda_{mon} = 0.01, 0.02, 0.05, 0.1$ for $t > 0$

Fig 6.6 overlays the end time mesh for (5.16) (red, taking 198 seconds), (5.15) (green, 267 secs) and (5.14) (blue, 423 secs) - they only differ significantly where the monitor is very small, which is to be expected as approximations to the weight integral will be poorer where there are fewer datapoints. (5.16) is used from here on.

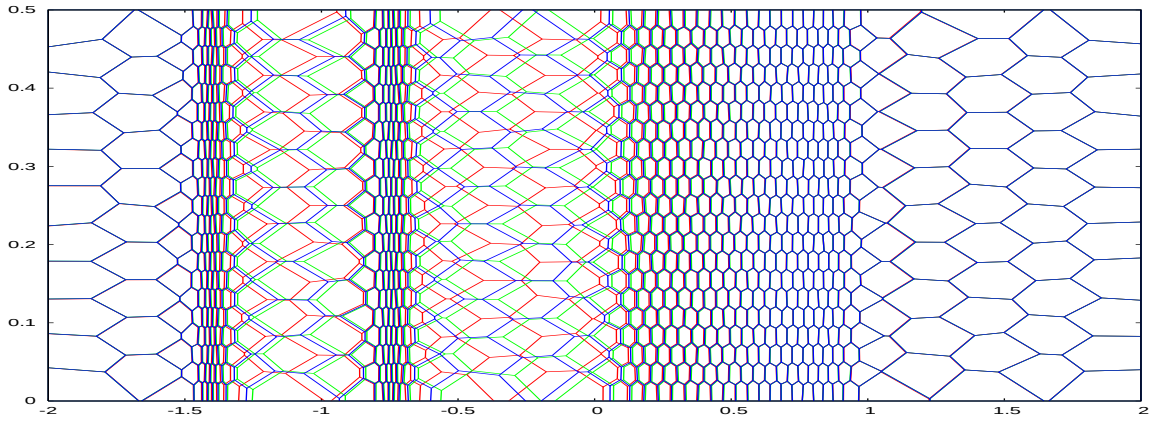


Figure 6.6: End time mesh for (5.16) (red), (5.15) (green) and (5.14) (blue)

Fig 6.7 compares the effect on the final mesh of increasing R_a or R_b - both enlarge the smallest cells:

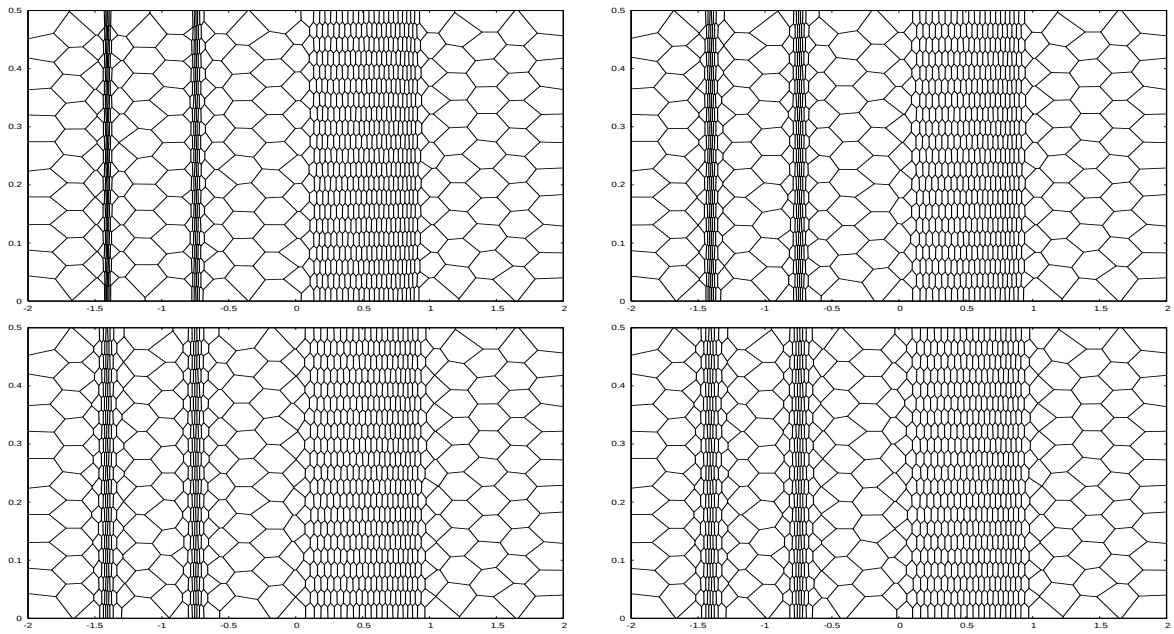


Figure 6.7: End time mesh for $R_a/R_0 = 0.25$ (left), 0.75 (right) and $R_b = 2.5$ (top), 4 (bottom)

Now the timestep is determined by both the wave speeds, which are the same in all cases (ignoring mesh dependency in the velocity), and the cell sizes, so should increase with R_a or R_b. Fig 6.8 compares the average timestep over the calculation with the minimum cell size at $t = 0$ for $R_a/R_0 \in \{0.25, 0.5, 0.75\}$, $R_b \in \{2.5, 3, 3.5, 4\}$ - the general trend is present although the relationship is not linear.

The reason for this is clearer from the end time monitor (which is related to the area via equidistribution (1.12)), shown in Fig 6.9.

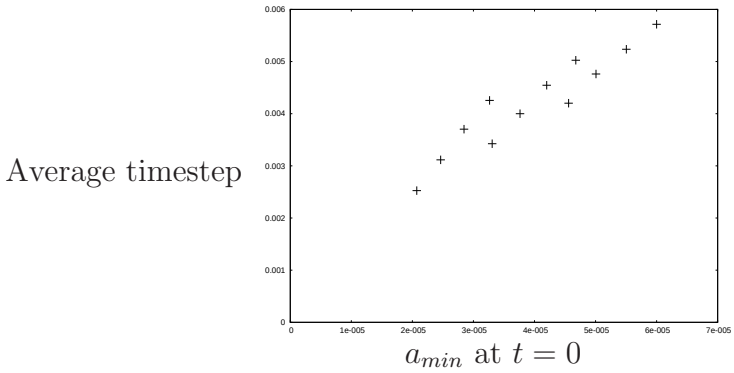


Figure 6.8: Plot of average timestep against minimum cell size at $t = 0$

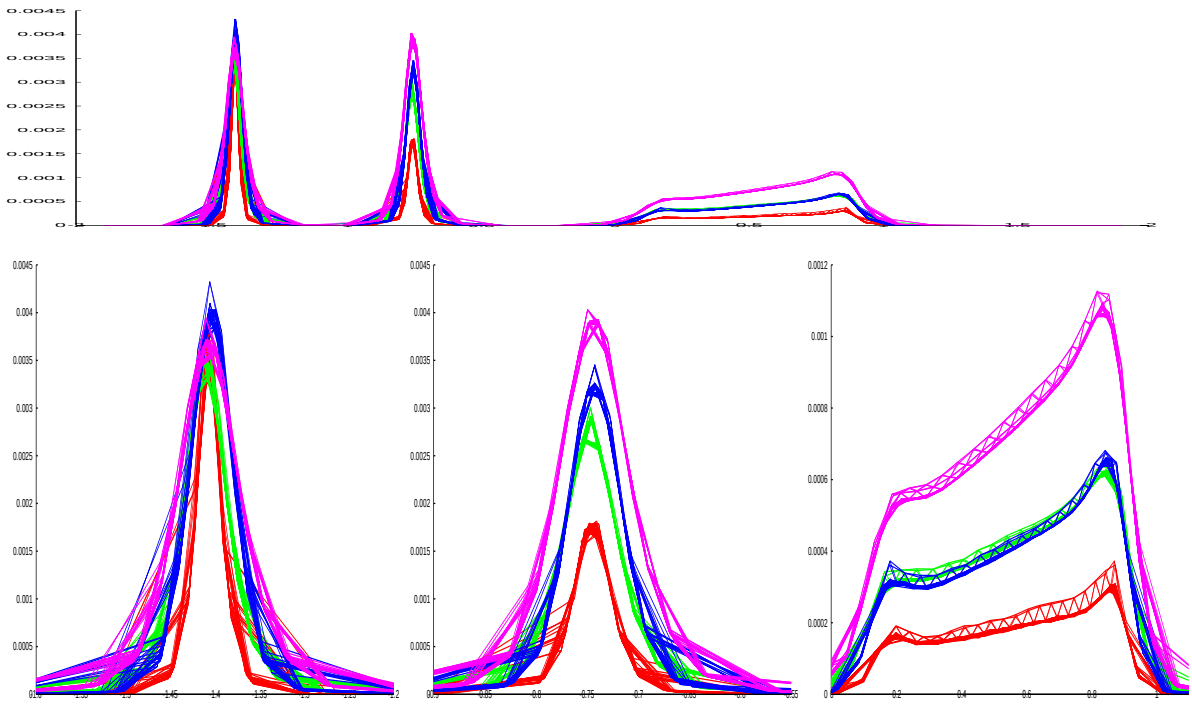


Figure 6.9: End time monitor for $(R_a/R_0, R_b) = (0.25,2.5)$ red, $(0.25,4.0)$ green, $(0.75,2.5)$ blue, $(0.75,4)$ pink for complete mesh (top) and individual features (bottom row)

We can now see that as R_a or R_b are increased, not only are the smallest cells enlarged, but the relative distributions change as well. The monitor (and hence cells) is redistributed from the shock to the contact and rarefaction, with the shock and contact being even for the pink case (bottom right Fig 6.7)). However this is at the expense of increasing the support numbers (the number of cells in a support), plotted in Fig 6.10.

For $R_b = 4$ (green and pink) they are excessively large, but the red is practical, being around 10 for large parts of the mesh (below $R_a/R_0 \lesssim 0.1$ or $R_b \lesssim 2.0$ mesh adaption fails). It is noticeable that the support numbers are highest not where the cells are smallest (over the shock or contact) but nearby, where the cell sizes are changing most rapidly.

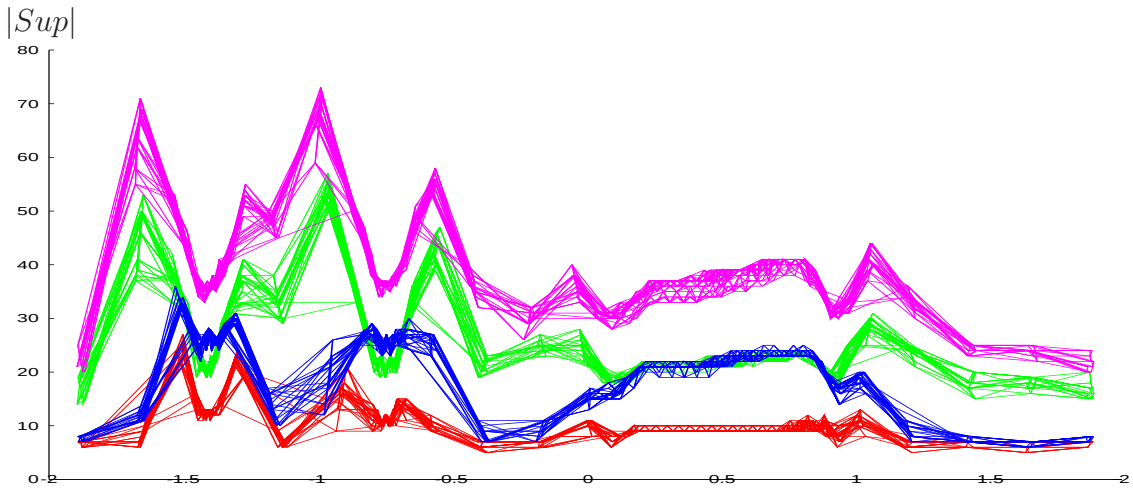


Figure 6.10: Support numbers for $(R_a/R_0, R_b) = (0.25,2.5)$ red, (0.25,4.0) green, (0.75,2.5) blue, (0.75,4) pink

Changing the weight w has a similar but less pronounced effect (after reducing λ_{mon} to 0.002 at $t = 0$ so the mesh can settle to the initial conditions). Fig 6.11 shows the

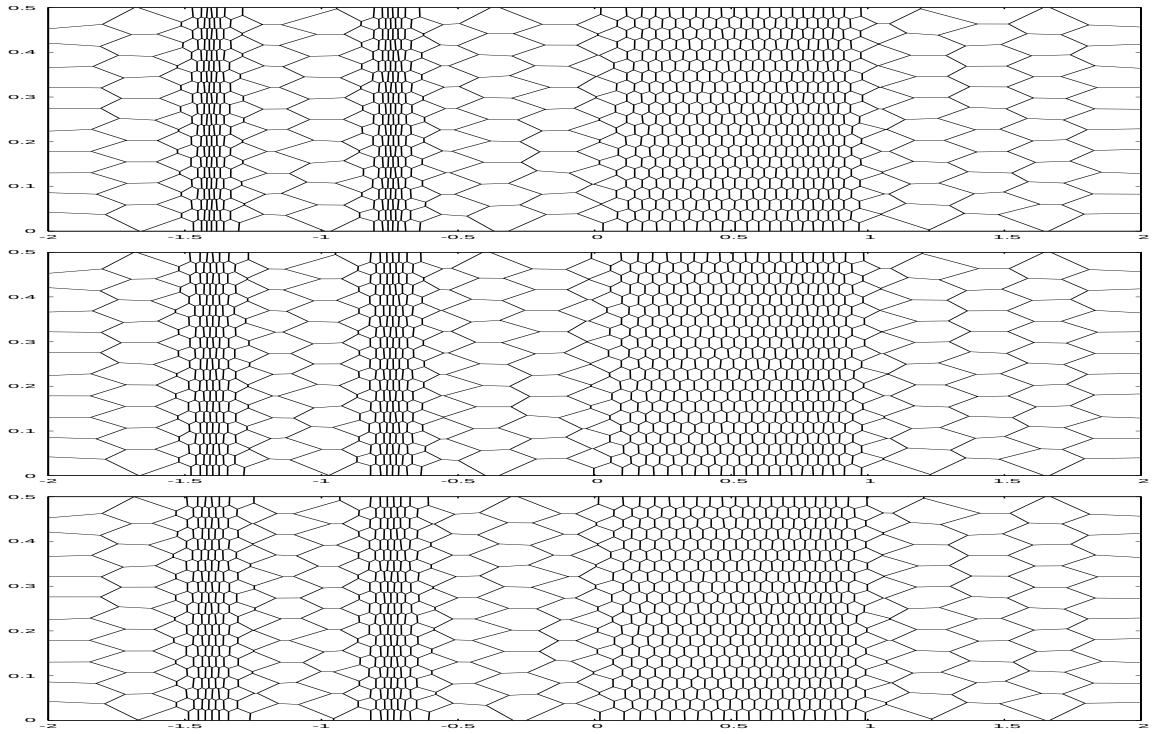


Figure 6.11: End time mesh for (top to bottom) $w = (1 - \lambda^2)^2, \sqrt{1 - \lambda^2}, 1$

meshes and Fig 6.12 the monitor at end time. As the weight becomes less sharply peaked cells are redistributed from the shock and contact to the rarefaction, but in this case the support numbers are not changed. However $w = 1$ triggered remaps on six occasions (out of 199 timesteps), so we choose $w = (1 - \lambda^2)^2$ from here on.

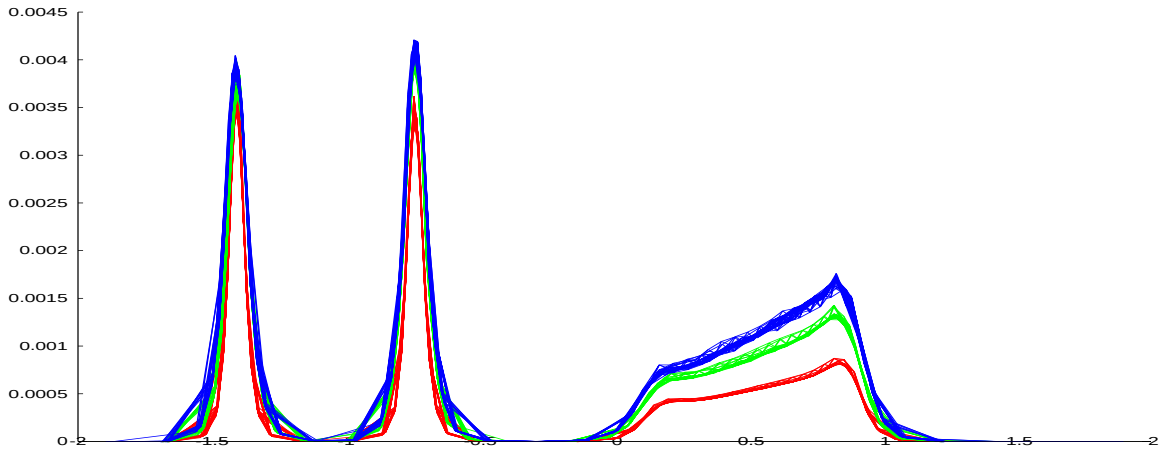


Figure 6.12: End time monitor for $w = (1 - \lambda^2)^2$ (red) $w = \sqrt{1 - \lambda^2}$ (green) and $w = 1$ (blue)

Having gained some understanding of λ_{mon}, R_a, R_b and w we turn now to the choice of E used in the monitor, which so far has been $E = E_1$. Fig 6.13 shows the least squares errors at end time for the baseline calculation.

