An Investigation of a Finite Volume Method Incorporating Radial Basis Functions for Simulating Nonlinear Transport.

Timothy John Moroney

B. Inf. Tech.
B. App. Sc. (Maths)
B. App. Sc. (Honours)

Applicable Mathematics and Advanced Computation Program
School of Mathematical Sciences
Queensland University of Technology
Australia

A thesis submitted for the degree of
Doctor of Philosophy
Queensland University of Technology

2006

Keywords

Finite volume method; Diffusion; Advection; Radial basis functions; Gaussian quadrature; Control volume-finite element; Jacobian-free; Newton-Krylov; Unstructured mesh; Triangular mesh; Tetrahedral mesh; Parallel preconditioner

Abstract

The objective of this PhD research programme is to investigate the effectiveness of a finite volume method incorporating radial basis functions for simulating nonlinear transport processes. The finite volume method is the favoured numerical technique for solving the advection-diffusion equations that arise in transport simulation. The method transforms the original problem into a system of nonlinear, algebraic equations through the process of discretisation. The accuracy of this discretisation determines to a large extent the accuracy of the final solution.

A new method of discretisation is presented that employs radial basis functions (RBFs) as a means of local interpolation. When combined with Gaussian quadrature integration methods, the resulting finite volume discretisation leads to accurate numerical solutions without the need for very fine meshes, and the additional overheads they entail.

The resulting nonlinear, algebraic system is solved efficiently using a Jacobian-free Newton-Krylov method. By employing the new method as an extension of existing shape function-based approaches, the number of nonlinear iterations required to obtain convergence can be reduced. Furthermore, information obtained from these iterations can be used to increase the efficiency of subsequent RBF-based iterations, as well as to construct an effective parallel preconditioner to further reduce the number of nonlinear iterations

required.

Results are presented that demonstrate the improved accuracy offered by the new method when applied to several test problems. By successively refining the meshes, it is also possible to demonstrate the increased order of the new method, when compared to a traditional shape function based-method. Comparing the resources required for both methods reveals that the new approach can be many times more efficient at producing a solution of a given accuracy.

Contents

List of Figures

List of Tables

List of Symbols

A	area
$A_{i\tau}$	area of control volume face $\sigma_{i\tau}$
$\mathbf{a}$	arbitrary two-dimensional vector
a_x	x coordinate of $\mathbf{a}$
a_y	y coordinate of $\mathbf{a}$
α	diffusivity constant (test problem context)
α	small parameter in line search (Newton-Krylov context)
β	exponent in forcing term (Newton-Krylov context)
$\mathbf{B}$	orthonormal basis matrix for $\mathbb{R}^N$
$\mathbf{b}$	arbitrary two-dimensional vector
b_x	x coordinate of $\mathbf{b}$
b_y	y coordinate of $\mathbf{b}$
C_i	complete line integral path around control volume V_i
$C_{i\tau}$	line integral path between $\mathbf{x}_{i\tau}$ and $\mathbf{x}_{i\tau+1}$
$\mathbf{c}$	polynomial coefficient vector
c^2	infinitely smooth RBF parameter
c_k	kth polynomial coefficient
Δ	twice a triangular element's volume (two dimensions)
Δ	six times a tetrahedral element's volume (three dimensions)
$\mathbf{D}$	diffusivity matrix

D_{xx}	diffusivity in x direction
D_{yy}	diffusivity in y direction
D_{zz}	diffusivity in z direction
E_i	ith element
$\mathbf{e}_k$	kth standard basis vector for $\mathbb{R}^n$
η_n	forcing term
$\boldsymbol{\varepsilon}$	error in approximately invariant subspace
ϵ_{mach}	machine precision
ε	discrete directional derivative shift value
$\mathbf{F}$	nonlinear finite volume discretisation function
$\mathbf{f}$	interpolated function values vector
f_i	nonlinear finite volume discretisation component function
g	arbitrary integrand
g_0	constant source
γ_j	RBF boundary coefficient
γ	coefficient in forcing term
$\bar{\mathbf{H}}_m$	$(m+1) \times m$ matrix of Arnoldi coefficients
$\mathbf{H}_m$	$\bar{\mathbf{H}}_m$ with last row removed
$\mathbf{h}$	Harmonic Ritz coordinate vector relative to $\mathbf{V}_m$
$h_{m+1,m}$	entry $(m+1, m)$ in $\bar{\mathbf{H}}_m$
h	heat transfer coefficient (PDE context)
h	interval/edge length ("big O" context)
h	nonlinear residual reduction function (Newton-Krylov context)
$\mathbf{I}$	identity matrix
$\mathbf{i}$	unit vector in x direction
$\mathbf{J}$	Jacobian matrix
$\tilde{\mathbf{J}}_{12}$	$\mathbf{U}^T\mathbf{J}\mathbf{W}$
$\tilde{\mathbf{J}}_{22}$	$\mathbf{W}^T\mathbf{J}\mathbf{W}$

$\mathbf{j}$	unit vector in y direction
$\mathcal{K}$	GMRES projection subspace
$\mathbf{k}$	unit vector in z direction
k	number of augmented Krylov subspace vectors
k	number of nonzero singular values (SVD context)
k^*	number of singular values greater than relative tolerance
$\mathcal{L}$	GMRES orthogonality subspace
ℓ	local cloud of nodes
$\mathbf{\Lambda}$	RBF coefficient matrix
$\boldsymbol{\lambda}$	RBF coefficient vector
λ_j	RBF coefficient
λ	Jacobian eigenvalue
$\mathbf{M}$	arbitrary preconditioner
$\mathbf{M}_1$	norm-minimising preconditioner
$\mathbf{M}_2$	deflation preconditioner
$\mathbf{m}_j$	jth column of $\mathbf{M}_1$
m	problem dimension (RBF context)
m	Krylov subspace dimension (GMRES context)
$m_{i\tau}$	midpoint of control volume face $\sigma_{i\tau}$
N	number of nodes in mesh
Nf_i	number of control volume faces possessed by control volume V_i
N_j	shape function
$\mathbf{n}$	outer normal
$\hat{\mathbf{n}}$	unit outer normal
n	current Newton iteration (Newton-Krylov context)
n	number of nodes used in an interpolation (RBF context)
n^*	number of boundary nodes used in an interpolation

n_{init}	number of nodes used in initial RBF interpolation
n_{final}	number of nodes used in final RBF interpolation
$\boldsymbol{\nu}$	unit outer normal acted upon by $\mathbf{D}$
Pe	cell Peclet number
$\mathbf{P}$	polynomial submatrix of $\boldsymbol{\Lambda}$
$\mathbf{P}_i$	sub-control volume vertex (three dimensions)
P_p	Legendre polynomial of degree p
$\mathbf{p}_i$	control volume outer face vector
p	polynomial (RBF context)
p	number of points in quadrature rule (integration context)
p	estimated order of method (numerical experiments)
$\boldsymbol{\varphi}$	finite volume solution vector
φ	finite volume conserved quantity
φ_i	finite volume solution at node $\mathbf{x}_i$
φ_∞	external temperature
$\boldsymbol{\delta\varphi}$	Newton search direction
$\boldsymbol{\Phi}$	radial submatrix of Λ
ϕ	radial basis function
π_m^k	space of all m-variate polynomials of degree less than or equal to p
ψ	function of conserved quantity φ
ψ_i	ψ computed at node $\mathbf{x}_i$
$\bar{\psi}_i$	control volume averaged value of ψ_i over V_i
$\mathbf{q}_i$	control volume inner face vector
q_k	kth standard basis polynomial of π_p^m
$\mathbf{R}_i$	sub-control volume vertex (two dimensions)
$\mathbf{R}_i$	sub-control volume face vertex (three dimensions)
$\mathbf{r}_0$	GMRES initial residual

S	volumetric source
S_i	S computed at node $\mathbf{x}_i$
$\bar{S}_i$	control volume averaged value of S_i over V_i
s_i	interpolation corresponding to element E_i
s	arc length
σ	surface
σ_i	surface of V_i
$\sigma_{i\tau}$	τth face of control volume surface σ_i
$\mathbf{T}$	$\mathbf{U}^T\mathbf{J}\mathbf{U}$
t	time
t_j	jth integration point
θ	Newton search direction scaling factor
θ_{c}	Current value of θ
θ_{m}	Previous value of θ
$\theta_{\max}$	Maximum safeguard value of θ
$\theta_{\min}$	Minimum safeguard value of θ
$\mathbf{U}$	matrix of orthonormal eigenvectors (Newton-Krylov context)
$\mathbf{U}$	matrix of left singular vectors (SVD context)
$\mathcal{U}$	span$\{u_1, u_2, ..., u_k\}$
$\mathbf{u}$	approximate Jacobian eigenvector
$\mathbf{u}_i$	approximate Jacobian eigenvector (Newton-Krylov context)
$\mathbf{u}_i$	left singular vector (SVD context)
u	parametric representation variable
$\mathbf{V}$	matrix of right singular vectors
$\mathbf{V}_m$	matrix of first m Arnoldi vectors
V	volume
V_i	ith control volume
ΔV_i	volume of V_i

V_{mesh}	total mesh volume
$\mathbf{v}$	velocity (PDE context)
$\mathbf{v}$	operand in Jacobian-vector product (Newton-Krylov context)
$\mathbf{v}_i$	right singular vector
v_x	x coordinate of $\mathbf{v}$
v_y	y coordinate of $\mathbf{v}$
v_z	z coordinate of $\mathbf{v}$
v	parametric representation variable
$\mathbf{W}$	diagonal matrix of singular values (SVD context)
$\mathbf{W}$	orthonormal basis matrix for $\mathcal{U}^{\perp}$ (Newton-Krylov context)
W	weighting function
w_i	singular value (SVD context)
w_i	Gaussian quadrature weight (integration context)
w	parametric representation variable
$\mathbf{x}$	rectangular coordinate vector
$\mathbf{x}(u)$	coordinate vector path parameterised by u
$\mathbf{x}_i$	node
$\mathbf{x}_{i\tau}$	starting point of control volume face $\sigma_{i\tau}$
$\tilde{\mathbf{x}}_i$	RBF centre
x	rectangular coordinate
$\mathbf{y}_0$	GMRES initial estimate vector
$\mathbf{y}$	SVD solution vector (SVD context)
$\mathbf{y}$	GMRES solution vector (Newton-Krylov context)
$\boldsymbol{\delta}\mathbf{y}$	GMRES correction vector
y	rectangular coordinate
z	rectangular coordinate

Statement of Original Authorship

The work contained in this thesis has not been previously submitted to meet requirements for an award at this or any other higher education institution. To the best of my knowledge and belief, the thesis contains no material previously published or written by another person except where due reference is made.

Signature: _____

Date: _____

Publications Arising from this Thesis

T.J. Moroney and I.W. Turner. A modified finite volume method incorporating radial basis functions for simulating diffusion. *ANZIAM J.* 46:C458–C471, 2005.

T.J. Moroney and I.W. Turner. A finite volume method based on radial basis functions for two-dimensional nonlinear diffusion equations. *Applied Mathematical Modelling*, 30(10):1118–1133, 2006.

T.J. Moroney and I.W. Turner. A three-dimensional finite volume method based on radial basis functions for the accurate computational modelling of nonlinear diffusion equations. *J. Comp. Physics* (submitted).

Acknowledgements

I acknowledge my principal supervisor Professor Ian Turner for his contribution to this research, and his constant support and encouragement throughout. No matter what the question, he always had the time to hear it, and the knowledge to answer it. He is indeed a wonderful supervisor.

I acknowledge my associate supervisor Dr Troy Farrell and director of research Professor Sean McElwain for their valuable feedback during the course of this research programme.

I acknowledge the postgraduate students and staff in the School of Mathematical Sciences for providing the ideal environment for mathematical research.

I acknowledge the High Performance Computing and Research Support Group for granting me access to the university supercomputer.

Finally, I acknowledge my family, particularly my parents, for their support throughout all of my university studies.

Chapter 1

Introduction

Partial differential equations occur ubiquitously in the fields of scientific and industrial simulation. Such equations can be used to describe physical conservation laws for the evolution of properties such as temperature, density, concentration, mass, energy, and so on. Mathematical models that make use of these equations can be used to simulate physical processes, and thus predict outcomes much faster and more cheaply than actual simulation in real time. Combined with visualisation techniques they can also be used to convey results of a simulation to those unfamiliar with the underlying physical processes.

Mathematical models may also lead to new scientific discoveries that would have been difficult or even impossible to achieve by simply observing real world phenomena. This is particularly true of applications or experiments where it is difficult to control certain physical parameters. There is no such difficulty in arbitrarily setting parameters in a mathematical model, and observing their impact on the outcome.

Practical, efficient means of solving partial differential equations are therefore highly sought. In some simple cases, analytic solution techniques

are possible. In a great many practical cases however, the equations are highly nonlinear and involve complicated boundary conditions. Furthermore, the domain over which they are to be solved is seldom a simple geometric region. These factors mean that in practice an analytic solution to such equations is all but impossible. Consequently, numerical techniques must be employed to obtain the solution.

1.1 Literature review

1.1.1 Finite volume method

There are many different numerical strategies for solving partial differential equations. In the past, the finite element method [17] has been the numerical method of choice for computational fluid dynamics (CFD) applications, and today it still remains in wide use. However, in more recent times the finite volume method has emerged as the preferred method in many industrial codes.

The finite volume method was introduced by Patankar in 1980 [65]. In his book, he developed the method for use in solving heat transfer and fluid flow problems. Today, the finite volume method is dominant throughout the field of CFD for solving the kinds of partial differential equations encountered in this area.

Examples of CFD applications that have employed the finite volume method include the modelling of saltwater intrusion into coastal aquifers [24, 23], generic heat and mass transport in porous media [67], low temperature convective wood drying [27, 68, 82, 84], combined microwave and convective drying in pine [90], compressible fluid flows [85] and other, general advection-diffusion problems [41, 43, 44, 46, 61, 64]. The finite volume

method is also gaining popularity for problems in solid mechanics [2, 3, 25, 26] and even for problems involving multi-physics phenomena [3].

The major advantage of the method over others such as the finite difference method and the finite element method lies in its intimate connection to the underlying physical process. Most partial differential equations are mathematical statements of underlying conservation laws—the conservation of mass, the conservation of energy, the conservation of momentum, and so on. In methods such as finite difference and finite element, these quantities are not necessarily conserved at the discrete level, meaning that the approximate solutions may, owing to numerical error, exhibit physically unrealistic behaviour. The finite volume method ensures that these quantities remain conserved, meaning that method is in agreement with the underlying laws of physics at all levels of the discretisation [65].

Another advantage of the method is its applicability to a variety of mesh structures and geometries. The formulation of the method lends itself to a very natural implementation of boundary conditions, even where the boundaries and associated boundary conditions are complicated. The mesh itself may be either structured or unstructured—the finite volume works equally well with either.

Current literature favours the use of unstructured meshes for the numerical treatment of real-world problems. In two dimensions, these meshes are usually composed of either triangles or quadrilaterals, while in three dimensions they are generally composed of tetrahedra or hexahedra, but there are exceptions. In [82, 84] for example, the previous work done in [67] was migrated to three dimensions by extending the two-dimensional, triangular mesh in the z-direction, along the length of the board, creating a mesh of triangular prisms.

The process of discretisation is at the heart of the finite volume method. Around each node in the mesh, a control volume is constructed over which the governing equations are solved discretely. It is well accepted in the literature that the accuracy of this discretisation determines to a large extent the accuracy of the entire method. There are two components of this discretisation. The first is the approximation of the flux through a control volume face by numerical integration over the face and the second is the evaluation of the functions and gradients themselves, at the points required for the integration. If both of these components are not sufficiently accurate, then the overall accuracy of the method will be impacted.

One approach commonly used in the literature is to compute the flux integral using the midpoint rule. It is shown in [85] that this produces an approximation accurate to second order, but only if the gradients themselves are evaluated precisely at the midpoint. However in some situations it may be desirable to achieve higher accuracy than that offered by the midpoint rule. For example, in [64, 79], Gaussian quadrature is used to achieve higher order accuracy for the integrations.

The evaluation of the functions and gradients also poses significant difficulties. A popular method, which has become known as the control volume-finite element method (CV-FE), incorporates ideas from the finite element method, in the form of shape functions, into the finite volume discretisation. The resultant interpolation is inexpensive to compute and is known to perform well under mostly isotropic conditions [41, 42].

For anisotropic problems however, the CV-FE method may fail to produce accurate results. This is caused by the first order interpolation being insufficient to capture the rapidly varying gradient over a single element. A number of different strategies for improving the accuracy offered by standard

CV-FE have been documented in the literature. In [86, 87] a two-node flux approximation technique was presented based on a cell-centred finite volume formulation. In this approach the face normal was decomposed into components in the direction joining the neighbouring node, and the direction orthogonal to this. It was found that acceptable accuracy was achieved only when using an orthogonal mesh.

In [88], techniques were presented for capturing information on the cross-diffusion term while still using a two-node approximation. A similar, cell-centred two-node approach was presented in [14] where it was successfully applied to two cavity flow problems. In [3], an approach to multi-physics modelling of both computational fluid dynamics and computational solid mechanics was presented, which incorporated a vertex-centred two-node scheme.

Later work in groundwater flow [23, 24] and wood drying [67, 68] also used also used vertex-centred finite volume methods, reflecting the increasing popularity of this approach in CFD applications. At the same time, methods of further improving the accuracy of the shape function-based (CV-FE) discretisations were being investigated. Several methods are compared and contrasted in [41, 42]. Some involve approximating the gradient at the mesh node itself, including Gradient Using a Representative Point (GRF) which is itself only first order, Gradient Using the Midpoint for FM (CMF1) and Gradient Using the Midpoint for FR (GMF2).

The conclusion drawn in [42] and in the later work [45] is that the accuracy of existing finite volume strategies is still doubtful under extreme anisotropic ratios. As such, only the use of very fine meshes can yield accurate results under these conditions. In an effort to overcome this limitation, a technique known as Flux Approximation using Decomposed Vectors (FADV) is proposed in [44]. In this method, vector algebra is utilised to obtain an expression for

the dot product of the gradient with the effective face normal. Improved Least Squares Gradient Reconstruction (ILSGR) is introduced as a method to compute the various quantities required by the FADV method. Finally, in [46] a method known as CVFE-LS is presented that corrects the problems of divergence of the underlying linear systems that were observed with previous methods.

The treatment of the advective component when solving advection-diffusion equations also requires careful consideration. In many cases, the usual interpolation strategies will fail to produce physically realistic solutions in cases of high advection [61]. Techniques have been proposed to overcome this limitation, the simplest of which is upwinding, but which is well known to introduce artificial diffusion, or smearing, into the solution [61]. More sophisticated, flux limiting approaches may be used to overcome this problem, see for example [82, 83, 84, 89]. In [82, 84], upwinding and flux-limiting are incorporated into a three-dimensional CV-FE method, along with higher order gradient approximations involving weighted shape function gradients and least squares Taylor expansions for estimating the leading order errors.

Such modifications made to the finite volume discretisation, although beneficial to the final accuracy of the solution, are not without cost. This cost is in the form of increased overall execution time, as reported in [43, 46, 83, 84] for example. The interpolation method chosen for the finite volume discretisation must therefore be of sufficient accuracy to justify the extra computational time required for its construction and evaluation.

1.1.2 Radial basis functions

The method of radial basis functions, or RBFs, is a method of scattered data interpolation applicable in any dimension. Presented in [70] as extension

of one-dimensional spline functions to several variables, an RBF interpolant takes the form

$$s(\mathbf{x}) = \sum_{j=1}^{n} \lambda_j \phi(\|\mathbf{x} - \mathbf{x}_j\|) + p(\mathbf{x}) \,, \tag{1.1}$$

where the $\mathbf{x}_j$ are the nodes over which the interpolation is constructed, the λ_j are the RBF coefficients, p is a polynomial, and ϕ is known as a radial basis function, which is where the method gets its name.

Different choices of ϕ give rise to different types of RBF interpolants. For example, the Gaussian function $\phi(r) = e^{-c^2 r^2}$ is used extensively in the field of neural networks [52, 55], an approach to computing that utilises networks of artificial neurons. Radial basis functions can provide the means of connecting these nodes via scattered data interpolation.

Other choices, such as the thin plate spline $\phi(r) = r^2 \log(r)$ and the linear spline $\phi(r) = r$ are linked to particular dimensions (two and three dimensions respectively), and so are popular choices for surface reconstruction and computer graphics [12].

One basis function than enjoys success in many different fields is the multiquadric, $\phi(r) = \sqrt{c^2 + r^2}$, first introduced in [35]. In the review paper [36], applications of multiquadrics to the fields of geodesy, geophysics, surveying, mapping, photogrammetry, remote sensing, signal processing, geography, digital terrain models and hydrology were presented. The reasons for the multiquadric's wide-reaching success are most likely its ease of implementation and its generally high accuracy. In fact, in tests of over 30 different methods of scattered data interpolation carried out in [30], the multiquadric was rated at the top for both ease of implementation and accuracy.

Given the versatility of the method of radial basis functions, it is no great surprise that it has also found application in the field of numerical solution of PDEs. However, it is not with the finite element or finite volume methods,

the dominant numerical methods for numerical transport simulation, that the method of RBFs has been applied. Rather, it is through collocation methods that RBFs, particularly multiquadrics, have been introduced.

Introduced in [47] and first analysed in [29], PDE collocation utilising RBFs has been employed to solve problems in the field of CFD [38, 47, 54] and elsewhere [39]. The primary advantage of this method over more traditional methods such as finite volume or finite element is that it does not require a mesh. However, it does not ensure that the discrete conservation laws are enforced, as does the finite volume method, and it also suffers from the more general limitations of radial basis function interpolations to be discussed in the next paragraphs.

In spite of its widespread use and acceptance in the literature, there are some difficulties associated with the method of radial basis functions as a means of interpolation. First, it is well known that the underlying matrix system that must be solved when constructing an RBF interpolant can be extremely ill-conditioned [18, 28, 48, 53, 70]. Some authors [4, 5], consider this ill-conditioning not as a limitation of the method itself, but as a consequence of the underlying vector space basis. Methods have been developed in [4] to utilise alternative bases in the solution process, however the process of actually computing an appropriate basis is nontrivial.

Aside from being ill-conditioned, the RBF matrix is also dense, which can therefore lead to high computational and storage requirements for large RBF interpolants. Quotes from different authors at various points in history lamenting this fact are given in [4], with the consensus that RBFs were unsuitable for large-scale interpolation problems. Techniques to overcome this computational limitation range from simply not using large numbers of nodes in an RBF interpolation, to using truncated basis functions with compact

support [48], through to sophisticated domain decomposition and change-of-basis techniques [4, 5]. For smaller sized problems, the truncated singular value decomposition [33] offers a way to overcome these ill-conditioned matrix systems without modification.

Another well-known problem of RBF-based interpolations it the relative lack of accuracy they offer near the boundaries of the domain. In [28] several modifications to the basic RBF method are presented that are designed to increase the accuracy of the interpolant near boundaries. These include incorporating higher degree polynomial terms into the interpolation, clustering more nodes near the boundaries, and even introducing a second set of points (the centres), different from the nodes, where the actual evaluations take place. For the small extra expense required for these modifications, the test results showed the accuracy was generally improved, although no tests were carried out in dimensions higher than two.

The multiquadric, like all such infinitely smooth basis functions, incorporates a free parameter usually denoted by c, c^2 or ε. The value of this parameter has implications on both the accuracy of the resulting interpolation, and of the conditioning of the underlying matrix problem [18, 53]. In [72], it is shown that the optimum value for this parameter in multiquadric, inverse multiquadric and Gaussian radial basis functions depends on many aspects of the problem, including the function to be interpolated, the number and positioning of the nodes and even the precision to which the calculations are performed.

An algorithm is presented in [72] that can assist in choosing a reasonable value for this parameter, but there is some expense involved in its application. On the positive side, it was found in [30] that in the particular case of the multiquadric, the interpolations are actually quite stable with respect to this

parameter, unlike with the inverse multiquadric for example.

The error analysis for RBF-based interpolations is not straightforward. Some results have been established [56, 57, 70, 77, 78], but they are not simple to apply. Certainly, there is no single result stating that RBF interpolations provide $\mathcal{O}(h^p)$ accuracy for some p. In [57] it is shown that the local error bound for certain types of RBFs, including multiquadrics and Gaussians, actually leads to arbitrarily high approximation orders for these functions, which the authors note limits the practical applicability of such measures.

1.1.3 Jacobian-free Newton-Krylov methods

The finite volume method transforms a partial differential equation into a system of algebraic equations through the process of discretisation. If the original PDE is nonlinear, then so too will be the algebraic system. Jacobian-free Newton-Krylov methods are an increasingly popular method for solving large scale, nonlinear, algebraic systems such as these.

An excellent survey of Jacobian-Free Newton-Krylov methods is presented in [51]. As its name suggests, the method involves the application of a nonlinear, Newton method in combination with a Krylov subspace linear, iterative method, with the entire process being carried out without needing to form the Jacobian matrix.

The basic Newton iteration for solving a system of nonlinear equations is based on the Taylor series for $\mathbf{F}(\boldsymbol{\varphi})$ around the current iterate $\boldsymbol{\varphi}^{(n)}$:

$$\mathbf{F}(\boldsymbol{\varphi}^{(n)} + \boldsymbol{\delta\varphi}^{(n)}) = \mathbf{F}(\boldsymbol{\varphi}^{(n)}) + \mathbf{J}(\boldsymbol{\varphi}^{(n)})\boldsymbol{\delta\varphi}^{(n)} + \mathcal{O}(\|\boldsymbol{\delta\varphi}^{(n)}\|^2)\,. \tag{1.2}$$

The expression for the next iterate $\boldsymbol{\varphi}^{(n+1)}$ can be found by setting the left

hand side to zero and solving, neglecting the higher order terms, for $\boldsymbol{\delta\varphi}^{(n)}$:

$$\mathbf{J}(\boldsymbol{\varphi}^{(n)})\ \boldsymbol{\delta\varphi}^{(n)} = -\mathbf{F}(\boldsymbol{\varphi}^{(n)}), \tag{1.3}$$

so that

$$\boldsymbol{\varphi}^{(n+1)} = \boldsymbol{\varphi}^{(n)} + \boldsymbol{\delta\varphi}^{(n)}\,. \tag{1.4}$$

The result is a linear system involving the Jacobian matrix: the matrix of partial derivatives of f_i with respect to φ_j. If each component function f_i involves only local values of φ_j, then this Jacobian matrix will be sparse.

The solution of linear systems, and in particular sparse linear systems, is undoubtedly one of the biggest areas of computational research. This is due to their pervasiveness in practically every field of science and technology. For almost as long as there have been linear systems to solve, there have been two schools of thought on how to solve them: direct methods or indirect (iterative) methods.

Direct method practitioners use Gaussian elimination or one of its variants, often exploiting bandedness or other sparsity patterns in the matrix to reduce fill-in. Iterative method practitioners on the other hand, have literally hundreds of methods to choose from when it comes to solving linear systems.

A survey of many such iterative methods is given in [76]. From the earliest and simplest iterative methods, such as Richardson's method, Jacobi's method and the successive over-relaxation method (SOR), came the more advanced projection methods such as the conjugate gradient and Lanczos methods. These were the first of what have been come to be known as Krylov subspace methods—methods that employ a projection onto a Krylov subspace to approximate the solution.

One of the most popular unsymmetric Krylov subspace methods today

is the Generalised Minimum Residual Method, GMRES, introduced in [75] and described in detail in [74]. It uses the Arnoldi method to compute an orthonormal basis for the Krylov subspace from which the best solution, in the least squares residual sense, is drawn. It also does not suffer from breakdown problems that affect some of its competitors. The cost of having these desirable properties is that it also requires the storage of all previous Arnoldi vectors. This can be expensive in terms of memory consumption when the dimension of the Jacobian matrix is extremely large.

In [74] the restarted GMRES method, GMRES(m) is discussed. In this variant the method is restarted after m iterations with the most recent solution used as the initial estimate for the next cycle. Although this overcomes the storage issue discussed above, the method is prone to stalling, where only small reductions in the residual are achieved in each GMRES cycle.

In [58] a modified GMRES(m) method that augments the Krylov subspace with approximate eigenvalues is presented. The method cleverly re-uses some of the previous Krylov subspace information in the subsequent iterations, rather than simply discarding this information. Variations and improvements on this initial idea have since also been explored [11, 13, 22, 59, 73] with a review of several methods given in [60]. These methods of augmenting approximate eigenvectors to the Krylov subspace have come to be known as deflated GMRES methods.

When Krylov subspace methods such as deflated GMRES are used to solve for the Newton steps in an inexact Newton method, the method is known as a Newton-Krylov method. The advantage in using Krylov subspace methods is that these methods only require the action of the matrix in the form of matrix-vector product [74, 75]. In the case of the Jacobian matrix, its action on a vector can be approximated using a first order finite difference quotient

[9, 49]

$$\mathbf{J}(\boldsymbol{\varphi})\mathbf{v} \approx \frac{\mathbf{F}(\boldsymbol{\varphi} + \varepsilon\mathbf{v}) - \mathbf{F}(\boldsymbol{\varphi})}{\varepsilon}\,, \tag{1.5}$$

where $\mathbf{v}$ is an arbitrary vector, the operand in the Jacobian-vector product. This approach means that the Jacobian matrix need not ever be explicitly computed, leading to a so-called Jacobian-free Newton-Krylov method [51]. The price paid for this convenience is that the convergence of the method may not be as fast or as predictable as when forming the Jacobian matrix explicitly [51]. For best results, it is important that ε be carefully chosen, using a value such as

$$\varepsilon = \frac{\sqrt{(1 + \|\boldsymbol{\varphi}\|\epsilon_{\text{mach}})}}{\|\mathbf{v}\|} \tag{1.6}$$

suggested in [51], or

$$\varepsilon = \frac{\sqrt{\epsilon_{\text{mach}}}}{\|\mathbf{v}\|} \max(|\boldsymbol{\varphi}^T\mathbf{v}|, \text{typ } \boldsymbol{\varphi}^T|\mathbf{v}|)\ \text{sign}(\boldsymbol{\varphi}^T\mathbf{v}) \tag{1.7}$$

suggested in [9], where ϵ_{mach} is the machine precision.

To improve the nonlinear convergence rate, second order or higher methods for approximating $\mathbf{Jv}$ are also possible. For example, using central differences yields, to second order,

$$\mathbf{J}(\boldsymbol{\varphi})\mathbf{v} \approx \frac{\mathbf{F}(\boldsymbol{\varphi} + \varepsilon\mathbf{v}) - \mathbf{F}(\boldsymbol{\varphi} - \varepsilon\mathbf{v})}{2\,\varepsilon} \tag{1.8}$$

but the use of such methods is practice is rare, likely owing to the extra evaluations of $\mathbf{F}$ they require [51].

The Newton-Krylov method can be made globally convergent by incorporating line searching, or backtracking [6, 19, 49, 66]. This ensures that the method either converges, or else fails in a limited number of well-understood ways, for any choice of initial estimate. Typically, the underlying linear sys-

tems are not solved to their full accuracy, but only to the level required to compute an appropriate Newton correction. These inexact Newton methods employ a forcing term [20, 49] to ensure that the linear systems are not over-solved.

Krylov subspace methods such as GMRES typically require some form of preconditioning to be most effective [11, 60, 74, 75]. Preconditioning in a Jacobian-free environment can be difficult, as there is no explicit Jacobian matrix available from which to construct a preconditioner. Therefore, some form of simpler, approximate Jacobian matrix is typically constructed for preconditioning purposes, hence the method being referred to only as "Jacobian-free", and not fully "matrix-free" [51].

In terms of parallelism, most attention is focussed on generating high quality, parallel preconditioners [51, 74]. There are several different approaches to writing parallel preconditioners for large linear systems. A good survey of methods is given in [7] and implementations for many of these in [74]. Some of the most well-known examples of parallel preconditioners are the polynomial preconditioners, the parallel ILU methods and the sparse approximate inverse methods [7]. There is still a high level of research in this area, in a bid to improve the robustness and parallel performance of these techniques [7], but so far no single method has demonstrated its superiority.

1.2 Objectives

The objective of this PhD research programme is to investigate the effectiveness of a finite volume method incorporating radial basis functions for simulating nonlinear transport processes. Applications of such techniques can be found in many fields, such as modelling wood drying, groundwater flow

and heat transfer. It is anticipated that this new method will improve both the accuracy and efficiency of existing finite volume discretisation techniques, permitting relatively coarse meshes to be used in obtaining solutions with a corresponding reduction in solution time and computational overheads.

The method of radial basis functions [70] will be incorporated into the finite volume discretisation. This method of scattered data interpolation has been found to perform well in a variety of different scenarios [30], but has yet to be tested in the context of a finite volume method. When coupled with Gaussian quadrature integration techniques, this new discretisation method will form the basis of the finite volume implementation under consideration.

An effective finite volume method must also employ some means of solving the nonlinear system of equations that results from the discretisation. Inexact Newton methods implemented using a Jacobian-free, Krylov-based linear solver are growing in popularity as a means for solving these kinds of problems [51]. One such method will be implemented in this thesis, with special attention paid to efficiency of implementation, particularly in the area of preconditioning.

The remainder of this section elaborates on these objectives in more detail.

Implement the finite volume method over unstructured two- and three-dimensional meshes

The finite volume method is a popular choice for solving partial differential equations (PDEs), finding application in the simulation of a wide range of important industrial problems in science and technology. The method involves the discretisation of the computational domain into a mesh of so-called control volumes, which together comprise the total volume of the domain. The

PDE is then solved discretely over each control volume, ultimately yielding an approximate solution at each node in the mesh.

The method has the desirable property of ensuring that the fundamental conservation laws of the system are satisfied, both over each control volume, and over the domain as a whole. The fact that the method captures the underlying physics, even at the discrete level, sets it apart from other popular methods such as the finite difference and finite element methods.

The finite volume method will be applied over unstructured triangular meshes in two dimensions, and unstructured tetrahedral meshes in three dimensions. This will require generating appropriate meshes using third party software, forming control volumes, and computing fluxes through every control volume face in the mesh. Certain numbering and ordering conventions will need to be established to ensure that these fluxes are computed consistently, thus ensuring that the underlying principles of the finite volume method are satisfied.

Develop an accurate finite volume discretisation method for two and three dimensions

At the heart of the finite volume method is the discretisation of the underlying partial differential equation. This discretisation allows the value of the function to be approximated at a representative point within each control volume. The accuracy of the discretisation effectively determines the overall accuracy of the finite volume method.

The method of shape functions, from finite element theory, is often used in finite volume discretisations. This method is simple to apply, but in some cases has been found to produce inaccurate results, particularly for problems with high anisotropy [41, 42]. Past work has sought to improve on this

strategy by increasing its spatial accuracy to higher order using Taylor polynomials and least squares methods [41, 46, 82, 84]. In this thesis the method of radial basis functions will be used, that, in combination with existing CV-FE practices will seek to provide an accurate yet efficient finite volume discretisation. In addition, Gaussian quadrature methods will be used for the control volume integrations, to ensure that the high interpolation accuracy offered by RBFs is not compromised by lower-order integration schemes such as the midpoint rule.

Use an efficient Jacobian-free Newton-Krylov method with parallel preconditioning to solve the underlying nonlinear system

The discretisation method discussed in the previous objective transforms the partial differential equation into a nonlinear system of algebraic equations. This nonlinear system must be solved in order to obtain the solution values at the mesh nodes.

Jacobian-free Newton-Krylov methods provide an efficient way to solve such systems without excessive memory requirements. By exploiting the fact that Krylov-based linear solvers do not require the Jacobian matrix explicitly, these methods avoid ever needing to generate the Jacobian matrix. However, a simpler, less expensive Jacobian matrix is still required for preconditioning purposes, and it is envisaged that the CV-FE method will be ideal for this purpose. Furthermore, parallel computing techniques will be used to greatly reduce the time required to generate such a preconditioner.

Test the accuracy and efficiency of the new method against the existing control volume-finite element approach

By solving test problems with known analytic solutions, the accuracy of the finite volume solutions may be compared against one another. In this way the effectiveness of the method at modelling the underlying physics can be judged. Especially for test problems with parameters that are known to pose difficulties to numerical solution methods, it is desirable to test whether the new RBF-based method will be more successful in this regard. Not only the will the accuracy of the methods be compared, but estimates of the order of accuracy they provide will be made, by using successively refined meshes and observing the reduction in the errors. The time and resource requirements of the methods will also be compared, thereby determining what savings can be made by using the new, more accurate method over coarse meshes, rather than existing methods over fine meshes. The effectiveness of the pre-conditioning strategies will also be judged by observing the reduction in the number of nonlinear function evaluations required to achieve convergence.

1.3 Original contributions

The original contributions of this thesis are summarised below:

- a finite volume discretisation incorporating radial basis functions and Gaussian quadrature for unstructured, two-dimensional, triangular meshes.
- the extension of this two-dimensional method to a three-dimensional, unstructured tetrahedral framework.
- a two-stage nonlinear solution strategy, whereby a traditional CV-FE

method is used initially to obtain a solution of "coarse" accuracy, which is then refined using a more accurate RBF-based method.

- a parallel preconditioning approach for improving the convergence rate by combining information from both shape function and RBF-based iterations.

- the C++ implementation of the complete finite volume code, beginning with the construction of control volumes around a given finite element mesh, and finishing with the computed solution output at each mesh node.