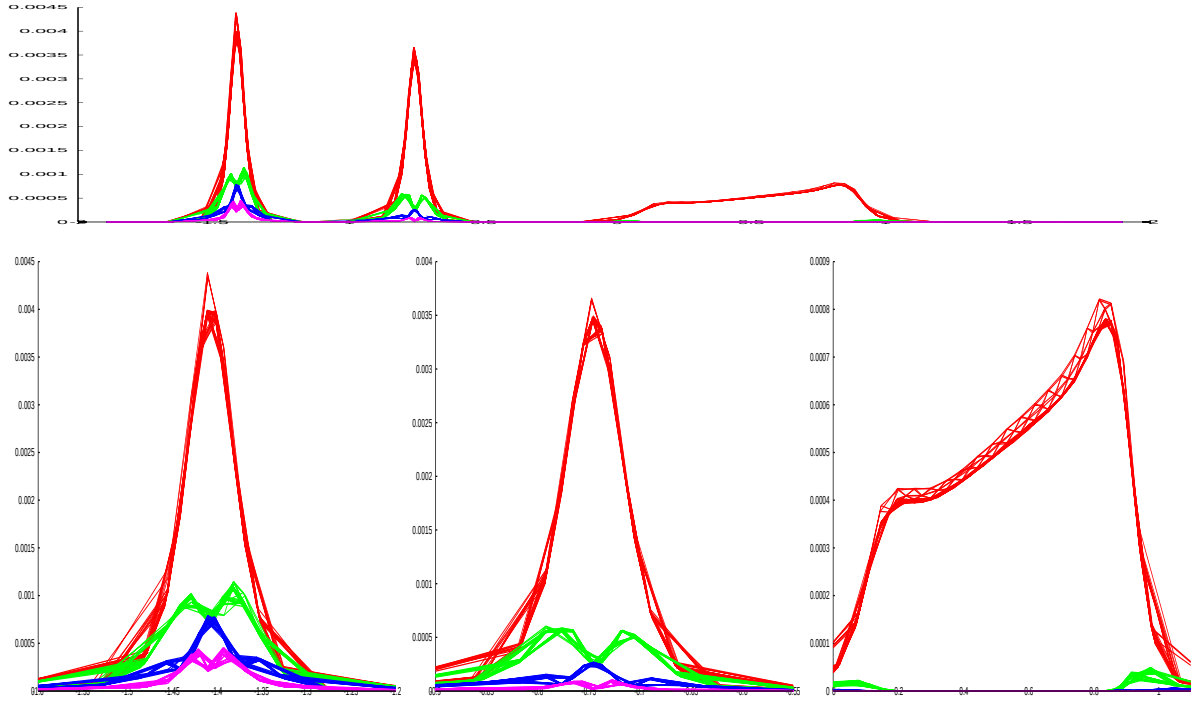


Figure 6.13: The least square errors at end time - E_1 (red), E_2 (green), E_3 (blue) and E_4 (pink) for complete mesh (top) and individual features (bottom)

The errors for the shock and contact are similar to the analytic errors for the fixed radius support (Fig 5.17). Now the discretization scheme is second-order in time and

space which means that it should be exact when the variables are all linear, regardless of their gradients. This suggests that in general the truncation error will be more closely represented by the deviation from linearity, i.e. the errors E_2, E_3 etc, rather than the gradient itself, approximated by $E_1 - E_2$. For example the gradient-type monitor used here so far has concentrated cells throughout the rarefaction fan which is wasteful as the truncation error will be greatest at the discontinuities in gradient at the ends, exactly as indicated by E_2 in green at the right of Fig 6.13. Potential problems are that E_2 is not singly peaked and the higher order errors become rapidly noisier but these turn out to be largely irrelevant. Fig 6.14 shows the final meshes for various errors:

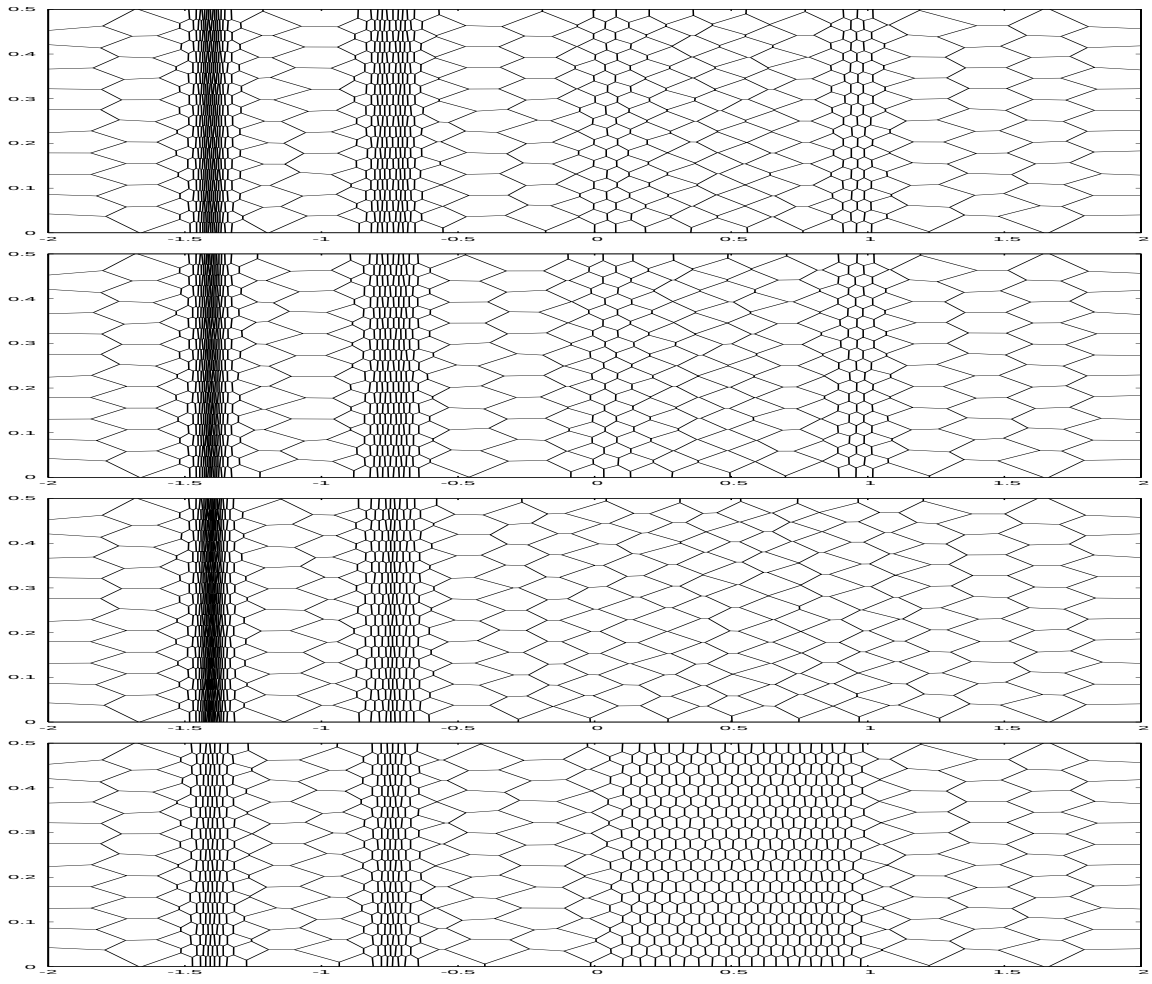


Figure 6.14: End time mesh for $E = E_2$, $E_2 + E_3$, E_3 and $E_1 + E_2 + E_3$ (top to bottom)

Unfortunately the higher order errors are biased towards the shock, and because the structure of the shock is more stable than the contact (characteristics flow into rather than with it) these differences are amplified with time. From Fig 6.13 this is true for all the higher order errors so no linear combination can correct for this. Comparing the end time monitor of the baseline E_1 (red) and $E_1 + E_2 + E_3$ (green) in Fig 6.15 only confirms

this.

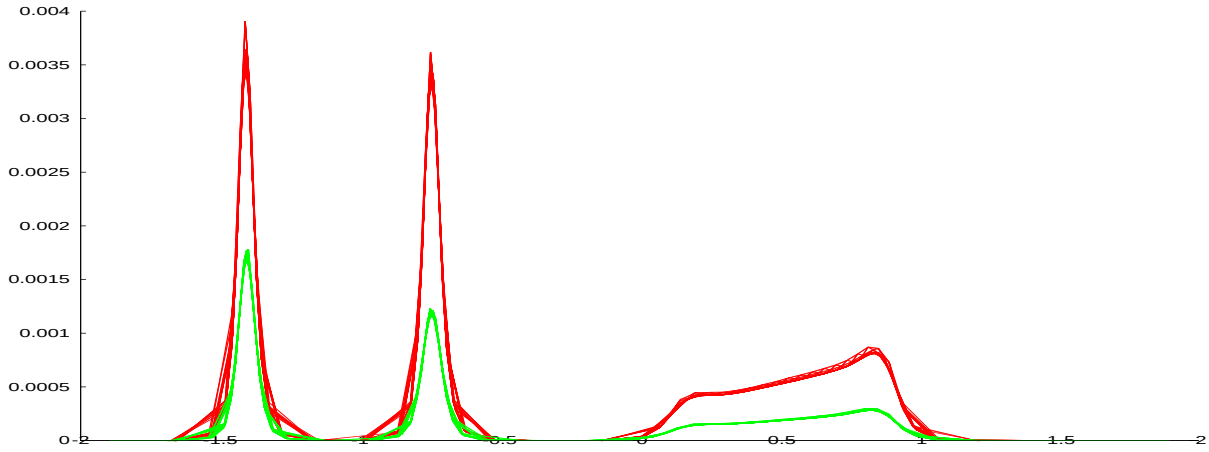


Figure 6.15: End time monitor for $E = E_1$ (red) and $E_1 + E_2 + E_3$ (green)

Increasing R_a or R_b to counteract the bias as in Fig 6.9 is not an option as the support numbers are already too large (somewhere between the green and pink in Fig 6.10), so we conclude that the higher order errors cannot help here and that $E = E_1$ is best. Unless otherwise stated $E = E_1$ from here on.

6.2 Huang and Sun's five circles problem

This static problem is taken from Huang and Sun [46], Example 2, in which

$$U(x,y) = \sum_{i=1}^{5} \tanh\left(30\left[(x-x_i)^2 + (y-y_i)^2 - \frac{1}{8}\right]\right) \tag{6.1}$$

describing five osculating circles with centres $(x_1, y_1) = (0, 0)$, $(x_i, y_i) = (\pm 0.5, \pm 0.5)$, $i = 2, 3, 4, 5$ in the domain $\Omega = [-2, 2] \times [-2, 2]$. We set $\boldsymbol{P} = (1, x, y)$ (linear approximation) and stop the iteration when the fractional change in monitor drops below 10^{-4}. We define $e_1(x, y)$ to be the difference between (6.1) and the least squares approximation on the converged mesh and $|Sup|$ to be the average size of the support on the converged mesh. Fig 6.16 shows the converged meshes for $E = E_1$ and $E = E_2$ where $N_x = N_y = 56$ (6385 cells), $R_a/R_0 = 0.1$, $R_b = 2.5$, $m = 1 + \sqrt{E}/\alpha$, weight (5.14) and $\beta = 0.5$. As with Huang and Sun, we see the mesh concentrates either directly over the circles (for $E = E_1$) or in bands either side ($E = E_2$), an effect we can now attribute to whether the monitor is singly or doubly peaked over each circle, as in Fig 5.17.

Fig 6.17 compares the variation of $\|e_1\|$, the L_2 norm of e_1, with $|Sup|$ over different values of the parameters: $R_a/R_0 \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$, $R_b \in \{1.5, 2.5, 3.5, 4.5, 5.5\}$,

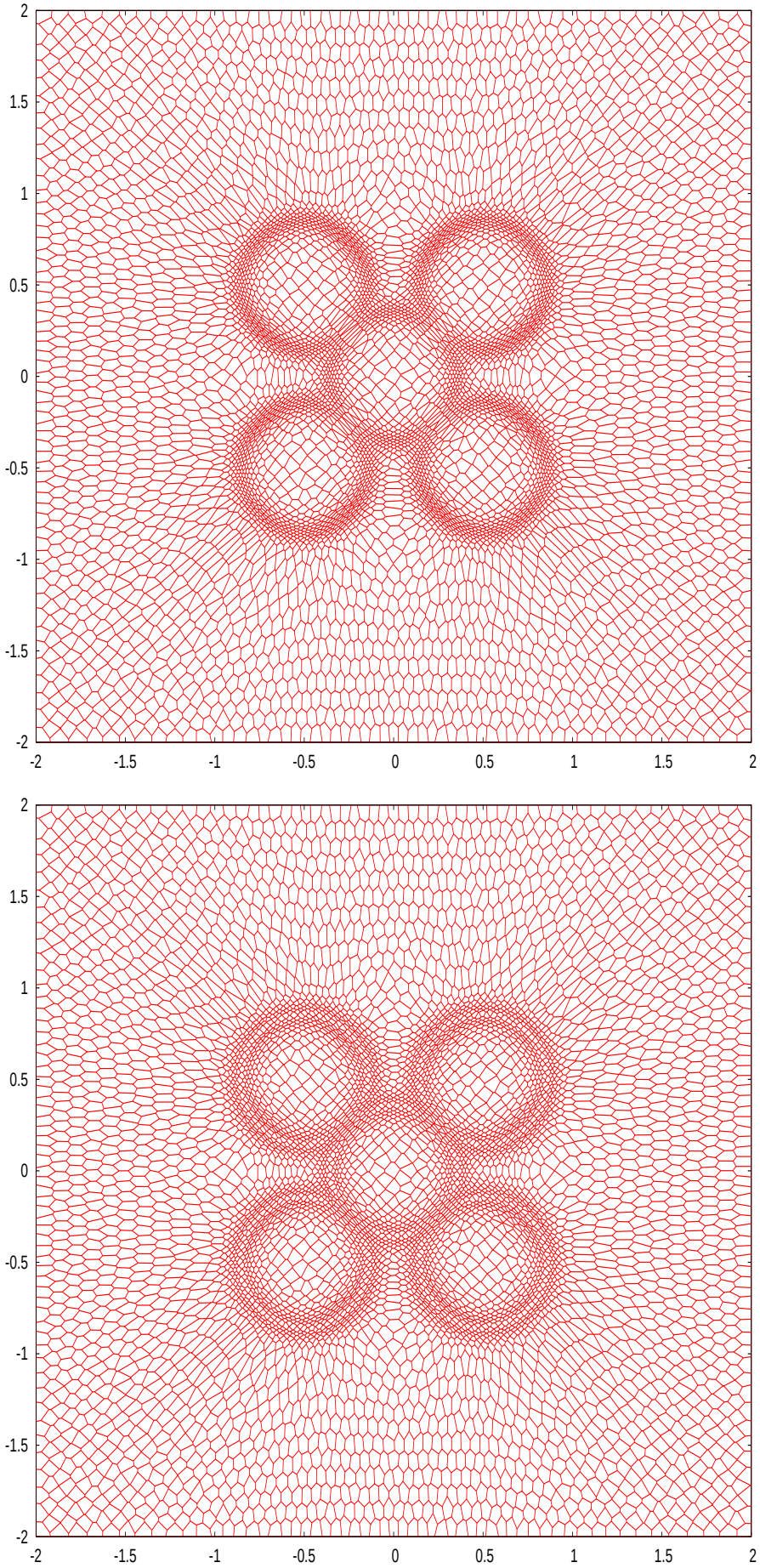


Figure 6.16: Converged mesh for $E = E_1$ (top) and $E = E_2$ (bottom)

$m \in \{\sqrt{1+E/\alpha}, 1+\sqrt{E}/\alpha\}$, $E \in \{E_1, E_2, E_1+E_2\}$, weights (5.14), (5.15) and (5.16).

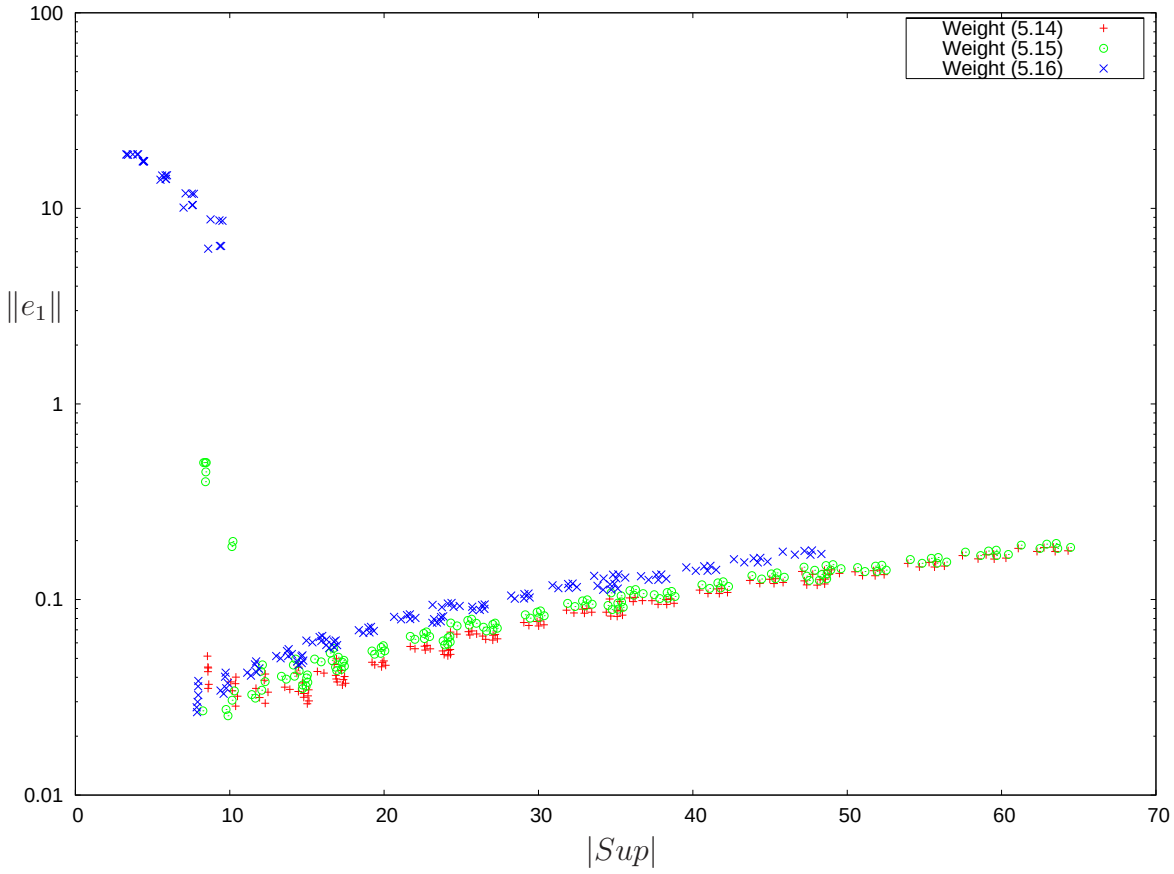


Figure 6.17: Variation of error with support size for various combinations of parameters

The general pattern is that $\|e_1\|$ increases with the support size, which is to be expected as the reconstruction becomes less tightly focused on the cell it is centred on (and where e_1 is evaluated). However if the support size drops below a certain threshold (around 10 here) the mesh fails to settle and the approximation becomes very poor. It is no surprise that for a given support size, $\|e_1\|$ is smallest for the most accurate weight - (5.14). Also, there is little difference between $m = \sqrt{1+E/\alpha}$ and $m = 1+\sqrt{E}/\alpha$, or between $E = E_1$, $E = E_2$ and $E = E_1 + E_2$. The smallest value of $\|e_1\|$ obtained here is about 2.54e-2 which is comparable with Huang and Sun's 1.77e-2 ([46], 81^2 cells, $(k, m) = (1, 0)$).

6.3 Rayleigh-Taylor instability

In this problem gravity acts downwards and the upper half of the rectangular domain is made denser than the lower. With zero velocity the system is in unstable equilibrium, but an initial perturbation starts the upper region falling down the left hand side, whilst a bubble of the lower region rises up the right hand side to replace it. Where they pass, the contact surface is unstable and starts to roll up forming a characteristic inverted mushroom shape.