1.4 Structure of thesis

This thesis comprises seven chapters, the contents of which are as follows:

Chapter 1 A brief introduction to transport processes and their numerical simulation is given. A literature review on the finite volume method and its implementation then follows. The objectives of this thesis are presented, and the original contributions listed. The chapter closes with an outline of the thesis structure.

Chapter 2 The finite volume method is introduced as a means for numerically solving partial differential equations arising from the simulation of transport processes. The discretisation process is discussed in detail, and the implications of its accuracy on the quality of the final computed solution considered. Methods for control volume integration are presented, consisting of the midpoint rule and Gaussian quadrature. A review of the shape function method for interpolation is followed by a general discussion of scattered data

interpolation, together with the implications of applying it as part of a finite volume discretisation. The method of radial basis functions is presented as a means for computing accurate and efficient interpolations within the finite volume framework. Methods for choosing the nodes in an RBF interpolation are considered, along with the impact that these have on the finite volume stencils. The means for enforcing the discrete conservation laws by ensuring consistent flux between adjacent control volumes is discussed, followed by some modifications that seek to improve the accuracy of the interpolations near boundaries of the solution domain.

A Jacobian-free Newton-Krylov method is offered as the means for computing the solution to the resulting nonlinear system. The most appropriate combinations of integration and interpolation methods are considered, and ways for reducing the impact of the most costly iterations are also presented. A method for adaptively switching between the cheaper shape function-based iterations and the more costly RBF-based iterations is introduced. The underlying Krylov iterative method is discussed, including the methods used for deflation and preconditioning. Finally, some comments are made on the possible parallelisation of this new finite volume strategy.

Chapter 3 The implementation aspects of this finite volume method in two dimensions are considered. First, an introduction to two-dimensional, unstructured meshes is given, followed by the means of forming control volumes and sub-control volumes in triangular meshes. Some conventions that help ensure correct and efficient processing of each element are then presented. The two-dimensional aspects of integration and interpolation are discussed, including one and two-dimensional Gaussian quadrature parameterisations and shape function and RBF interpolant expressions. Finally, the

two-dimensional finite volume discretisations are presented.

Chapter 4 The results of numerical experiments in two dimensions are presented. Three test problems are considered, incorporating diffusion, advection and nonlinearity, along with the set of unstructured, triangular meshes over which these problems have been solved. Results from solving each test problem over a single mesh with varying problem parameters are shown, and the accuracy of the new RBF-based method compared to the CV-FE method. Results over successively refined meshes are used to demonstrate the high order of the new method. The efficiency of the methods are also compared, including how effectively the preconditioning strategy performs, and how many nonlinear function evaluations are required to achieve convergence.

Chapter 5 The implementation aspects of this finite volume method in three dimensions are considered. First, an introduction to three-dimensional unstructured meshes is given, followed by the means of forming control volumes and sub-control volumes in tetrahedral meshes, which is a considerably more difficult process in three dimensions than it is in two. The conventions that ensure correct and efficient processing of each element are also discussed. The three-dimensional aspects of integration and interpolation are discussed, including two and three-dimensional Gaussian quadrature, shape functions and radial basis functions. Finally, the three-dimensional finite volume discretisations are presented.

Chapter 6 The results of numerical experiments in three dimensions are presented. Three test problems are considered, incorporating diffusion, advection and nonlinearity, along with a family of unstructured, tetrahedral meshes over which these problems have been solved. Results from solving

each test problem over a single mesh with varying problem parameters are shown, and the accuracy of the new RBF-based method compared to that which uses shape functions. Results from solving over refined meshes are used to demonstrate the high order of the new method. The efficiency of the methods are also compared, including how effectively the preconditioning strategy performs and how well it performs in a parallel environment. Timings for the various methods are given, demonstrating how the RBF-based method can achieve accurate results in less time than the shape function-based method. Finally, the effectiveness of the adaptive strategy for switching between shape functions and RBFs is investigated.

Chapter 7 A summary of the outcomes of this study, along with the new contributions that resulted from the research are presented. The chapter closes with some recommendations for future research in the area.

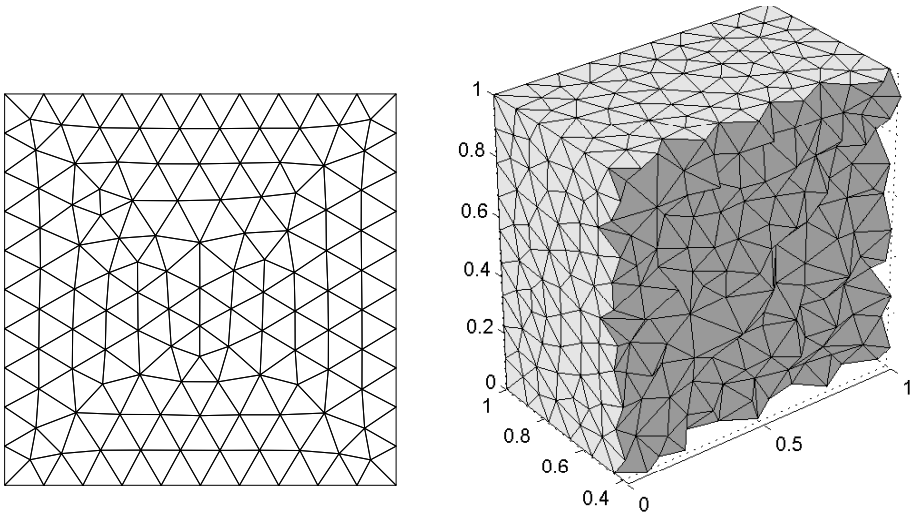
Chapter 2

The Finite Volume Method

2.1 Introduction

The *finite volume method* is a numerical method for solving partial differential equations. Introduced originally as a means for solving heat transfer problems [65], it has subsequently grown to be one of the most widely used numerical methods for solving partial differential equations. Its usage has extended beyond the original application of heat transfer, and is now the preferred method in the more general setting of computational fluid dynamics, where conservation is the most important property to be modelled [62].

The finite volume method is based on a discretisation of the original, continuous problem. To this end, it employs a mesh—a geometric structure consisting of *nodes*, and the corresponding connections between the nodes to form *elements*. The dimension of the underlying problem dictates the type of elements that make up the mesh. Commonly used element types are triangles and quadrilaterals in two dimensions and tetrahedra and hexahedra in three dimensions. Mixed meshes incorporating several different element types can also be used. Figures 2.1(a) and 2.1(b) show examples of the types



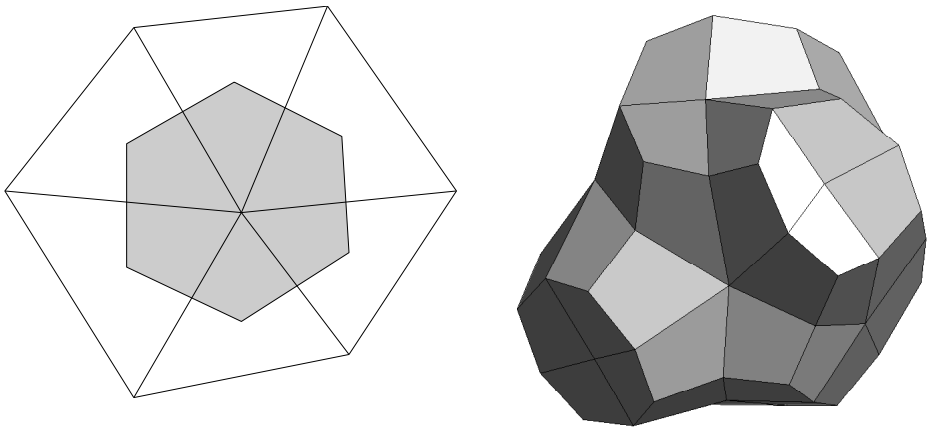
(a) (b)

Figure 2.1: Unstructured meshes: (a) triangular; (b) tetrahedral.

of meshes considered in this thesis—triangular meshes in two dimensions, and tetrahedral meshes in three dimensions, respectively.

Both of these are examples of *unstructured* meshes. The elements are not aligned in a regular fashion, but rather they are tiled irregularly across the domain. Unstructured meshes are important because they can model the complex and irregular domains often encountered when solving real-world problems.

Producing a good solution using the finite volume method depends in part on using a good mesh. Generating an unstructured mesh with good geometric properties on a complex two or three-dimensional domain is a difficult problem, and one that is best left to dedicated mesh-generation software. Fortunately, the popularity of another mesh-based method for solving PDEs, the *finite element method*, has seen the development of many high quality mesh generators, including many freeware releases, for example [31, 63, 69]. The so-called *finite element meshes* generated by these programs



(a) (b)

Figure 2.2: Examples of control volumes in (a) two dimensions (grey); (b) three dimensions.

can also be used for the finite volume method.

The finite volume method requires constructing *control volumes* around every node in the mesh. Examples of control volumes in two and three dimensions are shown in Figures 2.2(a) and 2.2(b) respectively. The method of actually forming control volumes from a finite element mesh is dimension-dependent, and will therefore be addressed separately in Chapters 3 and 5. Note that this is a *vertex-centred* approach to the finite volume method. *Cell-centred* finite volume methods are also possible, where the elements themselves form the control volumes, but they are not pursued here.

For now, suppose that there are N nodes in the mesh, and that around each node $\mathbf{x}_i$, there exists a control volume V_i with surface σ_i and volume ΔV_i, such that no V_i overlap and the total mesh volume V_{mesh} is given by

$$V_{\text{mesh}} = \sum_{i=1}^{N} \Delta V_i. \tag{2.1}$$

Now consider the basic time-dependent advection-diffusion equation

$$\frac{\partial \psi(\varphi)}{\partial t} = \boldsymbol{\nabla} \cdot (\mathbf{D}\boldsymbol{\nabla}\varphi - \mathbf{v}\varphi) + S. \tag{2.2}$$

The finite volume method proceeds by integrating (2.2) over each control volume V_i.

$$\iiint_{V_i} \frac{\partial \psi}{\partial t} \, \mathrm{d}V = \iiint_{V_i} \boldsymbol{\nabla} \cdot (\mathbf{D}\boldsymbol{\nabla}\varphi - \mathbf{v}\varphi) \, \mathrm{d}V + \iiint_{V_i} S \, \mathrm{d}V. \tag{2.3}$$

The Divergence Theorem then allows the first right hand side term in (2.3), representing the advective-diffusive flux, to be expressed as an integral over the surface σ_i

$$\iiint_{V_i} \boldsymbol{\nabla} \cdot (\mathbf{D}\boldsymbol{\nabla}\varphi - \mathbf{v}\varphi) \, \mathrm{d}V = \iint_{\sigma_i} (\mathbf{D}\boldsymbol{\nabla}\varphi - \mathbf{v}\varphi) \cdot \hat{\mathbf{n}} \, \mathrm{d}\sigma. \tag{2.4}$$

Defining the control-volume averaged values of ψ and S as $\bar{\psi}_i$ and $\bar{S}_i$ respectively:

$$\bar{\psi}_i = \frac{1}{\Delta V_i} \iiint_{V_i} \psi \, \mathrm{d}V \,, \tag{2.5}$$

$$\bar{S}_i = \frac{1}{\Delta V_i} \iiint_{V_i} S \, \mathrm{d}V \,, \tag{2.6}$$

(2.4) may be rewritten as

$$\underbrace{\frac{\mathrm{d}\bar{\psi}_i}{\mathrm{d}t}\Delta V_i}_{\text{temporal}} = \underbrace{\iint_{\sigma_i} (\mathbf{D}\boldsymbol{\nabla}\varphi - \mathbf{v}\varphi) \cdot \hat{\mathbf{n}} \, \mathrm{d}\sigma}_{\text{advective-diffusive flux}} + \underbrace{\bar{S}_i \, \Delta V_i}_{\text{source}} \,. \tag{2.7}$$

No approximations have been made at this stage, so (2.7) is an exact reformulation of (2.2) [85]. It is this formulation that leads to the finite volume discretisation of (2.2).

2.1.1 Discretisation

The process of *discretisation* lies at the heart of the finite volume method. It is this process that transforms equation (2.7), which still contains unknown quantities, into a usable approximation that can ultimately yield a solution. This solution consists of an approximate value φ_i at each node $\mathbf{x}_i$, usually written in vector form as

$$\boldsymbol{\varphi} = (\varphi_1, \varphi_2, \ldots, \varphi_N)^{\mathrm{T}}. \tag{2.8}$$

The discretisation of (2.7) relies on computing approximations of the temporal, advective-diffusive flux and source terms that involve only the (unknown) quantities $\varphi_i,\ i = 1, \ldots, N$. The accuracy of this discretisation effectively determines the overall accuracy of the finite volume method. In this thesis the focus is on finding accurate spatial discretisations, and therefore the steady-state form of (2.2) is used,

$$\boldsymbol{\nabla} \cdot (\mathbf{D}\boldsymbol{\nabla}\varphi - \mathbf{v}\varphi) + S = 0 \tag{2.9}$$

or, in control volume form

$$\underbrace{\iint_{\sigma_i} (\mathbf{D}\boldsymbol{\nabla}\varphi - \mathbf{v}\varphi) \cdot \hat{\mathbf{n}} \, \mathrm{d}\sigma}_{\text{advective-diffusive flux}} + \underbrace{\bar{S}_i \, \Delta V_i}_{\text{source}} = 0. \tag{2.10}$$

Note that in (2.10) some dependencies on φ and $\mathbf{x}$ have been suppressed for readability. In this thesis, both $\mathbf{D}$ and $\mathbf{v}$ are considered to depend on φ, while S, along with the solution φ itself, is considered to depend on $\mathbf{x}$. Thus

(2.10) could be expressed more thoroughly as

$$\iint_{\sigma_i} (\mathbf{D}(\varphi(\mathbf{x}))\boldsymbol{\nabla}\varphi(\mathbf{x}) - \mathbf{v}(\varphi(\mathbf{x}))\varphi(\mathbf{x})) \cdot \hat{\mathbf{n}} \, \mathrm{d}\sigma + \bar{S}_i \, \Delta V_i = 0. \tag{2.11}$$

In Chapters 3 and 5 this kind of explicit dependency will be included when the final discretisation forms in two and three dimensions are discussed. But for now, these dependencies will continue to be suppressed to aid readability.

Returning to (2.10), if suitable approximations for the advective-diffusive flux and source terms can be found, and if these approximations involve only the values of φ at the nodes, then (2.10) can be converted into its discrete, algebraic equivalent

$$f_i(\boldsymbol{\varphi}) = 0. \tag{2.12}$$

This scalar function f_i is expressed as being dependent on the vector $\boldsymbol{\varphi}$. In practice the interpolation method usually ensures that each f_i is actually dependent only on a "local" set of the φ_j, but in general it is nonetheless dependent on the entire vector $\boldsymbol{\varphi}$. Each such equation for node $\mathbf{x}_i$ forms part of an algebraic system of equations

$$\mathbf{F}(\boldsymbol{\varphi}) = \mathbf{0}\,, \tag{2.13}$$

where

$$\mathbf{F} = (f_1, f_2, \ldots, f_N)^{\mathrm{T}}, \tag{2.14}$$

the solution of which yields the finite volume solution vector $\boldsymbol{\varphi}$. In general, owing to the dependence of the diffusivity $\mathbf{D}$ and velocity $\mathbf{v}$ on φ, (2.13) will be a nonlinear system of algebraic equations.

In Sections 2.2 and 2.3 a discussion of the two main components of the discretisation of (2.10) is presented: integration and interpolation. There-

after, in Sections 2.4 and 2.5 the solution strategy for the nonlinear system (2.13) is given, to yield the final numerical solution.

2.2 Integration

In order to apply (2.10), discrete approximations of the advective-diffusive flux and source terms must be computed. Both terms involve spatial integration of quantities whose analytic representation may be arbitrarily complicated or involve the unknown function φ. Thus, it is natural to turn to numerical methods of integration in forming these discretisations.

First consider the advective-diffusive flux component

$$\iint_{\sigma_i} (\mathbf{D}\boldsymbol{\nabla}\varphi \cdot \hat{\mathbf{n}} - \mathbf{v}\varphi) \, \mathrm{d}\sigma \tag{2.15}$$

from (2.10). Since the analytic solution φ is unknown, so too the integrand of (2.15) is unknown. Thus, numerical methods must be sought to approximate it. Fortunately, the control volumes V_i and their surfaces σ_i are, by construction, relatively simple geometric figures. In particular, the control volume surfaces σ_i are composed of line segments in two dimensions, and quadrilaterals in three dimensions (see Figure 2.2), enabling (2.15) to be expressed as

$$\iint_{\sigma_i} (\mathbf{D}\boldsymbol{\nabla}\varphi \cdot \hat{\mathbf{n}} - \mathbf{v}\varphi) \, \mathrm{d}\sigma = \sum_{\tau=1}^{Nf_i} \iint_{\sigma_{i\tau}} (\mathbf{D}\boldsymbol{\nabla}\varphi \cdot \hat{\mathbf{n}} - \mathbf{v}\varphi) \, \mathrm{d}\sigma \,, \tag{2.16}$$

where σ_i comprises Nf_i faces $\sigma_{i\tau},\ \tau = 1, \ldots, Nf_i$. The problem of approximating (2.15) is thus reduced to approximating each term

$$\iint_{\sigma_{i\tau}} (\mathbf{D}\boldsymbol{\nabla}\varphi \cdot \hat{\mathbf{n}} - \mathbf{v}\varphi) \, \mathrm{d}\sigma \tag{2.17}$$

of (2.16).

2.2.1 Midpoint rule

One commonly used approach to approximate (2.17) is to employ the midpoint rule

$$\iint_{\sigma_{i\tau}} (\mathbf{D}\boldsymbol{\nabla}\varphi \cdot \hat{\mathbf{n}} - \mathbf{v}\varphi)\, \mathrm{d}\sigma \approx [\mathbf{D}\boldsymbol{\nabla}\varphi \cdot \hat{\mathbf{n}} - \mathbf{v}\varphi]_{\mathbf{m}_{i\tau}} A_{i\tau} \tag{2.18}$$

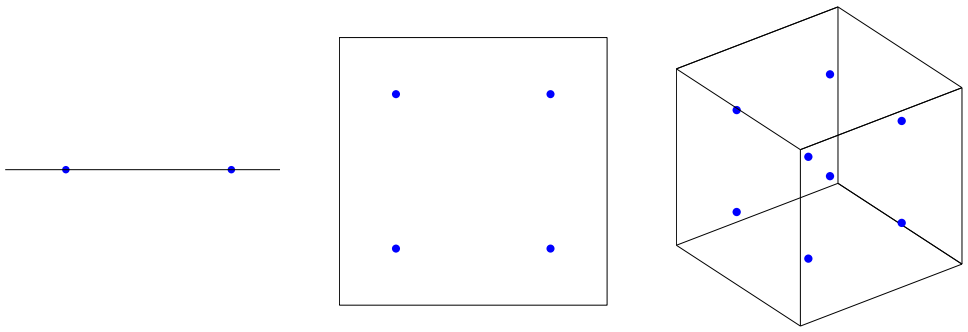
where $\mathbf{m}_{i\tau}$ is the midpoint of face $\sigma_{i\tau}$ and $A_{i\tau}$ is its area. If the value used for $[\mathbf{D}\boldsymbol{\nabla}\varphi \cdot \hat{\mathbf{n}} - \mathbf{v}\varphi]_{\mathbf{m}_{i\tau}}$ were exact, then this method would be accurate to second order [65].

Now consider the source component of (2.10). The representation of this quantity in terms of the control-volume averaged value $\bar{S}_i$ defined in (2.6) reflects the usual method of approximation: $\bar{S}_i$, is simply replaced by S_i, the value of the source term at node $\mathbf{x}_i$. This yields the approximation

$$\bar{S}\Delta V_i \approx S_i \Delta V_i \,. \tag{2.19}$$

In many applications the accuracy offered by quadrature schemes (2.18) and (2.19) is sufficient, and the methods are widely used. Nonetheless, there are some situations where a higher order method of quadrature is desired. One can consider (2.18) and (2.19) as one-point quadrature methods. To achieve higher accuracy, multiple-point quadrature methods may be used instead.

There are many popular numerical quadrature methods in one dimension that can be extended to apply in two and three dimensions. Generally though, the cost of such methods grows exponentially with dimension. For example, the one-dimensional, two-point quadrature method illustrated



(a) (b) (c)

Figure 2.3: "Two-point" quadrature in (a) one dimension; (b) two dimensions; (c) three dimensions.

in Figure 2.3(a) requires four points when extended to two dimensions, and eight when extended to three (Figures 2.3(b) and 2.3(c) respectively). Therefore it is important to employ a method of quadrature that can achieve high accuracy without requiring an excessive number of integrand evaluations, as these can be quite costly in the finite volume framework. *Gaussian quadrature* is one such method that has been successfully employed as part of the finite volume method (in [64, 79] for example) and is the method pursued in this thesis.

Note that when considering higher order methods of integration relating to the source term of (2.10), it is helpful to consider the original form

$$\bar{S}_i \Delta V_i = \iiint_{V_i} S \, \mathrm{d}V \tag{2.20}$$

as was derived in (2.3).

2.2.2 Gaussian quadrature

Gaussian quadrature [10, 21] in one dimension is a method of numerical quadrature that can be applied with an arbitrary number of integration points, or *abscissas*. Given a function g and a weighting function W over the interval $[a, b]$, p-point Gaussian quadrature uses the approximation

$$\int_a^b g(u)W(u)\,\mathrm{d}u \approx \sum_{j=1}^{p} w_j\, g(u_j). \tag{2.21}$$

The weights w_j and abscissas u_j are chosen so that the approximation is exact for polynomials up to degree $2p$. Specifically, the abscissas are precisely the roots of the corresponding orthogonal polynomial $P_p(u)$ over the interval $[a, b]$ with weighting function W. Tables of weights and abscissas for various forms of Gaussian quadrature are given in most numerical analysis texts (see [21] for example). In this thesis, *Gauss-Legendre quadrature* is used, taking $W(u) = 1$ and

$$P_p(u) = \frac{1}{2^p p!}\left[\frac{\mathrm{d}^p}{\mathrm{d}u^p}(u^2-1)^p\right], \tag{2.22}$$

are the *Legendre* polynomials. Abscissas and weights for two and three point Gauss-Legendre quadrature are given in Table 2.1. In terms of accuracy, it can be shown [21] that p-point Gaussian quadrature is $\mathcal{O}(h^{2p})$, where h is the length of the interval. The means of extending this method to two and three-dimensional settings in order to approximate (2.15) and (2.20) is discussed in Chapters 3 and 5 respectively.

2.3 Interpolation

The previous section dealt with approximating (2.10) using numerical quadrature. The methods outlined in that section require the evaluation

Table 2.1: Two and three-point Gauss-Legendre quadrature.

Points	Abscissas	Weights
2	$-\frac{\sqrt{3}}{3}$	1
	$\frac{\sqrt{3}}{3}$	1
3	$-\frac{\sqrt{3}}{\sqrt{5}}$	$\frac{5}{9}$
	0	$\frac{8}{9}$
	$\frac{\sqrt{3}}{\sqrt{5}}$	$\frac{5}{9}$

of the integrand at particular integration points. For the advective-diffusive flux component, these points lie on the control volume faces. For the source component, these are points within the control volume. In the case of the advective-diffusive flux component, the integrands are dependent on φ.

This dependence on φ poses somewhat of a problem, because when applying the finite volume method, the only points at which values of φ are available are the mesh nodes themselves. As was discussed in Section 2.1, the method is based on constructing control volumes around these nodes, and solving each discretised equation (2.10) for the values φ_i. These φ_i are the only unknowns to appear in equation (2.13) and as such are the only values for φ that can be used in the discretisation.

Diffusive flux approximations, regardless of whether they are computed using the midpoint rule, Gaussian quadrature, or some other quadrature method, not only require values for φ, but its gradient $\boldsymbol{\nabla}\varphi$ as well. In this case, some form of interpolation must be employed that takes the φ_i values at the nodes and produces interpolated values of both φ and $\boldsymbol{\nabla}\varphi$ at arbitrary points on the control volume surfaces.

Throughout this section it is assumed that all methods of interpolation are element-based (see Sections 2.3.5 and 2.3.6 for why element-based schemes are preferred). That is, each element in the mesh E_i has associated with it an interpolating function s_i that approximates φ throughout that element.

Furthermore, to simplify the notation, local numbering of element vertices is used throughout. That is, the vertices of E_i are labelled $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \ldots$ irrespective of their global position in terms of the vector $\boldsymbol{\varphi}$ (2.8).

Whatever method is used to construct s_i, its purpose is to approximate φ over element E_i:

$$s_i(\mathbf{x}) \approx \varphi(\mathbf{x}), \quad \mathbf{x} \in E_i \,. \tag{2.23}$$

Furthermore, if s is differentiable, then it may also be used to approximate the gradient $\boldsymbol{\nabla}\varphi$:

$$\boldsymbol{\nabla} s_i(\mathbf{x}) \approx \boldsymbol{\nabla}\varphi(\mathbf{x}), \quad \mathbf{x} \in E_i \,. \tag{2.24}$$

With the interpolating function s_i now in place, all occurrences of $\varphi(\mathbf{x})$ and $\boldsymbol{\nabla}\varphi(\mathbf{x})$ throughout the integration formulas in Section 2.2 may be replaced within element E_i by $s_i(\mathbf{x})$ and $\boldsymbol{\nabla} s_i(\mathbf{x})$ respectively. It remains to be determined which particular methods of interpolation might be suitable within the finite volume framework.

2.3.1 Shape functions

A popular method of interpolation used in the finite volume method is borrowed from finite element theory [3, 24, 27, 41, 42, 65]. The method of *shape functions* uses the vertices of a triangular, or tetrahedral element, to compute a linear interpolation of φ, satisfying

$$s_i(\mathbf{x}_j) = \varphi_j, \quad j = 1, \ldots, n \,, \tag{2.25}$$

where n, the number of element vertices, is three for two-dimensional, triangular elements, and four for three-dimensional, tetrahedral elements. Since s_i is linear, (2.25) uniquely determines the shape function interpolant for

element E_i.

In the finite element literature, this interpolant s_i is traditionally expressed as a linear combination of so-called shape functions, $N_j(\mathbf{x})$ [15]:

$$s_i(\mathbf{x}) = \sum_{j=1}^{n} \varphi_j N_j(\mathbf{x}) \,, \tag{2.26}$$

where the N_j depend only on the geometric properties of element E_j, and not on the values φ_j. Formulas for the N_j in two and three dimensions are given in Chapters 3 and 5 respectively

Shape function interpolations also allow approximation of the gradient, with $\boldsymbol{\nabla} s_i$ given by

$$\boldsymbol{\nabla} s_i = \sum_{j=1}^{n} \varphi_j \boldsymbol{\nabla} N_j(\mathbf{x}) \tag{2.27}$$

which is constant throughout the element.

Finite volume methods that employ shape functions as a means of interpolation are often called control volume-finite element, or CV-FE, methods. The most obvious limitation of the approach is the use of a constant gradient approximation within each element. For problems where the gradient does not vary greatly this approach can be satisfactory. However, in many problems, there are regions where the gradient can vary significantly and using a constant gradient approximation may not adequately capture this behaviour [41, 42].

In such cases, the only way to improve upon the accuracy of this method is to employ a very fine mesh, which can be costly in terms of memory requirements and computational time. Ideally, what is needed in these cases is a more sophisticated method of interpolation that gives accurate function and gradient approximations even when using a coarse mesh.

Some work has been done (see for example [41, 43, 44, 46]) on extending

CV-FE methods of flux approximation to second and higher orders of accuracy. Typically, a least-squares approach is used to formulate a higher-degree interpolating polynomial. Rather than extend or improve upon this work, it was decided in this thesis to investigate alternative methods of interpolation that might also yield high-order gradient approximations.

2.3.2 Scattered data interpolation

To go beyond the accuracy offered by shape functions, vertices other than the element's own may be used in forming the interpolant. The fitting of an interpolant to an arbitrary set of scattered nodes is a problem known as *scattered data interpolation*: given the set of nodes $\mathbf{x}_1,\ \mathbf{x}_2, \ldots,\ \mathbf{x}_n$, and the corresponding function values $\varphi_1,\ \varphi_2, \ldots,\ \varphi_n$ at these points, compute a function that interpolates these values.

Scattered data interpolation is required in many different areas of mathematics, so it is not surprising that there are numerous methods documented in the literature (see [30] for a comparison of many of these). However, not all of these methods are applicable in the context of the finite volume method. There are certain requirements that need to be met in order for a method of scattered data interpolation to be successful in the finite volume framework. The following four requirements have been identified as essential:

1. *It must be possible to obtain a gradient approximation from an interpolation of function values.*

To compute the advective-diffusive flux using (2.10), values for $\boldsymbol{\nabla}\varphi$ are needed. The interpolation itself though can involve only values of φ at the nodes. Therefore, it must be possible to obtain $\boldsymbol{\nabla}\varphi$ from an interpolation

based on φ only. This requires that the interpolant must itself be at least once differentiable.

2. *The method must achieve high accuracy on a coarse mesh.*

The method of shape functions discussed in Section 2.3.1 has as its major downside the inability to adequately handle problems where the gradient can vary significantly over a small region. One solution posed in that section was to use a fine mesh in such cases—an expensive option both in terms of memory requirements and computational effort. If instead of refining the mesh, a more sophisticated method of interpolation is used, involving more nodes than just the element's own vertices, this lost accuracy may be able to be recovered while still employing a coarse mesh.

3. *The method must be flexible with regard to the number of nodes used in the interpolation.*

Referring again to Section 2.3.1, the method of shape functions uses a fixed number of nodes (three nodes for triangular elements, four nodes for tetrahedral elements) in computing the interpolant. In order to improve upon the accuracy offered by shape functions, it may be necessary to involve more than just these nodes in each element's interpolation.

One might think that the ideal scenario would be to use every node in the mesh and produce a single, global interpolant. However this approach has several problems relating to efficiency. First, it makes the construction and evaluation of the interpolant very costly, particularly for meshes with large numbers of nodes. Second, it eliminates the sparsity of the underlying Jacobian matrix. Section 2.4 covers this aspect in more detail, but for now, note that the method used to solve (2.13) is most effective when this Jacobian matrix is sparse.

Thus, the conclusion is that utilising local interpolants is preferable to a single, global interpolant in the context of the finite volume method. When using local interpolants, there must still be a balance struck between using too few nodes per interpolation and suffering poor accuracy, and using too many nodes per interpolation and suffering reduced efficiency. Ideally this balance should be adjustable, which requires an interpolation method that is flexible with respect to the number of nodes that can be used in its construction.

4. *It must be possible to efficiently obtain an interpolation based on a set of shifted values.*

In Section 2.4, the solution of the nonlinear system (2.13) using an inexact Newton method will be discussed. It will be shown how the discrete directional derivative

$$\mathbf{J}(\boldsymbol{\varphi})\mathbf{v} \approx \frac{\mathbf{F}(\boldsymbol{\varphi} + \varepsilon\mathbf{v}) - \mathbf{F}(\boldsymbol{\varphi})}{\varepsilon} \tag{2.28}$$

where $\mathbf{v}$ is an arbitrary vector and ε is a small parameter, is used in place of an explicit Jacobian-vector product [9]. Many such directional derivatives must be evaluated in a single iteration of an inexact Newton method, with different values of $\mathbf{v}$ required at each iteration. It would not be practical if every such evaluation required every element's interpolation to be completely recomputed. Rather, it must be possible, given an initial interpolation based on

$$\boldsymbol{\varphi} = (\varphi_1, \varphi_2, \ldots, \varphi_N)^T,$$

to efficiently compute the interpolation based on

$$\boldsymbol{\varphi} + \varepsilon\mathbf{v} = (\varphi_1 + \varepsilon v_1, \varphi_2 + \varepsilon v_2, \ldots, \varphi_N + \varepsilon v_N)^T.$$

Note that the method of shape functions, outlined in Section 2.3.1, satisfies only the first and fourth of these requirements. In the next section, it will be shown that the method of *radial basis functions* [70] meets them all.

2.3.3 Radial basis functions

The method of radial basis functions (RBFs) is a method of scattered data interpolation applicable in any dimension. It can be thought of as an extension to higher dimensions of one-dimensional spline functions [70].

In one dimension, a linear spline function s, interpolating the values $\varphi_1, \varphi_2, \ldots, \varphi_n$ at nodes $x_1, x_2, \ldots, x_n$ is simply a function connecting the points (x_1, φ_1), (x_2, φ_2), ..., (x_n, φ_n) with line segments. It may be defined in a piecewise manner, or it may be defined as follows [70]:

$$s(x) = \sum_{j=1}^{n} \lambda_j |x - x_j| + c_0 + c_1 x, \quad x_1 \leq x \leq x_n \,, \tag{2.29}$$

where the conditions

$$s(x_j) = \varphi_j, \quad j = 1, 2, \ldots, n \tag{2.30}$$

and

$$\sum_{j=1}^{n} \lambda_j = \sum_{j=1}^{n} \lambda_j x_j = 0 \tag{2.31}$$

determine the λ_i, c_0 and c_1.

The conditions (2.30) are the interpolation conditions, specifying that s and φ agree at the nodes. Condition (2.31) is more interesting. There are several ways of viewing these equations [70]. First, there are $n+2$ unknowns in this interpolation: c_0, c_1 and the λ_i. Conditions (2.30) impose only n constraints on these unknowns. Equation (2.31) simply absorbs these additional

two degrees of freedom.

A second interpretation is that (2.31) is imposing orthogonality conditions on the solution vector $(\lambda_1, \lambda_2, \ldots, \lambda_n)^T$ such that $|s(x)| = \mathcal{O}(|x|)$ for large $|x|$ [70].

Finally, (2.31) can be interpreted simply as the set of conditions that along with (2.30) ensures that (2.29) is in fact the unique, one-dimensional linear spline connecting the points (x_1, φ_1), (x_2, φ_2), ..., (x_n, φ_n).

Generalising (2.29), (2.30) and (2.31) to $\mathbb{R}^m$ yields the method of radial basis functions [70]. Equation (2.29) becomes

$$s(\mathbf{x}) = \sum_{j=1}^{n} \lambda_j \phi(\|\mathbf{x} - \mathbf{x}_j\|) + \sum_{k=0}^{m} c_k q_k(\mathbf{x}) \,, \tag{2.32}$$

with the conditions (2.30) and (2.31) becoming

$$s(\mathbf{x}_j) = \varphi_j, \quad j = 1, 2, \ldots, n \tag{2.33}$$

and

$$\sum_{j=1}^{n} \lambda_j \, q_k(\mathbf{x}_j) = 0, \quad k = 0, 1, \ldots, m \,, \tag{2.34}$$

where the q_k form the standard basis for the space of m-variate linear polynomials (in two dimensions for example, $q_0 = 1, q_1 = x, q_2 = y$, while in three dimensions, $q_0 = 1$, $q_1 = x$, $q_2 = y$, $q_3 = z$). The function ϕ of (2.32) is known as a radial basis function, and is where the method gets its name. Equivalence with the one-dimensional linear spline requires taking $\phi(r) = r$, but as a generalisation ϕ may be selected from a wide range of choices, some of which are listed in Table 2.2 [70].

Note carefully the difference between the symbols φ and ϕ: φ represents the conserved physical quantity solved for with the finite volume method,

Table 2.2: Radial basis functions.

RBF	Name
$\phi(r) = r$	Linear
$\phi(r) = r^3$	Cubic
$\phi(r) = r^2 \log(r)$	Thin plate spline
$\phi(r) = e^{-c^2 r^2}$	Gaussian
$\phi(r) = \sqrt{c^2 + r^2}$	Multiquadric
$\phi(r) = 1/\sqrt{c^2 + r^2}$	Inverse multiquadric

while ϕ is a radial basis function. Despite possible confusion, this notation is adopted because it is standard in the literature.

The first three choices of ϕ in Table 2.2 are examples of piecewise smooth RBFs, while the second three are examples of infinitely smooth RBFs [53]. The infinitely smooth RBFs also involve a free parameter c^2 which must be selected by the user. The value of this parameter can affect both the accuracy and the conditioning of the resulting interpolation [53, 72].

The different properties possessed by these basis functions mean that some are more suited to particular applications than others. For example, the rapid decay of the Gaussian function as r increases makes it suitable for neural networking applications [52, 55]. Surface fitting applications in two and three dimensions often use the thin plate spline and linear basis functions respectively [12], these being analogous to cubic splines in one dimension [70].

In this thesis the multiquadric basis function is used for all RBF interpolations. There are two reasons it is preferred over the other possible choices. First, it has been found in the numerical experiments of this thesis, and elsewhere [30, 36], to consistently offer the best accuracy of all competing methods (RBF-based or not). Second, the value of c^2 is not so crucial with this choice, as the method is quite stable with respect to this parameter (which is not the case with the inverse multiquadric for example) [30]. In the context of the finite volume method, this is important, as an inappropriate

choice of c^2 can hinder the convergence of the method (see Sections 4.4.3 and 6.4.2).