The problem definition is as follows: $\Omega = [0, 0.5] \times [-1, 1]$, $\rho = 2, 1$ for $y > 0, < 0$ with reflective boundary conditions. Gravity $\boldsymbol{g} = (0, -1)$, which is included by modifying $\boldsymbol{S} \rightarrow \boldsymbol{S} + (0, \rho g_x, \rho g_y, u g_x + v g_y)^T$ in the predictor-corrector scheme. The initial velocity is $u = sign(-y) k e^{-2\pi|y|/l_0} \sin(2\pi x/l_0)$, $v = k e^{-2\pi|y|/l_0} \cos(2\pi x/l_0)$ where $k = 1.02$, $l_0 = 1$ and the pressure is set so as to be in equilibrium ($p = p_0 + g_y \rho y$, $p_0 = 2$). $N_x = 40$, $N_y = 10$ (851 cells), $\beta = 0.8$ and $C_{cfl} = 0.95$. For the monitor, $m = \sqrt{1 + E/\alpha}$, $U = \rho$, $R_a/R_0 = 0.25$, $R_b = 2.5$ ($R_0 = 0.037$) and $\lambda_{mon} = 0.02$.

Figs 6.18 and 6.19 show the evolution of the mesh. The perturbation also causes a pressure wave to travel upwards from the contact, reaching the top of the domain at about $t = 0.5$, and a smaller release wave to travel downwards, which is more visible in the 3D density plots in Figs 6.20 and 6.21. The calculation took 1121 timesteps in 922 seconds and required no remaps.

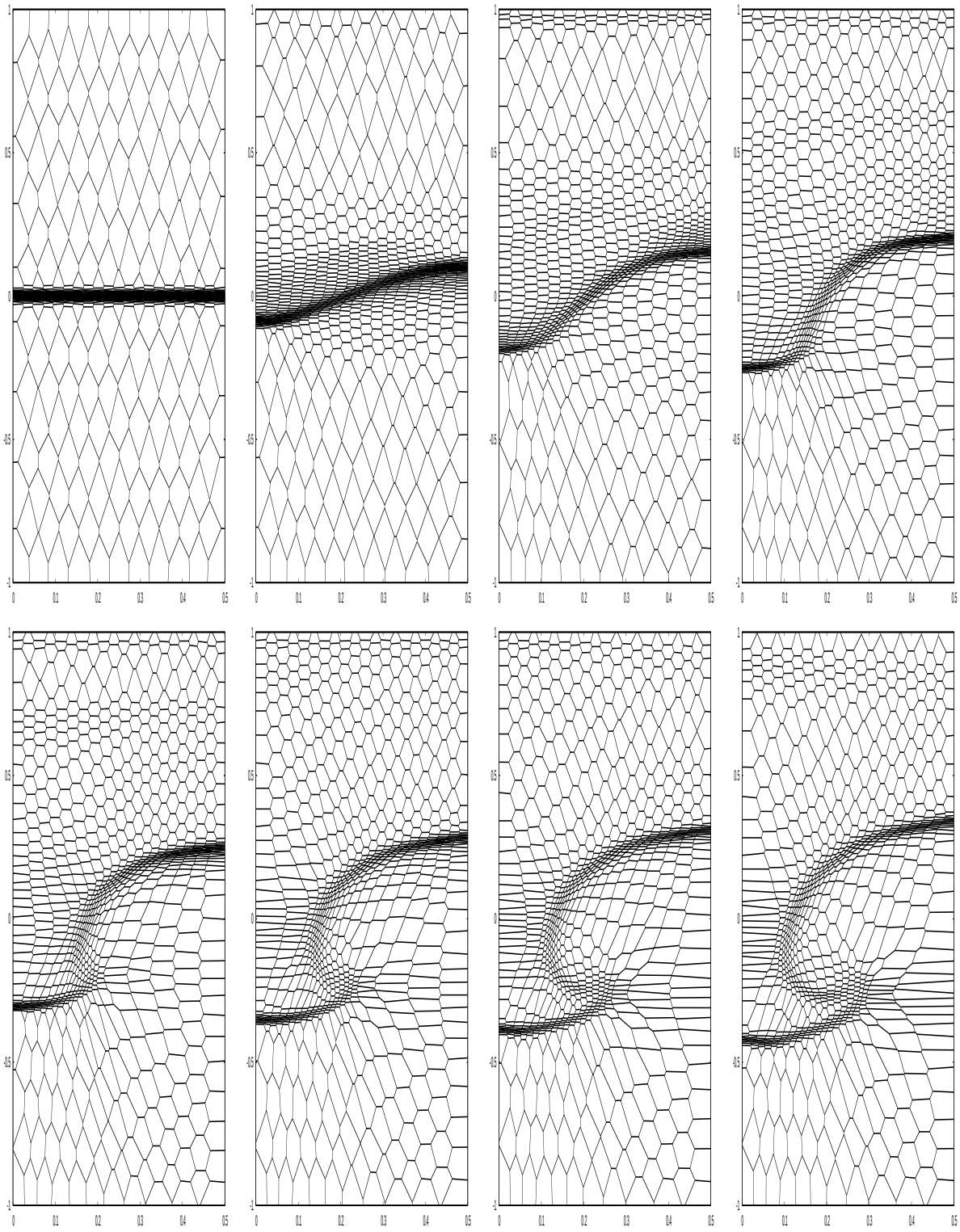


Figure 6.18: Mesh at $t = 0, 0.1, 0.2, 0.3$ (top row, left to right) and $t = 0.4, 0.5, 0.6, 0.7$ (bottom row, left to right)

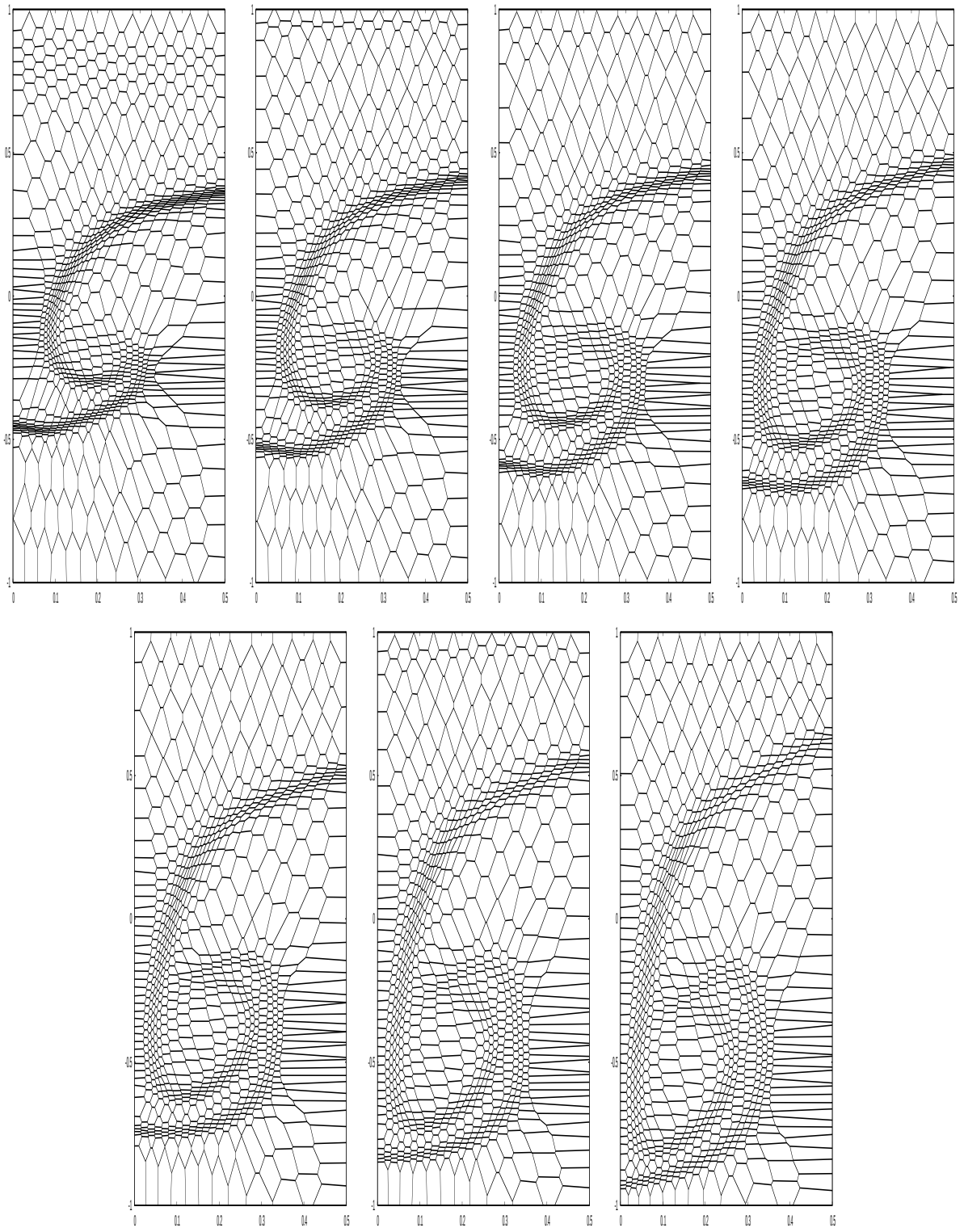


Figure 6.19: Mesh at $t = 0.8, 1, 1.2, 1.4$ (top row, left to right) and $t = 1.6, 1.8, 2$ (bottom row, left to right)

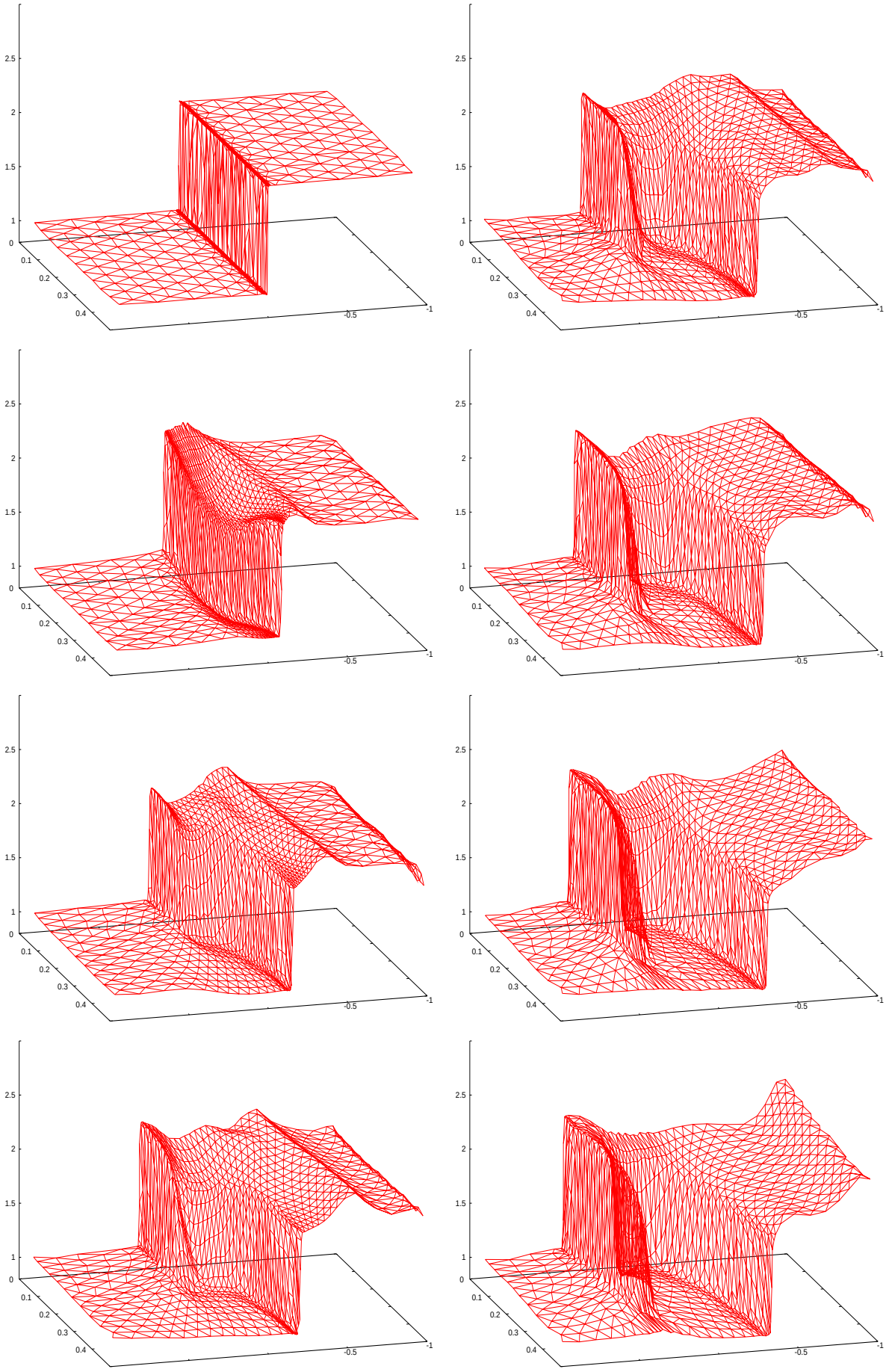


Figure 6.20: Density at $t = 0, 0.1, 0.2, 0.3$ (left column, top to bottom) and $t = 0.4, 0.5, 0.6, 0.7$ (right column, top to bottom)

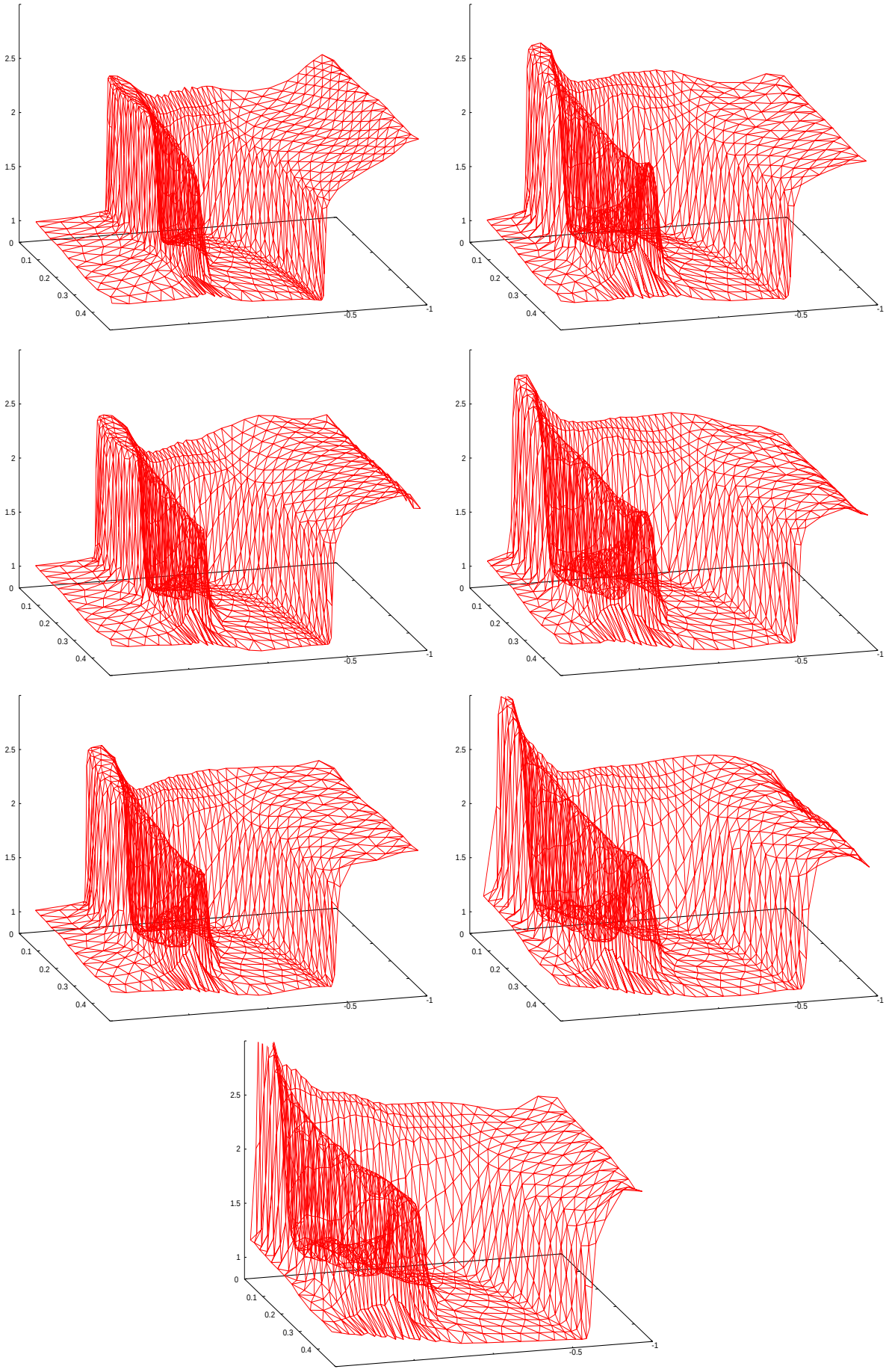


Figure 6.21: Density at $t = 0.8, 1, 1.2$ (left column, top to bottom), $t = 1.4, 1.6, 1.8$ (right column, top to bottom) and $t = 2$ (bottom)

Fig 6.22 compares the density contours at various times for $m = \sqrt{1 + E/\alpha}$ with $m = 1 + \sqrt{E}/\alpha$ - the differences are very slight, suggesting this problem is less sensitive to the monitor regularisation.