Forming the linear system

Conditions (2.33) and (2.34) can be written in matrix form as follows [70]:

$$\begin{pmatrix} \mathbf{\Phi} & \mathbf{P} \\ \mathbf{P}^T & \mathbf{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{\lambda} \\ \mathbf{c} \end{pmatrix} = \begin{pmatrix} \mathbf{f} \\ \mathbf{0} \end{pmatrix} \tag{2.35}$$

with the various matrix and vector quantities defined entry-wise by

$$\begin{aligned} (\mathbf{\Phi})_{ij} &= \phi\left(\|\mathbf{x}_i - \mathbf{x}_j\|\right), && i = 1, \ldots, n,\ j = 1, \ldots, n \\ (\mathbf{P})_{i,k+1} &= q_k(\mathbf{x}_i), && i = 1, \ldots, n,\ k = 0, \ldots, m \\ (\boldsymbol{\lambda})_i &= \lambda_i, && i = 1, \ldots, n \\ (\mathbf{c})_{k+1} &= c_k, && k = 0, \ldots, m \\ (\mathbf{f})_i &= \varphi_i, && i = 1, \ldots, n. \end{aligned}$$

This is a square matrix system, as there are as many conditions as there are unknowns. It can be shown that for any choice of ϕ from Table 2.2, the interpolation matrix

$$\mathbf{\Lambda} = \begin{pmatrix} \mathbf{\Phi} & \mathbf{P} \\ \mathbf{P}^T & \mathbf{0} \end{pmatrix} \tag{2.36}$$

is nonsingular, ensuring the existence of a unique interpolating function s [70].

Solution of linear system

Despite the theoretical guarantee of nonsingularity, it is well known that in practice the matrix $\mathbf{\Lambda}$ (2.36), and in particular the submatrix $\mathbf{\Phi}$, can be extremely ill-conditioned [70]. Even with relatively small systems (say

25 nodes) care must be taken that this ill-conditioning does not introduce significant error in the computed solution.

Some authors [4, 5], consider this ill-conditioning not as a limitation of the method itself, but as a consequence of the underlying vector space basis. Methods have been developed [4] to utilise alternative bases in the solution process, however the strategy is best suited to problems involving "many, many nodes" [5]. Within the finite volume framework, RBF interpolations are local, typically using 25 nodes at the most. Under these circumstances there are better alternatives for dealing with ill-conditioning.

With the small number of local nodes used in each element's RBF interpolant, a direct method of solving (2.35) is applicable. In the field of direct linear solvers, the *singular value decomposition* is renowned for its robustness on very ill-conditioned (even to the point of being numerically singular) matrices [32, 33, 34]. The singular value decomposition (SVD) decomposes a matrix into the product of an orthogonal matrix $\mathbf{U}$, a diagonal matrix $\mathbf{W}$ and the transpose of an orthogonal matrix $\mathbf{V}$ [32]:

$$\mathbf{\Lambda} = \mathbf{U}\mathbf{W}\mathbf{V}^T. \tag{2.37}$$

This decomposition generates a substantial amount of information, including orthonormal bases for the four fundamental subspaces of $\mathbf{\Lambda}$ and the so-called *singular values* w_i, the diagonal entries of $\mathbf{W}$, that determine the conditioning of the matrix [32]. Matrices that are numerically rank-deficient (singular) are characterised by having one or more singular values less than the machine precision.

Using (2.37), the solution to the system

$$\mathbf{\Lambda}\mathbf{y} = \mathbf{b} \tag{2.38}$$

can be written as

$$\mathbf{y} = \mathbf{V}\mathbf{W}^{-1}\mathbf{U}^T\mathbf{b}. \tag{2.39}$$

However, applying (2.39) directly is of no benefit over a more conventional LU decomposition. In particular, the presence of very small singular values has the potential to introduce catastrophic rounding error into the solution when these values are inverted [33, 71].

To see how the singular value decomposition can overcome this problem, it is helpful to consider (2.37) and (2.39) in their alternative forms

$$\mathbf{\Lambda} = \sum_{i=1}^{\dim(\mathbf{\Lambda})} w_i \mathbf{u}_i \mathbf{v}_i^T \tag{2.40}$$

and

$$\mathbf{y} = \sum_{i=1}^{\dim(\mathbf{\Lambda})} \frac{1}{w_i} \mathbf{v}_i \mathbf{u}_i^T \mathbf{b}\,. \tag{2.41}$$

Equation (2.40) shows how $\mathbf{\Lambda}$ may be expressed as a linear combination of rank-one matrices formed by taking the outer products of the left and right singular vectors $\mathbf{u}_i$ and $\mathbf{v}_i$. Equation (2.41) highlights the numerical difficulties in using this expression to invert $\mathbf{\Lambda}$ in the presence of small singular values: the division by these small values will magnify any accumulated roundoff error in the solution. If the values are of the order of the machine precision then every significant digit in the solution has the potential to be eroded [32, 71].

Suppose, as is the usual practice, that the singular values are ordered so that $w_1 \geq w_2 \geq \cdots \geq w_{\dim(\mathbf{\Lambda})}$. Then (2.40) shows that the contributions made to $\mathbf{\Lambda}$ by each term $w_i \mathbf{u}_i \mathbf{v}_i^T$ are nonincreasing with i. In fact, if the last $\dim(\mathbf{\Lambda}) - k$ singular values are zero, then (2.40) may be truncated at $i = k$,

giving

$$\mathbf{\Lambda} = \sum_{i=1}^{k} w_i \mathbf{u}_i \mathbf{v}_i^T \,. \tag{2.42}$$

The *truncated singular value decomposition* (TSVD) [33] exploits this fact by truncating (2.40) at $i = k^*$,

$$\mathbf{\Lambda} \approx \sum_{i=1}^{k^*} w_i \mathbf{u}_i \mathbf{v}_i^T \tag{2.43}$$

where the w_i satisfy

$$w_i \leq w_1 \epsilon_{\text{mach}}, \quad i > k^* \tag{2.44}$$

where ϵ_{mach} is the machine precision. This produces an approximation of $\mathbf{\Lambda}$ that is accurate to within ϵ_{mach} in relative two-norm [71], but does not suffer from numerical instability when inverting. This truncated singular value decomposition may then be used to solve (2.38):

$$\mathbf{y} \approx \sum_{i=1}^{k^*} \frac{1}{w_i} \mathbf{v}_i \mathbf{u}_i^T \mathbf{b} \tag{2.45}$$

without the associated problems with roundoff error.

Note that this approach of computing (and saving) the singular value decomposition of $\mathbf{\Lambda}$ satisfies property four of Section 2.3.2. The only impact of φ in the matrix system (2.35) is in the right hand side. The matrix $\mathbf{\Lambda}$ itself depends only on the underlying geometry. Therefore, additional right hand sides, corresponding to shifted values of φ may be efficiently processed using (2.43), without the need to recompute the decomposition.

Gradient approximation

It is shown in [70] that the function s of (2.32) is infinitely differentiable for any choice of ϕ from Table 2.2 on any region of $\mathbb{R}^m$ that excludes the interpolation points. Thus from (2.32), $\boldsymbol{\nabla} s$ is given by

$$\boldsymbol{\nabla} s = \sum_{j=1}^{n} \lambda_j \left(\frac{\mathbf{x} - \mathbf{x}_j}{\|\mathbf{x} - \mathbf{x}_j\|} \right) \phi'(\|\mathbf{x} - \mathbf{x}_j\|) + \sum_{k=1}^{m} c_k \mathbf{e}_k \,, \tag{2.46}$$

where $\mathbf{e}_k$ is the kth standard basis vector for $\mathbb{R}^n$. The restriction of not applying (2.46) at the interpolation points is not a concern in the finite volume framework, as all gradient evaluations take place over control volume faces and not at nodes.

2.3.4 Choosing the nodes in an RBF interpolation

In the previous section, the method of radial basis functions, as it applies to a single RBF interpolant s was discussed. When applied in the finite volume framework, there is actually a different interpolant, s_i, used for every element in the mesh. In this sense the approach is analogous to that taken with shape functions. However, with shape functions, the number of nodes used in the interpolation is fixed at three (for triangular elements). With an RBF interpolation, one is free to choose whichever and however many nodes to use in each element's interpolation. In this section, possible strategies for choosing these nodes are discussed.

Global nodes

In Section 2.3.2 reasons were given why it is unwise to fit a single, global interpolant to cover the entire mesh. The first of these reasons is the large

cost associated with constructing and then evaluating the resulting interpolant. In the case of radial basis functions, this cost is the solution of (2.35) and the evaluation of (2.32) and (2.46). The second reason is the dense Jacobian matrix that this approach generates (see Section 2.4). Therefore a global node-selection approach is not considered to be viable within the finite volume method framework.

Selected global nodes

It is tempting to try to overcome the problems mentioned in the previous paragraph by fitting a single, global interpolant that uses only a small sample of nodes in its construction. Such an approach is described in [12], where a greedy algorithm is used to select a subset of nodes that results in the desired accuracy. Unfortunately approaches of this kind are not applicable to the finite volume method, since the interpolation is being constructed as part of the finite volume discretisation. The very point of the discretisation is to replace the original PDE by the algebraic system of equations (2.13), involving all N nodes in the mesh. The solution of this system then gives the approximate value of φ at each node. Excluding nodes from the RBF interpolation is equivalent to excluding them from the discretisation altogether, meaning they are no longer involved in any way in the computation of the numerical solution.

Local nodes

Employing element-wise RBF interpolations requires choosing, for each element, which nodes are to be involved in that element's interpolation. The first point to note is that using just the element's own vertices produces an RBF interpolant identical to the corresponding shape function interpolant.

To see that this is so, recall that the shape function interpolant is simply a linear polynomial in $\mathbf{x}$. The second term in (2.32) is also a linear polynomial in $\mathbf{x}$, so that the shape function interpolant can be thought of as the special case of an RBF interpolant with $n = m + 1$, and

$$\lambda_1 = \lambda_2 = \ldots = \lambda_n = 0\,. \tag{2.47}$$

Now, the shape function polynomial satisfies the interpolation constraints (2.30), and (2.47) satisfies the orthogonality constraints (2.31), so by uniqueness the shape function interpolant and the RBF interpolant are equivalent for $n = m + 1$. The fact that shape functions can be considered a form of "simplification" of the more general RBF method has implications for the Jacobian-free Newton-Krylov method discussed in Section 2.4.

To achieve higher accuracy with radial basis functions than is possible with shape functions, it is therefore necessary to include more than just the element's own vertices in the interpolation. How many nodes are used, and how they are selected will play a part in determining the accuracy and efficiency of the resulting interpolant. In this thesis, the method outlined in Algorithm 1 below is used to choose a pre-determined number of nodes for each interpolation. First, the element's own vertices are included in the local node set. Then, the vertices of its neighbouring elements that have not yet been included are included, followed by the yet-to-be-included vertices of the neighbours' neighbours, and so on.

Algorithm 1 (Choosing the nodes in an RBF interpolation)**.**
INPUT:
`m` : Mesh
`i`: Identity of element
`size`: Size of point cloud desired

```
OUTPUT:
    ℓ_i: Local point cloud consisting of size nodes surrounding element i
BEGIN:
    ℓ_i = {}
    q = [i]
    while not q.empty()
        e = m.element(q.pop())
        for j = 1 to e.nodes()
            if size > 0 and ℓ_i.insert(e.node(j))
                size = size - 1
            end if
        end for
        for j = 1 to e.nodes()
            if size > 0 and e.has_neighbour(j)
                q.push_back(e.neighbour(j))
            end if
        end for
    end while
```

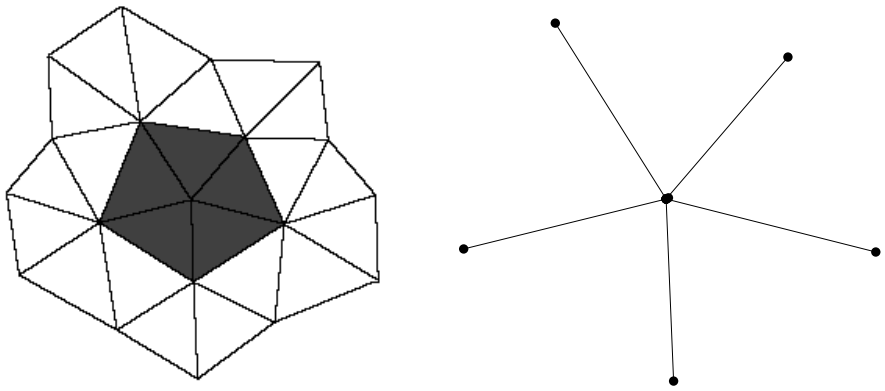
This algorithm has the desirable property of assuring that, for the most part, the element in question lies in the centre of its resulting "cloud of points". This is important, because it is well known that radial basis function interpolants are poorest when evaluated near the boundary of this cloud [28]. Algorithm 1 ensures that for interior elements (those not lying on or near the boundary of the mesh), the cloud will be mostly symmetric, with the element in the centre. For elements on or near the mesh boundary, the algorithm will tend to skew the cloud towards the interior as it encounters boundary elements that prevent it from searching further in that direction. Some ideas for overcoming this potential loss of accuracy near boundaries are presented in Section 2.3.7.

2.3.5 Stencils

In this section *stencil* obtained by using element-wise radial basis functions is compared with that of shape functions. To understand what is meant by stencil, consider the section of mesh (in two dimensions, for simplicity) illustrated in Figure 2.4(a). Suppose shape function interpolations are being used to evaluate the component function $f_i(\boldsymbol{\varphi})$ of (2.12) at the innermost node. This node lies at the intersection of five triangular elements. When integrating around the control volume that encloses this node, interpolants will be evaluated at points within each one of these five elements. The combined set of all the nodes involved in each of these five element-wise interpolations is called the stencil. This particular stencil is illustrated in Figure 2.4(b).

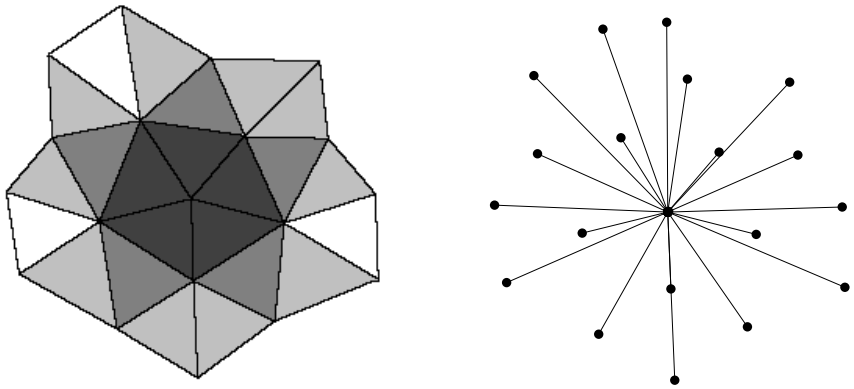
When using RBFs instead of shape functions, the additional nodes in each interpolation mean that the resulting stencil is larger. Figure 2.5(a) illustrates the possible extent of the RBF point clouds for the same five elements. Each cloud incorporates its own set of local nodes, and the union of each set produces the stencil shown in Figure 2.5(b).

The reader may be thinking at this point that a *node-wise* RBF interpolation strategy would yield the benefits of increased accuracy while preserving the shape function stencil. In this strategy, each *node* in the mesh would have an associated RBF interpolant, using precisely the point cloud dictated by the corresponding shape function stencil (like the one shown in Figure 2.4(b)). Unfortunately this approach fails in practice, for reasons that will become apparent in the next section.



(a) (b)

Figure 2.4: (a) node selection and (b) stencil when using shape functions.



(a) (b)

Figure 2.5: (a) node selection (shading shows distance from centre) and (b) stencil when using radial basis functions.

2.3.6 Ensuring consistent flux

One of the advantages of the finite volume method over other methods such as the finite difference method and the finite element method is its conservative nature. Most real-world applications of partial differential equations derive from underlying physical conservation laws. Equation (2.9) for example can derive from the law of conservation of mass or the conservation of energy. Equation (2.10) is a mathematical statement dictating in the steady state that the flux through the surface of any control volume is equal to the net production within it. Formulated correctly, the finite volume method will preserve this conservation law, even at the discrete level.

For this to be so, the discretisation method must ensure that for any two neighbouring control volumes, the flow through their shared face is consistent. That is, the flow as measured from the first control volume into the second through the adjoining control volume face must be the negative of the flow as measured from the second into the first.

Figure 2.6 shows how this requirement is met when using element-wise interpolants. The flux between the two adjoining control volumes is computed as an integral over their shared control volume faces. Along those faces, all interpolated values of φ and $\boldsymbol{\nabla}\varphi$ are the same, whether computed from the upper node's point of view, or from the lower's. The reason for this is that the points of evaluation on either side of the control volume face lie within the same element and therefore use the same interpolant.

The corresponding figure when using node-wise interpolants reveals the problem with this approach. In Figure 2.7, the interpolations used to compute φ and $\boldsymbol{\nabla}\varphi$ from above the control volume face are different to those used below it. As a result, the conservative nature of the method is lost.

One can correct this problem by averaging the interpolated values from ei-

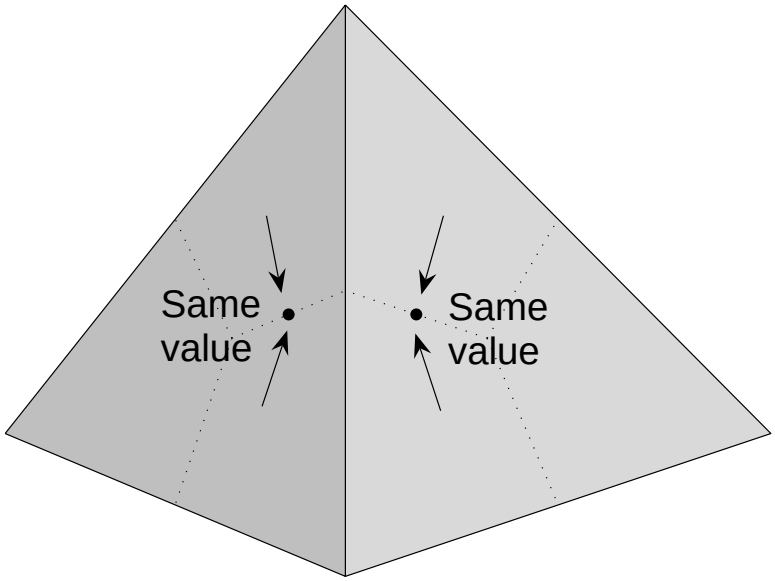


Figure 2.6: Computing fluxes using element-wise interpolation. Shaded regions represent different interpolants.

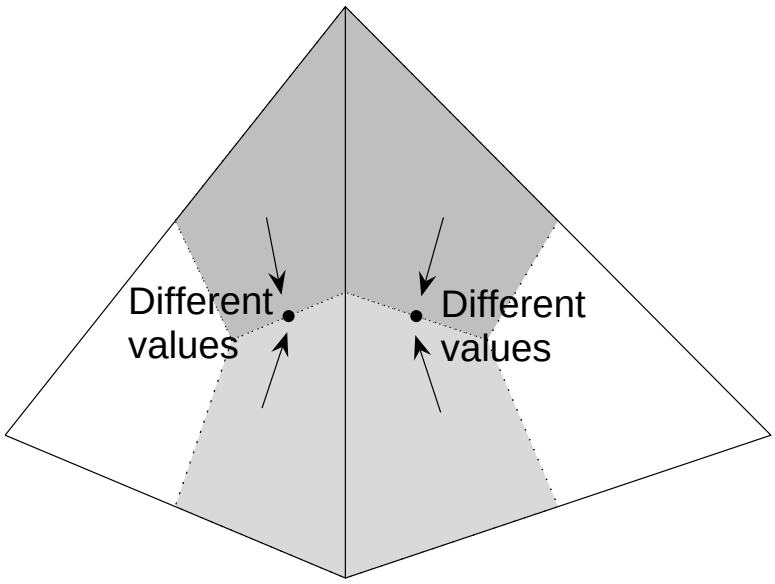


Figure 2.7: Computing fluxes using node-wise interpolation. Shaded regions represent different interpolants.

ther side of the control volume face. This overcomes the loss of conservatism, but at the cost of once again losing consistency with the shape function stencil. Faced with the fact that conservatism is lost in either case, in this thesis all interpolations are carried out element-wise, be they shape function-based or RBF-based.

2.3.7 Boundary accuracy

It is well known that the accuracy of radial basis function interpolations is poorest near the boundaries of the set of interpolated nodes. This problem is addressed in some detail in [28] with several modifications presented that attempt to overcome this limitation. It should be noted however, that the element-wise approach advocated throughout this thesis goes a long way to overcoming this problem of low accuracy near boundaries. Fitting local RBF interpolants ensures that points of evaluation occur in the interior of point clouds, and not on or near the boundaries. The only exception is when the element itself lies on or near the mesh boundary, in which case the resultant point cloud, selected according to Algorithm 1, may be skewed towards the interior of the mesh.

Two of the modifications to the basic RBF method presented in [28] are applicable in the context of a finite volume method. They are presented here for completeness, however their use in the finite volume method is not recommended, for reasons that will be discussed presently.

Higher degree polynomial term

The first applicable modification presented in [28] is the inclusion of a higher-degree polynomial term in the RBF interpolation. In this case, equation (2.32)

is replaced by

$$s(\mathbf{x}) = \sum_{j=1}^{n} \lambda_j \phi(\|\mathbf{x} - \mathbf{x}_j\|) + \sum_{k=0}^{\dim(\pi_p^m)} c_k q_k(\mathbf{x}) \tag{2.48}$$

where q_k is now the kth standard basis polynomial for the space π_p^m of all m-variate polynomials of degree less than or equal to p. The corresponding orthogonality conditions (2.34) are extended accordingly,

$$\sum_{j=1}^{n} \lambda_j \, q_k(\mathbf{x}_j) = 0, \quad k = 0, 1, \ldots, \dim(\pi_p^m). \tag{2.49}$$

Although results in [28] show that this modification can improve the boundary accuracy of certain one-dimensional problems, it has not been found to be effective in the context of two and three-dimensional finite volume methods. Additionally, it sacrifices the equivalence between the shape function interpolant and the RBF interpolant with $n = m + 1$ (Section 2.3.4), so is not recommended.

Not a Knot

The second modification from [28] that could be applied in the finite volume framework is the so-called *Not a Knot* or NaK condition. The idea behind this technique is that the $\mathbf{x}_i$ at which condition (2.33) is imposed need not be the same as the $\mathbf{x}_j$ used in the evaluation of (2.32). In other words, the *nodes* $\mathbf{x}_i$ at which the interpolation conditions are imposed, need not be the same as the *centres* $\tilde{\mathbf{x}}_j$ that are involved in the actual evaluation of the interpolant.

With this convention, and incorporating the higher degree polynomial

modification just discussed, equations (2.32) to (2.34) become

$$s(\mathbf{x}) = \sum_{j=1}^{n} \lambda_j \phi(\|\mathbf{x} - \tilde{\mathbf{x}}_j\|) + \sum_{k=0}^{\dim(\pi_p^m)} c_k q_k(\mathbf{x}) \tag{2.50}$$

with

$$s(\mathbf{x}_j) = \varphi_j, \quad j = 1, 2, \ldots, n \tag{2.51}$$

and

$$\sum_{j=1}^{n} \lambda_j \, q_k(\mathbf{x}_j) = 0, \quad k = 0, 1, \ldots, \dim(\pi_p^m). \tag{2.52}$$

In [28], the NaK approach of moving the outermost centres from inside to outside the domain resulted in some improvement in boundary accuracy for two-dimensional problems. But this was for interpolations based on 200 nodes, rather than the 25 or so nodes used in each RBF interpolation in the finite volume context. There does not appear to be any benefit in applying this method for small, local interpolations as are used in the finite volume method.

2.4 Solution of nonlinear system

The partial differential equation (2.9) is transformed into its finite volume equivalent (2.10) through the process of discretisation. In Sections 2.2 and 2.3, methods were presented that allow the advective-diffusive flux and source components of (2.10) to be computed. The values so computed depend on the values φ_i at the mesh nodes. Each discretised equation (2.12) is thus a single, nonlinear algebraic equation for the vector $\boldsymbol{\varphi} = (\varphi_1, \varphi_2, \ldots, \varphi_N)^{\mathrm{T}}$. The particular form that each component function f_i takes will be discussed in Chapters 3 and 5, while this section will focus on methods for finding the

solution to the nonlinear system of equations

$$\mathbf{F}(\boldsymbol{\varphi}) = \mathbf{0}\,. \tag{2.53}$$

2.4.1 Jacobian-free Newton-Krylov method

The basic Newton method for solving (2.53) is the sequence of iterations

$$\mathbf{J}(\boldsymbol{\varphi}^{(n)})\ \boldsymbol{\delta\varphi}^{(n)} = -\mathbf{F}(\boldsymbol{\varphi}^{(n)})\,, \tag{2.54}$$

$$\boldsymbol{\varphi}^{(n+1)} = \boldsymbol{\varphi}^{(n)} + \boldsymbol{\delta\varphi}^{(n)} \tag{2.55}$$

where

$$\mathbf{J} = \frac{\partial f_i}{\partial \varphi_j} \tag{2.56}$$

is the Jacobian matrix.

In row i of the Jacobian matrix, the nonzero values correspond to those nodes that contribute to the evaluation of the nonlinear component function f_i. This, in turn, is determined by the local point clouds used in the interpolations by the elements that share node $\mathbf{x}_i$. Denote by $\ell_1, \ell_2, \ldots, \ell_p$ the local point cloud sets for each of the elements sharing node i. Then the following can be said about the nonzero entries in the Jacobian matrix:

$$\mathbf{J}_{ij} \neq 0 \text{ iff } \mathbf{x}_j \in \ell_1 \cup \ell_2 \cup \ldots \cup \ell_p\,. \tag{2.57}$$

One consequence of using radial basis functions rather than shape functions in the finite volume discretisation is the increase in density of the Jacobian matrix. This increase in density is due to the increase in the number of nodes in each local point cloud set ℓ_i. The computational and storage expense in computing this Jacobian matrix can be considerable when the problem is

large. Fortunately, variations on the basic Newton method have been developed that avoid the need to compute this matrix explicitly. *Jacobian-free Newton-Krylov methods* [9, 49, 50, 51] offer a way to solve (2.53) without the need to ever explicitly compute $\mathbf{J}$.

Typically in such methods, (2.54) is solved only approximately, so that

$$\|\mathbf{F}(\boldsymbol{\varphi}) + \mathbf{J}(\boldsymbol{\varphi})\ \boldsymbol{\delta\varphi}^{(n)}\| \leq \eta_n \|\mathbf{F}(\boldsymbol{\varphi})\| \tag{2.58}$$

and η_n is known as a *forcing term* [20].

An iterative linear solver is used to compute the Newton step $\boldsymbol{\delta\varphi}^{(n)}$ in (2.58). In order that the Newton method is Jacobian-free, this underlying iterative method must not require the formation of $\mathbf{J}$ explicitly. The family of Krylov subspace iterative methods [74] have this property, and so are a popular choice of iterative solver. Section 2.5 covers the details of the Krylov subspace method used in this thesis.

The key point with Krylov subspace methods is that they require only the operation of the matrix on a vector in the form of a matrix-vector product [74]. In the case of the Jacobian-vector product $\mathbf{Jv}$, this can be computed without ever forming $\mathbf{J}$, by using a finite difference approximation [9]. Expanding $\mathbf{F}$ in a Taylor series about $\boldsymbol{\varphi}$,

$$\mathbf{F}(\boldsymbol{\varphi} + \varepsilon\mathbf{v}) = \mathbf{F}(\boldsymbol{\varphi}) + \mathbf{J}(\boldsymbol{\varphi})\varepsilon\mathbf{v} + \mathcal{O}(\varepsilon^2\|\mathbf{v}\|^2) \tag{2.59}$$

gives to first-order accuracy,

$$\mathbf{J}(\boldsymbol{\varphi})\mathbf{v} \approx \frac{\mathbf{F}(\boldsymbol{\varphi} + \varepsilon\mathbf{v}) - \mathbf{F}(\boldsymbol{\varphi})}{\varepsilon}. \tag{2.60}$$

Since the value $\mathbf{F}(\boldsymbol{\varphi})$ is already required in (2.58), (2.60) provides a first-

order estimate of $\mathbf{Jv}$ with only one additional function evaluation [51]. Second and higher order Jacobian approximations are possible, but they require additional function evaluations [51]. Note that in any case, it is the rate of convergence that is affected by the order of this approximation, not the accuracy of the final solution.

Choosing an appropriate value for ε requires balancing possible inaccurate derivative estimates obtained using too large a value, and roundoff error contamination when using too small a value [51]. The suggestion given in [9] is to choose ε as follows:

$$\varepsilon = \frac{\sqrt{\epsilon_{\text{mach}}}}{\|\mathbf{v}\|} \max(|\boldsymbol{\varphi}^T \mathbf{v}|, \text{typ } \boldsymbol{\varphi}^T |\mathbf{v}|) \text{ sign}(\boldsymbol{\varphi}^T \mathbf{v}) \tag{2.61}$$

where ϵ_{mach} is the machine precision, typ $\boldsymbol{\varphi}$ is the "typical size" of $\boldsymbol{\varphi}$ and $|\mathbf{v}|$ means $(|v_1|, |v_2|, \ldots, |v_N|)^{\text{T}}$.

The algorithm for the Jacobian-free Newton-Krylov method is given in Algorithm 2. It uses the NSOLI implementation [50], which also incorporates a line search (backtracking) strategy

$$\boldsymbol{\varphi}^{(n+1)} = \boldsymbol{\varphi}^{(n)} + \theta_n \boldsymbol{\delta\varphi}^{(n)} \tag{2.62}$$

in place of (2.55) to ensure that the function $\mathbf{F}$ decreases sufficiently at each iteration [19].

Algorithm 2 (NSOLI [50])**.**
INPUT:
- $\boldsymbol{\varphi}_0$: Initial estimate
- $\mathbf{F}$: Nonlinear function
- tol : Nonlinear tolerance = [atol, rtol]
- maxit : Maximum number of nonlinear iterations

OUTPUT:
- $\boldsymbol{\varphi}$: Solution

```
BEGIN:
    itc = 0
    iarm = 0
    φ = φ_0
    fnorm = ||F(φ)||
    stoptol = atol + rtol × fnrm
    η = η_0
    while (fnorm > stoptol and itc < maxit)
        solve for δφ such that ||F(φ) + J(φ) δφ|| ≤ η × fnorm
        θ_m = θ
        while (||F(φ + δφ)|| > [1 − α(1 − η)] × fnorm)
            if (iarm == 0)
                θ = θ_min × θ
            else if (iarm == maxarm)
                error("Too many reductions.")
            else
                fcnorm = ||F(φ + θδφ)||
                fmnorm = ||F(φ + θ_m δφ)||
                θ = linesearch(θ, θ_m, fnorm, fcnorm, fmnorm)
            end
            δφ = θδφ
            η = 1 − θ(1 − η)
            iarm = iarm + 1
        end
        φ = φ + δφ
        fnorm = ||F(φ)||
        update η
        itc = itc + 1
    end
```

2.4.2 Combining integration and interpolation

To this point two methods of numerical integration have been considered: the midpoint rule (Section 2.2.1), and Gaussian quadrature (Section 2.2.2). Furthermore, two methods of interpolation have been considered: shape functions (Section 2.3.1) and radial basis functions (Section 2.3.3). In this section, the strategy for best combining these methods in order to achieve an accu-

Table 2.3: Optimal pairings of integration and interpolation: the two stages of the nonlinear solution process.

Name	Speed	Accuracy	Integration	Interpolation
CV-FE	Faster	Lower	Midpoint rule	Shape functions
CV-RBF	Slower	Higher	Gaussian quadrature	Radial basis functions

rate finite volume solution to (2.53) without excessive computational cost is discussed.

It must be remembered that the potential for improved accuracy offered by Gaussian quadrature over the midpoint rule comes at the cost of increased computational requirements. Similarly, the potential improvement offered by using radial basis functions as a means of interpolation instead of shape functions is also offset by the increased computational cost associated with it. To balance this additional computational work, it is therefore recommended that an accurate means of interpolation is not paired with a relatively inaccurate means of integration, and vice versa. Therefore, the optimal pairing of the various methods appears to be shape functions with midpoint rule, and radial basis functions with Gaussian quadrature. This is summarised in Table 2.3.

Names shall be given to these two discretisation strategies so that they may be easily referred to throughout this thesis. The method incorporating shape functions and the midpoint rule shall be referred to as CV-FE, standing for control volume-finite element, as this is a standard term in the literature. Keeping with this pattern, the method incorporating radial basis functions and Gaussian quadrature shall be referred to as CV-RBF, standing for *control volume-radial basis function*. Likewise, where necessary, the nonlinear functions and Jacobian matrices of the two methods will be distinguished by labelling them as $\mathbf{F}_{FE}$ and $\mathbf{J}_{FE}$ for the CV-FE method, and $\mathbf{F}_{RBF}$ and $\mathbf{J}_{RBF}$

for the CV-RBF method.

Note also that the terms CV-FE and CV-RBF can refer to just the discretisation, or to the finite volume method as a whole, incorporating that particular method of discretisation. The context of the discussion will make it clear which meaning is intended.

One possible criticism of this approach could be that the CV-FE method may be being unfairly penalised in its potential accuracy by not using the more accurate Gaussian quadrature interpolation. In Sections 4.3.1 and 6.3.1 the method of Gaussian quadrature is paired with shape functions to demonstrate that in fact no significant accuracy is gained by such an approach.

2.4.3 Improving performance

When applying Algorithm 2, the most expensive step is the evaluation of $\mathbf{F}$. This is the step that actually invokes the finite volume discretisation that has just been discussed. The performance of the nonlinear solution process can be improved by minimising the cost of these evaluations, the number of evaluations required, or both.

Minimising the cost of each evaluation

Each evaluation of $\mathbf{F}$ requires the advective-diffusive fluxes and source components to be computed for every control volume in the mesh. In the case of the source component, there is one m-dimensional integral required per control volume (2.20). In the case of the advective-diffusive flux component, there is one $m-1$ dimensional integral required for every control volume face (2.16).

In both cases, the cost of the evaluation is much cheaper when applying CV-FE (Table 2.3) than when applying CV-RBF. There are two reasons for

this:

1. CV-FE iterations require fewer evaluations of the interpolant s_i, because they use the midpoint rule for integration instead of multiple-point Gaussian quadrature.

2. each evaluation is cheaper with CV-FE, because it uses shape functions instead of RBFs.

Note that it is not the up-front cost of computing the interpolant itself that is the significant factor in the evaluation. Since this occurs only once, its cost is averaged over the many times the interpolant is evaluated. It is the evaluation of the interpolant that accounts for the majority of the computational time in this process. (This would certainly be different if a single, global interpolant were used, in which case the solution of (2.43) would indeed be a significant upfront expense.)

Minimising the number of evaluations

There are three cases in Algorithm 2 that require the evaluation of $\mathbf{F}$, which are listed as follows:

1. at the beginning of a Newton step, to form the right hand side of (2.58) and also to use in the Jacobian approximation (2.60).

2. once for every Jacobian-vector product used in the underlying Krylov method.

3. once for every backtracking step required to ensure a sufficient decrease in $\mathbf{F}$ (see Algorithm 3 later in this section for more details on line searching).

Of the three, case 1 contributes least to the overall number of function evaluations, since it occurs only once per Newton iteration. Case 3 may be invoked several times per iteration, but only when line searching is required to correct an unsatisfactory Newton step (discussed later in this section). The majority of evaluations are incurred by case 2, so this is where the primary focus of improving efficiency should be directed.

The most natural way to keep the number of **F** evaluations under control is to reduce the total number of nonlinear iterations required to solve (2.53). More correctly, it is the total number of nonlinear iterations using CV-RBF that needs to be kept to a minimum. By comparison, iterations using CV-FE are not costly. This leads to the idea of solving (2.53) in two stages. Stage one, using CV-FE, is applied first, to produce a solution of moderate accuracy without great computational expense. Stage two, using CV-RBF, is then applied using the final CV-FE solution as its initial estimate, to "correct" the CV-FE solution, giving the final solution of greatest accuracy. Refer again to Table 2.3 for a summary of this two-stage nonlinear solution process.

The number of case 2 evaluations above can be further reduced by:

- using an effective preconditioner within the linear iterations.
- using an appropriate forcing term.

The number of case 3 evaluations above can be further reduced by:

- using an efficient line search algorithm.

The remainder of this section addresses each one of these points.

Preconditioning It is well-known that effective preconditioning is vital in reducing the number of linear iterations in a Newton-Krylov method [51].

However, preconditioning in a Jacobian-free Newton-Krylov method is complicated by the fact that $\mathbf{J}$ is not explicitly available. One approach that can be taken in these cases is to use the Jacobian matrix from a *simplification* of the problem to compute a preconditioner [51]. This simplified problem should give rise to a Jacobian matrix that is relatively inexpensive to compute, but which approximates the full Jacobian matrix sufficiently well that it can be used to construct an effective preconditioner.

The method of shape functions again offers a practical solution. The CV-FE Jacobian matrix is relatively inexpensive to compute, owing to its sparsity and to the simpler discretisation method used to generate its entries. Thus, it is feasible to compute and store this matrix at the beginning of the two-stage nonlinear solution process. This Jacobian matrix can then be used in preconditioning the underlying linear solver in both stage one and two of the nonlinear solution process. Since the CV-FE discretisation and the CV-RBF discretisation are two representations of the same physical process, it seems plausible that the same preconditioner could be effective for both methods.