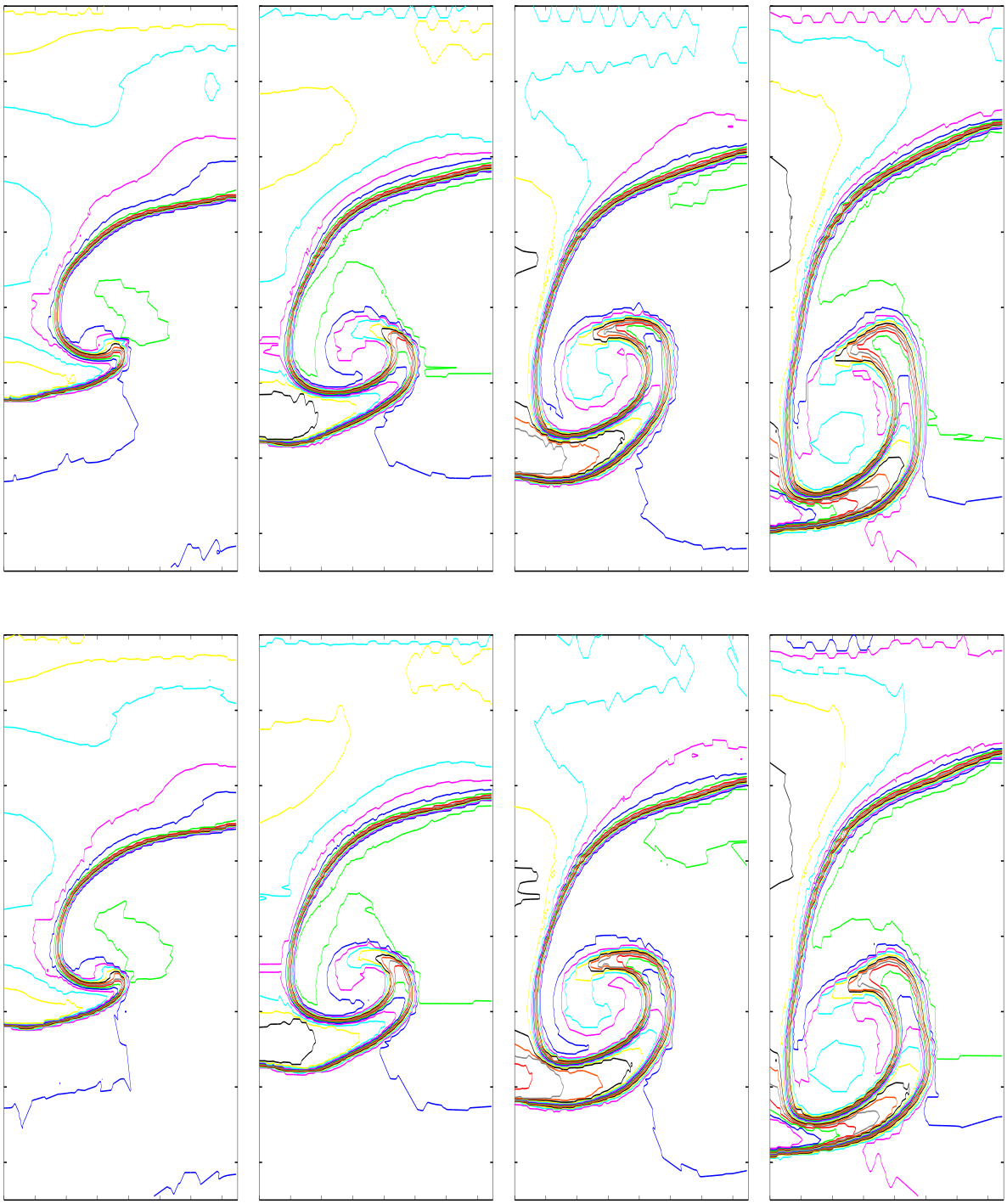


Figure 6.22: Density contour plots for $m = \sqrt{1 + E/\alpha}$ (top row) and $m = 1 + \sqrt{E}/\alpha$ (bottom row) at $t = 0.6, 1.0, 1.4, 1.8$ (left to right) for $0.9 \leqslant \rho \leqslant 2.8$ in 0.1 increments

Fig 6.23 shows the total number of edge flips experienced by each cell at the end time - cells passed by the contact surface average about ten and those in the roll up about twenty indicating significant mesh restructuring, but nearly all cells have changed connectivity several times.

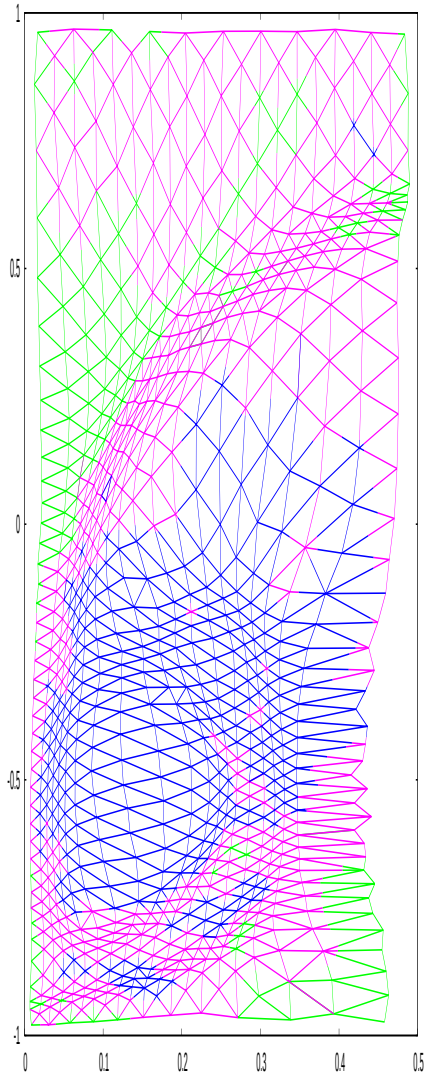


Figure 6.23: Total edge-flips per cell at the end time (green=0-5, pink=5-15, blue=15-31)

6.4 2D Riemann problem

This problem, introduced by Kurganov and Tadmor [54], is a 2D extension of the Riemann problem in which a square domain is divided into four quarters. As time evolves, the dense lower left quarter moves diagonally into the dense stationary upper right quarter creating a region of higher density between them.

The definition is as follows: $\Omega = [0,1] \times [0,1]$, (ρ, p, u, v) =(1.1,1.1,0.8939,0.8939) in the bottom left corner $(x, y \leqslant 0.5)$, (0.5065,0.35,0,0.8939) bottom right $(y \leqslant 0.5 < x)$, (0.5065,0.35,0.8939,0) top left $(x \leqslant 0.5 < y)$, (1.1,1.1,0,0) top right $(x, y > 0.50)$ with transmissive boundary conditions. $N_x = N_y = 50$ (5101 cells), $\beta = 0.8$ and $C_{cfl} = 0.75$. For the monitor, $m = 1 + \sqrt{E}/\alpha$ or $m = \sqrt{1 + E/\alpha}$, $U = \rho$, $R_a/R_0 = 0.25$, $R_b = 2.5$ ($R_0 = 0.025$), and $\lambda_{mon} = 0.02$.

Fig 6.24 shows the initial mesh in full on the left, and on the right zooms in on the centre. Moving away from the centre along the diagonals, the cells become quite stretched which is an artifact of the fixed dual-space connectivity, but the monitor ensures that as the shocks move, cells become aware of them in good time and shrink smoothly in response.

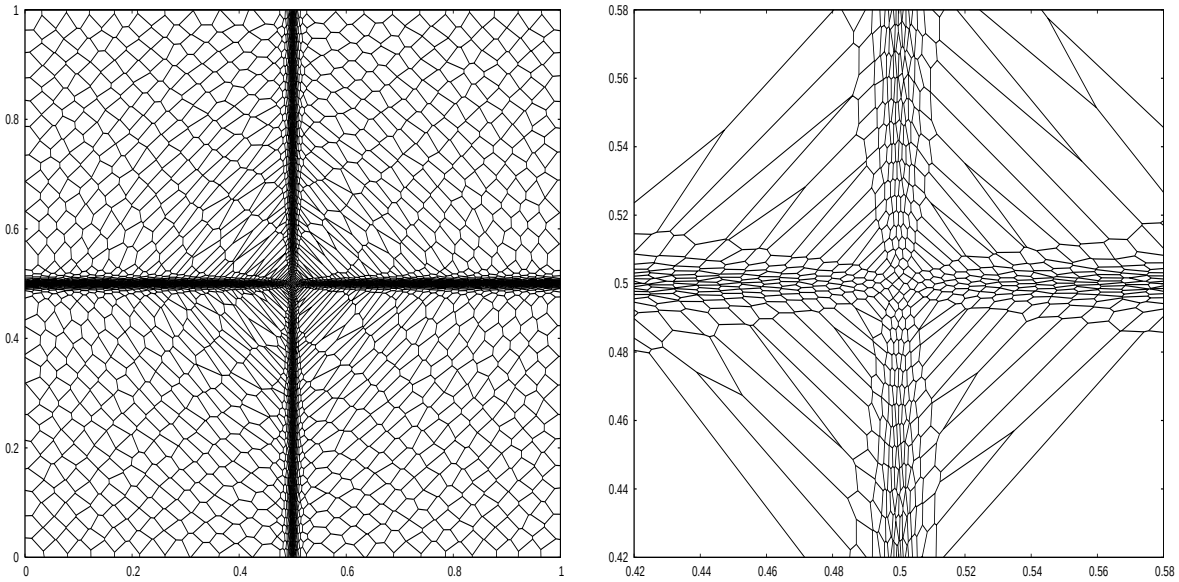


Figure 6.24: Initial mesh for 2D Riemann problem

Fig 6.25 shows the mesh and density at the end time $t = 0.2$ (13 remaps were required in 1143 timesteps taking 110 minutes for $m = \sqrt{1 + E/\alpha}$, and 22 remaps in 1653 timesteps taking 160 minutes for $m = 1 + \sqrt{E}/\alpha$).

Comparing with the uniform calculation of Morrell [65, Fig 4.4.1, 100^2 cells], both monitor regularisations achieve the correct shock location but in this problem $m = \sqrt{1 + E/\alpha}$

is clearly superior, capturing the valley down the middle of the high density region, and compares favourably with her ALE-AMR calculation [65, Fig 4.4.2, $\sim$ 4500 cells].

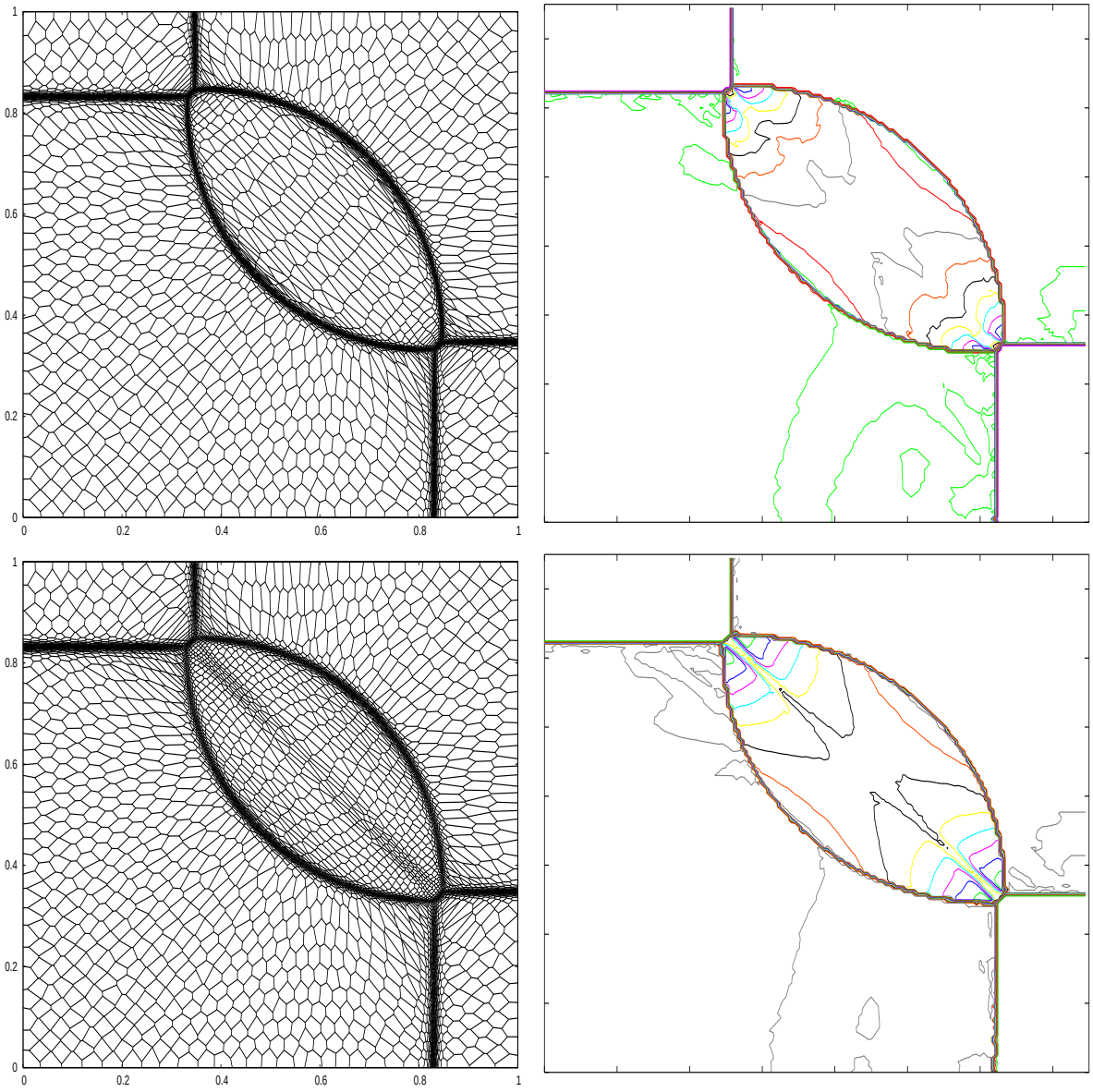


Figure 6.25: End time mesh and density (contours $0.55 \leqslant \rho \leqslant 2$ in 0.05 increments) for $m = 1 + \sqrt{E}/\alpha$ (top) and $m = \sqrt{1 + E/\alpha}$ (bottom)

6.5 LeVeque spherical blast

This problem, introduced by LeVeque [56], is a spherical explosion in an axisymmetric domain between reflective walls. The domain is $\Omega = [0,1] \times [0,1.4]$, rotated around the x axis with reflective boundary conditions everywhere except $y = 1.4$ where it is transmissive. Initially the fluid is at rest with $p = 1$ everywhere and $\rho = 5$ inside a circle (sphere) centre (0.4,0), radius 0.2, $\rho = 1$ outside. As the spherical shock starts to expand, the rarefaction wave travels inwards, resulting in a region of low density. Fluid then rushes in from all directions, and meets at the centre forming a second, smaller outward shock to follow the first. By the time they have both reflected off the boundaries a number of interactions have occurred forming a complex pattern of waves.

For the monitor, $m = 1 + \sqrt{E}/\alpha$, $R_a/R_0 = 0.5$, $R_b = 3.0$ ($R_0 = 0.015$), and $\lambda_{mon} = 0.01$. The mesh is $N_x = 120$, $N_y = 30$ (7351 cells) and the monitor variable is $U = \ln(\rho p)$. This is so the monitor identifies discontinuities in both the density and pressure, and is equally sensitive to the discontinuities in the low density region (which have much smaller magnitude) as those elsewhere. This is not recommended in general because for example 0.0001, 0.0002 and 1,2 have the same ratio but usually the difference between the former is due to noise and so of no interest. Fig 6.26 shows the mesh evolution (in total requiring 20 remaps in 1224 timesteps and taking 3 hours). It is noticeable that all of the mesh behind the initial shock is refined due to the gradients of ρ or p being non-zero (Fig 6.27), but in most of that area ρ and p vary quite smoothly and there is not much happening, suggesting this is a waste of mesh. Fig 6.28 plots the least squares errors at the end time. As with the Sod shock tube (Fig 6.14), the errors are distributed differently between the features. On a logscale (Fig 6.29) E_2 and E_3 are much noisier, and are proportionally much smaller between features than E_1, so if used for monitors should let the mesh derefine more in such places . Fig 6.30 compares the end time meshes for $E = E_1$, $E_1 + E_2 + E_3$ and $E_2 + E_3$ - with no component of E_1 the latter does indeed focus almost completely on the discontinuities.

Figs 6.31, 6.32 compare end time density and pressure contour plots. Despite the differences in mesh between the different monitors, these are very similar, and comparable with Azarenok [6, Figs 5.2-5.10] (who ran separate calculations for $U = \rho$ and $U = p$ and used a larger mesh, 100×140). For this problem, changing the monitor from $m = 1+\sqrt{E}/\alpha$ to $m = \sqrt{1 + E/\alpha}$ makes virtually no difference to the results (compare Figs 6.30, 6.33).

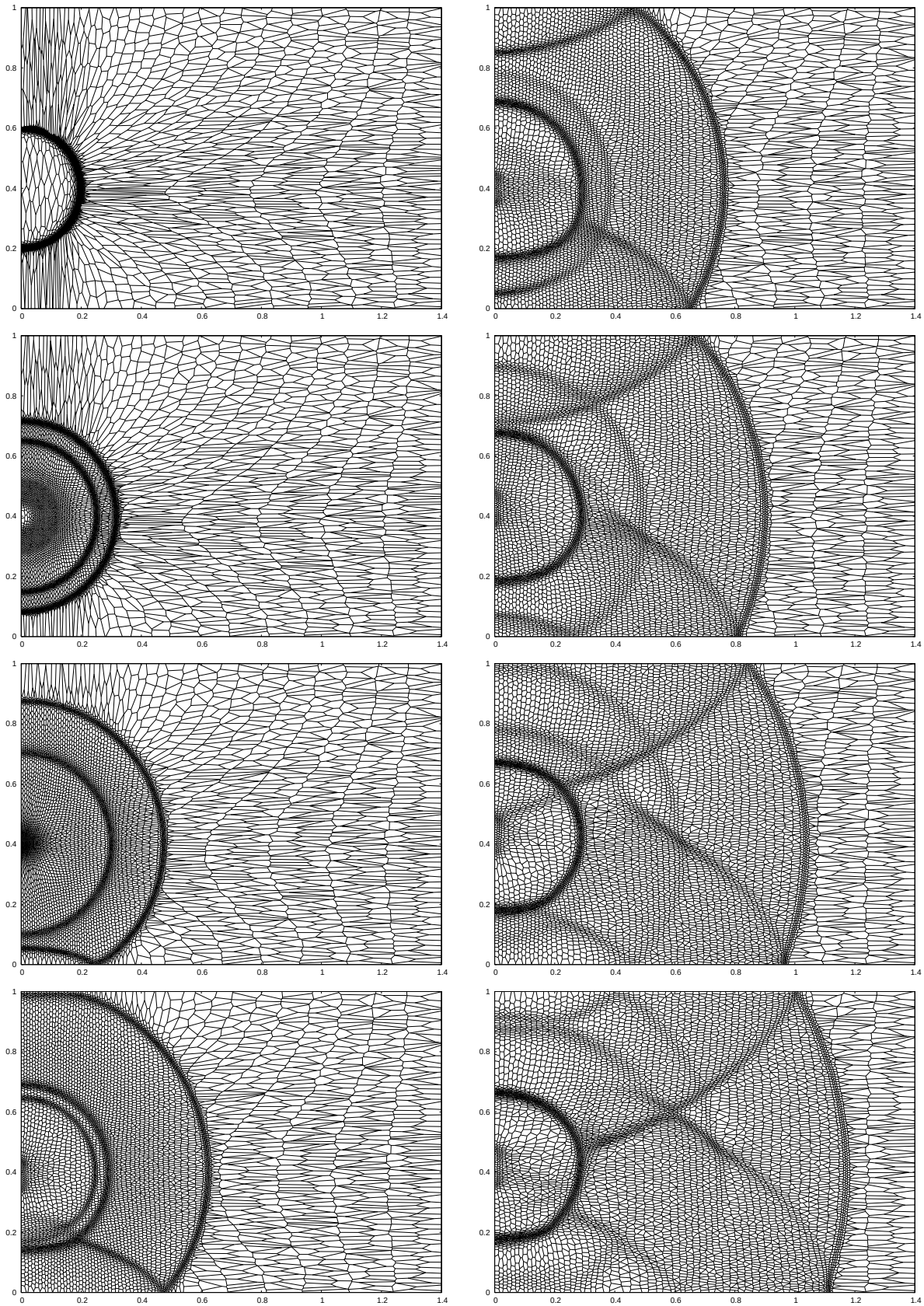


Figure 6.26: Mesh at $t = 0, 0.1, 0.2, 0.3$ (left column, top to bottom) and $t = 0.4, 0.5, 0.6, 0.7$ (right column, top to bottom)