Forcing term The choice of forcing term used in Algorithm 2 of Section 2.4.1 is also important in reducing the number of required evaluations of $\mathbf{F}$. Essentially, a poor choice of forcing term can result in "oversolving" of the underlying linear system (2.58) [66]. This occurs when the Newton step $\delta\boldsymbol{\varphi}^{(n)}$ is solved to a higher accuracy than is actually necessary to achieve the required reduction in $\|\mathbf{F}\|$. A good choice of forcing term helps avoid this wasted computational effort. The forcing term used in this thesis is *Choice 2* used in [66] and first described in [20]. It uses parameters $\gamma \in [0, 1]$ and

$\beta \in (1, 2]$, and has the form

$$\eta_n = \gamma \left(\frac{\|\mathbf{F}(\boldsymbol{\varphi}^{(n)})\|}{\|\mathbf{F}(\boldsymbol{\varphi}^{(n-1)})\|} \right)^{\beta}, \qquad k = 1, 2, \ldots \tag{2.63}$$

with the safeguard

$$\eta_n \leftarrow \min(\eta_n, \eta_{\max}). \tag{2.64}$$

As stated in [20], although this choice of forcing term does not directly ensure agreement between $\mathbf{F}(\boldsymbol{\varphi} + \boldsymbol{\delta\varphi})$ and its local linear model $\mathbf{F}(\boldsymbol{\varphi}) + \mathbf{J}(\boldsymbol{\varphi})\boldsymbol{\delta\varphi}$, in practice it does nevertheless tend to limit the oversolving of (2.58).

An efficient line search algorithm *Line searching* or *backtracking*, is the process of adjusting the size of the Newton step to ensure that the nonlinear residual decreases sufficiently [49]. This is necessary to make the method *globally convergent*, that is, either convergent or failing in a limited number of well-understood ways, for any initial estimate [49].

With the Newton step denoted by $\boldsymbol{\delta\varphi}^{(n)}$, the goal of a line search is to select a value θ_n such that

$$\|\mathbf{F}(\boldsymbol{\varphi}^{(n)} + \theta_n \boldsymbol{\delta\varphi}^{(n)})\| < (1 - \alpha\theta_n)\|\mathbf{F}(\boldsymbol{\varphi}^{(n)})\|. \tag{2.65}$$

Equation (2.65) represents a *sufficient decrease* in $\|\mathbf{F}\|$, and it ensures that the method is progressing towards the solution without oscillation [49]. The parameter α is a small, positive number that makes (2.65) as easy as possible to satisfy [49].

Denoting by h the function

$$h(\theta) = \|\mathbf{F}(\boldsymbol{\varphi}^{(n)} + \theta \boldsymbol{\delta\varphi}^{(n)})\|, \tag{2.66}$$

the ideal value of θ is the value that minimises (2.66) over $0 < \theta \leq 1$. In practice, a polynomial model is used in place of (2.66), and the minimum of this model found.

The line search used in Algorithm 2 fits a quadratic model to the points $(0, \|\mathbf{F}(\boldsymbol{\varphi}^{(n)})\|)$, $(\theta_\mathrm{c}, \|\mathbf{F}(\boldsymbol{\varphi}^{(n)} + \theta_\mathrm{c}\boldsymbol{\delta\varphi}^{(n+1)})\|)$ and $(\theta_\mathrm{m}, \|\mathbf{F}(\boldsymbol{\varphi}^{(n)} + \theta_\mathrm{m}\boldsymbol{\delta\varphi}^{(n+1)})\|)$ using the current step length θ_c and the previous step length θ_m. The newly computed θ is also *safeguarded* by the values θ_min and θ_max [49]. Algorithm 3 outlines the procedure.

Algorithm 3 (Linesearch)**.**
INPUT:
θ_c : Current step length
θ_m : Previous step length
f_0 : $\|\mathbf{F}(\boldsymbol{\varphi}^{(n)})\|$
f_c : $\|\mathbf{F}(\boldsymbol{\varphi}^{(n)} + \theta_\mathrm{c}\boldsymbol{\delta\varphi}^{(n+1)})\|$
f_m : $\|\mathbf{F}(\boldsymbol{\varphi}^{(n)} + \theta_\mathrm{m}\boldsymbol{\delta\varphi}^{(n+1)})\|$
OUTPUT:
θ_p : New step length
BEGIN:
model = quadratic_fit($[0, f_0]$, $[\theta_\mathrm{c}, f_c]$, $[\theta_\mathrm{m}, f_m]$)
θ_p = model.minimise()
$\theta_\mathrm{p} = \max(\theta_\mathrm{min}, \theta_\mathrm{p})$
$\theta_\mathrm{p} = \min(\theta_\mathrm{max}, \theta_\mathrm{p})$

2.4.4 Adaptive switching

Given that interpolations involving RBFs are much more expensive than those using shape functions, it is natural to consider whether in stage two of the nonlinear solution process every element in the mesh needs to be using RBFs as a means of interpolation. It may be that for some elements, the accuracy offered by shape functions is sufficient to achieve the desired accuracy, without needing to resort to RBFs. In this section a method is presented that can

adaptively switch between using shape functions and using RBFs to exploit this situation.

One possible approach would be to base the method on an analysis of the underlying problem, and employ shape functions for elements where the physics of the problem suggests that low accuracy approximations are sufficient to fully capture the solution behaviour. Using this approach does impact on the versatility of the method however. For each different problem to be solved, this analysis must be carried out to determine which elements require RBFs and which can make do with shape functions. Misclassification of elements would result in either wasted computational effort, or a loss in accuracy of the final solution.

An alternative approach, and the approach that is taken in this thesis, is for the method itself to classify every element in the mesh based on the information it has gathered during its execution. By monitoring the accuracy of the solution as the method progresses it may be possible to directly identify which elements are in need of additional accuracy, and which are not. This strategy thus offers two advantages:

1. It is non-intrusive. The user does not have to supply any additional information to use the method.

2. It may be more effective. The classification is directly based on the accuracy of the solution at various stages during the solution process.

A disadvantage of this method is that it requires additional computational effort to monitor the solution accuracy and compute the appropriate information on which to base the classification. In this work, the two-stage solution process outlined in Table 2.3 is once again exploited. Recall that this process uses an initial stage of CV-FE iterations, followed by a second stage

of CV-RBF iterations. After the first stage of the process, the solution has converged with respect to CV-FE, i.e. the residual norm $\|\mathbf{F}_{FE}\|$ is less than the desired tolerance. Nevertheless, the solution that has been obtained at this stage is not guaranteed to be within any given tolerance of the analytic solution. It is the accuracy of the CV-FE discretisation that determines the accuracy of the solution. To reiterate: the nonlinear solution process simply computes the solution of (2.53). Whether this solution is a good approximation of φ depends entirely on how well the finite volume discretisation models the original PDE (2.9).

When commencing stage two, this CV-FE solution is used as the initial estimate for the CV-RBF method. With respect to this more accurate discretisation strategy, the CV-FE solution will not generally be sufficiently accurate. That is, $\|\mathbf{F}_{RBF}\|$ will not be less than the desired tolerance. The idea behind the adaptive strategy is to examine each component of the new absolute residual vector $|\mathbf{F}_{RBF}|$, and observe exactly which components have increased significantly, and which have remained largely unchanged. Components that have increased significantly indicate that the CV-RBF discretisation is producing a significantly different result for these nodes. On the other hand, components that have not increased significantly may indicate that the CV-FE discretisation was already giving acceptable accuracy for these nodes, and that the introduction of RBFs has not dramatically improved the accuracy.

There are some additional complications in applying this method. First there is the issue of the nonlinear residual vector being a node-wise indication of accuracy, while the interpolations are carried out in an element-wise fashion. There needs to be some way of mapping this information from the nodes to the elements. A conservative approach is used in this implementation that, where necessary, sacrifices efficiency for accuracy. After the inter-

polation method for each node has been classified (using the process to be discussed in the next paragraph), the elements are assigned shape functions or RBFs as follows:

- if *all* of the element's vertices have been classified as requiring shape functions, then the element uses shape functions.
- if *any* of the element's vertices has been classified as requiring RBFs, then the element uses RBFs.

To actually decide whether shape functions or RBFs are more appropriate for a given node, the residual components are compared against a predetermined threshold value (see Section 6.4.4 for a comparison of different threshold values). All nodes with residual components less than the threshold value are classified as requiring shape functions. All nodes with residual components greater than the threshold value are classified as requiring RBFs. The threshold value itself provides the balance between losing too much accuracy by overusing shape functions and wasting too much computational time by overusing RBFs.

The element classification algorithm incorporating these features is given in Algorithm 4.

Algorithm 4 (Adaptive switching)**.**
INPUT:
`m` : Mesh (N nodes, E elements)
$\mathbf{F}_{RBF}$: RBF residual vector $\in \mathbb{R}^N$
`threshold` : Threshold value for using shape functions or RBFs

OUTPUT:
`classification` : Classification for every element in the mesh (0 represents shape functions, 1 represents RBFs)

```
BEGIN:
    element_nodes = { }
    for i = 1 to N
        vol = m.control_volume(i)
        if F_RBF[i] < cutoff
            for j = 1 to vol.num_elements()
                element_nodes.insert([vol.element(j), 0])
            end for
        else
            for j = 1 to vol.num_elements()
                element_nodes.insert([vol.element(j), 1])
            end for
        end if
    end for
    for i = 1 to E
        classification[i] = max(element_nodes[i])
    end for
```

One additional feature is also implemented in this thesis as part of the adaptive switching method: the ability to change the number of nodes used in the RBF interpolations. The idea is to use a smaller number of nodes initially, say n_{init}, only for the purposes of classifying elements as requiring either shape functions or RBFs. Then, once the elements have been classified, those elements requiring RBFs have their interpolations re-computed using a larger number of nodes, n_{final}. For example, the method might use $n_{\text{init}} = 10$ nodes per element initially in computing $\mathbf{F}_{RBF}$. This then allows the elements genuinely requiring RBF accuracy to be identified by means of Algorithm 4, after which their interpolations are modified to use the value n_{final}, say 25. This way, large RBF interpolations are only ever computed for elements that actually require them.

The overhead incurred by this method, including the application of Algorithm 4 and any additional computations of RBF interpolations, is not extravagant, but certainly must be taken into account when evaluating its

overall effectiveness. In Section 6.4.4, the accuracy and efficiency of this method with various combinations of parameters is tested, and compared to the standard, non-adaptive method.

2.5 Solution of linear system

In a Jacobian-free Newton-Krylov method for solving (2.53), the Newton step $\boldsymbol{\delta\varphi}^{(n)}$ is determined according to (2.54). A Krylov subspace method is used to solve this linear system to within the tolerance required by (2.58). As was mentioned in the previous section, one of the motivations for using a Krylov subspace method is that the Jacobian matrix need not be computed explicitly. Krylov subspace methods require only $\mathbf{Jv}$, the action of the matrix $\mathbf{J}$ on the vector $\mathbf{v}$, and (2.60) can be used to compute these products.

Despite this, in practice the method cannot usually be fully matrix-free [51]. Effective preconditioning is crucial to the performance of the Krylov solver, and this typically requires at least an approximation to the Jacobian matrix to be explicitly available. Often a Jacobian matrix obtained from a simplification of the problem can be obtained cheaply, and this may be sufficient for preconditioning purposes [51].

2.5.1 GMRES-DR

The Krylov method used in this thesis is GMRES *with Deflated Restarting*, or GMRES-DR [59]. This method is an extension of the basic GMRES method which is described below.

GMRES

The Generalised Minimum Residual Method (GMRES) [75] is a popular iterative method for solving nonsymmetric, sparse linear systems of equations [74, 76]. It belongs to a group of methods known as *projection methods*, so named because they involve projecting the problem onto a subspace, subject to certain restrictions, such that an approximate solution is obtained.

Consider the linear system (2.54), with $\mathbf{y}$ in place of $\boldsymbol{\delta\varphi}^{(n)}$ for notational convenience:

$$\mathbf{J}\mathbf{y} = -\mathbf{F}, \tag{2.67}$$

where $\mathbf{J} \in \mathbb{R}^{N\times N}$ and $\mathbf{y}, \mathbf{F} \in \mathbb{R}^{N\times 1}$. Suppose there is an initial estimate for the solution, $\mathbf{y_0}$, and that two subspaces of $\mathbb{R}^N$ have been selected, $\mathcal{K}$ and $\mathcal{L}$, each of arbitrary dimension m. Let the initial residual vector $\mathbf{r_0}$ be defined as

$$\mathbf{r_0} = -(\mathbf{F} + \mathbf{J}\mathbf{y}_0). \tag{2.68}$$

The general projection method computes a correction $\delta\mathbf{y}$ to the approximate solution $\mathbf{y_0}$ such that

$$\delta\mathbf{y} \in \mathcal{K} \tag{2.69}$$

and

$$\mathbf{r_0} - \mathbf{J}\delta\mathbf{y} \perp \mathcal{L}. \tag{2.70}$$

Equations (2.69) and (2.70) state that the correction lies in $\mathcal{K}$ and the new residual is orthogonal to $\mathcal{L}$. Note that there are m degrees of freedom in selecting the correction which lies in $\mathcal{K}$, and m orthogonality constraints corresponding to the m linearly independent basis vectors of $\mathcal{L}$ [74]. The new solution,

$$\mathbf{y} = \mathbf{y_0} + \delta\mathbf{y} \tag{2.71}$$

lies in the affine space

$$\mathbf{y} \in \mathbf{y_0} + \mathcal{K}. \tag{2.72}$$

GMRES uses the Krylov subspace

$$\mathcal{K} \equiv \mathcal{K}_m = \text{span}\{\mathbf{r}_0, \mathbf{J}\mathbf{r}_0, \mathbf{J}^2\mathbf{r}_0, \ldots, \mathbf{J}^{m-1}\mathbf{r}_0\} \tag{2.73}$$

with the subspace $\mathcal{L}$ taken as $\mathbf{J}\mathcal{K}$:

$$\mathcal{L} \equiv \mathcal{L}_m = \text{span}\{\mathbf{J}\mathbf{r}_0, \mathbf{J}^2\mathbf{r}_0, \mathbf{J}^3\mathbf{r}_0, \ldots, \mathbf{J}^m\mathbf{r}_0\}. \tag{2.74}$$

Computationally, each iteration of GMRES introduces one more vector into the Krylov basis by multiplying the most recently included vector by $\mathbf{J}$ and orthonormalising it against all other basis vectors. This is known as the *Arnoldi process*, and the resultant vectors are known as *Arnoldi vectors* [74]. Because the basis grows by one vector for each GMRES iteration, the memory and computational requirements of the method can be substantial if the problem dimension is large. For this reason, a variant called *restarted* GMRES is often used in practice, whereby the entire method is restarted every m iterations, using the most recently computed solution as the initial estimate for the next cycle [74]. This method is also denoted by GMRES(m).

Restarting

A phenomenon that is often observed with restarted GMRES is stalling, whereby the residual norm "levels out" at a certain value and subsequent cycles have little effect in reducing it further. It is known [58, 74] that the presence of both large and small magnitude eigenvalues in the matrix spectrum can contribute to this undesirable situation.

To overcome this problem, the system (2.67) can be modified to incorporate *preconditioning* [74]. A preconditioner matrix $\mathbf{M}$ is introduced, with a view to modifying the matrix spectrum to make the problem more amenable to solution using GMRES(m). Preconditioners may be applied either on the left, where (2.67) is replaced by

$$\mathbf{MJy} = -\mathbf{MF}, \tag{2.75}$$

or on the right, where (2.67) is replaced by

$$\mathbf{JM}(\mathbf{M}^{-1}\mathbf{y}) = -\mathbf{F}\,. \tag{2.76}$$

Of the two approaches, only preconditioning on the right preserves the residual

$$\mathbf{r} = -\mathbf{F} - \mathbf{JM}(\mathbf{M}^{-1}\mathbf{y}) = -(\mathbf{F} + \mathbf{Jy}) \tag{2.77}$$

and is therefore the preferred approach for methods such as GMRES that monitor the residual norm [76]. Section 2.5.2 discusses the particular preconditioners that were employed in this thesis.

Re-using Krylov subspace information

When GMRES(m) is restarted, all the information generated in the previous m iterations, apart from the most recently computed solution, is discarded. It was shown in [58] how the method could be improved by saving more information than just the most recent solution. The most effective approach found was to use the Krylov subspace to compute *approximate eigenvectors* of the matrix. These approximate eigenvectors would then be included in the Krylov subspace in the next cycle.

Denoting these additional vectors by $\mathbf{u}_i, i = 1, 2, \ldots, k$ the so-called *augmented Krylov subspace* takes the following form (modified to include right preconditioning):

$$\mathcal{K}_{k+m} = \text{span}\{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_k, \mathbf{r}_0, \mathbf{JM}\mathbf{r}_0, (\mathbf{JM})^2\mathbf{r}_0, \ldots, (\mathbf{JM})^{m-1}\mathbf{r}_0\} \quad (2.78)$$

GMRES-based methods that employ a Krylov subspace of this form are known as *deflated* GMRES [13, 58, 59]. The method used in this thesis is called GMRES-DR, standing for GMRES with *deflated restarting* [59]. It modifies the standard Krylov subspace by augmenting it with k approximate eigenvectors corresponding to the k smallest eigenvalues (in magnitude) of $\mathbf{JM}$. The motivation for this is discussed in [58], where it is shown that by injecting these vectors into the Krylov subspace, the corresponding eigenvalues are deflated from the spectrum, and convergence then proceeds according to this modified spectrum.

GMRES-DR uses a harmonic projection onto $\mathcal{K}$ to compute the $\mathbf{u}_i$, which are known as *Harmonic Ritz vectors*. The projection exploits the fundamental Arnoldi equation [74]:

$$\mathbf{JM}^{-1}\mathbf{V}_m = \mathbf{V}_{m+1}\bar{\mathbf{H}}_m \quad (2.79)$$

where $\mathbf{V}_m \in \mathbb{R}^{N\times m}$ is the matrix with the first m orthonormalised Krylov basis vectors as its columns, and $\bar{\mathbf{H}}_m \in \mathbb{R}^{(m+1)\times m}$ is the matrix of Arnoldi coefficients.

The harmonic projection is onto the space spanned by the columns of $\mathbf{V}_m$ and orthogonal to that spanned by the columns of $\mathbf{JM}^{-1}\mathbf{V}_m$ [13, 58]. In matrix form:

$$(\mathbf{JM}^{-1}\mathbf{V}_m)^T(\mathbf{JM}^{-1} - \lambda\mathbf{I}_N)\mathbf{V}_m\mathbf{h} = \mathbf{0} \quad (2.80)$$

where

$$\mathbf{u} = \mathbf{V}_m \mathbf{h} \tag{2.81}$$

is the Harmoinc Ritz vector, and $\mathbf{h} \in \mathbb{R}^{m \times 1}$. Rearranging slightly, and substituting (2.79), yields

$$(\mathbf{V}_{m+1}\bar{\mathbf{H}}_m)^T(\mathbf{V}_{m+1}\bar{\mathbf{H}}_m)\mathbf{h} = \lambda(\mathbf{V}_{m+1}\bar{\mathbf{H}}_m)^T\mathbf{V}_m\mathbf{h} \tag{2.82}$$

which, after some further manipulation (see [60]) simplifies to

$$(\mathbf{H}_m + h_{m+1,m}^2 \mathbf{H}_m^{-T}\mathbf{e}_m\mathbf{e}_m^T)\mathbf{h} = \lambda\mathbf{h}, \tag{2.83}$$

a standard eigenvalue problem involving the matrix $\mathbf{H}_m \in \mathbb{R}^{m \times m}$, the first m rows of $\bar{\mathbf{H}}_m$ [58]. Compared to the scale of the original problem, it is a simple matter to compute the eigenvectors $\mathbf{h}_i \in \mathbb{R}^{m \times 1}$ of this small problem. Equation (2.81) is the means by which these m-dimensional eigenvectors are mapped to the full Harmonic Ritz vectors $\mathbf{u}_k \in \mathbb{R}^{N \times 1}$ to be used in (2.78).

GMRES-DR offers two advantages over other, similar methods based on augmenting eigenvectors. The first is its use of the Harmonic projection to compute the approximate eigenvectors, rather than the alternative Rayleigh-Ritz projection. The difference between the two is the space used to impose the orthogonality constraints. While the Rayleigh-Ritz projection uses $\mathbf{V}_m$ itself in the orthogonality condition, the Harmonic projection uses $\mathbf{J}\mathbf{M}^{-1}\mathbf{V}_m$. The reason for preferring the Harmonic projection is to do with the accuracy of the small eigenvalue approximations [58]. The Harmonic Ritz values tend to give much better approximations of the small eigenvalues of $\mathbf{J}$ than do the (ordinary) Ritz values computing using the Rayleigh-Ritz projection. This can be explained by recognising that the Harmonic Ritz values are actually

Ritz values of $\mathbf{J}^{-1}$, albeit computed with respect to a subspace based on $\mathbf{J}$ [58].

The second advantage offered by GMRES-DR is its augmentation of the approximate eigenvectors at the front of the Krylov basis, rather than at the back. Although this comes as a change to the usual Arnoldi process, it is nonetheless shown in [59] that the space spanned by this augmented Krylov basis is still a Krylov subspace. The advantage in augmenting at the front is that a Jacobian-vector product is saved for each augmented vector. Mathematically equivalent methods that augment these vectors at the back of the basis do require these additional Jacobian-vector products, which can be a significant additional cost (see Section 2.4.3).

Note also that full re-orthogonalisation has been used in this implementation of GMRES-DR. There is a small additional cost associated with this approach, but it is included to ensure that the approximations of the Harmonic pairs are as accurate as possible.

The GMRES-DR method is described in full in Algorithm 5 [60].

Algorithm 5 (GMRES-DR)**.**
$\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0, \quad \beta = \|\mathbf{r}_0\|_2, \quad \mathbf{v}_1 = \mathbf{r}_0/\beta,$
for $j = 1$ to m
    $\mathbf{w} = \mathbf{A}\mathbf{M}^{-1}\mathbf{v}_j$
    for $i = 1$ to j
        $h_{ij} = \langle \mathbf{w}, \mathbf{v}_i \rangle$
        $\mathbf{w} = \mathbf{w} - h_{ij}\mathbf{v}_i$
    end for
    for $i = 1$ to j
        $c = \langle \mathbf{w}, \mathbf{v}_i \rangle$
        $h_{ij} = h_{ij} + c$
        $\mathbf{w} = \mathbf{w} - c\,\mathbf{v}_i$
    end for
    $h_{j+1,j} = \|\mathbf{w}\|_2$

if $h_{j+1,j} = 0$
$m = j$; break
end if
$\mathbf{v}_{j+1} = \mathbf{w}/h_{j+1,j}$
end for
$\mathbf{g} = \beta \mathbf{e}_1$
$\mathbf{y}_m = \min \|\mathbf{g} - \bar{\mathbf{H}}_m \mathbf{y}\|_2$
$\mathbf{x}_m = \mathbf{x}_0 + \mathbf{M}^{-1} \mathbf{V}_m \mathbf{y}_m$
$\mathbf{r}_m = \mathbf{V}_{m+1}(\mathbf{g} - \bar{\mathbf{H}}_m \mathbf{y}_m)$
if solution satisfactory then stop
converged = false
while not converged
$\mathbf{f} = \bar{\mathbf{H}}_m^{-T} \mathbf{e}_m$
$\hat{\mathbf{H}}_m = \mathbf{H}_m + h_{m+1,m}^2 \mathbf{f} \mathbf{e}_m^T$
Compute k eigenvalues and eigenvectors of $\hat{\mathbf{H}}_m$
Orthonormalise eigenvectors to obtain $\mathbf{Q}_k$
Form $\hat{\mathbf{Q}}_k = \begin{pmatrix} \mathbf{Q}_k \\ \mathbf{0}^T \end{pmatrix}$
Orthonormalise $\mathbf{g} - \bar{\mathbf{H}}_m$ against the columns of $\hat{\mathbf{Q}}_k$ to form $\mathbf{q}_{k+1}$
$\mathbf{Q}_{k+1} = (\hat{\mathbf{Q}}_k, \mathbf{q}_{k+1})$
$\mathbf{V}_{k+1} = \mathbf{V}_{m+1} \mathbf{Q}_{k+1}$
$\bar{\mathbf{H}}_k = \mathbf{Q}_{k+1}^T \bar{\mathbf{H}}_m \mathbf{Q}_k$
for $j = k + 1$ to m
$\mathbf{w} = \mathbf{A}\mathbf{M}^{-1} \mathbf{v}_j$
for $i = 1$ to j
$h_{ij} = \langle \mathbf{w}, \mathbf{v}_i \rangle$
$\mathbf{w} = \mathbf{w} - h_{ij} \mathbf{v}_i$
end for
for $i = 1$ to j
$c = \langle \mathbf{w}, \mathbf{v}_i \rangle$
$h_{ij} = h_{ij} + c$
$\mathbf{w} = \mathbf{w} - c\, \mathbf{v}_i$
end for
$h_{j+1,j} = \|\mathbf{w}\|_2$
if $h_{j+1,j} = 0$
$m = j$; break
end if
$\mathbf{v}_{j+1} = \mathbf{w}/\mathbf{h}_{j+1,j}$
end for
$\mathbf{g} = \mathbf{V}_{m+1}^T \mathbf{r}$

$\mathbf{y}_m = \min \|\mathbf{g} - \bar{\mathbf{H}}_m \mathbf{y}\|_2$
$\mathbf{x}_m = \mathbf{x}_0 + \mathbf{M}^{-1}\mathbf{V}_m \mathbf{y}_m$
$\mathbf{r}_m = \mathbf{V}_{m+1}(\mathbf{g} - \bar{\mathbf{H}}_m \mathbf{y}_m)$
if solution satisfactory
converged = true
end if
end while

One further improvement to GMRES-DR is possible when using the method in the context of a Newton-Krylov method. Usually, the first cycle of GMRES-DR is performed without any $\mathbf{u}_i$ vectors, since initially there is no information available from which to compute them. However, in a Newton-Krylov method there is an outer level of nonlinear iterations that can provide this information. In all but the first nonlinear iteration, the first set of $\mathbf{u}_i$ vectors can be computed from the last GMRES cycle of the previous nonlinear iteration.

2.5.2 Preconditioning

As stated in the previous section, the presence of either large or small eigenvalues in the matrix spectrum can be a hinderance to the convergence of GMRES [58, 74]. The effectiveness of a given preconditioning strategy can be judged by how it influences the eigenvalue spectrum with respect to these largest and smallest values. In this thesis, two different preconditioners are used in combination. The first is a norm-minimising, approximate inverse preconditioner which is most effective in dealing with large eigenvalues [74]. The second is a deflation-based preconditioner, designed to explicitly deflate small eigenvalues [11]. Both of these preconditioners use combinations of CV-FE information from stage one and CV-RBF information from stage two in their construction.

Norm-minimising preconditioner

The first preconditioner is a sparse approximate inverse based on Frobenius-norm minimisation, as described in [74] and elsewhere. The idea is to build a matrix $\mathbf{M}_1$, with some imposed sparsity pattern, that minimises

$$\|\mathbf{I} - \mathbf{J}\mathbf{M}_1\|_F. \tag{2.84}$$

where $\mathbf{I}$ is the identity matrix. Using the Frobenius norm is the key to this method, for it allows problem (2.84) to be decomposed into N independent least squares problems [74]:

$$\|\mathbf{I} - \mathbf{J}\mathbf{M}_1\|_F^2 = \sum_{j=1}^{N} \|\mathbf{e}_j - \mathbf{J}\mathbf{m}_j\|_2^2 \,, \tag{2.85}$$

where $\mathbf{e_j}$ is the jth unit vector, and $\mathbf{m}_j$ is the jth column of $\mathbf{M}_1$. Each vector $\mathbf{m}_j$ will have nonzero entries only where the sparsity pattern of $\mathbf{M}_1$ dictates, so each problem (2.85) is solvable using standard direct decomposition methods (QR factorisation for example [81]).

In this thesis, $\mathbf{M}_1$ is computed using a combination of CV-FE and CV-RBF information, and is applied during both stages of the nonlinear solution process. The *values* of the nonzero entries are determined according to (2.85), using CV-FE information. The *locations* of these entries are determined according CV-RBF information: the sparsity pattern of $\mathbf{M}_1$ matches that of the Jacobian matrix $\mathbf{J}_{RBF}$. The location of these nonzero entries is predetermined by means of the stencils computed in Algorithm 1, so there is no difficulty in forming $\mathbf{M}_1$ before the nonlinear solution process has begun.

Algorithm 6 (Constructing norm-minimising preconditioner)**.**
INPUT:
ℓ_1, ℓ_2, …, ℓ_E: Point clouds from applying Algorithm 1 to each element
$\mathbf{J}_{FE}$: $N \times N$ CV-FE Jacobian matrix
OUTPUT:
$\mathbf{M}_1$: $N \times N$ preconditioning matrix
BEGIN:
for i = 1 to E
nbs$_i$ = {}
for j = 1 to ℓ_i.size
nbs$_i$.insert(ℓ_i.element(j))
end for
end for
for j = 1 to N
m$_j$ = nbs$_j$.size
$\mathbf{A}_i = N \times j$ zero matrix
i = 1
for p ∈{p|**J**(p,j) nonzero}
$\mathbf{A}_i(i,j) = \mathbf{J}(p,j)$
i = i + 1
end for
$\mathbf{Q}, \mathbf{R}$ = QR($\mathbf{A}_i$)
solve $\mathbf{R}\mathbf{m}_j = \mathbf{Q}^T\mathbf{e}_j$
end for

Deflation

The second preconditioner used in this thesis involves building a matrix that explicitly deflates the smallest eigenvalues from the matrix spectrum. An approach for constructing such a matrix is described in [11]. Suppose that the eigenvectors corresponding to the k smallest eigenvalues of $\mathbf{J}$ have been computed, orthonormalised, and stored as the columns of the matrix $\mathbf{U}$. Then

$$\mathbf{M}_2 = \mathbf{I}_N + \mathbf{U}(\mathbf{T}^{-1} - \mathbf{I}_k)\mathbf{U}^T, \tag{2.86}$$

where

$$\mathbf{T} = \mathbf{U}^T \mathbf{J} \mathbf{U} \tag{2.87}$$

defines a deflation matrix $\mathbf{M}_2$ that will map the k smallest eigenvalues of $\mathbf{J}$ to unity [11].

To see that this is so, consider the space

$$\mathcal{U} = \text{span}\{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_k\} \tag{2.88}$$

and let the matrix $\mathbf{U} \in \mathbb{R}^{N \times k}$ have as its columns an orthonormal basis for $\mathcal{U}$, and the matrix $\mathbf{W} \in \mathbb{R}^{N \times (N-k)}$ have as its columns an orthonormal basis for $\mathcal{U}^\perp$. Now consider $\mathbf{B} = [\mathbf{U}, \mathbf{W}] \in \mathbb{R}^{N \times N}$ and note that

$$\mathbf{B}^T \mathbf{B} = \mathbf{B} \mathbf{B}^T = \mathbf{I}_N \,. \tag{2.89}$$

Thus,

$$\mathbf{B}^T \mathbf{J} \mathbf{B} = \begin{bmatrix} \mathbf{U}^T \\ \mathbf{W}^T \end{bmatrix} \mathbf{J} \, [\mathbf{U}, \ \mathbf{W}] = \begin{bmatrix} \mathbf{U}^T \mathbf{J} \mathbf{U} & \mathbf{U}^T \mathbf{J} \mathbf{W} \\ \mathbf{W}^T \mathbf{J} \mathbf{U} & \mathbf{W}^T \mathbf{J} \mathbf{W} \end{bmatrix} \tag{2.90}$$

Were the $\mathbf{u}_i$ exact eigenvectors of $\mathbf{J}$, they would span an invariant subspace. However, since they are only approximations, they span what is called an *approximately invariant subspace* [40]:

$$\mathbf{J} \mathbf{U} = \mathbf{U} \mathbf{T} + \boldsymbol{\varepsilon} \,. \tag{2.91}$$

Thus

$$\mathbf{W}^T \mathbf{J} \mathbf{U} = \mathbf{W}^T \mathbf{U} \mathbf{T} + \mathbf{W}^T \boldsymbol{\varepsilon} = \mathbf{W}^T \boldsymbol{\varepsilon} \approx \mathbf{0} \,, \tag{2.92}$$

so from (2.87) and (2.90), a matrix similar to $\mathbf{J}$ is given by

$$\mathbf{B}^T\mathbf{J}\mathbf{B} = \begin{bmatrix} \mathbf{T} & \tilde{\mathbf{J}}_{12} \\ \mathbf{W}^T\boldsymbol{\varepsilon} & \tilde{\mathbf{J}}_{22} \end{bmatrix} \approx \begin{bmatrix} \mathbf{T} & \tilde{\mathbf{J}}_{12} \\ \mathbf{0} & \tilde{\mathbf{J}}_{22} \end{bmatrix} \tag{2.93}$$

where $\tilde{\mathbf{J}}_{12} = \mathbf{U}^T\mathbf{J}\mathbf{W}$ and $\tilde{\mathbf{J}}_{22} = \mathbf{W}^T\mathbf{J}\mathbf{W}$.

A similar result can be derived for $\mathbf{M}_2$. Referring to (2.86), it follows that

$$\mathbf{B}^T\mathbf{M}_2\mathbf{B} = \mathbf{B}^T\mathbf{B} + \mathbf{B}^T\mathbf{U}(\mathbf{T}^{-1} - \mathbf{I}_k)\mathbf{U}^T\mathbf{B}\,. \tag{2.94}$$

Next, expressions for $\mathbf{B}^T\mathbf{U}$ and $\mathbf{U}^T\mathbf{B}$ are derived:

$$\mathbf{B}^T\mathbf{U} = \begin{bmatrix} \mathbf{U}^T\mathbf{U} \\ \mathbf{W}^T\mathbf{U} \end{bmatrix} = \begin{bmatrix} \mathbf{I}_k \\ \mathbf{0} \end{bmatrix}, \tag{2.95}$$

$$\mathbf{U}^T\mathbf{B} = [\mathbf{U}^T\mathbf{U},\ \mathbf{U}^T\mathbf{W}] = [\mathbf{I}_k,\ \mathbf{0}]\,. \tag{2.96}$$

Therefore

$$\begin{aligned} \mathbf{B}^T\mathbf{U}(\mathbf{T}^{-1} - \mathbf{I}_k)\mathbf{U}^T\mathbf{B} &= \begin{bmatrix} \mathbf{I}_k \\ \mathbf{0} \end{bmatrix} [\mathbf{T}^{-1} - \mathbf{I}_k]\ [\mathbf{I}_k,\ \mathbf{0}] \\ &= \begin{bmatrix} \mathbf{T}^{-1} - \mathbf{I}_k & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \end{aligned} \tag{2.97}$$

and by combining (2.89), (2.94) and (2.97), a matrix similar to $\mathbf{M}_2$ is found to be

$$\mathbf{B}^T\mathbf{M}_2\mathbf{B} = \mathbf{I}_N + \begin{bmatrix} \mathbf{T}^{-1} - \mathbf{I}_k & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{T}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{N-k} \end{bmatrix}. \tag{2.98}$$

Combining (2.93) and (2.98) yields

$$\mathbf{B}^T\mathbf{J}\mathbf{B}\cdot\mathbf{B}^T\mathbf{M}_2\mathbf{B} = \begin{bmatrix} \mathbf{T} & \tilde{\mathbf{J}}_{12} \\ \mathbf{0} & \tilde{\mathbf{J}}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{T}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{N-k} \end{bmatrix}. \tag{2.99}$$

Performing the multiplication on the right hand side, again using (2.89), (2.99) becomes

$$\mathbf{B}^T\,\mathbf{J}\mathbf{M}_2\,\mathbf{B} = \begin{pmatrix} \mathbf{I}_k & \tilde{\mathbf{J}}_{12} \\ \mathbf{0} & \tilde{\mathbf{J}}_{22} \end{pmatrix} \tag{2.100}$$

which shows that the k smallest eigenvalues of $\mathbf{J}\mathbf{M}_2$ have all been mapped to unity.

The final issue to consider is where the k approximate eigenvectors $\mathbf{u}_i$, needed to form this preconditioner, come from. In fact, they are the very same $\mathbf{u}_i$ that are computed during a cycle of GMRES-DR (2.81). It may seem unnecessary to form an explicit preconditioner based on these vectors, when they are already included in the augmented Krylov subspace (2.78). The reason for doing so, is that once these vectors are included in the preconditioner, subsequent cycles of GMRES-DR begin to deflate the *next* k smallest eigenvalues. In fact, this process could be continued, forming preconditioners $\mathbf{M}_3, \mathbf{M}_4, \ldots$, each explicitly deflating further sets of small eigenvalues.

Dual preconditioning strategy

The two preconditioning techniques presented here have been found to work well in combination (see Section 4.4.1). The numerical experiments show that when operating on a Jacobian matrix $\mathbf{J}$ possessing many large eigenvalues, the effect of the approximate inverse preconditioner $\mathbf{M}_1$ is to scale these large eigenvalues towards unity. Unfortunately, the small eigenvalues are not necessarily eliminated, and in fact their magnitudes may even be

reduced. The deflation matrix $\mathbf{M}_2$ is used to handle this. When operating on the matrix $\mathbf{JM}_1$, the effect of this preconditioner is to deflate the smallest eigenvalues while leaving the others unchanged (Section 2.5.2). Thus the combination of the two preconditioners in the order $\mathbf{JM}_1\mathbf{M}_2$ has the desired overall effect on the spectrum.

In terms of the two-stage nonlinear solution process (Table 2.3), the combination $\mathbf{M}_1\mathbf{M}_2$ is only applied during stage two. Only $\mathbf{M}_1$ is applied at the commencement of stage one, since no GMRES-DR iterations have been processed yet and thus no $\mathbf{u}_i$ vectors are available. Experimentally, it has been found that $\mathbf{M}_1$ alone is sufficient for satisfactory convergence of stage one. When this stage is complete, the most recent $\mathbf{u}_i$ vectors are used to construct $\mathbf{M}_2$ for use during stage two. Note again, that CV-FE information (the $\mathbf{u}_i$) is being used to precondition the CV-RBF iterations. This is an example of the technique discussed in Section 2.4.3 of using a simplification of the problem to construct an effective preconditioner.

2.6 Parallelisation

Discretising PDEs over fine, three-dimensional meshes can result in very large Jacobian matrices. Processing a Newton-Krylov method for such a problem, even if done in the Jacobian-free manner outlined in Section 2.4.1, can be extremely time consuming when running on a single processor. With the advent of parallel computing, this problem can potentially be alleviated by spreading the computational load over many processors simultaneously.

Programming for a parallel environment is inherently more difficult than programming in serial, as it involves thinking in a way that is different from that of serial programming [80]. Two particular issues that are present in in

parallel programming are

1. ensuring that algorithms are amenable to workload-sharing over multiple processors.

2. ensuring that algorithms behave correctly when running on multiple processors simultaneously.

The first issue concerns writing algorithms whose workload can be broken up into smaller portions to be processed simultaneously. After all processes have finished processing their individual portions of data, the partial results are combined to produce the final result. This strategy assumes that each process may work completely independently of all others.

The second issue concerns writing algorithms that safely share data among multiple processes. Generally there is no issue with multiple processes simultaneously reading the same data. However, any data structure that is being written to by one process, and either read from or written to by any other process must be accessed in a controlled fashion to ensure that the data remain consistent.

The implementation written for this thesis uses the *OpenMP* [8] application programming interface (API) for parallel programming under a *shared memory architecture*. The API consists of various functions and directives that control how variables in memory are accessed by separate processes, how many processes are active at any one time, and how the workload is distributed between them. Through the use of *parallel for loops* (relating to point 1 above), and *critical sections* (relating to point 2 above), high-level parallelism is achieved in the Newton-Krylov method for solving (2.53).

In particular, it is the preconditioning component of the Newton-Krylov method that is the primary source of parallelism in this implementation. As

discussed in [51, 74], effective preconditioners can be very expensive to generate, particularly for large matrices. However, once generated, they can be efficiently applied to each subsequent iteration of the Newton-Krylov method and improve the rate of convergence accordingly. Therefore, parallel algorithms for generating effective preconditioners are highly valued by those wishing to solve large problems using Newton-Krylov methods.

Of the two preconditioners presented in Section 2.5.2, the first, $\mathbf{M}_1$ was specifically designed to be run in a parallel environment. The fact that problem (2.84) can be split into N independent least-squares problems (2.85) allows the computation to be shared over multiple processors [74], with each process responsible for computing one or more distinct columns of the matrix $\mathbf{M}_1$.

The second preconditioner, $\mathbf{M}_2$ of Section 2.5.2 is not inherently parallelisable, as it results from eigenvector information generated during GMRES-DR iterations. It is the generation of the Krylov subspace, an inherently serial operation, that comprises the most work in this algorithm. Once this subspace has been generated, there is little extra work required to compute Harmonic Ritz vectors, and assemble them into the form required in (2.86).

In summary, it is through the preconditioning of the Jacobian-free Newton-Krylov method that parallelism is achieved in this finite volume implementation. In particular, the generation of the norm-minimising, approximate inverse preconditioner $\mathbf{M}_1$ of Section 2.5.2 is carried out in parallel, allowing it to be applied to problems whose size would prohibit the generation of such a preconditioner in serial. Section 6.4.1 exhibits the parallel speedup obtained with this approach for one particular three-dimensional problem.

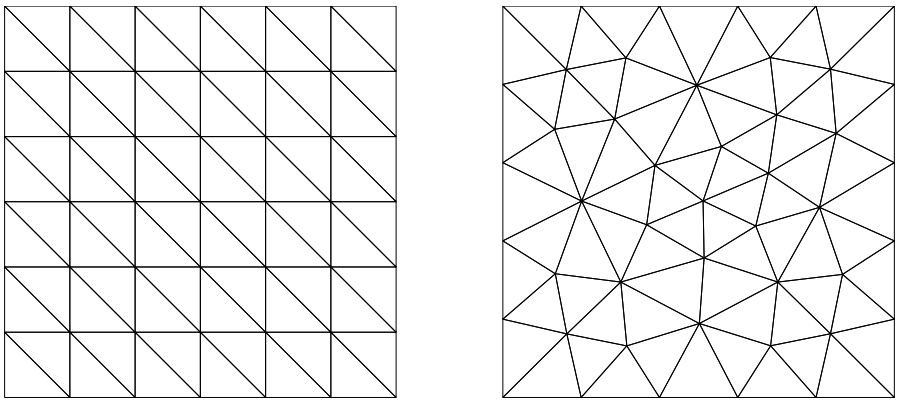
Chapter 3

Two-dimensional considerations

In this chapter the details of implementing the finite volume method in two dimensions are presented. Note that throughout this chapter, three-dimensional terms such as "volume", "area" and "face" will often be used when referring to the analogous two-dimensional concepts of "area", "length" and "edge". This is for consistency with the discussion in Chapter 2, and because these terms are standard in the finite volume literature. On the other hand, whenever mathematical equations are presented, they will always be written in terms of two-dimensional concepts (for example, using line integrals rather than surface integrals).

3.1 Meshing

In Section 2.1 the mesh-based nature of the finite volume method was discussed. The standard finite element mesh used with the finite volume method consists of a set of nodes, and a corresponding set of connections between the nodes, forming elements. There are many different ways to connect a set of nodes to form the elements of a mesh. Common element types in two-



(a) (b)

Figure 3.1: Triangular meshes: (a) structured; (b) unstructured.

dimensional meshes are triangles and quadrilaterals. Most mesh generation software can generate at least one of these two types of meshes. Less common is the ability to generate mixed meshes, where a combinations of different element types are used. In this chapter, the assumption is that all meshes are triangular, although the concepts could easily be generalised.

Meshes may also be either *structured* or *unstructured*. A structured mesh is created using a regular grid of elements. An example is shown in Figure 3.1(a). Structured meshes are most useful when the underlying domain is composed of simple geometric regions that lend themselves to a regular grid. Some numerical PDE solution methods, such as the finite difference method, are almost exclusively applied over structured meshes. For problems where creating structured meshes is awkward, these methods can be difficult to apply.

The alternative to a structured mesh is an unstructured mesh. In an unstructured mesh there is no underlying grid. An example is shown in Figure 3.1(b). The freedom offered by unstructured meshes allows them to

closely fit irregular and complex boundaries, which are often encountered in real-world problems. Additionally, the mesh can be *refined* in regions where the underlying physical problem warrants closer attention.

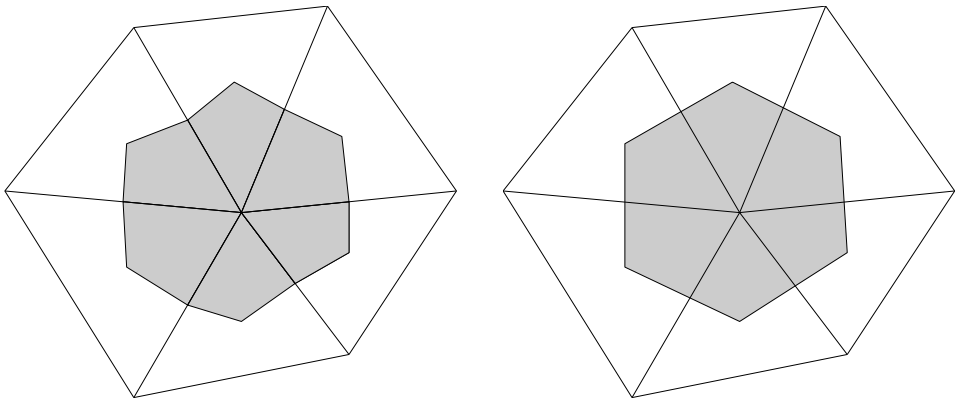
The cost of using unstructured meshes is added complexity, both in the mesh generation software and in the mesh-based PDE solver. Generating good quality, unstructured meshes is a difficult problem, and certainly one best left to dedicated software. For this thesis, all meshes were generated using the *Gmsh* software [31].

3.2 Forming control volumes

Finite volume discretisation relies on constructing control volumes around each node in the mesh. In this section, the method used in the code written for this thesis, for taking the nodes and elements defined in a two-dimensional, finite element mesh and constructing control volumes is presented.

Figure 3.2 illustrates a small section of mesh, with six elements all sharing a common vertex. Around this node, a control volume has been constructed using two different methods. In Figure 3.2(a), the centroids of the elements have been connected to the midpoints of the faces. In Figure 3.2(b) the centroids of the elements have been connected directly to one another. Both methods have their advantages and disadvantages, which will be addressed in Section 3.4.2.

Figure 3.3 illustrates a section of mesh where the central node lies on a boundary. When constructing *boundary control volumes* such as the one shown here, method (b) must be altered slightly. Since boundary elements do not have neighbours on all sides, the element centroids of these boundary ele-



(a) (b)

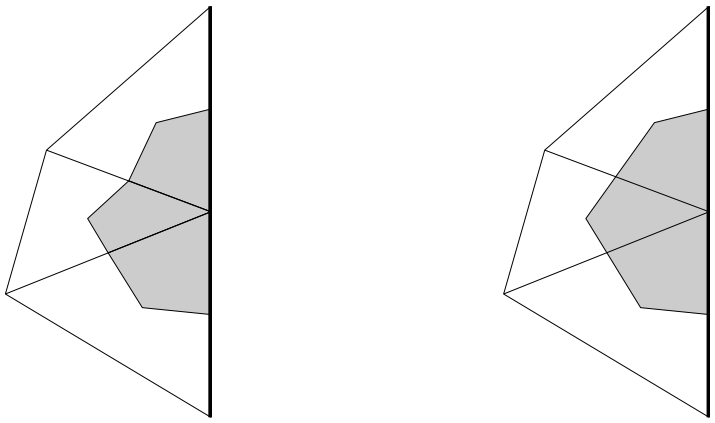
Figure 3.2: Forming control volumes: (a) joining centroids and midpoints; (b) joining centroids and centroids.

ments are connected to the boundary face midpoint. This is true irrespective of whether method (a) or method (b) is used elsewhere.

3.2.1 Sub-control volumes

The volume of a control volume is the sum of the contributions from each of the elements that share its node. Each of these contributions is known as a *sub-control volume*. The control volumes illustrated in Figure 3.2 are composed of six sub-control volumes. The boundary control volumes illustrated in Figure 3.3 are composed of three sub-control volumes.

An element may also be considered as being composed of sub-control volumes. Unlike with control volumes however, the number of sub-control volumes comprising a triangular element does not vary: it is always three. Figure 3.4(a) shows the three sub-control volumes of a particular element. Figure 3.4(b) shows a single sub-control volume. Figure 3.5 shows how an entire mesh is partitioned into control volumes and sub-control volumes,



(a) (b)

Figure 3.3: Forming boundary control volumes: (a) joining centroids and midpoints; (b) joining centroids and centroids.

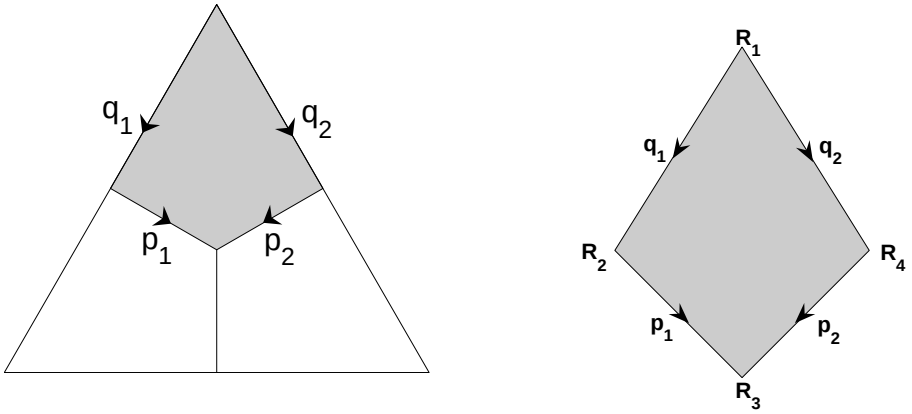
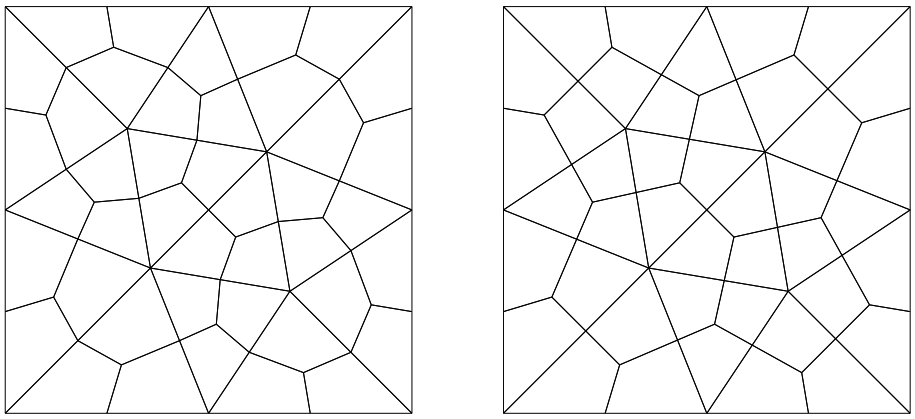


Figure 3.4: A sub-control volume (a) as one of three comprising an element; (b) on its own.



(a) (b)

Figure 3.5: A mesh partitioned into control volumes using (a): method (a); (b): method (b).

using methods (a) and (b).

It is helpful to consider the four faces of a sub-control volume as being divided into pairs. The first pair of faces, which in this thesis are called the *inner faces*, are those that are internal to a non-boundary control volume. They are marked in Figure 3.4 by the vectors $\mathbf{q}_1$ and $\mathbf{q}_2$, with the convention that these vectors point away from the element vertex. The other pair of faces, which in this thesis are called the *outer faces*, form part of the surface of each control volume. They are marked in Figure 3.4 by the vectors $\mathbf{p}_1$ and $\mathbf{p}_2$, with the convention that these vectors point to the element centroid. The numbering of the $\mathbf{p}_i$ and $\mathbf{q}_i$ is such that the vector

$$\mathbf{n} = (\mathbf{p}_1 \cdot \mathbf{j})\,\mathbf{i} - (\mathbf{p}_1 \cdot \mathbf{i})\mathbf{j}\,, \tag{3.1}$$

when positioned as shown in Figure 3.6, points anticlockwise around the element.

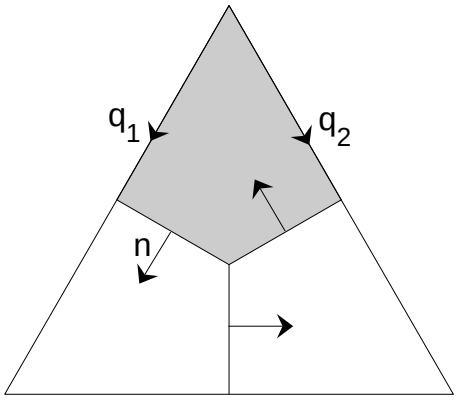


Figure 3.6: Correct, anticlockwise ordering

3.3 Conventions

In Section 2.3.6 the underlying conservation laws governing many of the physical processes modelled by PDEs were discussed. Recall from that discussion that in order to ensure these conservation laws hold at the discrete level, the fluxes computed through control volume faces need to be consistent from both directions. In this section the implementation aspects of ensuring this consistency in two dimensions are discussed, along with other numbering conventions.

A triangular element consists of three nodes, three faces and three sub-control volumes. A simple numbering scheme is used to ensure one can easily associate nodes, faces and sub-control volumes with one another. Each node, face and sub-control volume is numbered from 1 to 3 as shown in Figure 3.7. Node 1 belongs to sub-control volume 1 and lies opposite face 1, with the corresponding conventions for nodes 2 and 3. Note that this is a *local* numbering scheme particular to a given element. *Globally*, every node, face and sub-control volume in the mesh has a unique identifier, with no particular

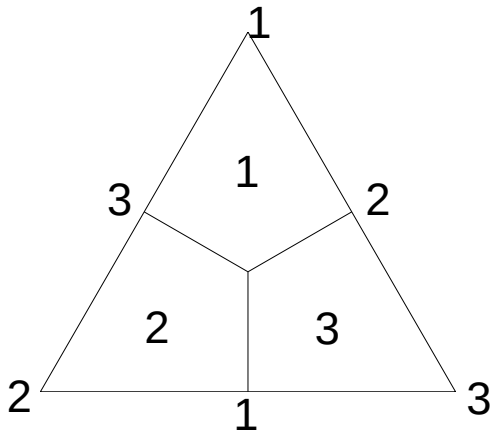


Figure 3.7: Numbering convention for nodes, faces and sub-control volumes.

relationship between them.

The finite volume implementation written for this thesis processes the entire mesh element-wise. Within a given element, the flux through each sub-control volume face is computed just once, in the direction of the normal, $\mathbf{n}$, given in (3.1). With the appropriate modification in sign, this common value is used to update both control volumes that share that face.

Additional fluxes must be processed when the element lies on a boundary. Figure 3.8 shows the additional normals on the "inner" faces for a particular boundary element. Only within boundary elements are these inner faces processed. For non-boundary elements, the inner faces are, as the name suggests, internal to the control volume, so fluxes through these faces do not need to be accounted for.

3.4 Integration

This section addresses how the integration methods discussed in Section 2.2 are applied in two dimensions. In (2.16) the integral over a control volume surface is expressed as the sum of integrals over each of its faces. In two di-

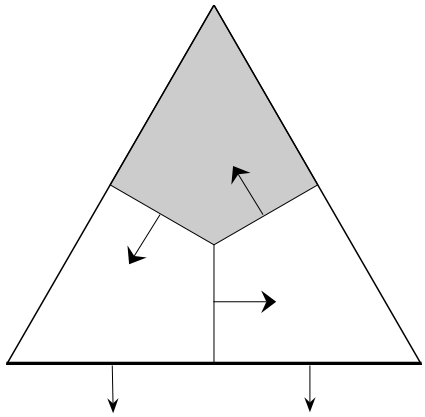


Figure 3.8: Outer normals for a boundary element.

mensions, these surface integrals are more correctly expressed as line integrals (refer to Figure 3.9):

$$\begin{aligned}\int_{C_i} (\mathbf{D}\boldsymbol{\nabla}\varphi \cdot \hat{\mathbf{n}} - \mathbf{v}\varphi)\ \mathrm{d}s &= \sum_{\tau=1}^{Nf_i} \int_{C_{i\tau}} (\mathbf{D}\boldsymbol{\nabla}\varphi \cdot \hat{\mathbf{n}} - \mathbf{v}\varphi)\ \mathrm{d}s \\ &= \sum_{\tau=1}^{Nf_i} \int_{\mathbf{x}_{i\tau}}^{\mathbf{x}_{i\tau+1}} (\mathbf{D}\boldsymbol{\nabla}\varphi \cdot \hat{\mathbf{n}} - \mathbf{v}\varphi)\ \mathrm{d}s. \qquad (3.2)\end{aligned}$$

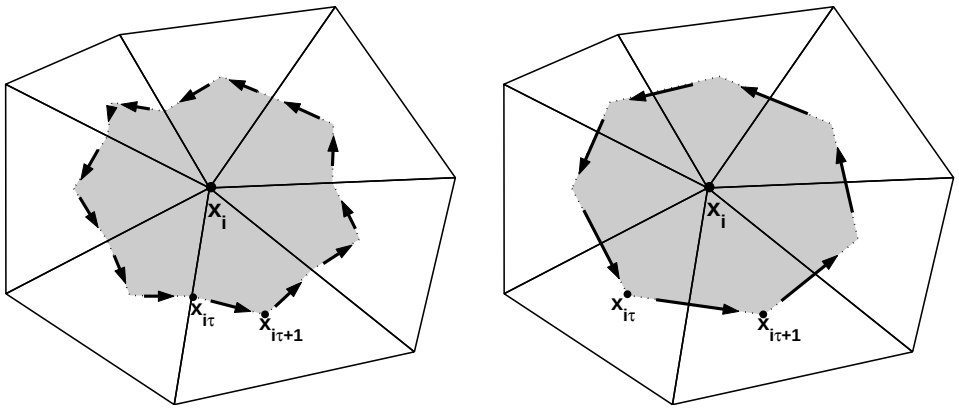
3.4.1 Midpoint rule

The midpoint rule may be used to compute (3.2):

$$\int_{\mathbf{x}_{i\tau}}^{\mathbf{x}_{i\tau+1}} (\mathbf{D}\boldsymbol{\nabla}\varphi \cdot \hat{\mathbf{n}} - \mathbf{v}\varphi)\ \mathrm{d}s \approx \|\mathbf{x}_{i\tau+1} - \mathbf{x}_{i\tau}\|\, [\mathbf{D}\boldsymbol{\nabla}\varphi \cdot \hat{\mathbf{n}} - \mathbf{v}\varphi]_{\mathbf{m}_{i\tau}} \qquad (3.3)$$

yielding the approximation

$$\int_{C_i} (\mathbf{D}\boldsymbol{\nabla}\varphi \cdot \hat{\mathbf{n}} - \mathbf{v}\varphi)\ \mathrm{d}s \approx \sum_{\tau=1}^{Nf_i} \|\mathbf{x}_{i\tau+1} - \mathbf{x}_{i\tau}\|\, [\mathbf{D}\boldsymbol{\nabla}\varphi \cdot \hat{\mathbf{n}} - \mathbf{v}\varphi]_{\mathbf{m}_{i\tau}} \qquad (3.4)$$



(a) (b)

Figure 3.9: Line integral paths for control volume methods (a) and (b).

where

$$\mathbf{m}_{i\tau} = \frac{1}{2}(\mathbf{x}_{i\tau} + \mathbf{x}_{i\tau+1}) \tag{3.5}$$

is the face midpoint.

As for the source component, it may be approximated using (2.19), which requires computing the volume ΔV_i. This is achieved by summing the volumes of each of V_i's sub-control volumes. Each sub-control volume is a convex quadrilateral (see Figure 3.4(b)), whose volume can be computed using the following formula [91]:

$$\text{Volume} = \frac{1}{2}\left\|(\mathbf{q}_1 + \mathbf{p}_1) \times (\mathbf{p}_2 - \mathbf{p}_1)\right\| \tag{3.6}$$

with the two-dimensional cross product norm in (3.6) defined by

$$\|\mathbf{a} \times \mathbf{b}\| = \|(a_x, a_y)^{\mathrm{T}} \times (b_x, b_y)^{\mathrm{T}}\| = |a_x b_y - a_y b_x|. \tag{3.7}$$

3.4.2 Gaussian quadrature

Gaussian quadrature was discussed in Section 2.2.2 as a means of computing more accurate approximations to both the advective-diffusive flux and source terms of (2.10).

First the advective-diffusive flux component is considered. In this case, each integral

$$\int_{C_{i\tau}} (\mathbf{D}\boldsymbol{\nabla}\varphi - \mathbf{v}\varphi) \cdot \hat{\mathbf{n}} \, \mathrm{d}s = \int_{\mathbf{x}_{i\tau}}^{\mathbf{x}_{i\tau+1}} (\mathbf{D}\boldsymbol{\nabla}\varphi - \mathbf{v}\varphi) \cdot \hat{\mathbf{n}} \, \mathrm{d}s \tag{3.8}$$

must be computed using the Gaussian quadrature formula (2.21). To do so, the following parameterisation for $C_{i\tau}$ is introduced:

$$\mathbf{x}(u) = \frac{1}{2}(1-u)\mathbf{x}_{i\tau} + \frac{1}{2}(1+u)\mathbf{x}_{i\tau+1}, \quad -1 \leq u \leq 1 \tag{3.9}$$

so that

$$\|\dot{\mathbf{x}}(u)\| = \frac{1}{2}\|\mathbf{x}_{i\tau+1} - \mathbf{x}_{i\tau}\|. \tag{3.10}$$

Applying (3.9) and (3.10) to (3.8) yields

$$\int_{C_{i\tau}} (\mathbf{D}\boldsymbol{\nabla}\varphi \cdot \hat{\mathbf{n}} - \mathbf{v}\varphi) \, \mathrm{d}s = \int_{-1}^{1} (\mathbf{D}\boldsymbol{\nabla}\varphi \cdot \hat{\mathbf{n}} - \mathbf{v}\varphi) \, \|\dot{\mathbf{x}}(u)\| \mathrm{d}u \tag{3.11}$$

$$= \frac{1}{2}\|\mathbf{x}_{i\tau+1} - \mathbf{x}_{i\tau}\| \int_{-1}^{1} (\mathbf{D}\boldsymbol{\nabla}\varphi \cdot \hat{\mathbf{n}} - \mathbf{v}\varphi) \, \mathrm{d}u. \tag{3.12}$$

The issue of whether to use two-point or three-point Gaussian quadrature is determined by whether method (a) or method (b) from Section 3.2.1 has been used to form control volumes. In the case of method (a), Figure 3.10 illustrates how two line integrals are required to compute the flux between adjoining control volumes. Two-point Gaussian quadrature is applied to compute each integral. From Table 2.1, the abscissas and weights are such

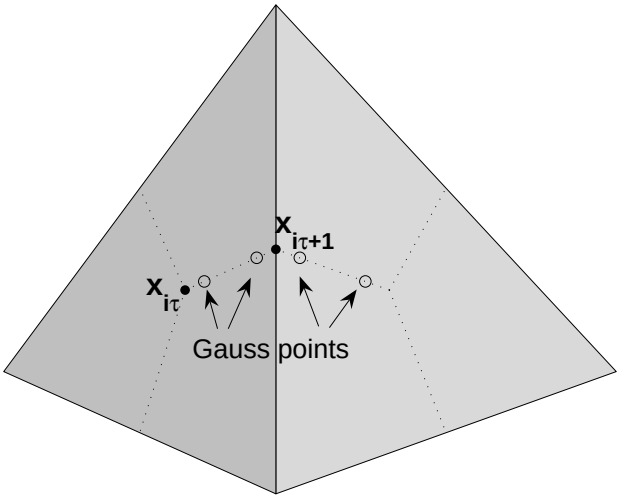


Figure 3.10: Computing flux using two-point Gaussian quadrature.

that

$$\int_{-1}^{1} f(u)\,\mathrm{d}u \approx f\left(\frac{-\sqrt{3}}{3}\right) + f\left(\frac{\sqrt{3}}{3}\right). \tag{3.13}$$

Applying (3.13) to (3.12) yields

$$\begin{aligned}\int_{C_{i\tau}} (\mathbf{D}\boldsymbol{\nabla}\varphi\cdot\hat{\mathbf{n}} - \mathbf{v}\varphi)\,\mathrm{d}s \approx \frac{1}{2}\|\mathbf{x}_{i\tau+1} - \mathbf{x}_{i\tau}\| \\ \times\ \left\{[\mathbf{D}\boldsymbol{\nabla}\varphi\cdot\hat{\mathbf{n}} - \mathbf{v}\varphi]_{\mathbf{x}(u_1)} + [\mathbf{D}\boldsymbol{\nabla}\varphi\cdot\hat{\mathbf{n}} - \mathbf{v}\varphi]_{\mathbf{x}(u_2)}\right\},\end{aligned} \tag{3.14}$$

where

$$\begin{aligned}\mathbf{x}(u_1) &= \mathbf{x}\left(\frac{-\sqrt{3}}{3}\right) \\ &= \frac{1}{2}\left(1+\frac{\sqrt{3}}{3}\right)\mathbf{x}_i + \frac{1}{2}\left(1-\frac{\sqrt{3}}{3}\right)\mathbf{x}_{i\tau+1} \\ &= \frac{1}{2}\left[(\mathbf{x}_{i\tau}+\mathbf{x}_{i\tau+1}) - \frac{\sqrt{3}}{3}(\mathbf{x}_{i\tau+1}-\mathbf{x}_{i\tau})\right],\end{aligned} \tag{3.15}$$

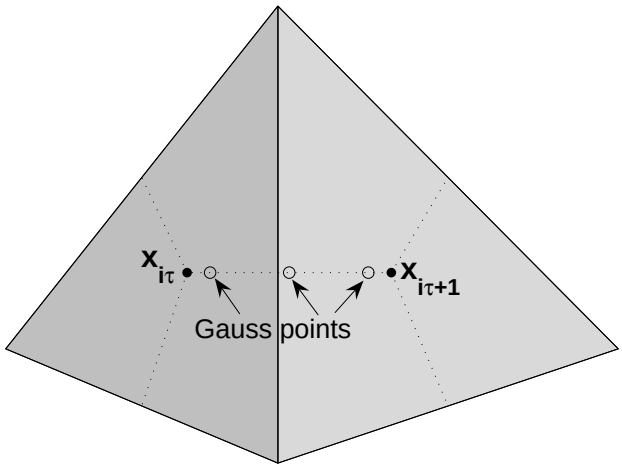


Figure 3.11: Computing flux using three point Gaussian quadrature.

and similarly

$$\mathbf{x}(u_2) = \frac{1}{2}\left[(\mathbf{x}_{i\tau} + \mathbf{x}_{i\tau+1}) + \frac{\sqrt{3}}{3}(\mathbf{x}_{i\tau+1} - \mathbf{x}_{i\tau})\right]. \tag{3.16}$$

In the case of method (b), Figure 3.11 illustrates how only one line integral is required to compute the flux between adjoining control volumes. Three-point Gaussian quadrature is applied to compute this integral. From Table 2.1,

$$\int_{-1}^{1} f(u)\,\mathrm{d}u \approx \frac{5}{9} f\left(\frac{-\sqrt{3}}{\sqrt{5}}\right) + \frac{8}{9} f(0) + \frac{5}{9} f\left(\frac{\sqrt{3}}{\sqrt{5}}\right). \tag{3.17}$$

Applying (3.17) to (3.12) yields

$$\begin{aligned}\int_{C_{i\tau}} (\mathbf{D}\boldsymbol{\nabla}\varphi \cdot \hat{\mathbf{n}} - \mathbf{v}\varphi)\,\mathrm{d}s \approx \frac{1}{2}\|\mathbf{x}_{i\tau+1} - \mathbf{x}_{i\tau}\| \times \Big\{ &[\mathbf{D}\boldsymbol{\nabla}\varphi \cdot \hat{\mathbf{n}} - \mathbf{v}\varphi]_{\mathbf{x}(u_1)} \\ &+ [\mathbf{D}\boldsymbol{\nabla}\varphi \cdot \hat{\mathbf{n}} - \mathbf{v}\varphi]_{\mathbf{x}(u_2)} + [\mathbf{D}\boldsymbol{\nabla}\varphi \cdot \hat{\mathbf{n}} - \mathbf{v}\varphi]_{\mathbf{x}(u_3)} \Big\}\end{aligned} \tag{3.18}$$

where

$$\begin{aligned}\mathbf{x}(u_1) &= \mathbf{x}\left(\frac{-\sqrt{3}}{\sqrt{5}}\right) \\ &= \frac{1}{2}\left(1+\frac{\sqrt{3}}{\sqrt{5}}\right)\mathbf{x}_{i\tau} + \frac{1}{2}\left(1-\frac{\sqrt{3}}{\sqrt{5}}\right)\mathbf{x}_{i\tau+1} \\ &= \frac{1}{2}\left[(\mathbf{x}_i + \mathbf{x}_{i\tau+1}) - \frac{\sqrt{3}}{\sqrt{5}}(\mathbf{x}_{i\tau+1} - \mathbf{x}_{i\tau})\right], \end{aligned} \tag{3.19}$$

$$\begin{aligned}\mathbf{x}(u_2) &= \mathbf{x}(0) \\ &= \frac{1}{2}(\mathbf{x}_{i\tau+1} + \mathbf{x}_{i\tau}) \end{aligned} \tag{3.20}$$

and, in a similar manner to (3.19),

$$\mathbf{x}(u_3) = \frac{1}{2}\left[(\mathbf{x}_i + \mathbf{x}_{i\tau+1}) + \frac{\sqrt{3}}{\sqrt{5}}(\mathbf{x}_{i\tau+1} - \mathbf{x}_{i\tau})\right]. \tag{3.21}$$

It is of interest to compare the accuracy of the Gaussian quadrature integration schemes for control volume methods (a) and (b). In the case of control volume method (a), the two-point scheme can be shown to be $\mathcal{O}(h^4)$, where h is the control volume face length [21]. In the case of method (b), the three-point scheme can likewise be shown to be $\mathcal{O}(h^6)$, however the control volume face length h in this case is approximately twice that for method (a) (compare Figures 3.10 and 3.11).

Figure 3.12 is a comparison of the accuracy offered by the two and three-point Gaussian quadrature schemes, along with the midpoint rule, for meshes of various refinements. These plots were produced by using the exact gradients taken from the analytic solution of Test Problem 1 in Chapter 4. These values were then used in the various quadrature methods to compute

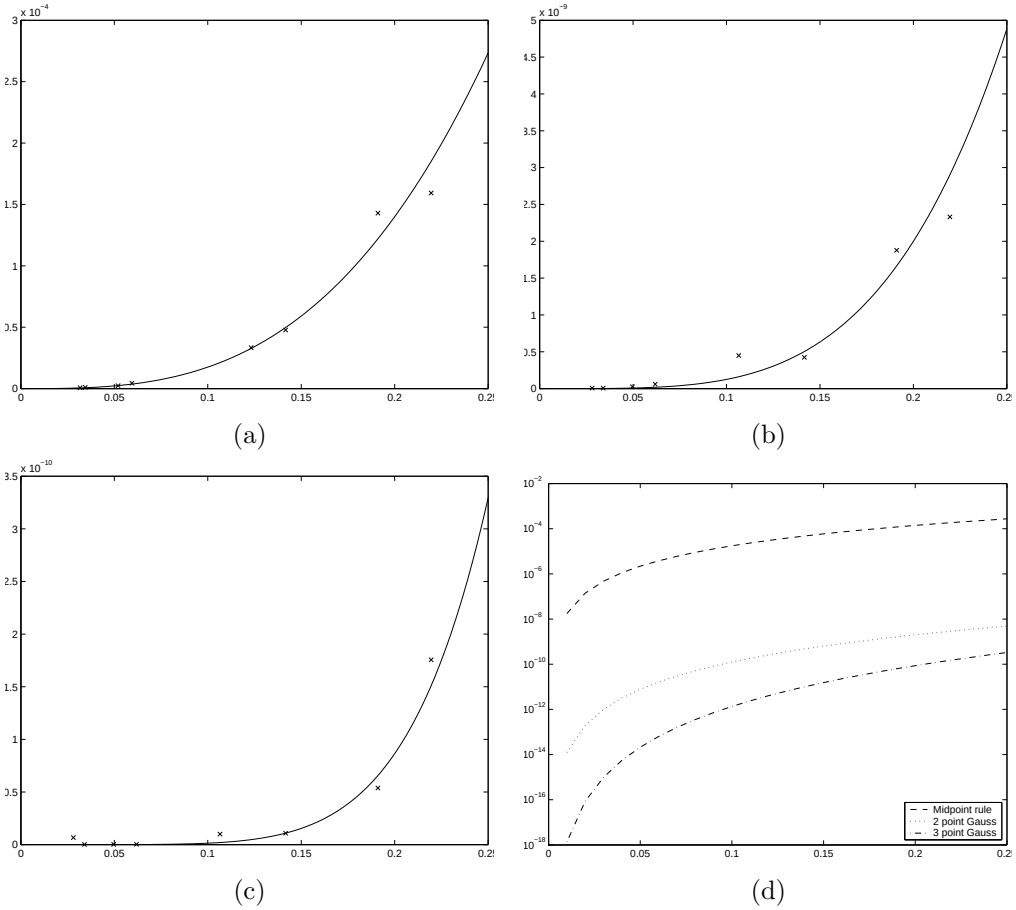


Figure 3.12: Accuracy of line integrals versus face length: (a) Midpoint rule; (b) Two-point Gaussian quadrature; (c) Three-point Gaussian quadrature; (d) Comparison of all three methods on a vertical log scale.

the fluxes across each control volume face. The graphs of $\mathcal{O}(h^2)$, $\mathcal{O}(h^4)$ and $\mathcal{O}(h^6)$ are superimposed over the midpoint (a), two-point (b) and three-point (c) rules respectively. The fourth plot (d) compares the three methods on a vertical-log scale, demonstrating that the higher-order, three-point method is the most accurate, despite its application over larger face lengths.

Therefore, throughout this thesis, method (b) is used, whereby control volumes in two dimensions are constructed by connecting element centroids to centroids. This method is used both because it is more accurate, and

because it requires one fewer flux evaluation per control volume face (or pair of faces) than does method (a).