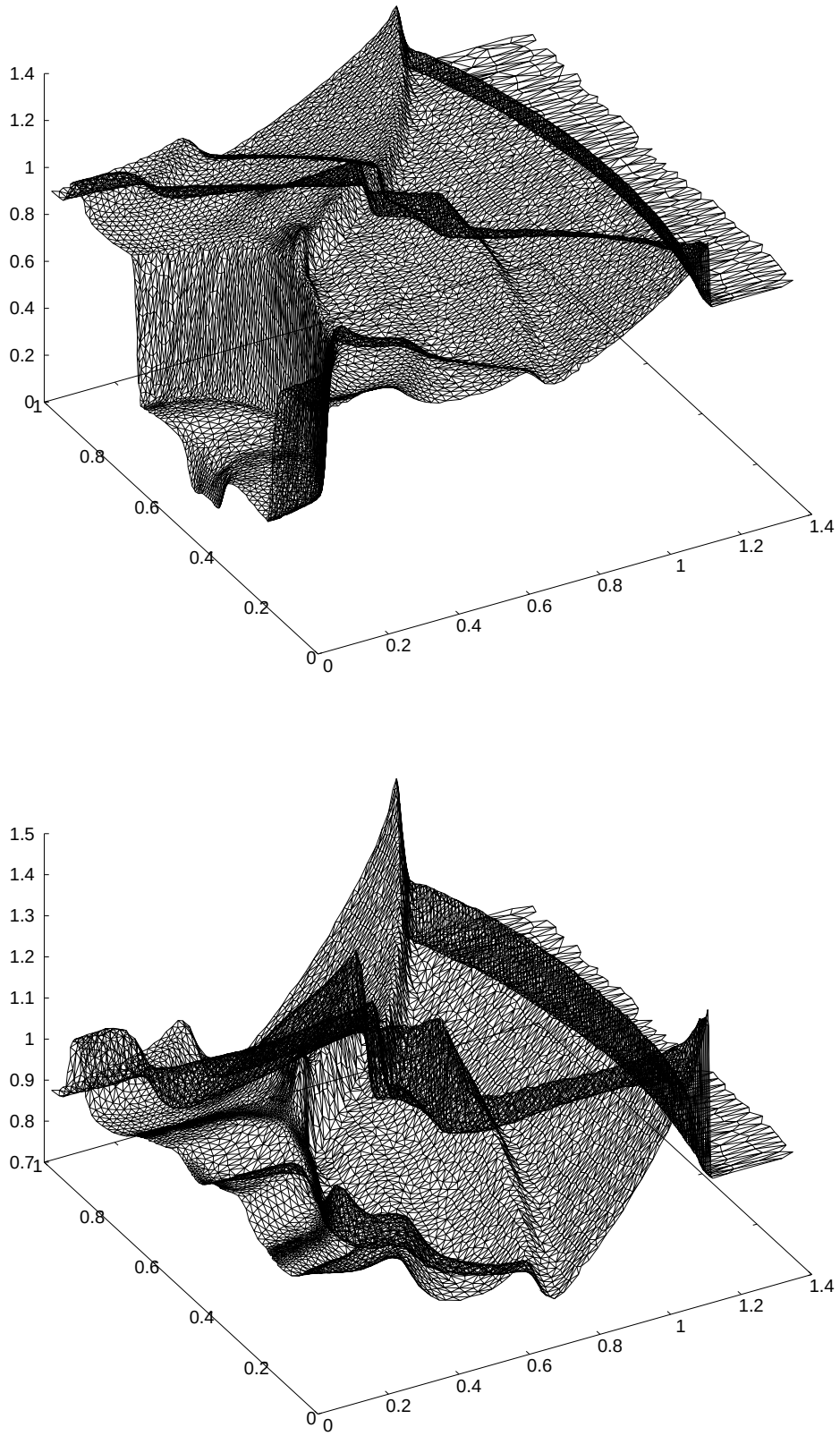


Figure 6.27: Density and pressure at $t = 0.7$

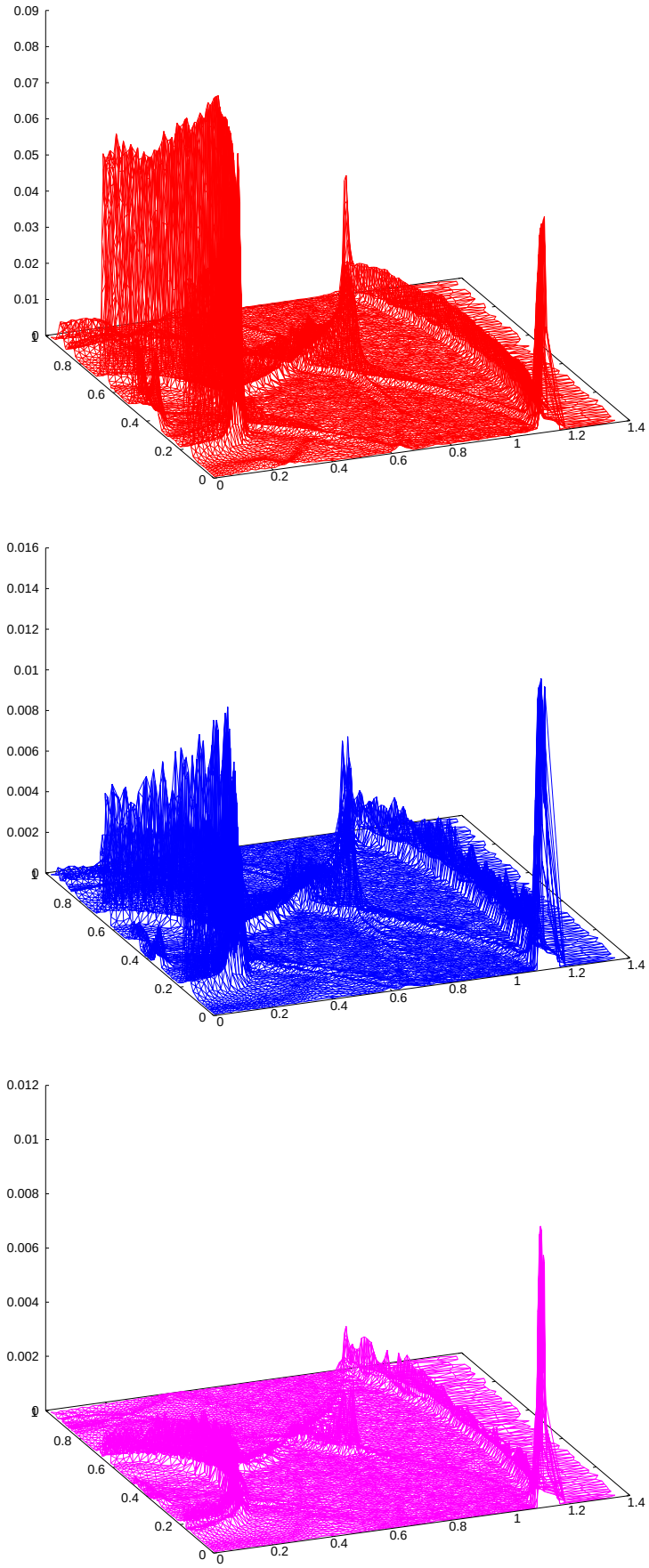


Figure 6.28: E_1, E_2 and E_3 (top to bottom) at $t = 0.7$

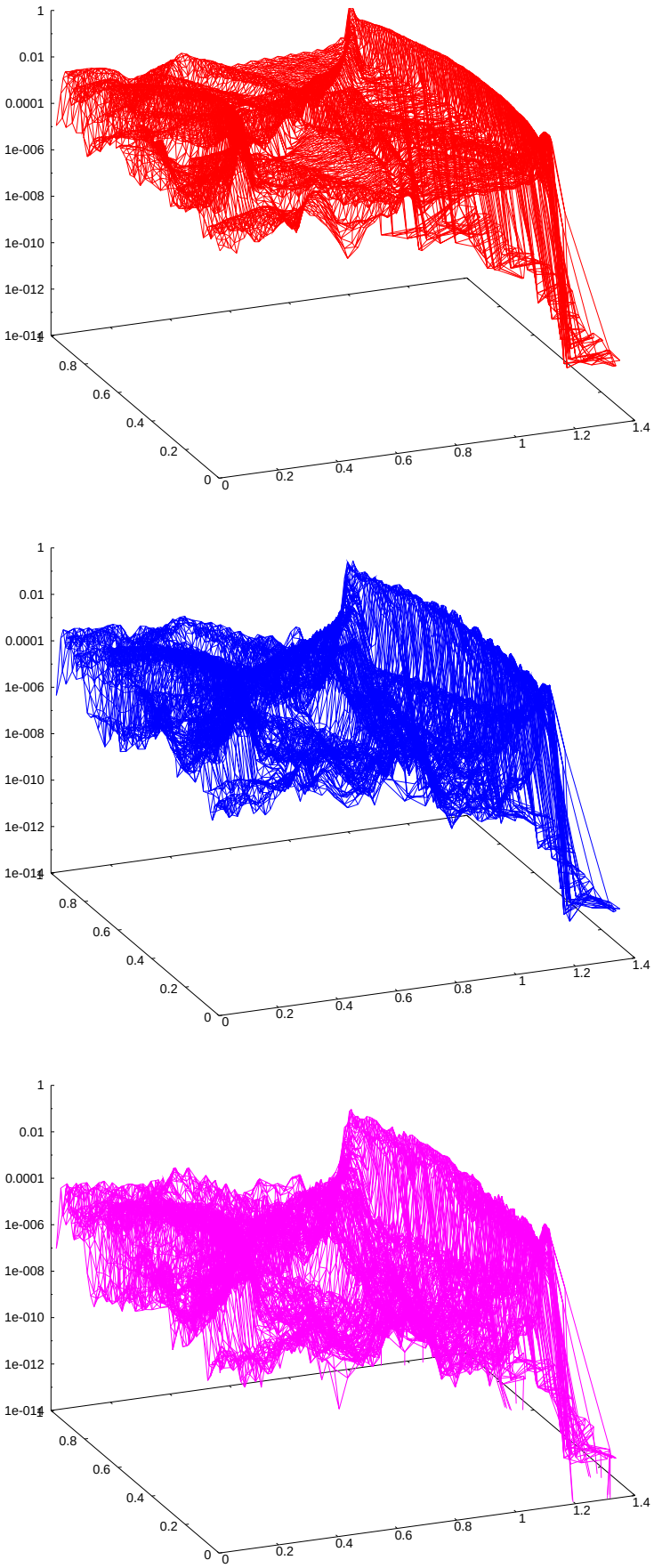


Figure 6.29: E_1, E_2 and E_3 in log-scale (top to bottom) at $t = 0.7$

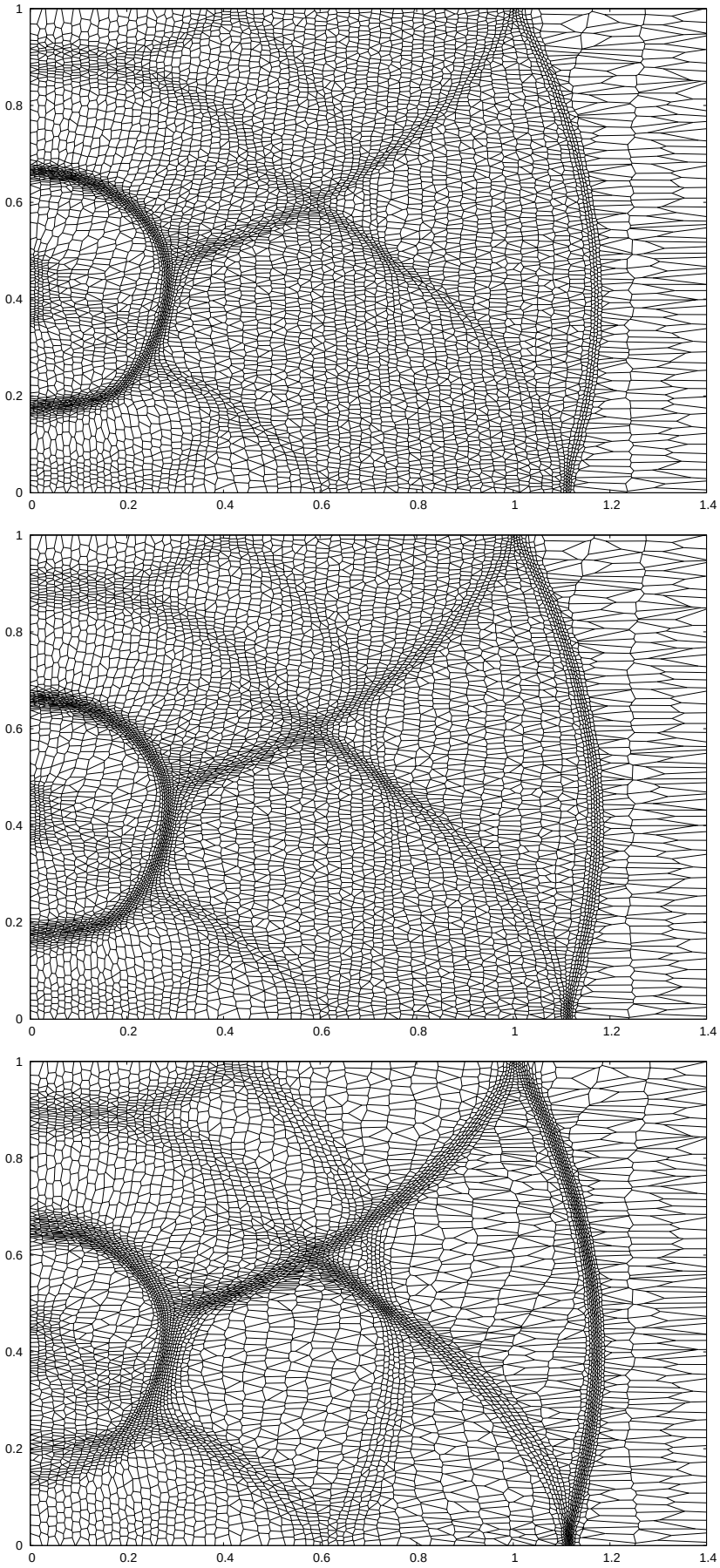


Figure 6.30: End time meshes for $E = E_1$ (top), $E = E_1 + E_2 + E_3$ (middle) and $E = E_2 + E_3$ (bottom)

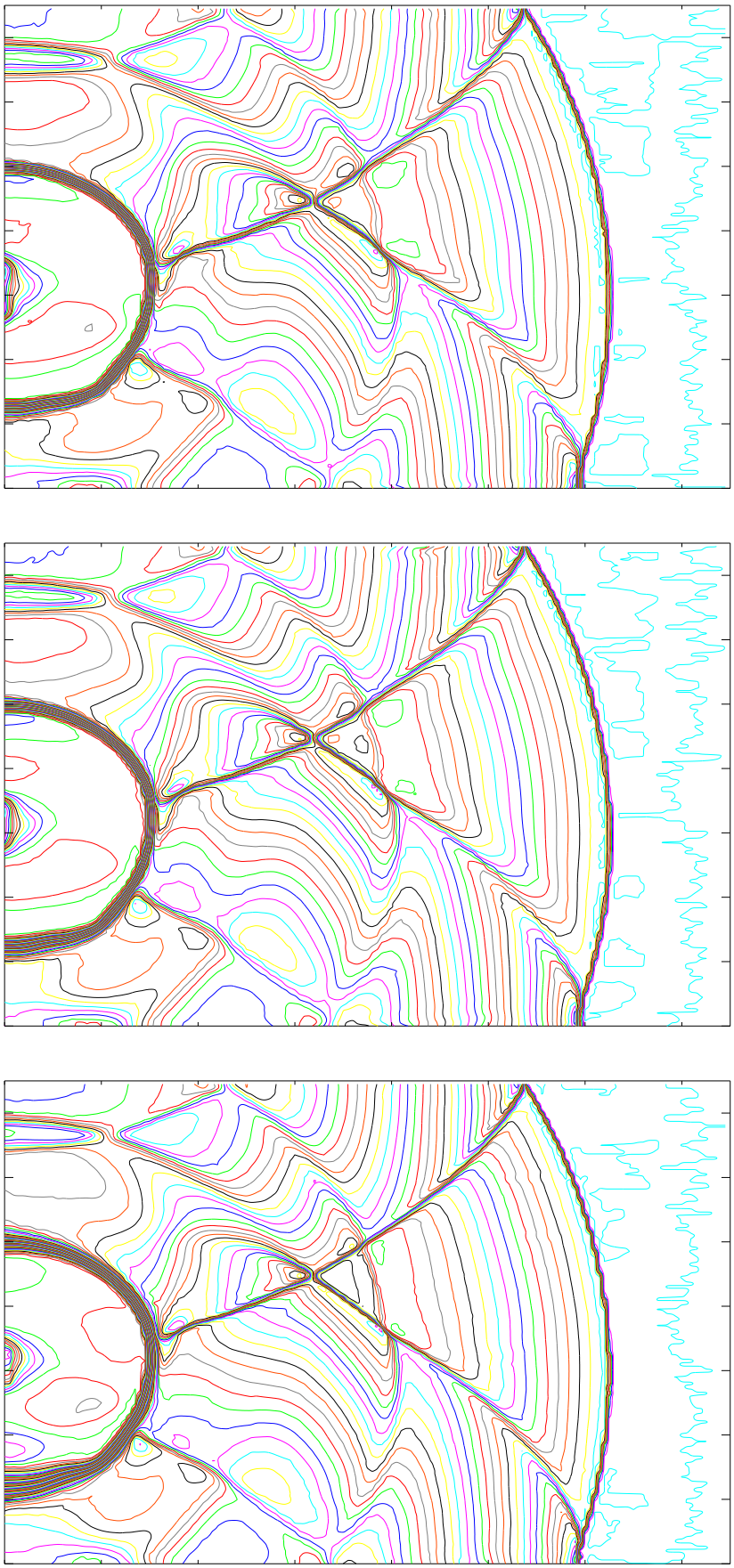


Figure 6.31: Density contour plots for $E = E_1$ (top), $E = E_1 + E_2 + E_3$ (middle) and $E = E_2 + E_3$ (bottom) for $0.2 \leqslant \rho \leqslant 1.3$, increment 0.0125

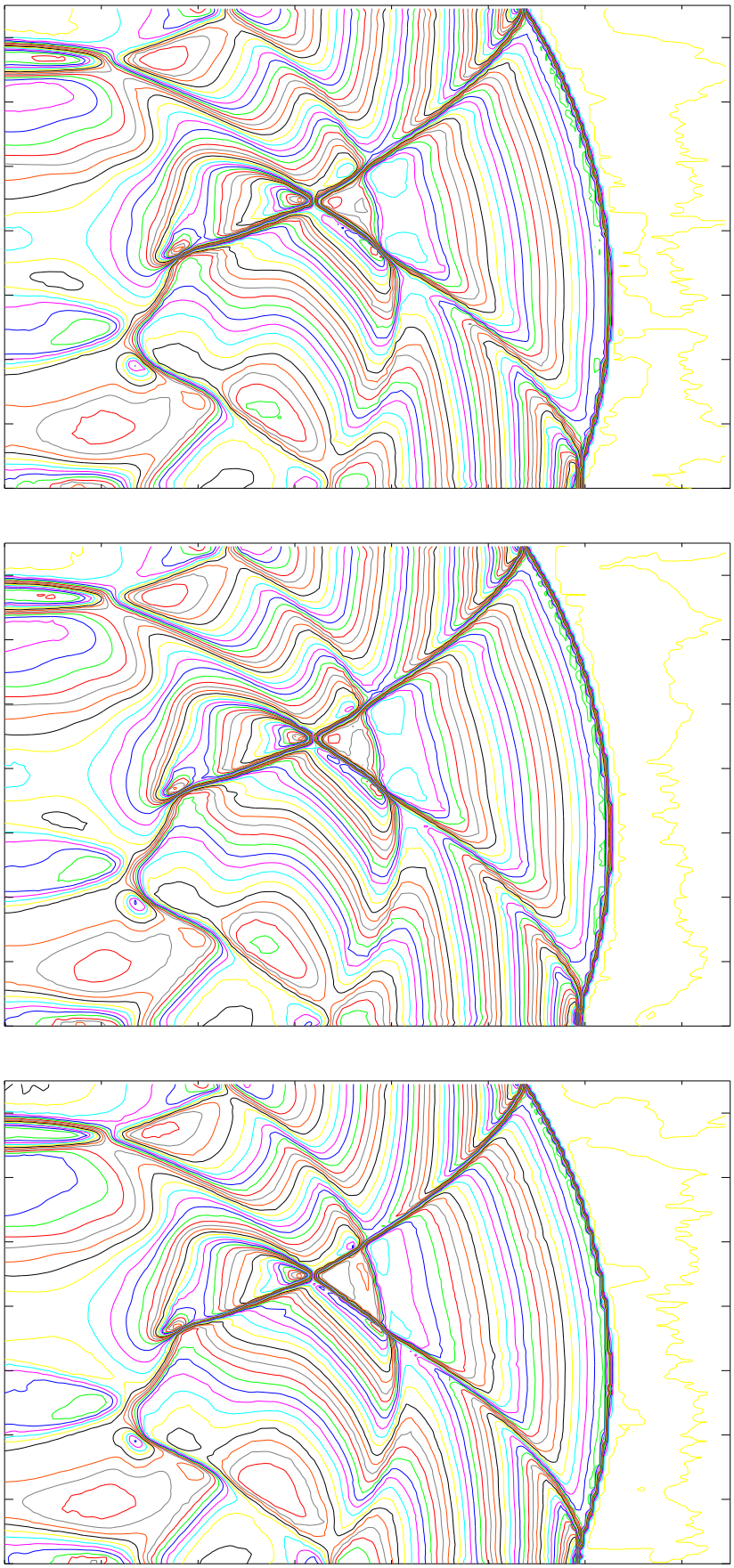


Figure 6.32: Pressure contour plots for $E = E_1$ (top), $E = E_1 + E_2 + E_3$ (middle) and $E = E_2 + E_3$ (bottom) for $0.775 \leqslant p \leqslant 1.5$, increment 0.0125

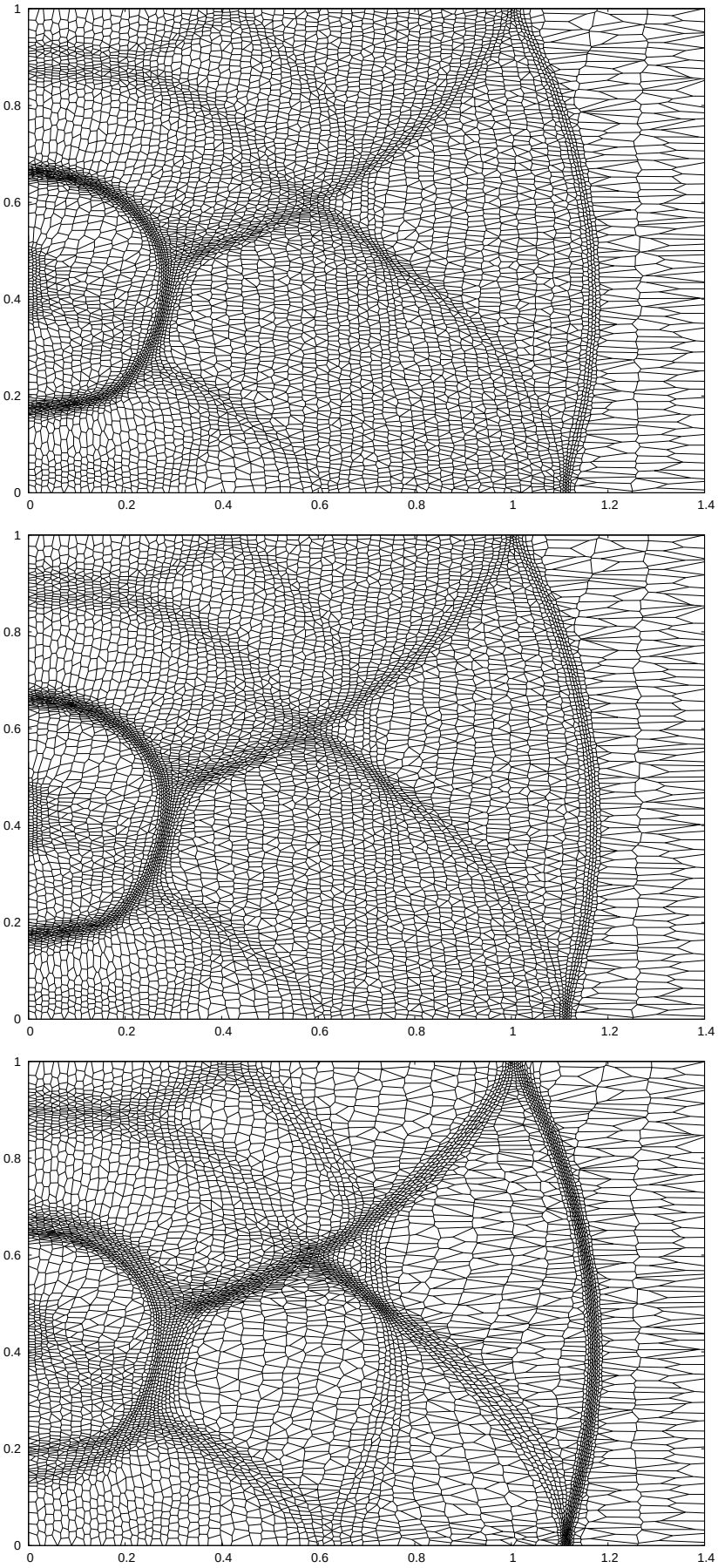


Figure 6.33: End time meshes for $m = \sqrt{1 + E/\alpha}$ and $E = E_1$ (top), $E = E_1 + E_2 + E_3$ (middle) and $E = E_2 + E_3$ (bottom)

6.6 Elliptical shock problem

The last problem is a variation on the circular Sod problem in which the discontinuity is elliptical. Our interest here is to run the problem onto very late times, so that the shock bounces around the domain several times, and see how the mesh reacts. The physical domain $\Omega = [-1, 1] \times [0, 1]$, $(\rho, p) = (1, 1)$ inside, $(0.125, 0.1)$ outside the ellipse $\left(\frac{x}{0.4}\right)^2 + \left(\frac{y-0.5}{0.3}\right)^2 = 1$, $N_x = 100$, $N_y = 40$ (8141 cells), $\beta = 0.8$, $m = 1 + \sqrt{E}/\alpha$, $E = E_1$, $R_a/R_0 = 0.25$, $R_b = 2.5$ ($R_0 = 0.017$), and $\lambda_{mon} = 0.05$. Figs 6.34, 6.35 and 6.36 show the mesh evolution, and Figs 6.37, 6.38 and 6.39 show the density. It took 2815 timesteps (32 remaps) and 6 hours to reach the end time $t = 2.0$. The mesh has managed to stay with the shock throughout, but more notable is the fact that the contact surface is still visible in the mesh at the end time. As with the previous problem, changing the monitor from $m = 1 + \sqrt{E}/\alpha$ to $m = \sqrt{1 + E/\alpha}$ makes virtually no difference to the results (Fig 6.40).

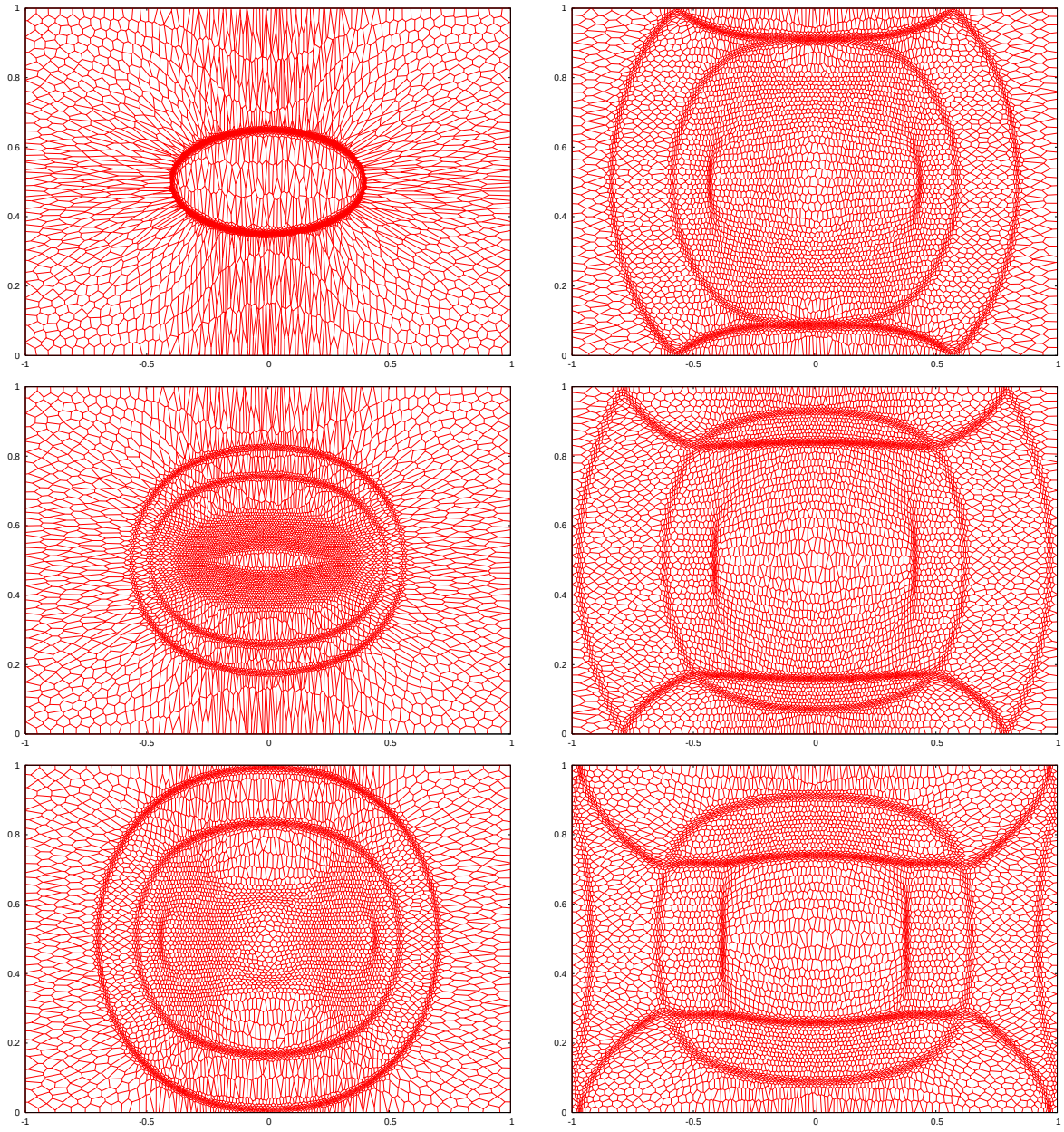


Figure 6.34: Mesh at $t = 0, 0.1, 0.2$ (left column, top to bottom) and $t = 0.3, 0.4, 0.5$ (right column, top to bottom)

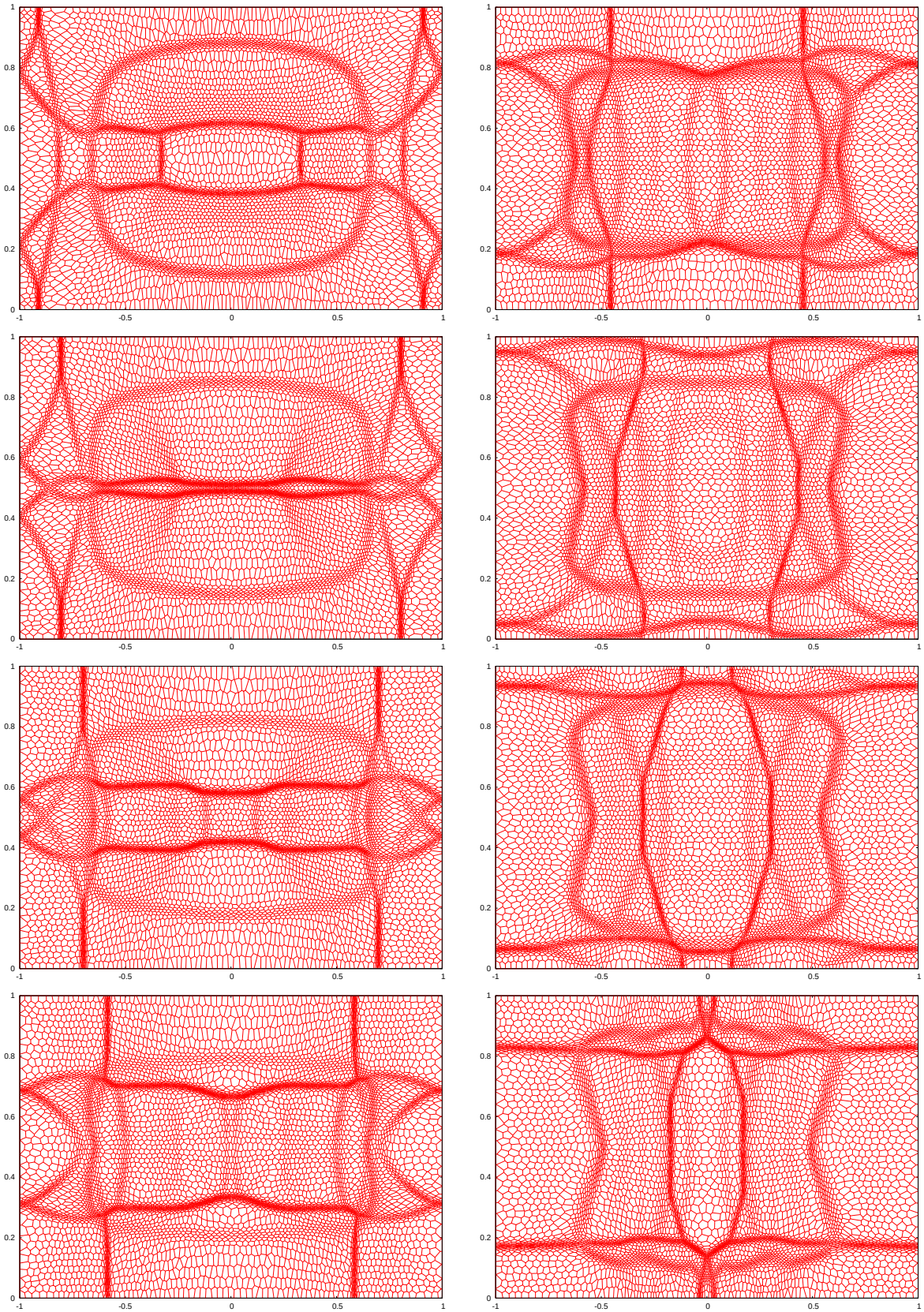


Figure 6.35: Mesh at $t = 0.6, 0.7, 0.8, 0.9$ (left column, top to bottom) and $t = 1, 1.1, 1.2, 1.3$ (right column, top to bottom)