As for the source component of (2.10), it may be computed using Gaussian quadrature in two dimensions. The extension of (2.21) to two dimensions is given in [10]:

$$\int_{-1}^{1}\int_{-1}^{1} g(u,v)\,\mathrm{d}u\,\mathrm{d}v \approx \sum_{j=1}^{p}\sum_{k=1}^{p} w_j w_k\, g(u_j,u_k) \tag{3.22}$$

with weights w_i and abscissas u_i still given by Table 2.1. Applying (3.22) to (2.20) requires mapping each sub-control volume to $[-1,1]\times[-1,1]$. This can be achieved with the following parameterisation involving the sub-control volume vertices $\mathbf{R}_i$ (see Figure 3.4(b)):

$$\begin{aligned}\mathbf{x}(\xi,\eta) = {} & \frac{1}{4}(1-\xi)(1-\eta)\mathbf{R}_1 + \frac{1}{4}(1+\xi)(1-\eta)\mathbf{R}_2 \\ & + \frac{1}{4}(1+\xi)(1+\eta)\mathbf{R}_3 + \frac{1}{4}(1-\xi)(1+\eta)\mathbf{R}_4, \\ & -1 \le \xi \le 1, \quad -1 \le \eta \le 1\,,\end{aligned} \tag{3.23}$$

with the Jacobian of the transformation given by

$$\left|\frac{\partial \mathbf{x}}{\partial(\xi,\eta)}\right| = \left\|\frac{\partial \mathbf{x}}{\partial \xi} \;\times\; \frac{\partial \mathbf{x}}{\partial \eta}\right\|. \tag{3.24}$$

Applying (3.23) and (3.24) to (2.20) yields

$$\iint_{V_i} S\,\mathrm{d}A = \int_{-1}^{1}\int_{-1}^{1} S(\varphi(\mathbf{x}(\xi,\eta)))\left\|\frac{\partial \mathbf{x}}{\partial \xi} \;\times\; \frac{\partial \mathbf{x}}{\partial \eta}\right\| \mathrm{d}\xi\,\mathrm{d}\eta \tag{3.25}$$

$$\approx \sum_{j=1}^{p}\sum_{k=1}^{p} w_j w_k\, S(\varphi(\mathbf{x}(u_j,u_k)))\left\|\frac{\partial \mathbf{x}}{\partial \xi} \;\times\; \frac{\partial \mathbf{x}}{\partial \eta}\right\|_{(u_j,u_k)}. \tag{3.26}$$

3.5 Interpolation

3.5.1 Shape functions

In two dimensions, the method of shape functions produces a linear interpolant $s_i(x, y)$ over each triangular element. If the vertices of the element are (x_1, y_1), (x_2, y_2) and (x_3, y_3), the shape function interpolant is given by

$$s_i(x, y) = \sum_{j=1}^{3} N_j(x, y)\varphi_j \tag{3.27}$$

where the shape functions $N_j(x, y)$ are given by

$$N_1(x, y) = \frac{x_2y_3 - x_3y_2 + (y_2 - y_3)x + (x_3 - x_2)y}{\Delta}, \tag{3.28}$$

$$N_2(x, y) = \frac{x_3y_1 - x_1y_3 + (y_3 - y_1)x + (x_1 - x_3)y}{\Delta}, \tag{3.29}$$

$$N_3(x, y) = \frac{x_1y_2 - x_2y_1 + (y_1 - y_2)x + (x_2 - x_1)y}{\Delta}, \tag{3.30}$$

and

$$\Delta = \begin{vmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{vmatrix} \tag{3.31}$$

is twice the triangular element's volume.

With s_i given by (3.27), $\boldsymbol{\nabla} s_i$ is thus computed as

$$\boldsymbol{\nabla} s_i(x, y) = \sum_{j=1}^{3} \boldsymbol{\nabla} N_j(x, y)\varphi_j \tag{3.32}$$

which is constant throughout the element.

3.5.2 Radial basis functions

The method of radial basis functions in two dimensions takes a set of scattered points $\{(x_j, y_j), j = 1, \ldots, n\}$, along with the corresponding function values $\{\varphi_j, j = 1, \ldots, n\}$, and fits an interpolating function $s_i(x, y)$ given by

$$s_i(x, y) = \sum_{j=1}^{n} \lambda_j \phi(r_j) + c_0 + c_1 x + c_2 y \tag{3.33}$$

with the conditions

$$s_i(x_j, y_j) = \varphi_j, \quad j = 1, 2, \ldots, n \tag{3.34}$$

and

$$\sum_{j=1}^{n} \lambda_j = \sum_{j=1}^{n} \lambda_j x_j = \sum_{j=1}^{n} \lambda_j y_j = 0 \tag{3.35}$$

where

$$r_j = \sqrt{(x - x_j)^2 + (y - y_j)^2}\,. \tag{3.36}$$

Recall from Section 2.3.3 that the multiquadric basis function $\phi(r) = \sqrt{c^2 + r^2}$ is used throughout this thesis. Figure 3.13 illustrates the multiquadric function in two dimensions, and the effect on its shape of the parameter c^2. In Figure 3.13(a) with $c^2 = 0$, the function is not a true multiquadric, but is actually the piecewise-smooth linear function given in Table 2.2. It is also not differentiable at the origin. Figures 3.13(b), 3.13(c) and 3.13(d) are multiquadrics with nonzero, and increasing values of c^2. The increase in c^2 results in a "flattening" of the multiquadric. This flattening is also associated with an increase in the condition number of the coefficient matrix $\mathbf{\Lambda}$ (2.36)

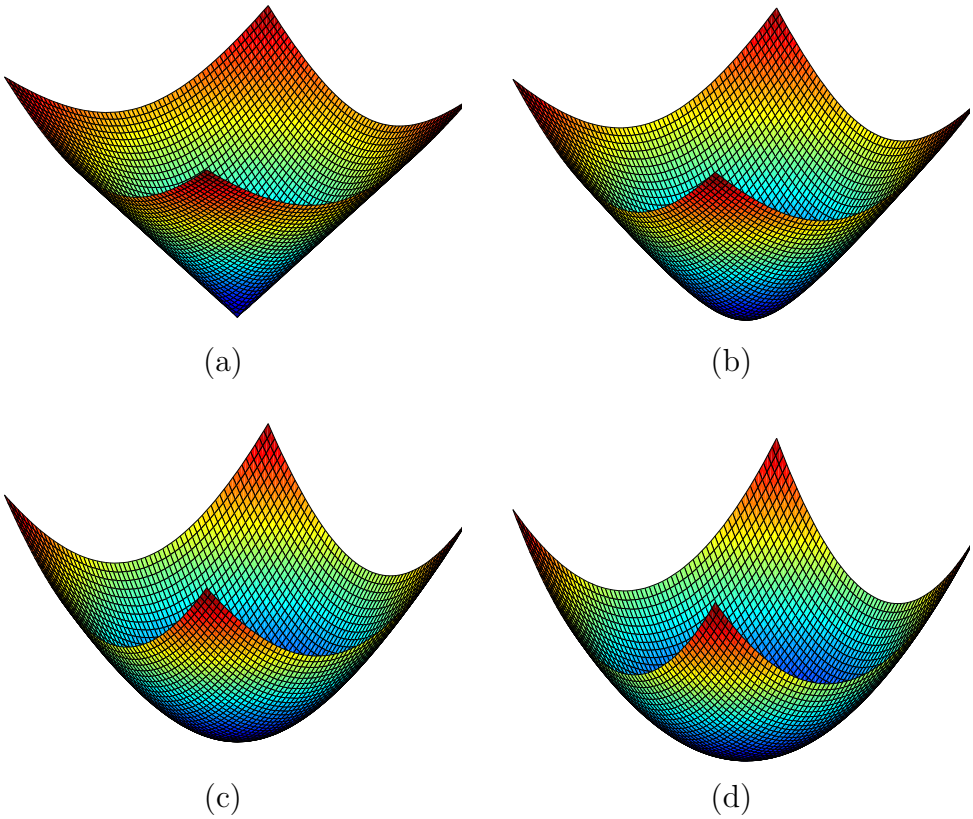


Figure 3.13: Multiquadrics in two dimensions with c^2 equal to: (a) 0 (linear); (b) 1; (c) 10; (d) 100.

[18, 53].

3.6 Finite volume discretisation

With the methods for integration and interpolation in place, the definition of the nonlinear component functions of the finite volume discretisation (2.12)

in two dimensions can be completed. In the case of CV-FE, f_i is given by

$$f_i(\boldsymbol{\varphi}) = \sum_{\tau=1}^{Nf_i} \|\mathbf{x}_{i\tau+1} - \mathbf{x}_{i\tau}\| \left[\mathbf{D}(s_i(\mathbf{m}_{i\tau}))\boldsymbol{\nabla} s_i(\mathbf{m}_{i\tau}) \cdot \hat{\mathbf{n}} - \mathbf{v}(s_i(\mathbf{m}_{i\tau}))s_i(\mathbf{m}_{i\tau})\right] + S(\mathbf{x}_i)\Delta V_i \tag{3.37}$$

where s_i is the shape function interpolant (3.27) and ΔV_i is the volume of control volume, V_i given by (3.6). In the case of CV-RBF, f_i is given by

$$\begin{aligned} f_i(\boldsymbol{\varphi}) = \frac{1}{2} \sum_{\tau=1}^{Nf_i} & \|\mathbf{x}_{i\tau+1} - \mathbf{x}_{i\tau}\| \\ & \times \sum_{j=1}^{3} \left[\mathbf{D}(s_i(\mathbf{x}(u_j)))\boldsymbol{\nabla} s_i(\mathbf{x}(u_j)) \cdot \hat{\mathbf{n}} - \mathbf{v}(s_i(\mathbf{x}(u_j)))s_i(\mathbf{x}(u_j))\right] \\ & + \sum_{j=1}^{2}\sum_{k=1}^{2} w_j w_k \, S(s_i(\mathbf{x}(u_j, u_k))) \left\| \frac{\partial \mathbf{x}_i}{\partial \xi} \times \frac{\partial \mathbf{x}}{\partial \eta} \right\|_{(u_j, u_k)} \end{aligned} \tag{3.38}$$

where s_i is the RBF interpolant (3.33).

In the next chapter, investigations into the accuracy and the efficiency of these different finite volume discretisation strategies are carried out, by using these methods to solve a number of test problems. The effectiveness of the Jacobian-free Newton-Krylov method and associated preconditioning techniques discussed in Chapter 2 perform are also investigated. Tests of the adaptive switching strategy and preconditioner parallelisation are deferred until Chapter 6 where they are tested on larger, three-dimensional problems.

Chapter 4

Numerical experiments in two dimensions

In this chapter the effectiveness of the two-dimensional schemes proposed in the previous chapter are investigated, by applying them to three different test problems. The first two sections of this chapter outline the test problems used, and the meshes used in solving them. The final two sections present the results of these tests, concentrating on the aspects of accuracy and efficiency respectively. Within these sections, the results are divided into subsections concentrating on specific aspects of the results, including the order of accuracy of the methods, the effectiveness of preconditioning, and the efficiency of the Jacobian-free Newton-Krylov method. To keep the number of results manageable, some subsections present results from only one test problem, while others present results from all three.

4.1 Test problems

In order that solid conclusions about the effectiveness of these methods can be drawn, the tests are performed exclusively on problems that have known analytic solutions. In general, only limited analytic solution techniques applicable to multi-dimensional advection-diffusion problems are available in the literature, so the problems in this section have been carefully crafted to ensure that they are amenable to analytic solution techniques. Two kinds of test problems are used:

- linear problems that give rise to analytic solution techniques.
- nonlinear problems constructed from known solutions.

In all cases, the test problems are based on the steady-state advection-diffusion problem

$$\boldsymbol{\nabla} \cdot (\mathbf{D}\boldsymbol{\nabla}\varphi - \mathbf{v}\varphi) + S = 0 \tag{4.1}$$

on the domain $(0,1) \times (0,1)$. For simplicity in discussing and visualising the solutions, the problem is interpreted as a heat transfer problem with $\mathbf{D}$ the thermal diffusivity, $\mathbf{v}$ the velocity of the medium, and S the volumetric source. Different choices for $\mathbf{D}$, $\mathbf{v}$ and S, along with different combinations of boundary conditions, give rise to the various test problems. The thermal diffusivity $\mathbf{D}$ is taken to have the form

$$\mathbf{D} = \begin{pmatrix} D_{xx} & 0 \\ 0 & D_{yy} \end{pmatrix} \tag{4.2}$$

where D_{xx} and D_{yy} are the diffusivities in the x and y directions respectively, and are potentially dependent on φ.

Table 4.1: Physical parameters for Test Problem 1.

Parameter	Description	Value
D_{xx}	Thermal diffusivity in x direction	5 $m^2 s^{-1}$
D_{yy}	Thermal diffusivity in y direction	$\{5, 50, 500, 5000\}$ $m^2 s^{-1}$
g_0	Source	10 $K m^{-2} s^{-1}$
h	Heat transfer coefficient	2 $W m^{-2} K^{-1}$
φ_∞	External temperature	20 K

4.1.1 Test Problem 1

The first test problem is the linear, steady state diffusion problem obtained by taking a constant $\mathbf{D}$ and setting $\mathbf{v} = \mathbf{0}$ along with $S = g_0$ (constant) in (4.1). The boundaries $x = 0$ and $y = 0$ are insulated, while the boundaries $x = 1$, $y = 1$ are subject to Newtonian cooling with external temperature φ_∞ and heat transfer coefficient h. This leads to the following boundary value problem:

$$\begin{aligned} D_{xx}\frac{\partial^2\varphi}{\partial x^2} + D_{yy}\frac{\partial^2\varphi}{\partial x^2} + g_0 &= 0 && \text{on } (0,1)\times(0,1) \\ \frac{\partial\varphi}{\partial x} &= 0 && \text{at } x = 0 \\ D_{xx}\frac{\partial\varphi}{\partial x} &= h(\varphi_\infty - \varphi) && \text{at } x = 1 \\ \frac{\partial\varphi}{\partial y} &= 0 && \text{at } y = 0 \\ D_{yy}\frac{\partial\varphi}{\partial y} &= h(\varphi_\infty - \varphi) && \text{at } y = 1 \end{aligned} \tag{4.3}$$

The parameter values used for this problem are listed in Table 4.1. For some tests, the parameter D_{yy} was varied by factors of ten to provide increasing challenges to the numerical solution methods.

Using the method of eigenfunction expansion, an analytic solution to

problem (4.3) is possible [37]. The spectral representations are

$$\delta(x-\xi) = \sum_{n=1}^{\infty} \frac{1}{N_x(\mu_n, x)} X(\mu_n, x) X(\mu_n, \xi) \tag{4.4}$$

$$\delta(y-\eta) = \sum_{m=1}^{\infty} \frac{1}{N_y(\lambda_m, y)} Y(\lambda_m, y) Y(\lambda_m, \eta) \tag{4.5}$$

where

$$X(\mu_n, x) = \mu_n \cos(\mu_n x)\,, \tag{4.6}$$

$$N_x(\mu) = \frac{\mu^2}{2}\left(1 + \frac{hD_{xx}}{h + \mu {D_{xx}}^2}\right) \tag{4.7}$$

and the μ_n are the positive roots of

$$\tan(\mu_n) = \frac{h}{\mu_n D_{xx}} \tag{4.8}$$

with similar expressions for Y, N_y and λ_m. Thus the solution is given by

$$\begin{aligned} \varphi(x,y) = \sum_{n=1}^{\infty}\sum_{m=1}^{\infty} \frac{\alpha(n,m) X(\mu_n, x) Y(\lambda_m, y)}{(D_{xx}\mu_n^2 + D_{yy}\lambda_m^2) N_x(\mu_n) N_y(\lambda_m)}\,, \\ 0 \le x \le 1, \quad 0 \le y \le 1 \end{aligned} \tag{4.9}$$

where

$$\alpha(n,m) = \int_0^1 \int_0^1 X(\mu_n, \xi) Y(\lambda_m, \eta) g_0 \, \mathrm{d}\eta \, \mathrm{d}\xi\,.$$

This solution is illustrated in Figure 4.1. The effect of increasing D_{yy} can be observed on the solution. As D_{yy} increases, the problem becomes more "one-dimensional" in nature owing to the strong diffusion in the y-direction. Such a configuration is known to pose a good challenge for numerical methods (see for example [41, 42, 82, 84]).

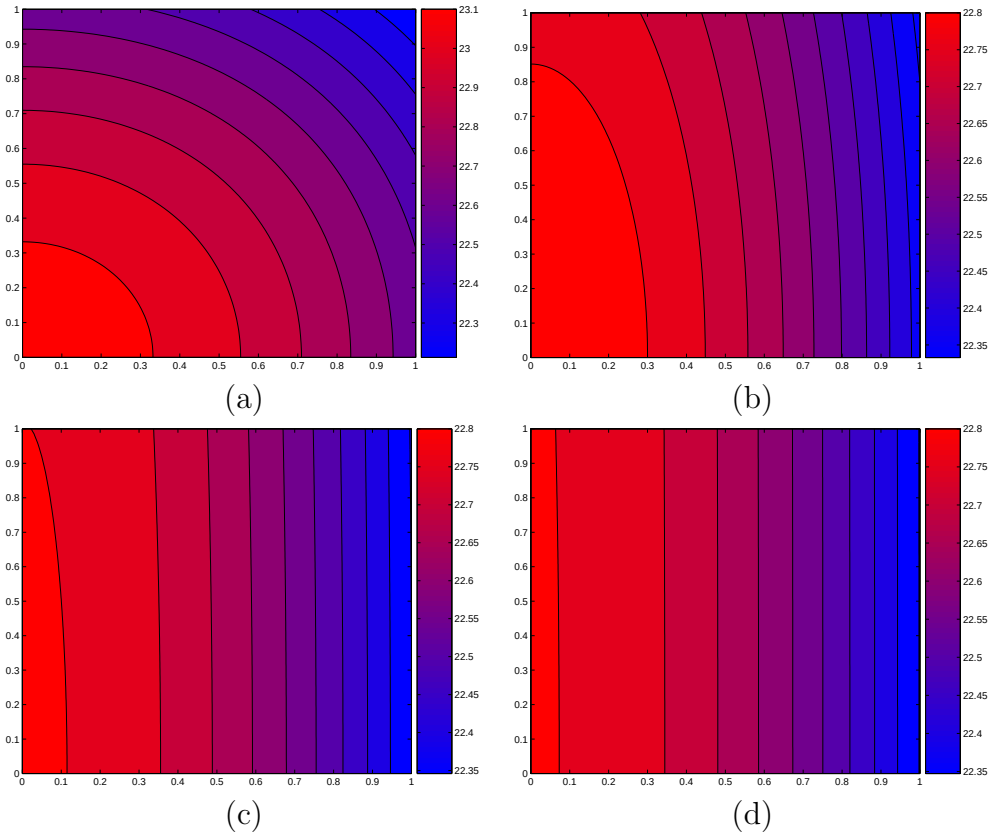


Figure 4.1: Analytic solution of (4.3) where (a) $D_{yy} = 5$; (b) $D_{yy} = 50$; (c) $D_{yy} = 500$; (d) $D_{yy} = 5000$. Colour bar shows temperature.

Table 4.2: Physical parameters for Test Problem 2.

Parameter	Description	Value
α	Thermal diffusivity	$1\ m^2 s^{-1}$
v_x	Speed	$\{1, 10, 100, 10000, 100000\}\ ms^{-1}$

4.1.2 Test Problem 2

The second test problem, taken from [64], uses the advective component $\mathbf{v} = v_x \mathbf{i}$ with $D_{xx} = D_{yy} = \alpha$ and $S = 0$. Boundaries $y = 0$ and $y = 1$ have a zero Dirichlet condition prescribed, boundary $x = 0$ has the Dirichlet condition $\varphi(x, y) = \sin(\pi y)$ prescribed and boundary $x = 1$ is insulated. This leads to the following boundary value problem:

$$\begin{aligned} \alpha\left(\frac{\partial^2\varphi}{\partial x^2} + \frac{\partial^2\varphi}{\partial y^2}\right) - v_x\frac{\partial\varphi}{\partial x} &= 0 && \text{on } (0,1)\times(0,1) \\ \varphi &= \sin(\pi y) && \text{at } x = 0 \\ \frac{\partial\varphi}{\partial x} &= 0 && \text{at } x = 1 \\ \varphi &= 0 && \text{at } y = 0 \\ \varphi &= 0 && \text{at } y = 1 \end{aligned} \tag{4.10}$$

The parameter values used for this problem are listed in Table 4.2. For some tests, the parameter v_x was varied by factors of ten to provide an increasing challenge to the numerical solution methods.

The analytic solution of (4.10) is given in [64] as:

$$\varphi(x, y) = \frac{\sin(\pi y)(r_2 e^{(r_1 x + r_2)} - r_1 e^{(r_1 + r_2 x)})}{r_2 e^{r_2} - r_1 e^{r_1}}, \quad 0 \le x \le 1, \quad 0 \le y \le 1 \tag{4.11}$$

where

$$r_1 = \frac{v_x}{2\alpha} + \sqrt{\frac{v_x^2}{4\alpha^2} + \pi^2} \tag{4.12}$$

and

$$r_2 = \frac{v_x}{2\alpha} - \sqrt{\frac{v_x^2}{4\alpha^2} + \pi^2}. \tag{4.13}$$

It should be noted however that this formulation of the solution is susceptible to numerical overflow as v_x becomes large. Therefore, in this thesis the following rearrangement of (4.11) is used, which does not suffer from this problem:

$$\varphi(x,y) = \frac{\sin(\pi y)(r_2 e^{r_2 - r_1(1-x)} - r_1 e^{r_2 x})}{(r_2 e^{r_2 - r_1} - r_1)}, \quad 0 \le x \le 1, \quad 0 \le y \le 1 \tag{4.14}$$

This solution is illustrated in Figure 4.2 for $v_x = 1$, $v_x = 10$ and $v_x = 100$. As v_x increases, the advective process begins to dominate the diffusive process, resulting in the solution being "stretched" in the x direction. Again, this is known to pose a good challenge for numerical methods [61].

4.1.3 Test Problem 3

The third test problem uses nonlinear diffusivity $D_{xx} = D_{yy} = \varphi^{1.3}$, along with $\mathbf{v} = \mathbf{0}$ in (4.1). The source term is constructed by substituting the following imposed solution, based on an example in [49], into (4.1)

$$\varphi(x,y) = 10xy(1-x)(1-y)e^{-(x^2+y^2)}, \quad 0 \le x \le 1, \quad 0 \le y \le 1\,, \tag{4.15}$$

with the Dirichlet condition $\varphi = 0$ prescribed on all boundaries. This solution is shown in Figure 4.3. It shows the hot interior of the solution, with the maximum temperature at around $(0.4, 0.4)$, cooling further away from this point and zero on the boundaries.

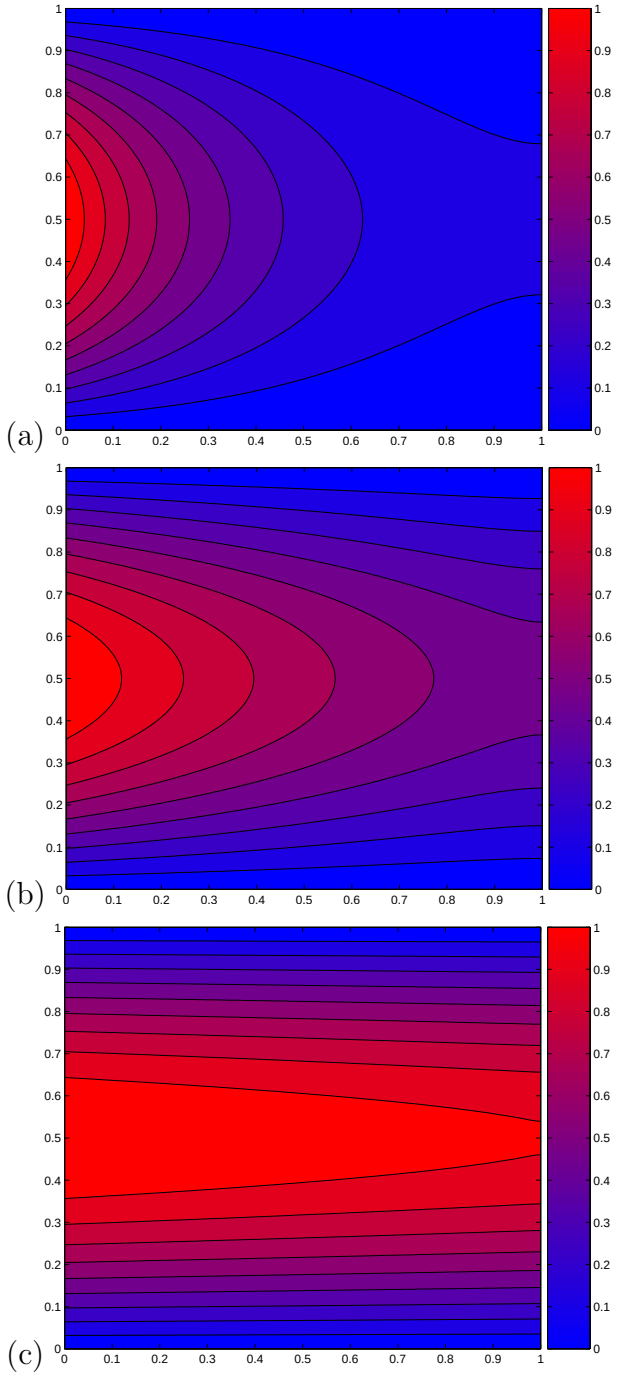


Figure 4.2: Analytic solution of (4.10) where (a) $v_x = 1$; (b) $v_x = 10$; (c) $v_x = 100$. Colour bar shows temperature.

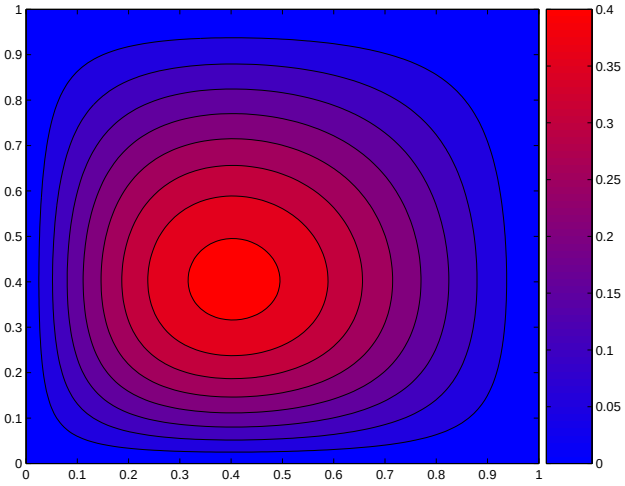


Figure 4.3: Analytic solution (4.15). Colour bar shows temperature.

Table 4.3: Properties of meshes illustrated in Figure 4.2.

Mesh	Nodes	Elements	Edge lengths		
			Minimum	Maximum	Average
(a)	13	16	0.320	0.500	0.410
(b)	47	72	0.138	0.231	0.183
(c)	144	246	0.067	0.134	0.099
(d)	450	822	0.036	0.079	0.054

4.2 Meshes

The unstructured, triangular meshes used for these test problems were generated using the mesh generation software *Gmsh* [31]. Four examples of the meshes used are shown in Figure 4.2. In each case the mesh was constructed on a unit square with a uniform edge length requested throughout the mesh. However, since these are unstructured meshes, there was some variation in edge lengths present in the final meshes. Table 4.3 lists the relevant properties of these meshes for comparison. It highlights the variation in edge lengths produced in the unstructured mesh generation process, with the maximum and minimum edge lengths in these meshes differing by up to a factor of two.

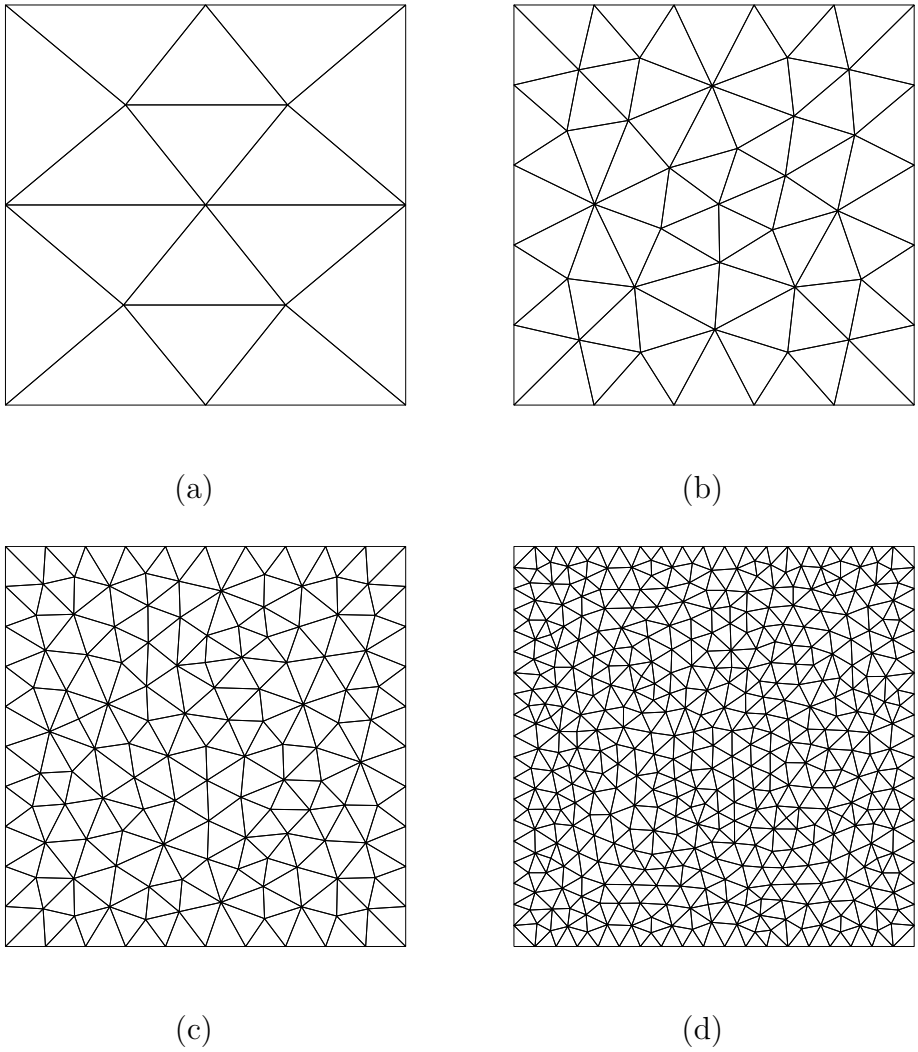


Figure 4.4: Unstructured meshes with average edge length (a) 0.410; (b) 0.183; (c) 0.099 (d) 0.054.

4.3 Results concerning accuracy

In this section the accuracy of the new CV-RBF method is compared to that of the CV-FE method. Results from solving the three test problems of Section 4.1 are shown, and the results compared to the analytic solutions, both graphically and by analysing computed solution errors.

4.3.1 Results for a single mesh

For the results in this section, the unstructured mesh comprising 144 nodes and 246 elements, illustrated in Figure 4.2(c) was used to solve each of the three test problems. In computing the solution errors, the following formula was used

$$\begin{aligned} \text{error} &= \frac{\|\boldsymbol{\varphi}^{(e)} - \boldsymbol{\varphi}^{(a)}\|_2}{\|\boldsymbol{\varphi}^{(e)}\|_2} \\ &= \frac{\sqrt{\sum_{i=1}^{N}(\varphi_i^{(e)} - \varphi_i^{(a)})^2}}{\sqrt{\sum_{i=1}^{N}(\varphi_i^{(e)})^2}} \end{aligned} \tag{4.16}$$

where superscript (e) symbolises the (exact) analytic solution and superscript (a) the (approximate) numerical solution.

For the CV-RBF method, the RBF parameters used are listed in Table 4.4. The reason for selecting the multiquadric as the basis function has been discussed in Section 2.3.3. Appropriate values for parameters c^2 and n were obtained through preliminary numerical investigation. Later, in Section 4.4.3, the effect of varying these parameters will be investigated.

Table 4.4: RBF parameters for solving Test Problems 1, 2 and 3.

Parameter	Description	Value
ϕ	Radial basis function	Multiquadric
c^2	Multiquadric parameter	3.0
n	Number of nodes per interpolation	25

Table 4.5: Errors using CV-FE and CV-RBF for Test Problem 1.

	$D_{yy} = 5$	$D_{yy} = 50$	$D_{yy} = 500$	$D_{yy} = 5000$
CV-FE	3.53E-05	1.71E-05	1.02E-04	8.35E-04
CV-RBF	9.33E-08	3.55E-08	6.86E-08	1.02E-06

Test Problem 1

The results of solving Test Problem 1 for increasing values of parameter D_{yy} are given in Table 4.5. The accuracy of the numerical solution can be seen to decrease as the parameter D_{yy} is increased. This is consistent with the notion that increasing this parameter makes the problem more difficult to solve numerically.

The table illustrates the improvement in accuracy offered by the CV-RBF method over CV-FE. For each value of D_{yy}, the new method offers at least two orders of magnitude improvement on the solution error, as measured by (4.16).

Figures 4.5 to 4.8 show the temperature contours for both CV-FE and CV-RBF as compared to the analytic solution. For smaller values of D_{yy} (Figures 4.5 and 4.6), both approximate solutions are essentially indistinguishable from the analytic solution, despite the CV-FE solution being several orders of magnitude less accurate than CV-RBF. For the larger values of D_{yy} however, the CV-FE solution starts to visibly depart from the analytic solution.

With $D_{yy} = 500$ (refer to Figure 4.7), the most notable departure from correctness in the CV-FE solution is in the left-most contour, where the approximate solution fails to capture the correct physical behaviour. In ad-

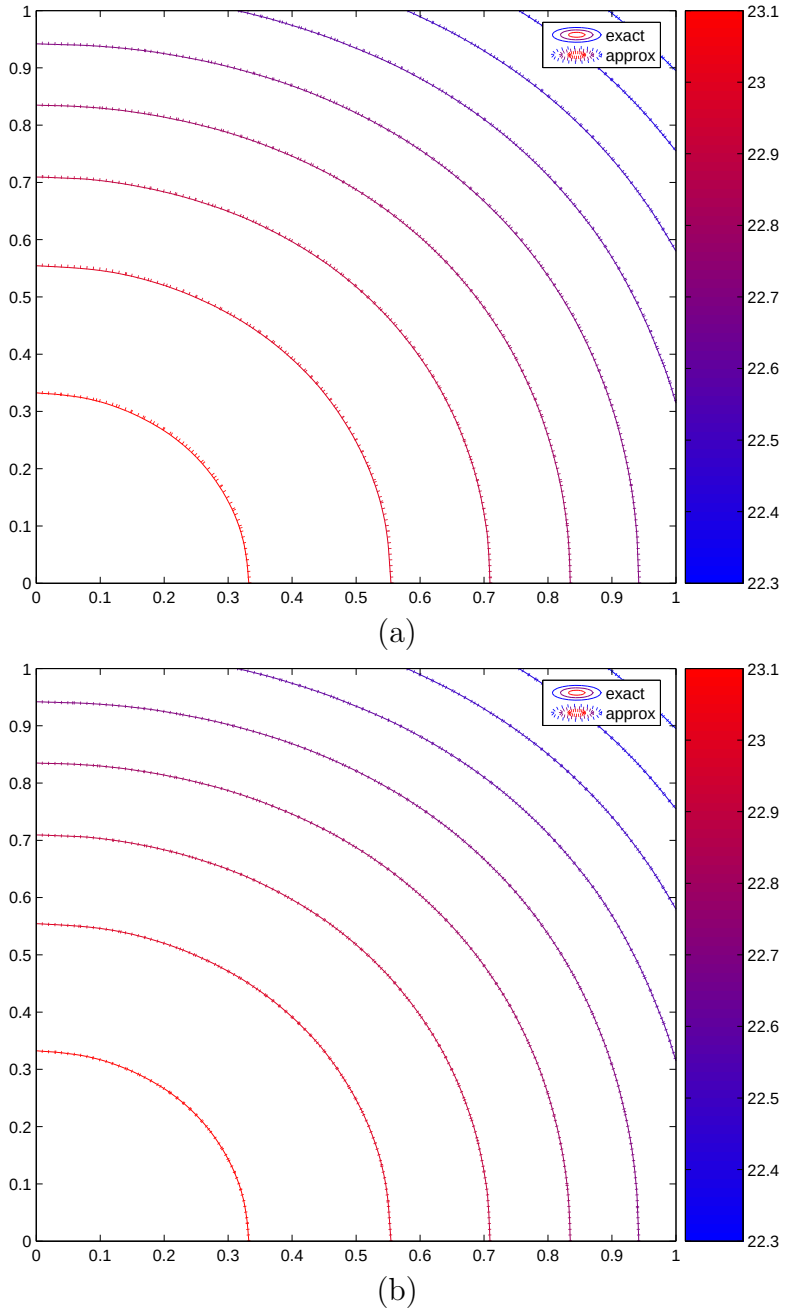
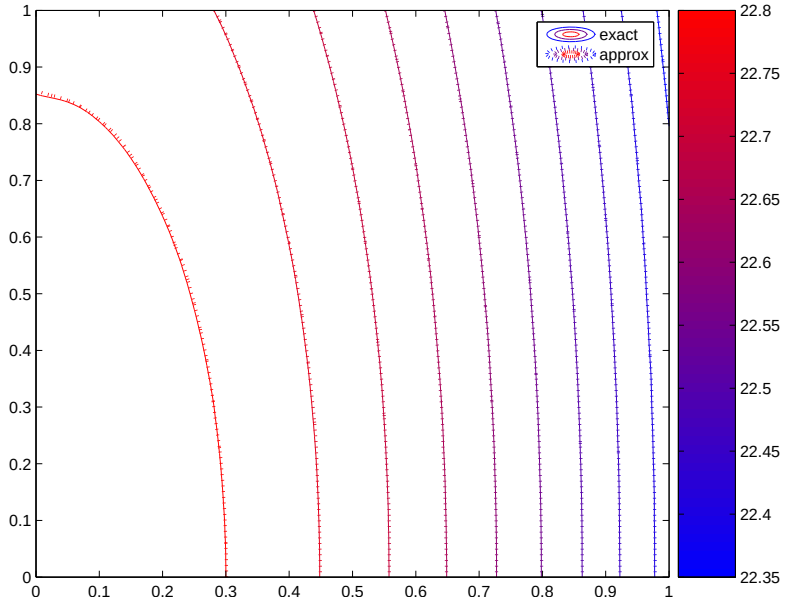
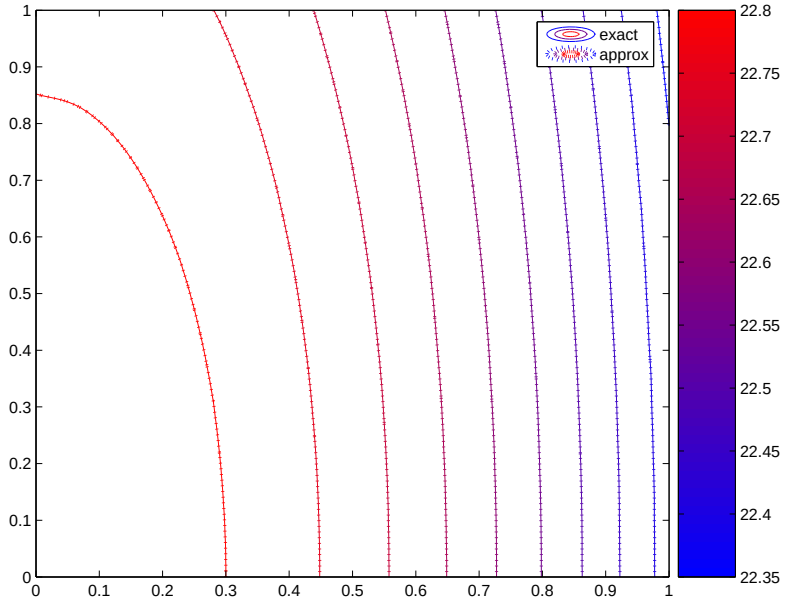


Figure 4.5: Temperature contours for Test Problem 1 with $D_{yy} = 5$ using (a) CV-FE; (b) CV-RBF. Colour bar shows temperature.