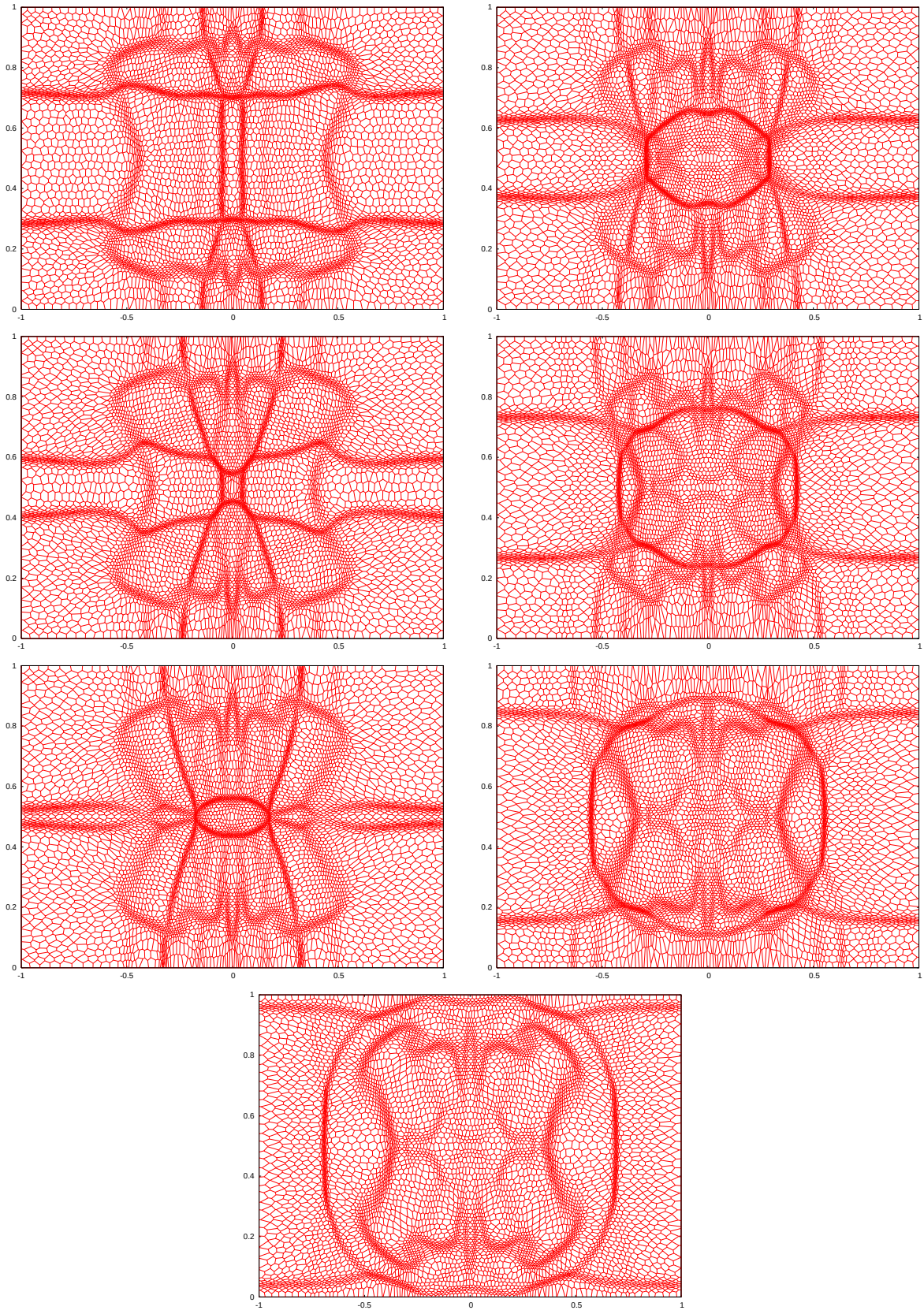


Figure 6.36: Mesh at $t = 1.4, 1.5, 1.6$ (left column, top to bottom), $t = 1.7, 1.8, 1.9$ (right column, top to bottom) and $t = 2$ (bottom)

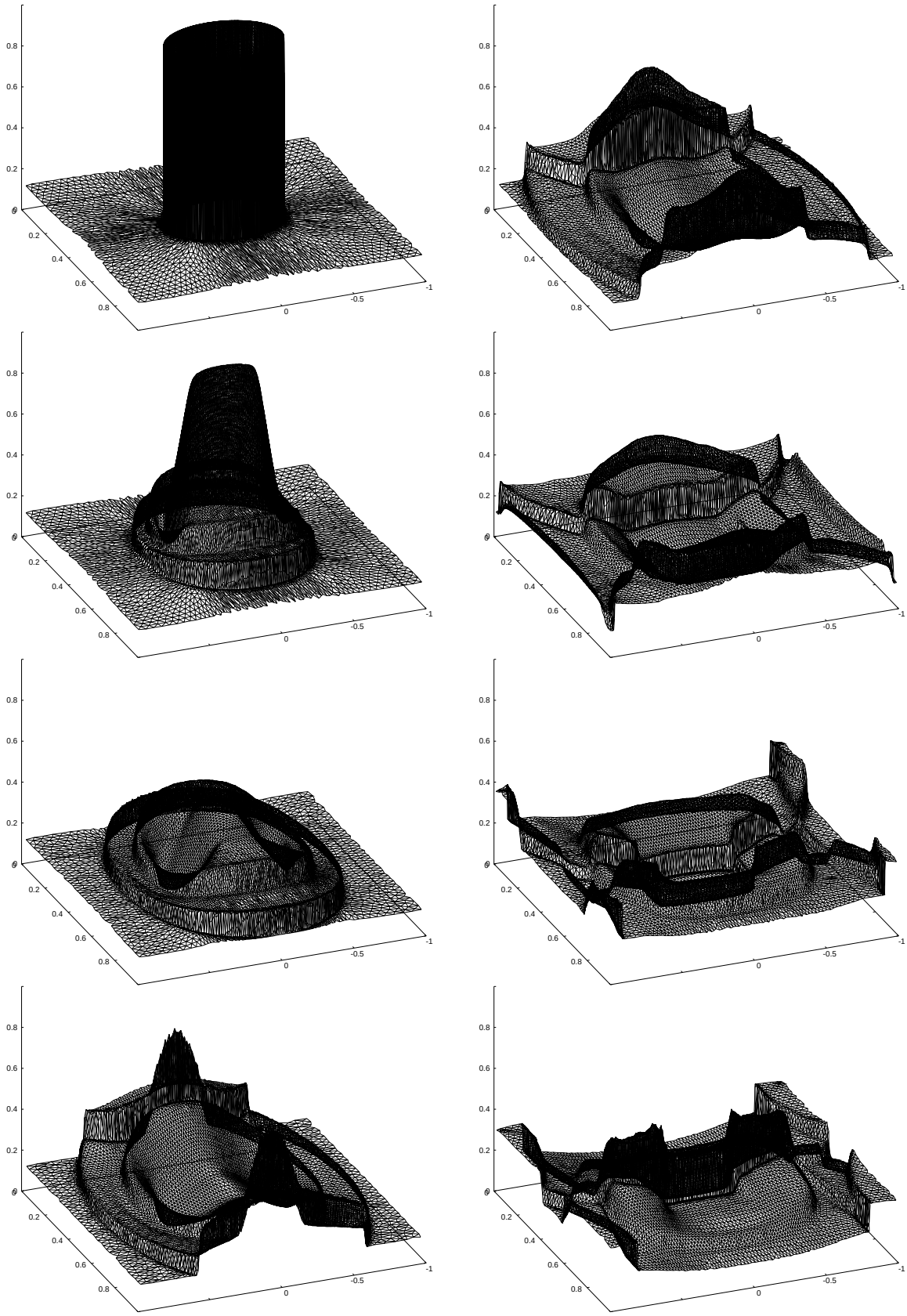


Figure 6.37: Density at $t = 0, 0.1, 0.2, 0.3$ (left column, top to bottom) and $t = 0.4, 0.5, 0.6, 0.7$ (right column, top to bottom)

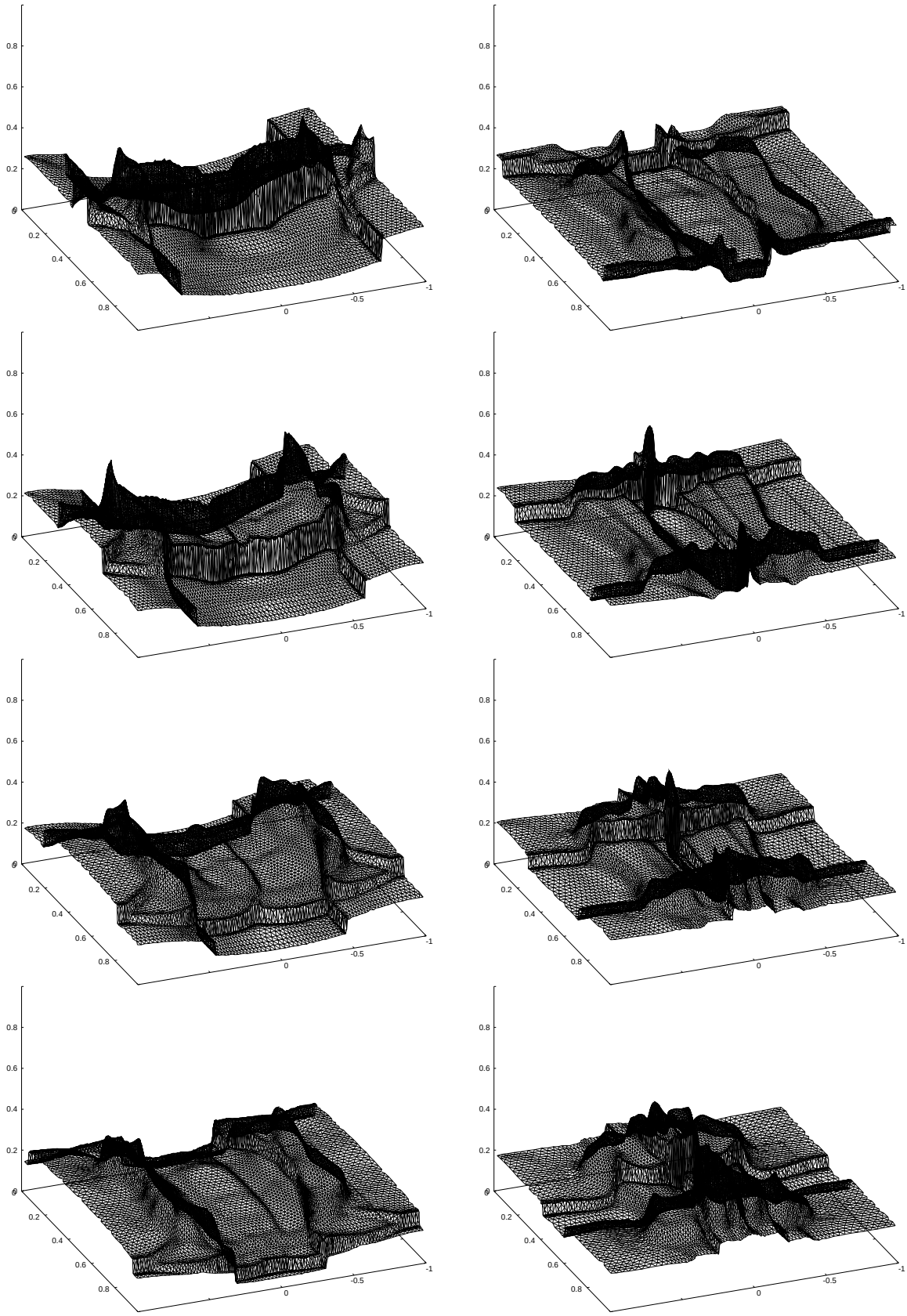


Figure 6.38: Density at $t = 0.8, 0.9, 1, 1.1$ (left column, top to bottom) and $t = 1.2, 1.3, 1.4, 1.5$ (right column, top to bottom)

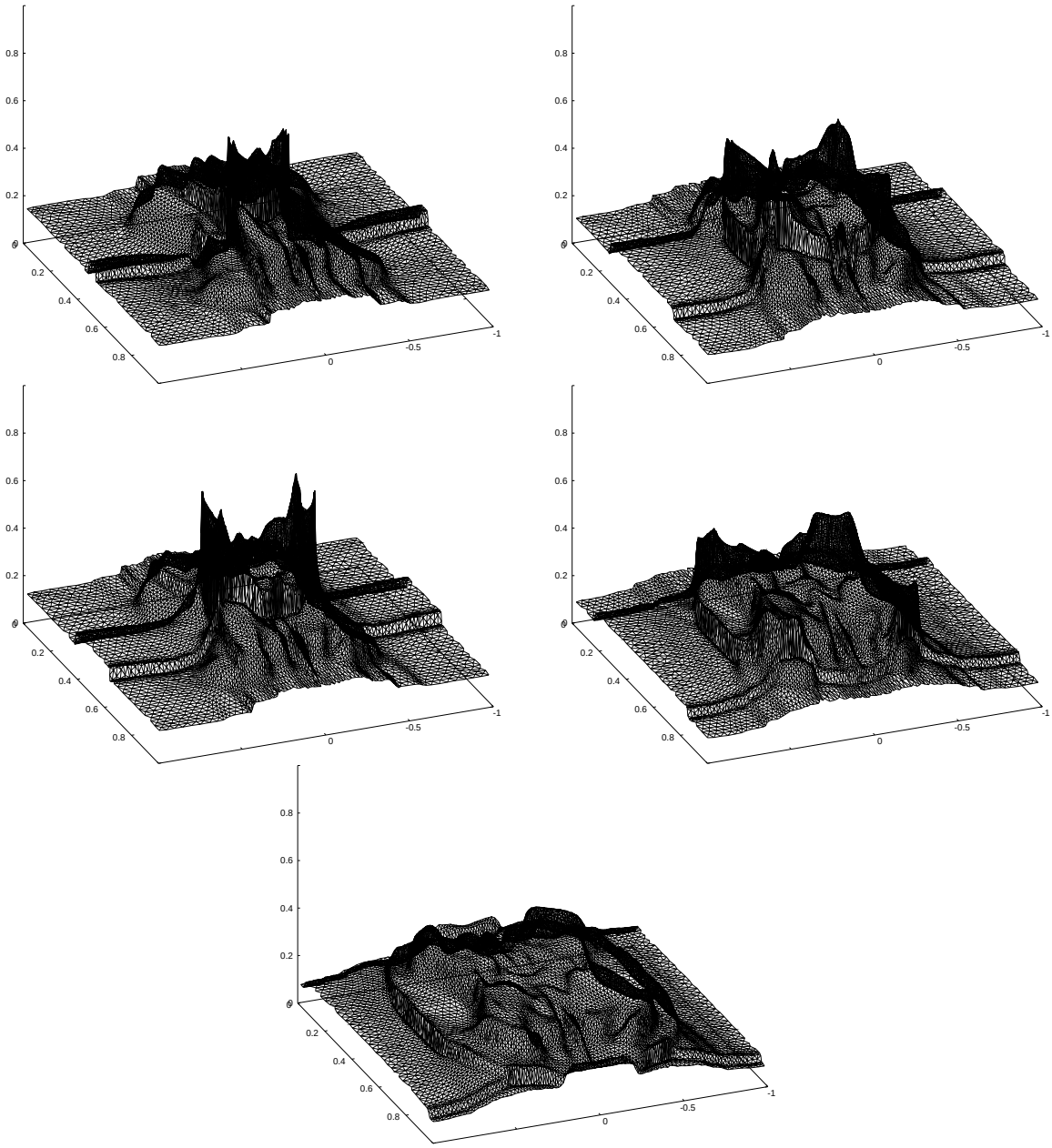


Figure 6.39: Density at $t = 1.6, 1.7$ (left column, top to bottom), $t = 1.8, 1.9$ (right column, top to bottom) and $t = 2$ (bottom)

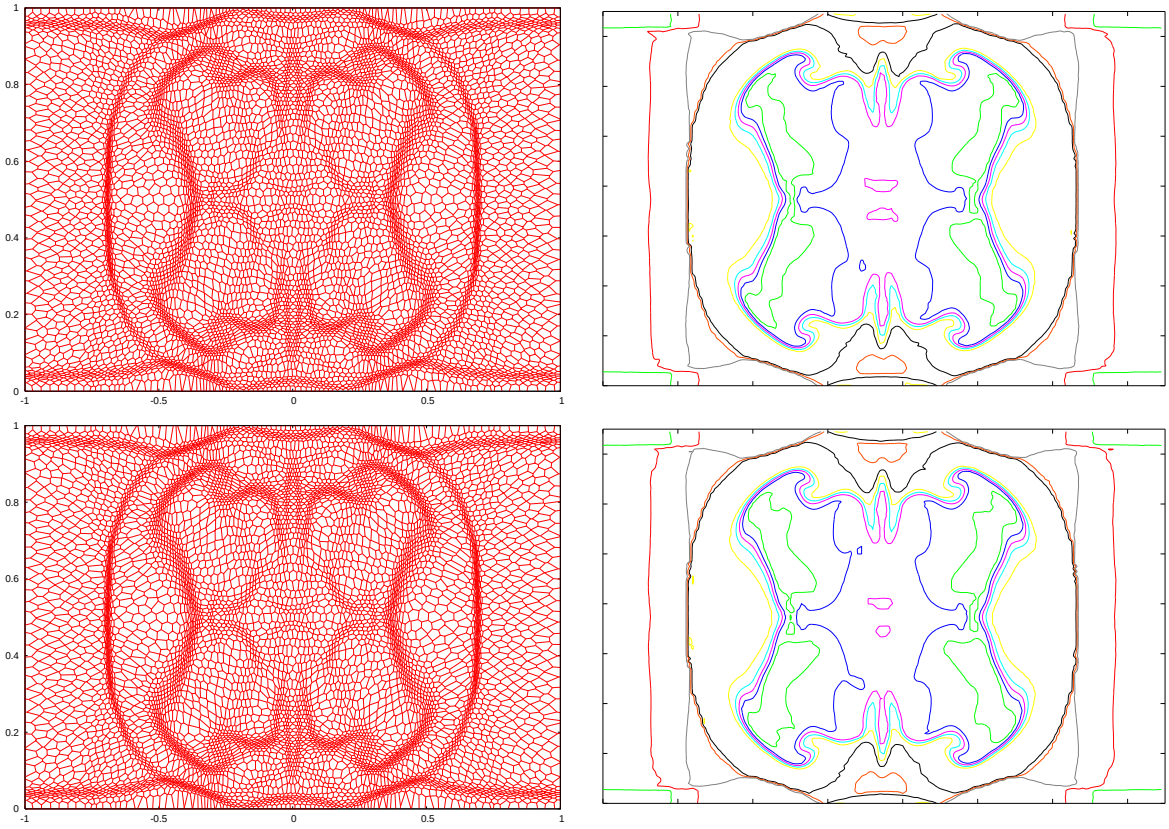


Figure 6.40: End time mesh (left) and density contour plot (right, contours from 0.075 to 0.325 in 0.025 intervals) for $m = \sqrt{1 + E/\alpha}$ (top) and $m = 1 + \sqrt{E}/\alpha$ (bottom)

Chapter 7

Conclusions and further work

The objective of this thesis was to develop a moving mesh algorithm for the Euler equations based on the Geometric Method for the solution of the optimal transport problem.

The Geometric Method was originally developed in the context of weather simulation to solve numerically the semi-geostrophic equations, which have since been recognised as an optimal transport problem. The method of discretisation results in a mesh of polyhedral cells with specified (positive) areas so could be used for mesh generation. The mesh depends continuously on the areas, so that as the specifications change, the mesh adapts smoothly and in such a way as to minimise the total movement suggesting its use as a moving mesh algorithm.

We set the scene in chapter 1 by surveying current moving mesh methods paying particular attention to variational formulations similar to optimal transportation such as the harmonic method and equidistribution. In chapter 2 we then introduced optimal transportation theory and the semi-geostrophic equations, and the discrete version which the Geometric Method solves. Other solution methods were then briefly mentioned. Chapter 3 went into the algorithmic details of the original solution method of Chynoweth and the later improvements of Purser. After repeating Purser's discussion of various extensions, we then applied the algorithm to some simple test cases to assess its performance. When the specified areas were fixed it displayed quadratic convergence, but when the specified areas become position dependent limit cycles were found to exist.

In chapter 4 we introduced the discretization method for the Euler equations, which employed the finite volume approach. Space-time cells were constructed by joining cells at one time level to the next, and any tetrahedral gaps created by connectivity changes were split up and assigned to adjacent cells. Then the predictor-corrector scheme and the mesh adaption loop were described and the least squares gradient estimator for the unstructured grid. A novel monotonic limiter on the unstructured grid was described, based on quadratic programming, then the timestep criteria and finally the remapping algorithm required when the mesh changes too much within a timestep.