(a)



(b)

Figure 4.6: Temperature contours for Test Problem 1 with $D_{yy} = 50$ using (a) CV-FE; (b) CV-RBF. Colour bar shows temperature.

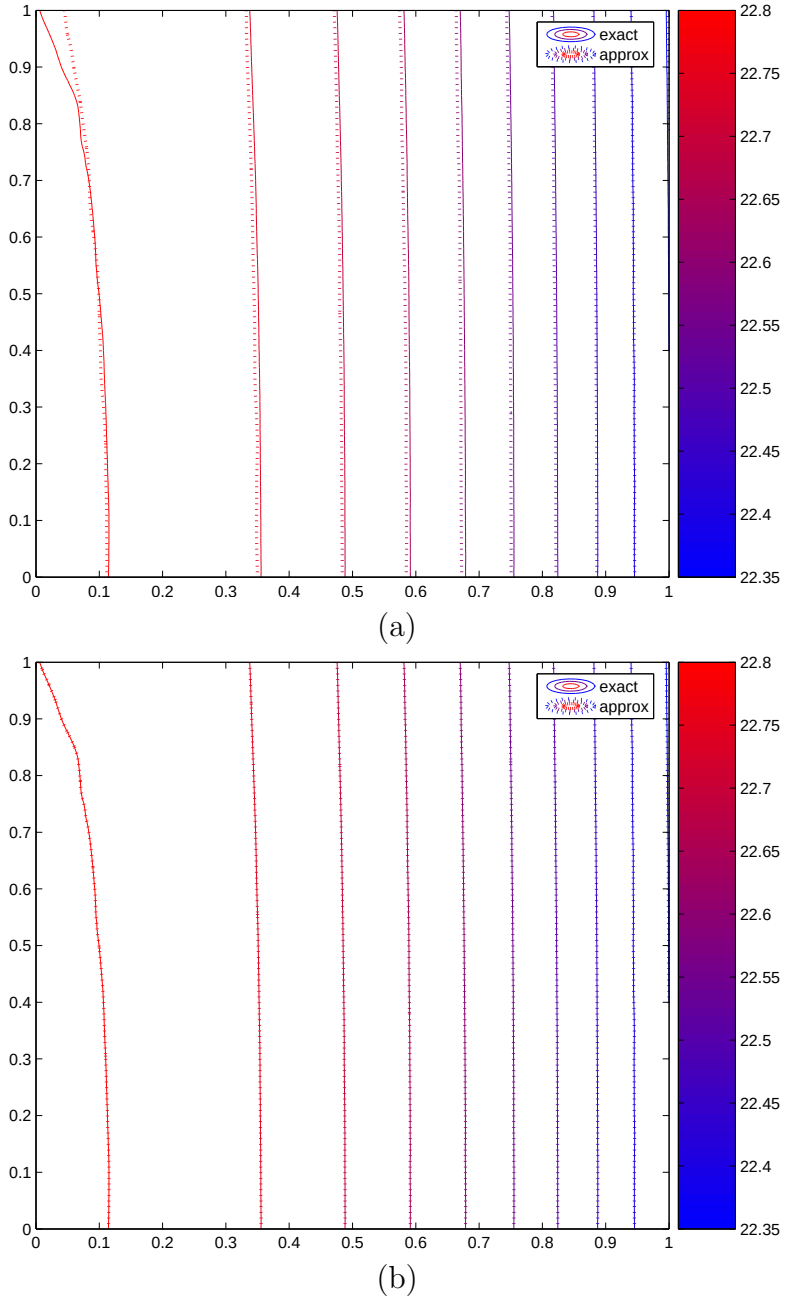
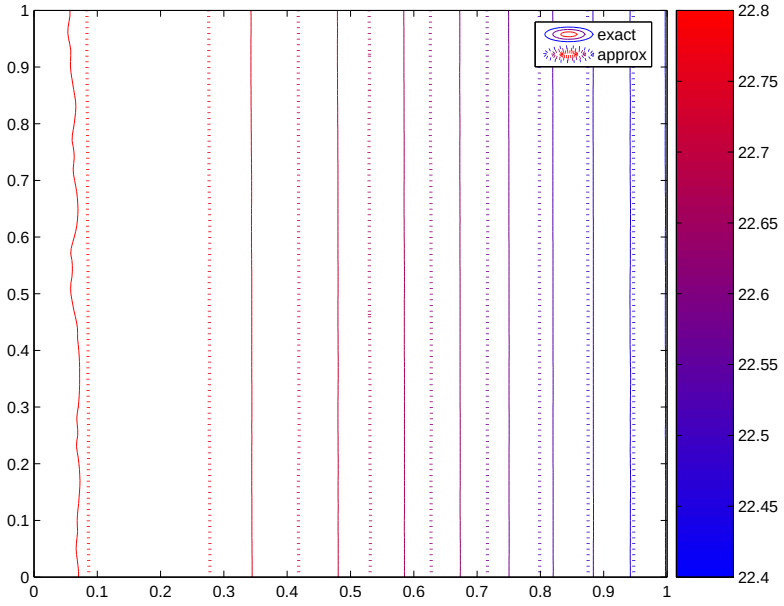
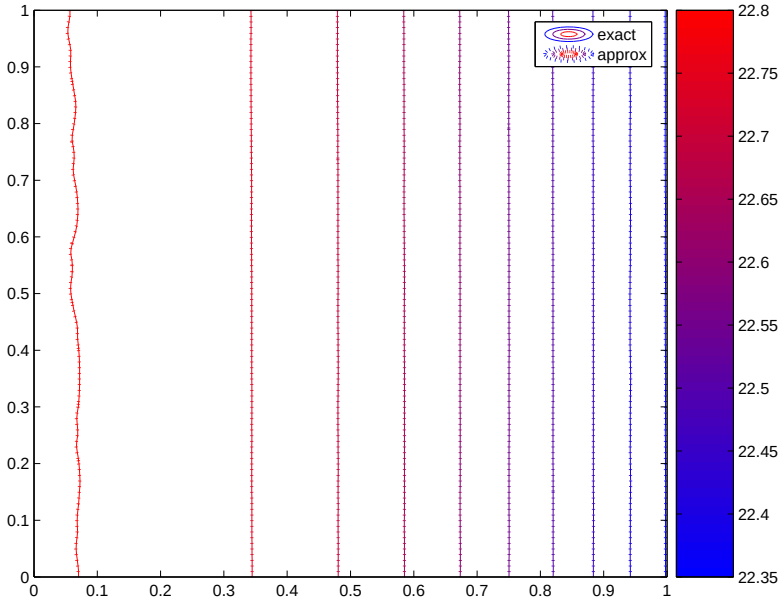


Figure 4.7: Temperature contours for Test Problem 1 with $D_{yy} = 500$ using (a) CV-FE; (b) CV-RBF. Colour bar shows temperature.



(a)



(b)

Figure 4.8: Temperature contours for Test Problem 1 with $D_{yy} = 5000$ using (a) CV-FE; (b) CV-RBF. Colour bar shows temperature.

Table 4.6: Errors using CV-FE and CV-RBF for Test Problem 2.

	$v_x = 1$	$v_x = 10$	$v_x = 100$
CV-FE	9.93E-04	3.98E-03	3.90E-03
CV-RBF	3.40E-05	2.13E-05	3.07E-05

dition, all other contours are slightly shifted out of their correct position observed in the analytic solution. The CV-RBF solution exhibits none of these discrepancies.

In Figure 4.8 with $D_{yy} = 5000$, the CV-FE solution now has all of its contours visibly unaligned with respect to the analytic solution. For this problem, the diffusion in the y direction is 1000 times stronger than in the x. The much subtler diffusion in x is not being adequately captured by the CV-FE method, and as a result the temperature variation in the x direction shown by the numerical solution is not correct. The CV-RBF solution however is able to handle this problem, with the numerical solution still visibly indistinguishable from the analytic solution.

It should be noted that the waviness present in the leftmost contour of the analytic solution, and of the CV-RBF solution, is an artefact of the visualisation process. It is a result of the transformation made by the visualisation software, when plotting contours, from a triangular mesh to a rectangular grid.

Test Problem 2

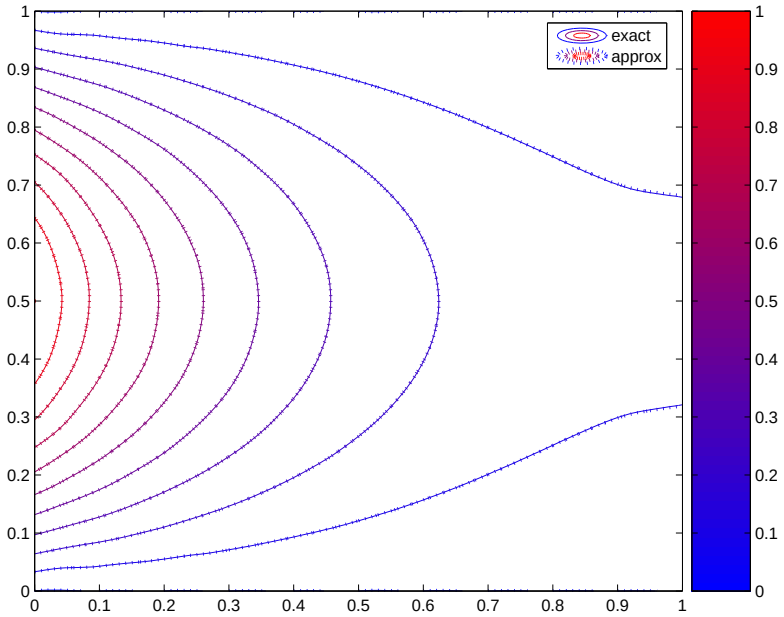
Table 4.6 shows the results of solving Test Problem 2 for the first three values of v_x given in Table 4.2, using both CV-FE and CV-RBF. It is again evident that CV-RBF offers improvement over CV-FE, although the size of the improvement is less than that observed for Test Problem 1—up to two orders of magnitude for $v_x = 10$ and $v_x = 100$, and between one and two for $v_x = 1$.

Raw accuracy is not always the only measurement that is relevant when comparing solution strategies however. While it is certainly true that the CV-RBF method has offered improved accuracy, more can be said of this solution the approximate temperature contours are compared with those of the analytic solution.

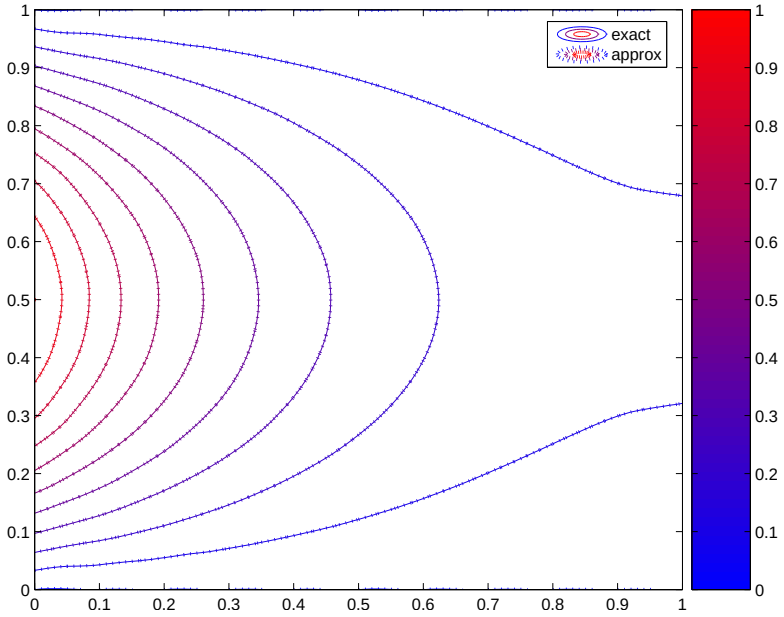
Figures 4.9 to 4.11 depict the approximate temperature contours, along with those of the analytic solution for comparison. In Figure 4.9 ($v_x = 1$), both the CV-FE and CV-RBF solutions are visibly indistinguishable from the analytic solution. As v_x increases, the solution profile changes to reflect the increased emphasis on the advective component, and as a consequence the CV-FE solution starts to noticeably depart from the analytic solution. In Figure 4.10, the (left to right) flow in the interior (around $y = 0.5$) is slightly under-predicted by the CV-FE computed solution. The RBF solution on the other hand correctly predicts the flow.

When v_x is further increased to 100 in Figure 4.11, the CV-RBF solution is again visibly indistinguishable from the analytic solution. The CV-FE solution now exhibits a different problem however. Rather than under-predicting the flow, the CV-FE temperature contours appear to oscillate around the exact contours.

Figure 4.12 is a closer view of the region $0 \leq x \leq 0.7$ and $0.3 \leq y \leq 0.7$ in Figure 4.11. The oscillatory nature of the CV-FE solution is now much clearer: the dotted contours of the numerical solution fluctuate around the solid, exact temperature contours. This type of behaviour is typical for CV-FE solutions of advection-diffusion problems of this kind [61]. Associated with this phenomenon is a dimensionless quantity known as the cell *Peclet number*, which measures how dominant the advective process is over the diffusive process. For this problem, the cell Peclet number is given by

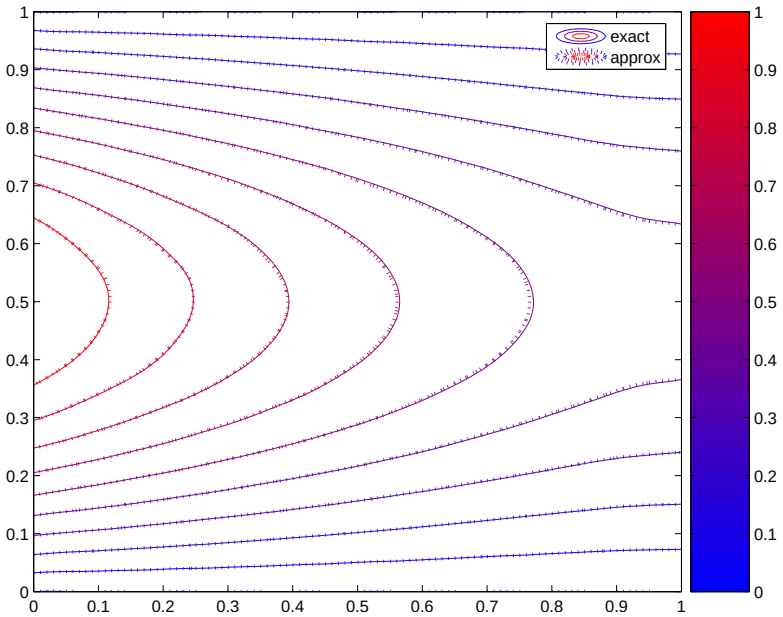


(a)

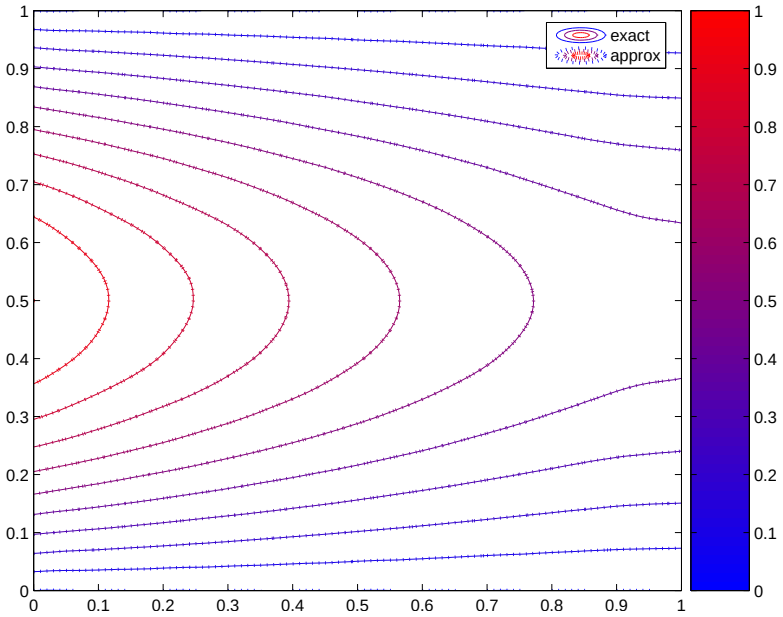


(b)

Figure 4.9: Temperature contours for Test Problem 2 with $v_x = 1$ using (a) CV-FE; (b) CV-RBF. Colour bar shows temperature.

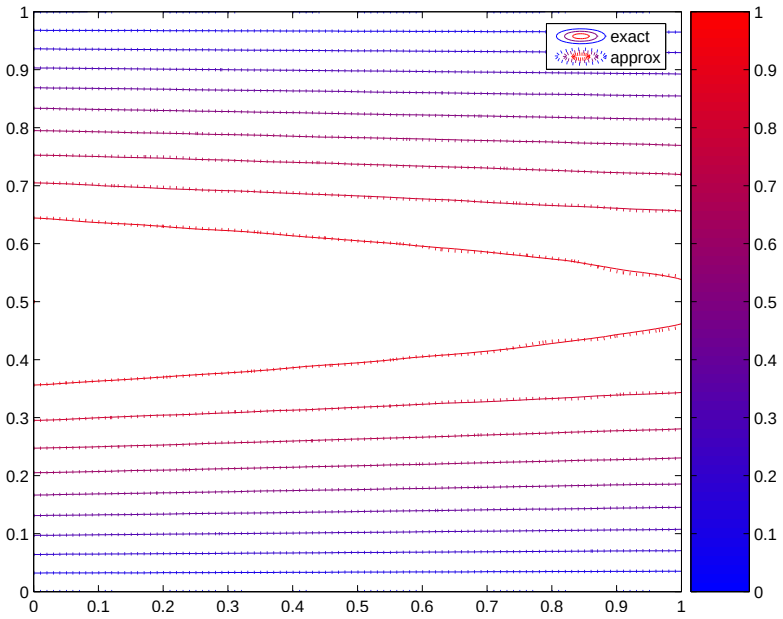


(a)

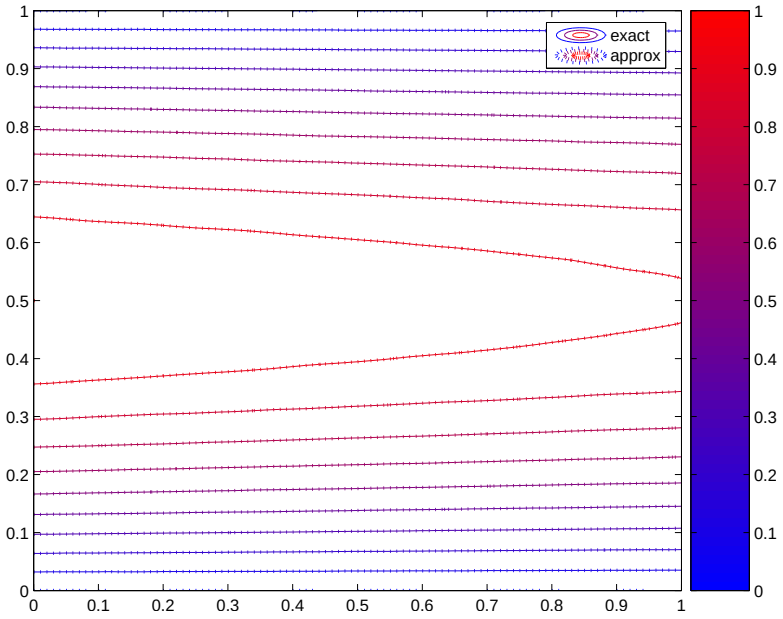


(b)

Figure 4.10: Temperature contours for Test Problem 2 with $v_x = 10$ using (a) CV-FE; (b) CV-RBF. Colour bar shows temperature.

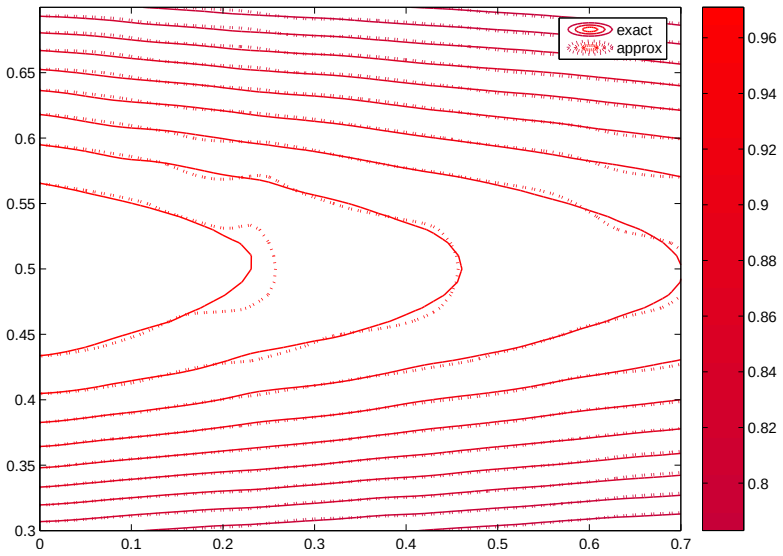


(a)

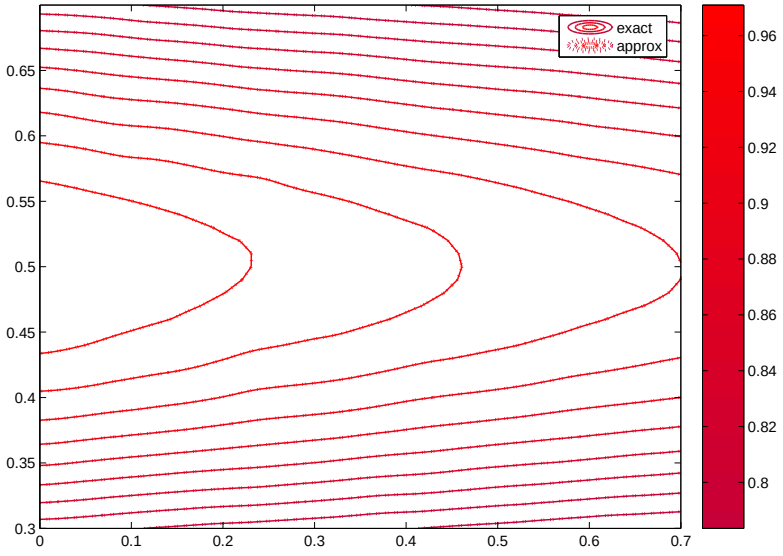


(b)

Figure 4.11: Temperature contours for Test Problem 2 with $v_x = 100$ using (a) CV-FE; (b) CV-RBF. Colour bar shows temperature.



(a)



(b)

Figure 4.12: Zoomed temperature contours for Test Problem 2 with $v_x = 100$ using (a) CV-FE; (b) CV-RBF. Colour bar shows temperature.

$$Pe = \frac{hv_x}{D_{xx}} \tag{4.17}$$

where h is the average element edge length. Oscillatory behaviour is associated with cell Peclet numbers greater than two [61]. For this problem, with $D_{xx} = 1$ and the value of h given by 0.099 (see Table 4.3), oscillations may be observed when $v_x > 20.2$. This is consistent with the results observed in Figure 4.12(a) with $v_x = 100$.

For this problem, it is worthwhile examining the effect of further increasing the parameter v_x, according to Table 4.2. Figures 4.13 and 4.14 show the temperature contours corresponding to $v_x = 10000$ and $v_x = 100000$ (again for $0 \leq x \leq 0.7$ and $0.3 \leq y \leq 0.7$) respectively. The oscillations in the CV-FE solution are now greatly exaggerated, with associated cell Peclet numbers of $\frac{0.099 \times 10000}{1} = 990$ and $\frac{0.099 \times 100000}{1} = 9900$ respectively.

When using CV-FE, there are several techniques that can be applied to eliminate this nonphysical solution behaviour. The first option is to employ a finer mesh, thereby reducing the value of h in (4.17). This can be quite an expensive option however, particularly if the advective-diffusive ratio is high, as this will require a very fine mesh to ensure that the cell Peclet number is sufficiently small.

Alternatively, one can employ the technique of *upwinding* [61]. In this approach, the usual shape function interpolation technique is dispensed with for the advective flux calculations, and instead the *upwinded* value of φ is used. That is, the value of φ is taken at the element vertex furthest in the direction opposite to the flow, so that the information from this node is "carried downwind".

Figure 4.15 shows the numerical solution for $v_x = 100$ when using CV-FE with upwinding. The full effect of the advective component is no longer

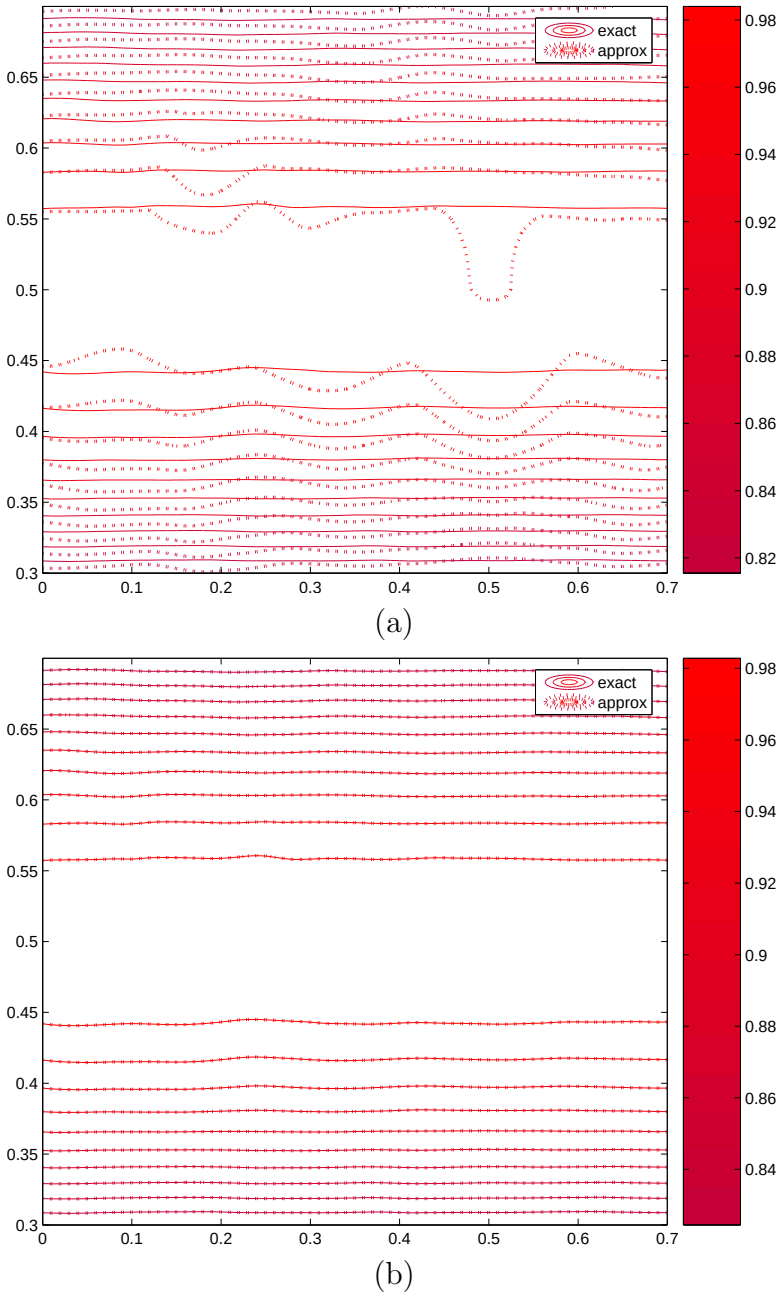
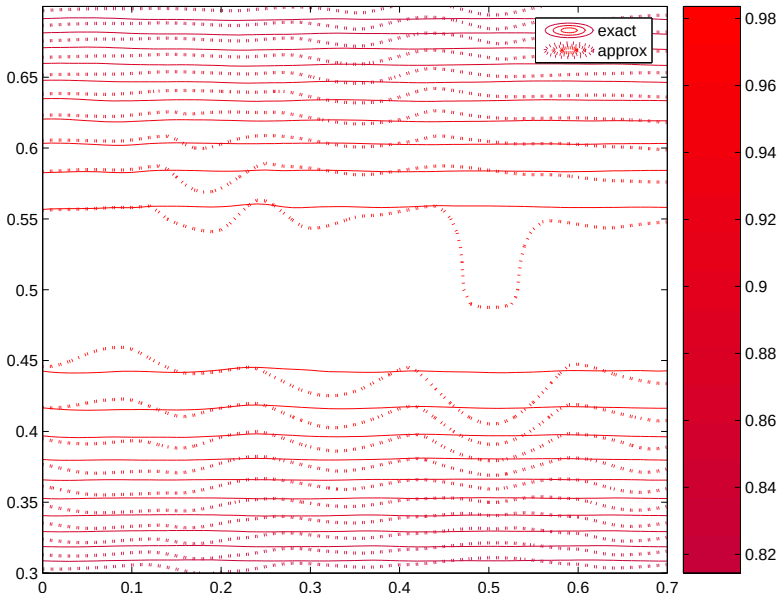
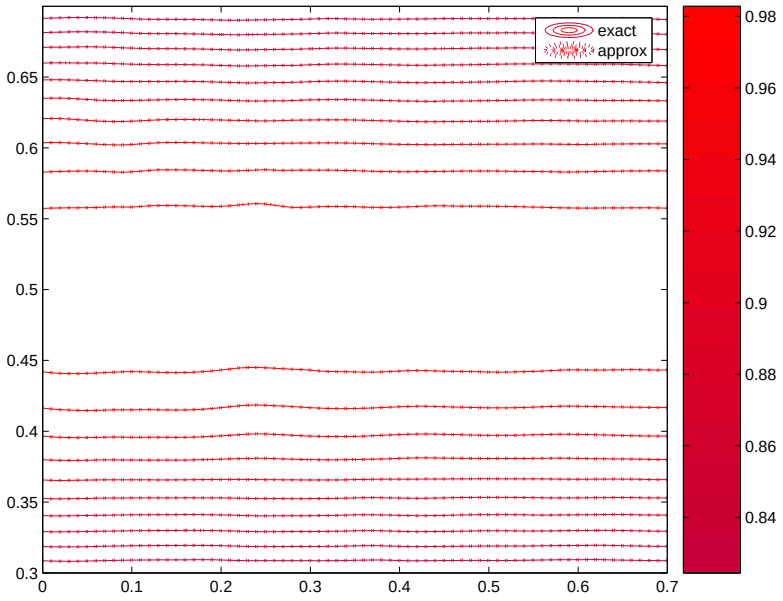


Figure 4.13: Zoomed temperature contours for Test Problem 2 with $v_x = 10000$ using (a) CV-FE; (b) CV-RBF. Colour bar shows temperature.



(a)



(b)

Figure 4.14: Zoomed temperature contours for Test Problem 2 with $v_x = 100000$ using (a) CV-FE; (b) CV-RBF. Colour bar shows temperature.

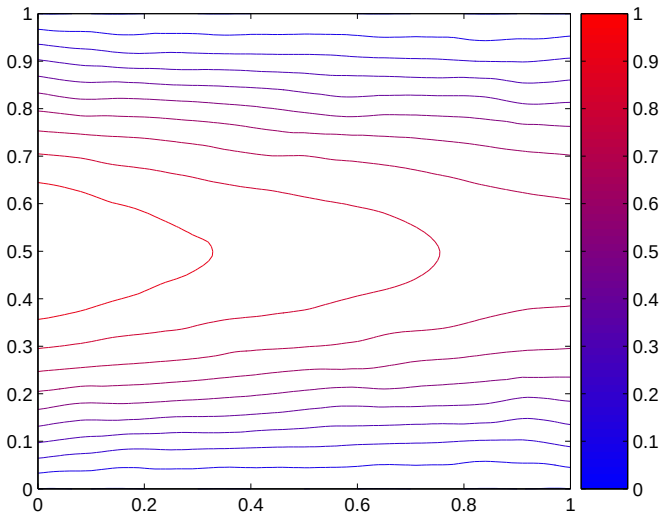


Figure 4.15: Temperature contours for Test Problem 2 with $v_x = 100$ using upwinding. Colour bar shows temperature.

exhibited in the solution. In appearance, the solution lies somewhere between Figures 4.10 and 4.11, showing that the advective-diffusive ratio is not being correctly modelled in this solution. More sophisticated techniques such as *flux-limiting* can be employed to combat this so-called "artificial diffusion" introduced by upwinding [83]. Here though, note that in none of the CV-RBF figures is any oscillatory behaviour visible in the numerical solution. So employing an RBF-based method instead of shape functions can be a viable alternative to mesh refinement or flux limiting.