In chapter 5 we selected a monitor function based on derivative estimates and applied it to a series of mesh adaption problems involving discontinuities. Each time it failed the algorithm was modified, first by changing the support to a fixed radius and introducing weighting, then making the support adaptive, and finally by switching the monitor to the least squares error itself. Further analysis showed that for smooth data the least squares errors can be linked to geometric properties such as the gradient. For data with a line discontinuity, error profiles were generated for various weights with a fixed support, and for an adaptive support the associated continuous problem was solved completely yielding conditions on some of the parameters and shedding some light on previous difficulties.

Chapter 6 then applied the complete algorithm to various test problems. The first, the Sod shock tube, was used as a test bed to investigate the effects of the various functions and parameters introduced in chapter 5. Two test problems, a 2D Riemann problem and a spherical explosion, demonstrate comparability with published solutions. The last, an elliptical shock, is run to late times, so that the regions of concentrated mesh change topology significantly during the course of the simulation, merging and separating.

Thus we conclude that the method is viable.

7.1 Further work

Two areas require immediate attention to make the algorithm fully functional. The first is the under/overshoots visible in the Sod shock tube. The claim that this is due to the predictor not using any neighbour information needs verification and if so, correction. The second is to ascertain whether the boundary conditions need to be explicitly enforced in the limiter for the solution to the Rayleigh-Taylor instability. Another area of concern is whether there is a satisfactory limiter that is both monotone and second order, perhaps based on limiting at the vertices of the space-time cell.

More generally, the complexity of the monitor functions required to satisfy the sensitivity of the mesh iteration suggest that this naive iteration requires some form of modification e.g. by damping the mesh movement or incorporating the monitor directly into the cost function for the optimal transport problem.

Beyond this there are two major avenues of exploration. The first is extension to three dimensions. A good starting point would be to investigate current convex hull algorithms to see if any can more naturally incorporate the changing nature of the solution as the specified volumes change. The second is extension to multiple fluids. Some progress has

been made already in this area using power diagrams, but no attempt has been made yet to vary the position of the sites to improve the solution e.g. to match up interfaces between adjacent cells.

Bibliography

[1] M.Abramowitz and I.A.Stegun, *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*, Dover, New York, 1964.

[2] H.T.Ahn and M.Shashkov, *Multi-material interface reconstruction on generalized polyhedral interfaces*, J. Comp. Phys., **226**(2), p.2096-2132, 2007.

[3] O. Aichholzer, *The path of a triangulation*, in Proc. 13th European Workshop on Computational Geometry CG '97, pp.1-3, 1997.

[4] O. Aichholzer, L.S. Alboul and F. Hurtado, *On flips in polyhedral surfaces*, Int. Jour. of Foundations of Computer Science, **13**(2), pp.303-311, 2002.

[5] F.Aurenhammer and R.Klein, *Voronoi diagrams*, in Handbook of Computational Geometry, ed. J.-R.Sack and J.Urrutia, Elsevier, Amsterdam, pp.201-290, 2000.

[6] B.N.Azanerok, S.A.Ivanenko, and T.Tang, *Adaptive mesh redistribution method based on Godunov's scheme*, Commun. in Math. Sci. **1**(1), pp.153-180, 2003.

[7] M.J.Baines, *Moving finite elements*, Oxford University Press, Oxford, 1994.

[8] G.J.Ball, *A Free-Lagrange method for unsteady compressible flow: simulation of a confined cylindrical blast wave*, Shock Waves **5**, pp.311-325, 1996.

[9] T.Barth and D.C.Jespersen, *The design and application of upwind schemes on unstructured meshes*, AIAA paper 89-0366, in: 27th AIAA Aerospace Sciences Meeting, Reno, NV, USA, 1989.

[10] D.J. Benson, *Computational methods in Lagrangian and Eulerian hydrocodes*, J. Comput. Methods Appl. Mech. Engrg., **99**, pp.235-294, 1992.

[11] M.de Berg, M.van Kreveld, M.Overmars, and O.Schwarzkopf, *Computational geometry: algorithms and applications*, Springer-Verlag, Berlin, 1997.

[12] M.Berger, M.J.Aftosmis and S.M.Murman, *Analysis of slope limiters on irregular grids*, AIAA paper 2005-0490, in: 43rd AIAA Aerospace Sciences Meeting, Reno, NV, USA, 2005.

[13] M.Berger and J.Oliger, *Adaptive mesh refinement for hyperbolic partial differential equations*, J. Comp. Phys., **53**, pp.484-512, 1984.

[14] J.U.Brackbill, *An adaptive grid with direction control*, J. Comp. Phys., **108**, pp.38-50, 1993.

[15] Y.Brenier, *Polar factorization and monotone rearrangement of vector valued functions*, Commun. Pure Appl. Math. **44**, pp.375-417, 1991.

[16] C.J.Budd and J.F.Williams, *Parabolic Monge-Ampere methods for blow-up problems in several spatial dimensions*, Phys. A **39**(19), pp. 5425-5444, 2006.

[17] W.Cao, W.Huang and R.D.Russell, *Comparison of two-dimensional r-adaptive finite element methods using various error indicators*, Mathematics and Computers in Simulation, **56**(2), pp.127-143, May 2001.

[18] W.Cao, W.Huang and R.D.Russell, *Approaches for generating moving adaptive meshes: location versus velocity*, Appl. Num. Math., **47**(2), pp.121-138, Nov 2003.

[19] G.F.Carey and H.T.Dinh, *Grading functions and mesh redistribution*, SIAM J. Numer. Anal., **22**(5), pp.1028-1040, 1985.

[20] D.R.Chand and S.S.Kapur, *An algorithm for convex polytopes*, J. ACM **17**(7), pp.78-86, 1970.

[21] R.Chartrand, K.Vixie, B.Wohlberg and E.Bollt, *A gradient descent solution to the Monge-Kantorovich problem* Los Alamos National Laboratory preprint LA-UR-04-6305, http://math.lanl.gov/Research/Publications/Docs/chartrand-2007-gradient.pdf, 2005.

[22] S.Chynoweth, *The semi-geostrophic equation and the Legendre transform*, Ph. D thesis, University of Reading, Whitenights, PO Box 220, Reading, RG6 2AX, U.K., 248pp, 1987.

[23] M.J.P.Cullen, *A mathematical theory of large-scale atmosphere/ocean flow*, Imperial College Press, 2006.

[24] M.J.P.Cullen and R.J.Purser, *An extended Lagrangian theory of semi-geostrophic frontogenesis*, J. Atmos. Sci, **41**, 1984.

[25] O.Devillers, S.Pion and M.Teillaud, *Walking in a triangulation*, Int. Jour. Foundations Comp. Sci., **13**(2), pp.181-199, 2002.

[26] J.K.Dukowicz, *Efficient volume computation for three-dimensional hexahedral cells*, J. Comp. Phys., **74**, pp.493-496, 1988.

[27] J.K.Dukowicz and J.W.Kodis, *Accurate conservative remapping (rezoning) for arbitrary Lagrangian-Eulerian computations*, SIAM J. Sci. Comput., **8**, p.305, 1987.

[28] A.S.Dvinsky, *Adaptive grid generation from harmonic maps on Riemannian manifolds*, J. Comp. Phys., **95**, pp.450-476, 1991.

[29] J.Eells and J.H.Sampson, *Harmonic mappings of Riemannian manifolds*, Amer. J. Math, **86**(1), pp.109-160, 1964.

[30] L.C.Evans, *Partial differential equations and Monge-Kantorovich mass transfer*, in Current developments in mathematics, Boston, Int. Press, pp.65-126, 1999.

[31] W.Gangbo and R.J.McCann, *The geometry of optimal transportation*, Acta Math., **177**, pp.113-161, 1996.

[32] R.Garimella, M.Kucharik, M.Shashkov, *An efficient linearity and bound preserving conservative interpolation (remapping) on polyhedral meshes*, Computers & Fluids, **36**, pp.224-237, 2007.

[33] P.E.Gill, W.Murray and M.H.Wright, *Practical optimization*, Academic Press, p.151 Eqn(4.97), 1981.

[34] S.K.Godunov, A.V.Zabrodin, M.Ya.Ivanov, A.N.Kraiko and G.P. Prokopov, *Numerical solution of multi-dimensional problems in gas dynamics*, Nauka, Moscow, 1976.

[35] J.B.Goodman and R.J.LeVeque, *On the accuracy of stable schemes for 2D scalar conservation laws*, Mathematics of Computation, **45**(171), pp.15-21, 1985.

[36] J.Grandy, *Conservative remapping and regions overlays by intersecting arbitrary polyhedra*, J. Comp. Phys., **148**, pp.433-466, 1999.

[37] S.Haker, A.Tannenbaum, L.Zhu and S.Angenent, *Optimal mass transport for registration and warping*, Int. Jour. of Computer Vision, **60**(3), pp.225-240, 2004.

[38] R.Hamilton, *Harmonic maps of manifolds with boundary*, Lecture Notes in Mathematics Vol.41, Springer-Verlag, New York, 1975.

[39] S.Hanke, T.Ottmann and S.Schuierer, *The edge-flipping distance of triangulations*, J. Universal Computer Science, **2**(8), pp.570-579, 1996.

[40] C.W.Hirt and B.D.Nichols, *Volume of fluid (VOF) method for the dynamics of free boundaries* J. Comp. Phys., **39**, pp.201-225, 1981.

[41] B.J.Hoskins, *The geostrophic momentum approximation and the semi-geostrophic equation* J. Atmos. Sci., **34**, pp.1859-1867, 1975.

[42] W.Huang, *Practical apsects of formulation and solution of moving mesh partial differential equations*, J. Comp. Phys., **171**, pp.753-775, 2001.

[43] W.Huang, *Variational mesh adaption: isotropy and equidistribution*, J. Comp. Phys., **174**, pp.903-924, 2001.

[44] W.Huang, *Mathematical principles of anisotropic mesh adaption*, Commun. Comput. Phys., **1**(2), pp.276-310, April 2006.

[45] W.Huang, Y.Ren, R.D.Russell, *Moving mesh partial differential equations (MMPDEs) based on the equidistribution principle* SIAM J. Numer. Anal., **31**(3), pp.709-730, June 1994.

[46] W.Huang and W.Sun *Variational mesh adaption II: error estimates and monitor functions*, J. Comp. Phys., **184**, pp.619-648, 2003.

[47] M.Hubbard, *Multidimensional slope limiters for MUSCL-type finite volume schemes on unstructured grids*, J. Comp. Phys., **155**, pp.54-74, 1999.

[48] F.Hurtado, M.Noy and J.Urrutia, *Flipping edges in triangulations*, in Proc. 12th Annual ACM Symposium on Computational Geometry, pp.214-223, 1996.

[49] G.-S. Jiang and C.-W. Shu, *Efficient implementation of weighted ENO schemes*, J. Comp. Phys., **126**, p.202, 1996.

[50] L.V.Kantorovich, *On the translocation of masses*, C. R. (Doklady) Acad. Sci. URSS (N. S.), **37**, pp.199-201, 1942.

[51] L.V.Kantorovich, *On a problem of Monge*, Uspekhi Mat. Nauk, **3**, pp.225-226, 1948.

[52] P.Knupp, *Mesh generation using vector fields*, J. Comp. Phys., **119**, p.142, 1995.

[53] P.Knupp, L.G.Margolin and M.Shashkov, *Reference Jacobian optimization-based rezone strategies for Arbitrary Lagrangian Eulerian methods*, J. Comp. Phys., **176**(1), pp.93-128, Feb 2002.

[54] A.Kurganov and E.Tadmor, *Solution of two-dimensional Riemann problems for gas dynamics without Riemann problem solvers*, Numerical Methods Partial Differential Equations, Vol. 18, 5, pp.584-604, 2002.

[55] P.Lancaster and K.Salkauskas, *Surfaces generated by moving least squares methods*, Mathematics of Computation, **37**(155), pp.141-158, 1981.

[56] J.O.Langseth and R.J.LeVeque, *A wave propagation method for three-dimensional hyperbolic conservation laws*, J. Comp. Phys., **165**(1), pp.126-166, Nov 2000.

[57] G.Lapenta, *Variational grid adaptation based on the minimization of local truncation error: time-independent problems*, J. Comp. Phys., **193**(1), pp.159-179, Jan 2004.

[58] R.J.LeVeque, *Numerical methods for conservation laws*, Birkhauser, 1990.

[59] G.Lieberman, *Second order parabolic equations*, Wold Scientific, New Jersey, 1996.

[60] X.-D.Liu, *A maximum principle satisfying modification of triangle based adaptive stencils for the solution of scalar hyperbolic conservation laws*, SIAM J. Numer. Anal., **30**, p.701, 1993.

[61] X.-D.Liu and P.D.Lax, *Positive schemes for solving multi-dimensional hyperbolic systems of conservation laws II*, J. Comp. Phys., **187**(1), pp.428-440, Feb 2003.

[62] H.Liu and G.Liao, *A note on harmonic maps*, Appl. Math. Lett., **9**(4), pp.95-97, 1996.

[63] J.J.Monaghan, *Smoothed particle hydrodynamics*, Ann. Rev. Astron. Astrophys., **30**, pp.543-574, 1992.

[64] G.Monge, *Mémoire sur la théorie des déblais at des remblais*, Histoire de lAcadémie Royale des Sciences de Paris, pp.666-704, 1781.

[65] J.M.Morrell, *A cell by cell anisotropic adaptive mesh Arbitrary Lagrangian Eulerian method for the numerical solution of the Euler equations*, Ph. D thesis, University of Reading, Whitenights, PO Box 220, Reading, RG6 2AX, U.K., 203pp, 2007.

[66] J.A.Nelder and R.Mead, *A simplex method for function minimisation*, Computer Journal, **7**, pp.308-313, 1965.

[67] V.Oliker, *Mathematical aspects of design of beam shaping surfaces in geometrical optics*, in Trends in Nonlinear Analysis, ed. by M. Kirkilionis, S. Kromker, R. Rannacher, F. Tomi, Springer-Verlag, pp.193-224, 2003.

[68] V.Oliker and T.Glimm, *Optical design of single reflector systems and the Monge-Kantorovich mass transfer problem*, J. Math. Sci., **117**(3), pp.4096-4108, 2003.

[69] V.Oliker and S.Kochengin, *Computational algorithms for constructing reflectors*, Computing and Visualization in Science, **6**, pp.15-21, 2003.

[70] J.O'Rourke, *Computational geometry in C (second edition)*, Cambridge University Press, Cambridge, 1997.

[71] A.V.Pogorelov, *Monge-Ampere equations of elliptic type*, Noordhoff, Groningen, p.114, 1964.

[72] F.P.Preparata and S.J.Hong, *Convex hulls of finite sets of points in two and three dimensions*, Comm. A.C.M. **20**, pp. 87-93, 1977.

[73] W.H.Press, B.P.Flannery, S.A.Teukolsky and W.T.Vetterling, *Numerical recipes: the art of scientific computing (third edition)*, Cambridge University Press, UK, 2007.

[74] R.J.Purser, *"Panel Beater": A proposed fast algorithm for semi-geostrophic finite-element codes*, Met. Office 11 Technical Note 11 (draft - never published), Met Office, FitzRoy Road, Exeter, Devon EX1 3PB, UK, June 1988.

[75] S.Rachev and Rüschendorf, *Mass transportation problems, vol. II: applications*, Springer-Verlag, New York, 1998.

[76] T.U.Rehman and A.Tannenbaum, *Multigrid optimal mass transport for image registration and morphing*, in Computational Imaging V, ed. by Bouman, C.A., Miller, E.L., Pollak, I. Proceedings of the SPIE, Volume 6498, p.649–810, 2007.

[77] R.T.Rockafellar, *Convex analysis*, Princeton University Press, 1970.

[78] F.Santos, *Geometric bistellar flips. The setting, the context and a construction*, arXiv:math.CO/0601746 v1, 30 Jan 2006.

[79] R.Schoen and S.-T.Yau, *On univalent harmonic maps between surfaces*, Invent.Math., **44**, pp.265-278, 1978.

[80] S.P.Schofield, R.V.Garimella, M.M.Francois and R.Loubre, *Material order-independent interface reconstruction using power diagrams*, Int. Jour. for Numer. Meth. in Fluids, **56**(6), pp.643-659, June 2007.

[81] M.Kucharik and M.Shashkov, *Extension of efficient, swept-integration-based conservative remapping method for meshes with changing connectivity*, Int. Jour. for Numer. Meth. in Fluids, **56**(8), pp.1359-1365, Mar 2008.

[82] D.Siersma and M.van Manen, *Power Diagrams and their applications*, arXiv:math.MG/0508037 v2, 2 Aug 2005.

[83] S.P.Spekreijse, *Multigrid solution of monotone second-order discretizations of hyperbolic conservation laws*, Mathematics of Computation, **49**(179), pp.135-155, 1987.

[84] A.Suresh, *The reconstruction problem revisited*, NASA report TM-1999-209082, 1999.

[85] B.Swartz, *Good neighborhoods for multidimensional Van Leer limiting*, J. Comp. Phys., **154**, pp.237-241, 1999.

[86] P.D.Thomas and C.K.Lombard, *Geometric conservation law and its application to flow computations on moving grids*, AIAA J., **17**, pp.1030-1037, 1979.

[87] J.F.Thompson, Z.A.Warsi and C.W.Mastin, *Numerical grid generation*, North-Holland, Amsterdam, 1985.

[88] V.A.Titarev and E.F.Toro, *ADER: arbitrary high order Godunov approach*, J. Sci. Comp., **17**, pp.609-618, Dec 2002.

[89] E.F.Toro, *Riemann solvers and numerical methods for fluid dynamics*, Springer Verlag, Berlin, 1999.

[90] B.Van Leer, *Towards the ultimate conservative difference scheme IV: a new approach to numerical convection*, J. Comp. Phys., **23**, pp.276-299, 1977.

[91] C.Villani, *Topics in optimal transportation*, Vol. 58 of *Graduate Studies in Mathematics*, Amer. Math. Soc., Providence, RI, 2003.

[92] S.Waldron, *Orthogonal polynomials on the disc*, J. Approx. Theory, **150**, pp.117-131, 2008.

[93] Z.U.A.Warsi, *Conservation form of the Navier-Stokes equations in general non-steady coordinates*, AIAA J. **19**, p.240, 1981.

[94] N.P.Waterson and H.Deconinck, *Design principles for bounded higher-order convection schemes - a unified approach*, J. Comp. Phys., **224**, pp.182-207, 2007.

[95] A.Winslow, *Numerical solution of the quasi-linear Poisson equation in a nonuniform triangle mesh*, J. Comp. Phys., **1**, pp.149-172, 1967.

[96] O.C.Zienkiewicz and J.Z.Zhu, *The superconvergent patch recovery and a posteriori error estimators. Part 1. The recovery technique*, Int. J. Numer. Methods Eng., **33**, pp.1331-1364, 1992.