Test Problem 3

The temperature contours for Test Problem 3 are shown in Figure 4.16. The errors associated with these solutions are: 2.36E-02 for CV-FE and 4.81E-06 for CV-RBF, as shown in Table 4.7. Once again the CV-RBF method offers much improved accuracy over CV-FE. This is also evident in Figure 4.16

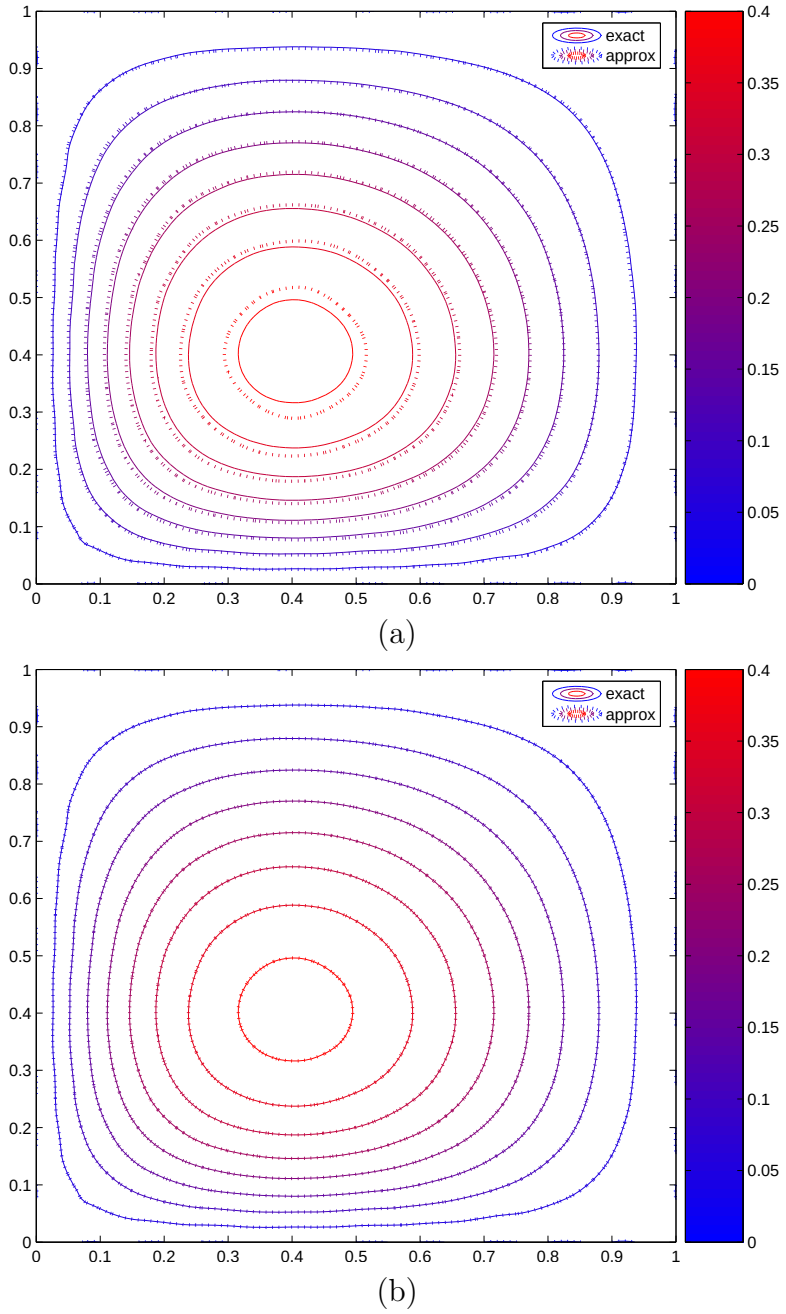


Figure 4.16: Temperature contours for Test Problem 3 using (a) CV-FE; (b) CV-RBF. Colour bar shows temperature.

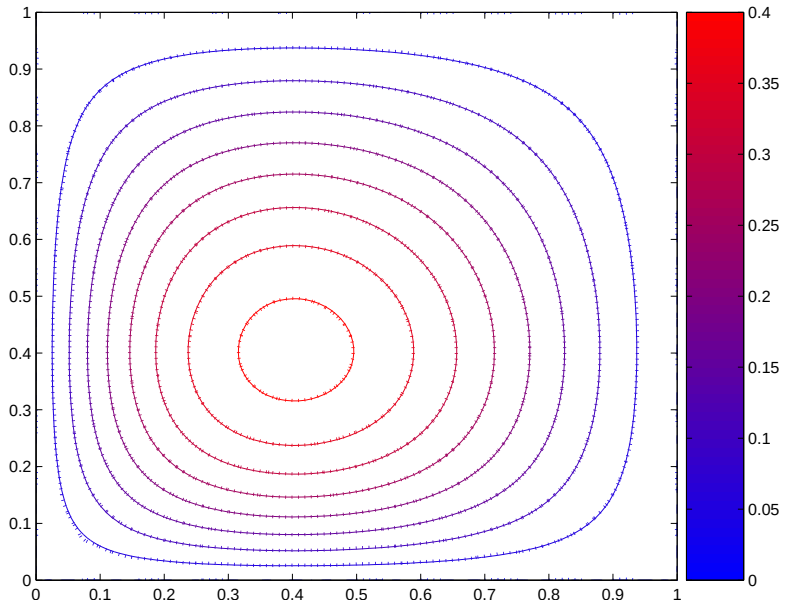


Figure 4.17: Temperature contours for Test Problem 3 using CV-FE on a fine mesh. Colour bar shows temperature.

Table 4.7: Errors using CV-FE and CV-RBF for Test Problem 3.

	Test Problem 3
CV-FE	2.36E-02
CV-RBF	4.81E-06

where the CV-FE solution fails to capture the solution peak near $(x, y) = (0.4, 0.4)$, while the CV-RBF solution is once again indistinguishable from the analytic solution.

It should be noted that when using CV-FE it is certainly possible to achieve the same level of accuracy as that offered by CV-RBF, but it requires a much finer mesh. For example, the solution in Figure 4.17 was obtained using CV-FE and is virtually indistinguishable from the analytic solution. However the mesh that was used to solve the problem consisted of 5996 nodes and 11694 elements, and as a result the method required more than ten times the execution time of the CV-RBF on mesh (c) in Table 4.3.

In three dimensions (see Chapter 6), where efficiency concerns are greater than in two, much more will be made of these comparisons between the resources required for CV-FE and CV-RBF to generate solutions of comparable accuracy. It is the increased accuracy offered by the CV-RBF discretisation that allows the CV-RBF method to generate highly accurate solutions on a much coarser mesh than is possible with CV-FE.

Shape functions with Gaussian quadrature

In Section 2.4.2, the CV-FE and CV-RBF methods were introduced, and the justification for their respective methods of integration and interpolation given. In particular, the CV-FE method employs midpoint rule integration, rather than the more expensive Gaussian quadrature used by CV-RBF. In this section, results are presented that show how this choice does not harm the accuracy of the CV-FE method, and so is justified on the basis of efficiency.

Recall from Section 2.3.4 that the method of shape functions can be considered a special case of the RBF method. In two dimensions, with triangular elements, each shape function interpolation involves the three nodes that make up the element's vertices. If an RBF interpolation is constructed using those same three nodes, the method of shape functions is recovered. Thus, it is possible to test the effectiveness of a shape function scheme utilising Gaussian quadrature by simply using CV-RBF with $n = 3$ nodes per interpolation. The results from applying this method to Test Problems 1, 2 and 3 are shown in Tables 4.8, 4.9 and 4.10 respectively. The results given for CV-FE in the

Table 4.8: Errors using CV-FE and CV-RBF (n=3) for Test Problem 1.

	$D_{yy} = 5$	$D_{yy} = 50$	$D_{yy} = 500$	$D_{yy} = 5000$
CV-FE	3.53E-05	1.71E-05	1.02E-04	8.35E-04
CV-RBF($n = 3$)	3.30E-05	3.12E-05	1.36E-04	3.38E-03

Table 4.9: Errors using CV-FE and CV-RBF (n=3) for Test Problem 2.

	$v_x = 1$	$v_x = 10$	$v_x = 100$
CV-FE	9.93E-04	3.98E-03	3.90E-03
CV-RBF(n=3)	1.68E-03	3.26E-03	2.51E-03

Table 4.10: Errors using CV-FE and CV-RBF (n=3) for Test Problem 3.

	Test Problem 3
CV-FE	2.36E-02
CV-RBF(n=3)	1.33E-02

earlier Tables 4.5, 4.6 and 4.7 are repeated here comparison purposes.

It is evident from these results that in no case does using Gaussian quadrature with the shape function-based interpolation method offer any significant improvements in accuracy. Accordingly, the pairing of shape functions with midpoint integration in the CV-FE method is justified, as it produces equivalent levels of accuracy without the additional overhead of Gaussian quadrature. For the remainder of this section, any reference to an interpolation method with $n = 3$ nodes per element will always refer to CV-FE.

In summary, the results in this section have demonstrated that the CV-RBF method offers greater accuracy than does CV-FE for the three test problems studied. In the next section an investigation will be made into whether the CV-RBF method also has a higher order of accuracy than CV-FE over meshes of increasing refinement.

4.3.2 Results over several meshes

To this point the CV-FE and CV-RBF methods have been tested on a single example of an unstructured mesh. It is also of interest to investigate how these methods perform as the mesh is refined—that is, as the number of nodes is increased, and the average control volume face length is decreased. To do

this, the error in the solution when solved over meshes of various refinements is measured. A plot of this error against the average control volume face length shows how the error decreases with mesh refinement.

Since unstructured meshes are being used, some degree of variation will be present in the results, owing to the variation in the element edge lengths throughout the meshes. Nevertheless, if this investigation is to be meaningful, the overall trend should be that as the mesh is refined, the error of the method decreases. Furthermore, it is important to test whether the observed order of the method is increased as the number of nodes used per RBF interpolation is increased.

The following error measurement is used for all of the plots in this section:

$$\begin{aligned} \text{error} &= \|\boldsymbol{\varphi}^{(e)} - \boldsymbol{\varphi}^{(a)}\|_\infty \\ &= \max|\varphi_i^{(e)} - \varphi_i^{(a)}|, \quad i = 1 \ldots N \end{aligned} \tag{4.18}$$

where superscript (e) symbolises the (exact) analytic solution, superscript (a) the (approximate) numerical solution, and N is the number of nodes in the mesh.

Figures 4.18 and 4.19 show the error plots from solving Test Problems 1, 2 and 3, using CV-FE ($n = 3$), and CV-RBF with $n = 6$, $n = 12$ and $n = 25$ nodes per interpolation. The order, p, of the method is estimated by fitting the curve

$$\text{error} = \text{constant} \times h^p \tag{4.19}$$

using logarithmic regression, where h is the average face length of the control volume exhibiting the maximum error in (4.18). The value of p computed by the regression model is used as an estimate of the method's order of accuracy.

One aspect that is immediately evident from these figures is that they all

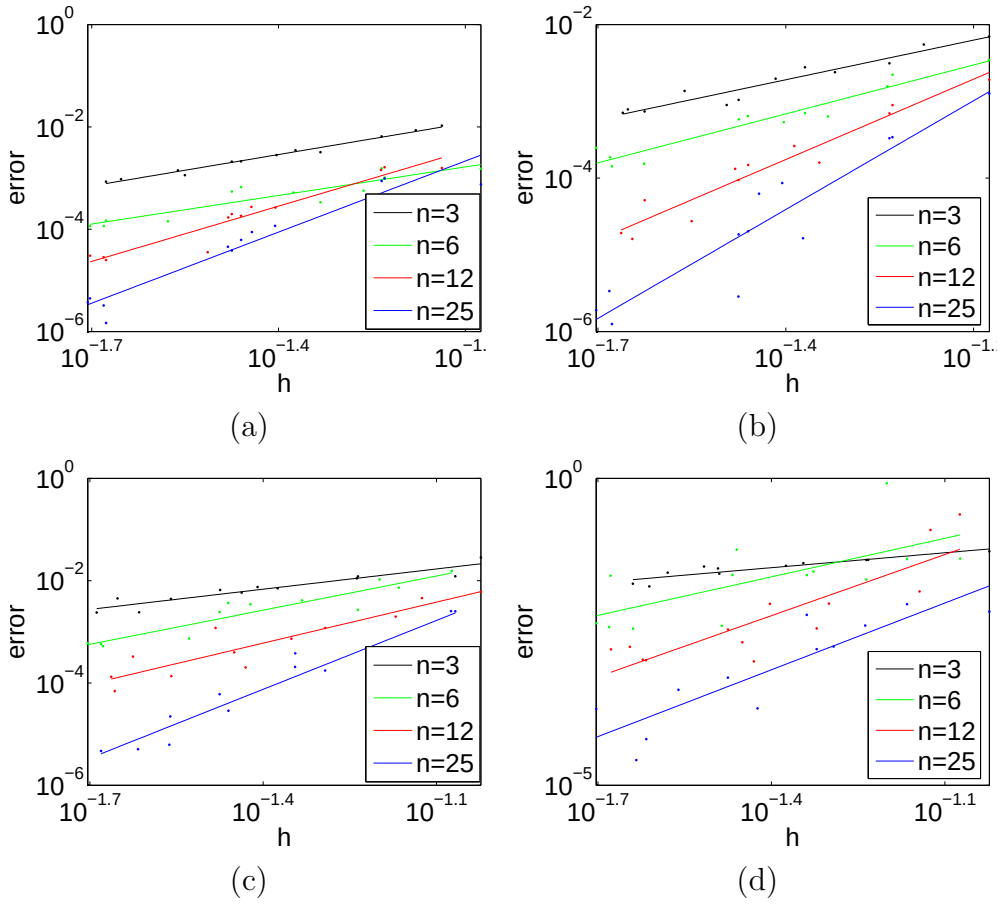


Figure 4.18: Error versus mesh refinement for Test Problem 1 with (a) $D_{yy} = 5$; (b) $D_{yy} = 50$; (c) $D_{yy} = 500$; (d) $D_{yy} = 5000$. Least squares line also shown.

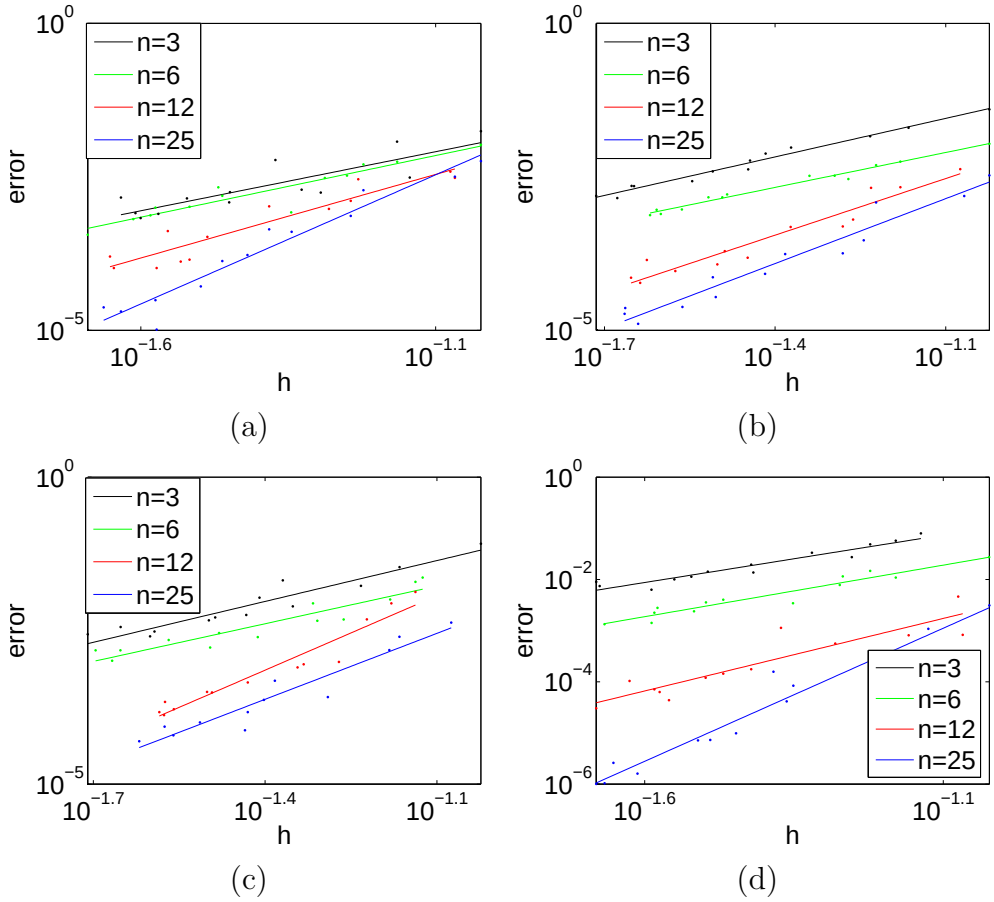


Figure 4.19: Error versus mesh refinement for (a) Test Problem 2 with $v_x = 1$; (b) Test Problem 2 with $v_x = 10$; (c) Test Problem 2 with $v_x = 100$; (d) Test Problem 3. Least squares line also shown.

Table 4.11: Estimated orders of CV-FE and CV-RBF methods for solving Test Problem 1

	n (nodes used per interpolation)			
	3	6	12	25
$D_{yy} = 5$	2.1	1.9	3.6	4.6
$D_{yy} = 50$	1.7	2.1	3.5	4.7
$D_{yy} = 500$	1.3	2.2	2.7	4.5
$D_{yy} = 5000$	0.8	2.1	3.3	3.6

Table 4.12: Estimated orders of CV-FE and CV-RBF methods for solving Test Problem 2

	n (nodes used per interpolation)			
	3	6	12	25
$v_x = 1$	1.9	2.0	2.7	4.2
$v_x = 10$	2.1	1.9	3.1	3.5
$v_x = 100$	2.2	2.0	4.0	3.6

exhibit a reduction in the error as the average control volume face length is reduced. Furthermore, the size of the error (indicated by the vertical axis) is reduced for all three test problems by using the CV-RBF method. This is in agreement with the findings of the previous section.

Tables 4.11, 4.12 and 4.13 list the estimated orders of accuracy of the methods for each test problem. They show that, by and large, the CV-FE method achieves approximately second order accuracy, with the exception of Test Problem 1 when the anisotropy ratio is large (either 100 or 1000). The CV-RBF method with $n = 6$ also appears to achieves second order accuracy, and this is true even in the two cases where the CV-FE method fails to.

Using $n = 12$ or $n = 25$ nodes per RBF interpolation results in an increase

Table 4.13: Estimated orders of CV-FE and CV-RBF methods for solving Test Problem 3

	n (nodes used per interpolation)			
	3	6	12	25
Test Problem 3	1.9	2.0	2.8	5.2

in the estimated order of the method. With $n = 12$, varying results are observed, but estimates as high as fourth order are observed in the case of Test Problem 2 (Table 4.12), and the majority of measurements are order 3 or higher. With $n = 25$ the method appears to achieve fifth order accuracy for Test Problem 3 (Table 4.13), and similar results are observed for Test Problem 1 (Table 4.11) except when $D_{yy} = 5000$.

Although there are some variations in these estimates across the test problems, perhaps owing to the use of unstructured meshes with varying edge lengths, the general trend is that using CV-RBF with $n = 12$ and particularly with $n = 25$ results in an increase in the estimated order of the method. Using CV-RBF with $n = 6$ does not necessarily give an increased order of accuracy (although it does improve the error), so larger values of n, such as $n = 25$ are recommended.

4.4 Results concerning efficiency

4.4.1 Preconditioning

Jacobian structure

Figure 4.20 shows the sparsity patterns for four different Jacobian matrices generated on the unstructured mesh illustrated in Figure 4.2(c). Each matrix corresponds to a different value of n, the number of nodes used in each element's interpolation.

From Table 4.3, the unstructured mesh depicted in Figure 4.2(c) has 144 nodes, so these are all 144×144 matrices. The impact of n on the sparsity of these matrices is evident from the figure. When using $n = 3$ (shape functions), the matrix is only 5.6% dense, with a lot of zero entries within

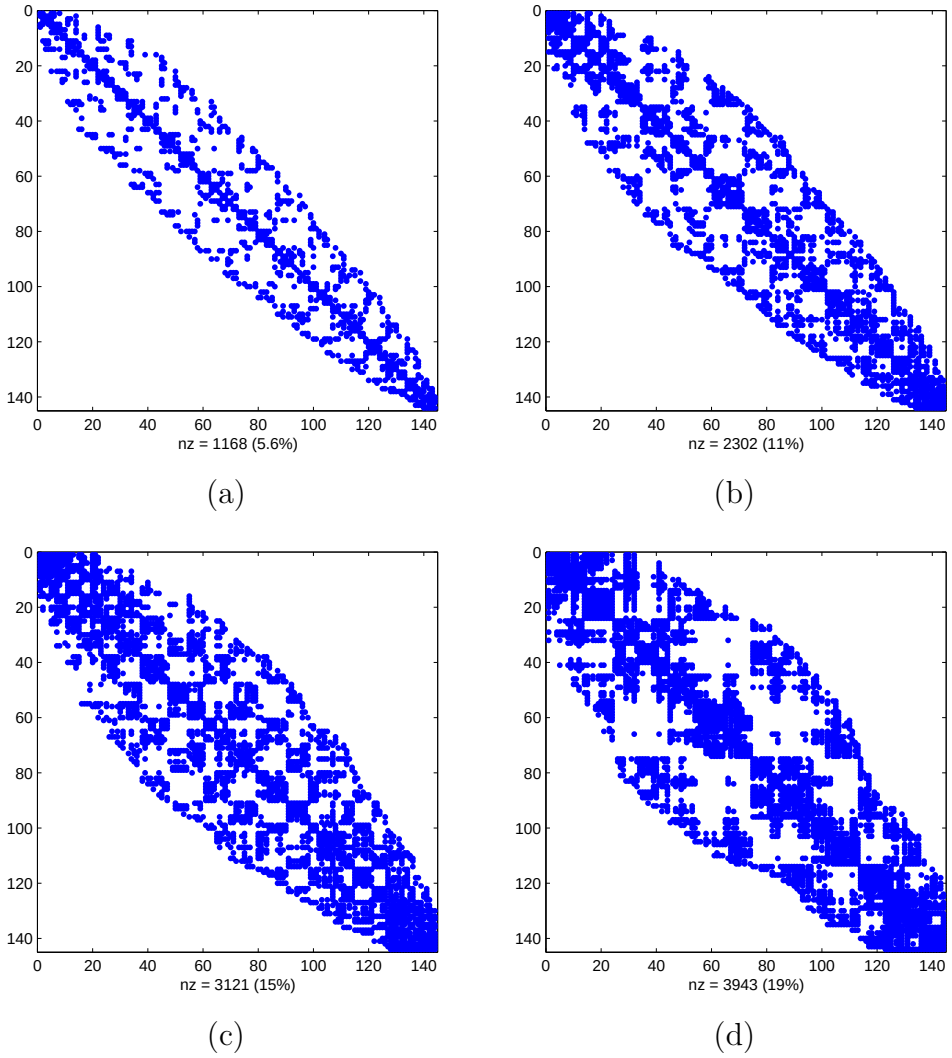
the bandwidth. Using $n = 6$ results in more of this bandwidth being filled in, along with a slight increase in bandwidth. Increasing n further to 12 and finally to 25 results in more nonzero entries being included in the matrix, along with the bandwidth widening. With $n = 25$ the Jacobian matrix is now almost 20% dense.

Jacobian spectra

In finding the solution to the nonlinear system (2.53) using a Newton-Krylov method, each step requires the solution of the linear system (2.54) involving the Jacobian matrix $\mathbf{J}$. It was discussed in Section 2.5.2 that preconditioning is crucial for this process to be efficient. In this section, the two methods of preconditioning outlined in that section are tested for their effectiveness. All tests are based on solving Test Problem 1, using the unstructured mesh with 144 nodes and 246 elements shown in Figure 4.2(c).

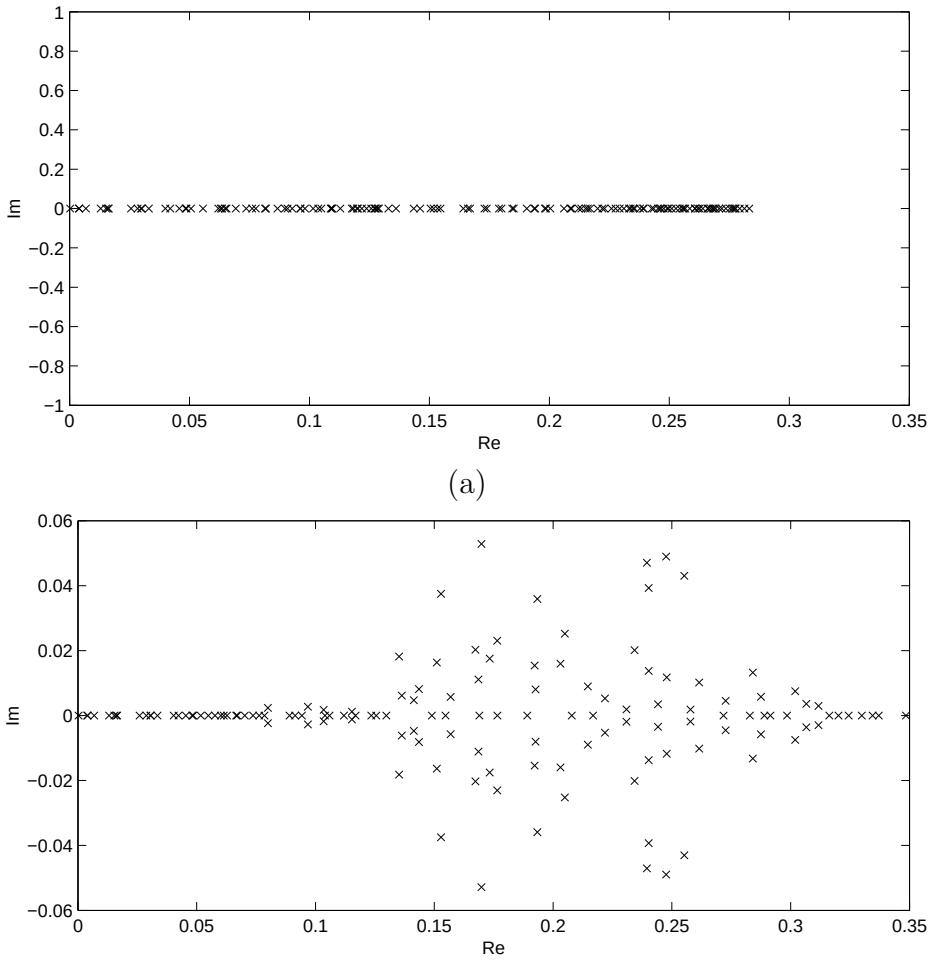
Recall from Section 2.4.3 that one way to apply preconditioning in a Jacobian-free environment is to use an approximate Jacobian matrix, drawn from a simplification of the problem. In the case of the CV-RBF Jacobian, $\mathbf{J}_{RBF}$, the Jacobian matrix from the corresponding CV-FE discretisation, $\mathbf{J}_{FE}$, is a good candidate. Figure 4.21 shows the eigenvalue spectra of the two Jacobian matrices for Test Problem 1 when $D_{yy} = 5$ (see Table 4.1) using CV-FE and CV-RBF with $n = 25$. (As this problem is linear, there is only one Newton step needed to solve it, so is is appropriate to speak of *the* Jacobian matrix.)

In both cases the eigenvalues are scattered fairly uniformly up to around 0.3. Although the CV-RBF Jacobian has many complex eigenvalues, their imaginary parts are an order of magnitude or so less than their real parts. Furthermore, the smallest eigenvalues of both matrices are real, and are of



(a) (b) (c) (d)

Figure 4.20: Jacobian matrices for mesh (c) of Figure 4.2 when using (a) $n = 3$; (b) $n = 6$; (c) $n = 12$; (d) $n = 25$. Note: `nz` is number of nonzero entries.



(a)

(b)

Figure 4.21: Eigenvalues of (a) $\mathbf{J}_{FE}$; (b) $\mathbf{J}_{RBF}$ for Test Problem 1 with $D_{yy} = 5$.

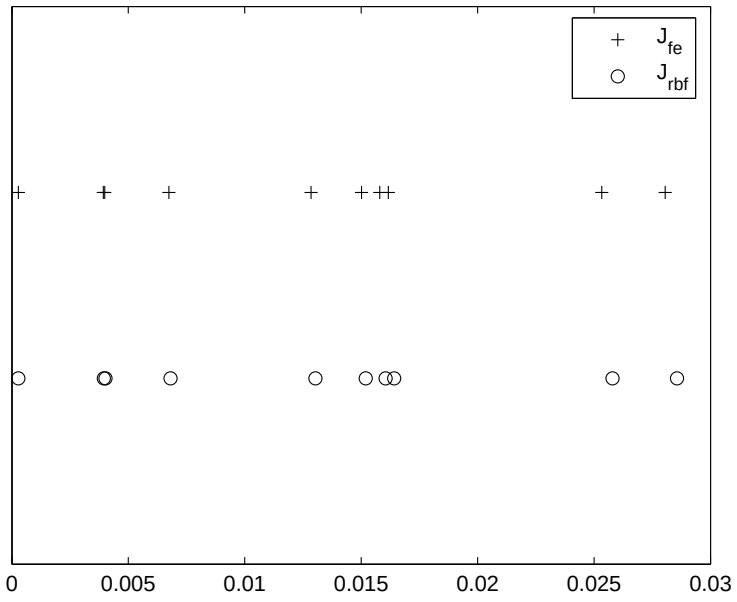


Figure 4.22: Ten smallest eigenvalues of $\mathbf{J}_{FE}$ and $\mathbf{J}_{RBF}$ for Test Problem 1 with $D_{yy} = 5$.

similar magnitudes. Figure 4.22 confirms this observation, comparing the ten smallest eigenvalues of the two Jacobian matrices. This suggests that information relating to the smallest eigenvalues could be extracted from $\mathbf{J}_{FE}$ and used successfully to precondition $\mathbf{J}_{RBF}$ as outlined in Section 2.5.2.

For some problems, deflation using the matrix $\mathbf{M}_2$ discussed in Section 2.5.2 is not sufficient to achieve satisfactory convergence using GMRES-DR. Recall that large eigenvalues in the spectrum are also troublesome. To combat this, the other preconditioner $\mathbf{M}_1$ discussed in Section 2.5.2 must also be applied. Furthermore, as outlined in that section, it must be applied before any subsequent deflation. Thus, in these cases it is necessary to examine the spectra of $\mathbf{J}\mathbf{M}_1$ rather than $\mathbf{J}$, to see if small eigenvalues are still of similar magnitudes.

Figure 4.23 illustrates the eigenvalue spectra of $\mathbf{J}_{FE}\mathbf{M_1}$ and $\mathbf{J}_{RBF}\mathbf{M_1}$. It is apparent that the norm-minimising preconditioner $\mathbf{M}_1$ has been effective at shifting many eigenvalues towards one, with the side effect that most are now complex. Several small, real eigenvalues remain however.

Figure 4.24 compares the remaining eigenvalues less than 0.5 of the two preconditioned Jacobian matrices. It shows that these eigenvalues are very similar in magnitude, which leads to the conclusion that a deflation preconditioner built using information from $\mathbf{J}_{FE}\mathbf{M}_1$ could be effective on the matrix $\mathbf{J}_{RBF}\mathbf{M}_1$.

Figure 4.25 shows the superimposed eigenvalue spectra of $\mathbf{J}_{RBF}\mathbf{M}_1$ and $\mathbf{J}_{RBF}\mathbf{M}_1\mathbf{M}_2$. Here $\mathbf{M}_2$ is constructed using the eigenvectors of $\mathbf{J}_{FE}\mathbf{M}_1$ corresponding to the smallest five eigenvalues. The figure shows that, rather than modifying only the smallest five eigenvalues, the effect of $\mathbf{M}_2$ has been to modify most or even all of the eigenvalues in some way. This is explained by the fact that the eigenvectors used in the construction of $\mathbf{M}_2$ were not exact, but only approximations, derived from a different matrix, albeit one with similar spectral properties.

The most important observation to be made from Figure 4.25 is that the overall effect on the spectral properties of the matrix has been beneficial, with several of the smallest eigenvalues having been successfully deflated. Figure 4.26 confirms this, showing a magnification of Figure 4.25 from 0 to 0.5. The effect of $\mathbf{M}_2$ has been to deflate the smallest three eigenvalues from $\mathbf{J}_{RBF}\mathbf{M}_1$. The smallest remaining eigenvalue is now approximately 0.45, compared to the previous smallest of about 0.01.

This first set of figures illustrates the effectiveness of the preconditioning strategy in a case where the original Jacobian matrix was reasonably well conditioned. The next set of figures correspond to Test Problem 1 when

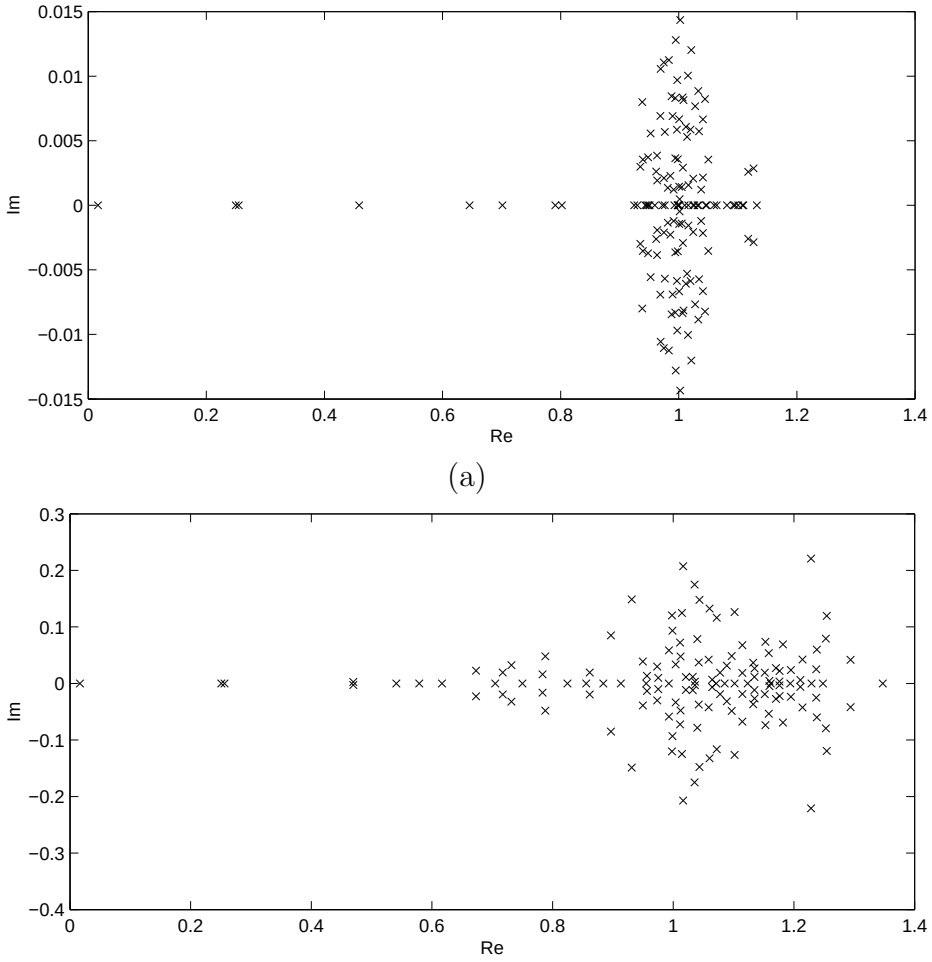


Figure 4.23: Eigenvalues of (a) $\mathbf{J}_{FE}\mathbf{M}_1$; (b) $\mathbf{J}_{RBF}\mathbf{M}_1$ for Test Problem 1 with $D_{yy} = 5$.

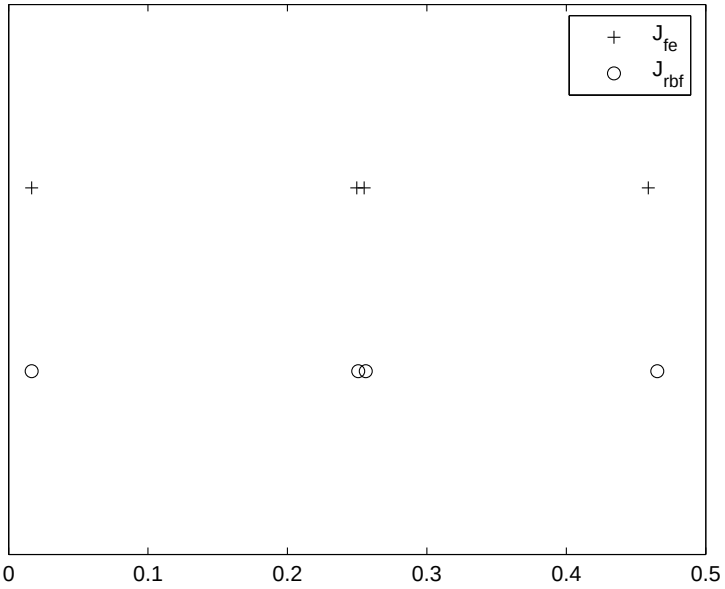


Figure 4.24: Eigenvalues less than 0.5 of $\mathbf{J}_{FE}\mathbf{M}_1$ and $\mathbf{J}_{RBF}\mathbf{M}_1$ for Test Problem 1 with $D_{yy} = 5$.

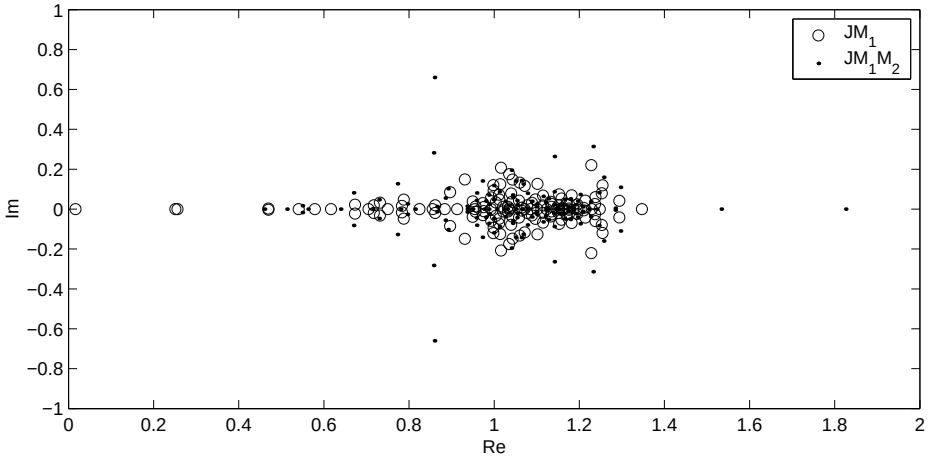


Figure 4.25: Superimposed eigenvalue spectra of $\mathbf{J}_{RBF}\mathbf{M}_1\mathbf{M}_2$ for Test Problem 1 with $D_{yy} = 5$.

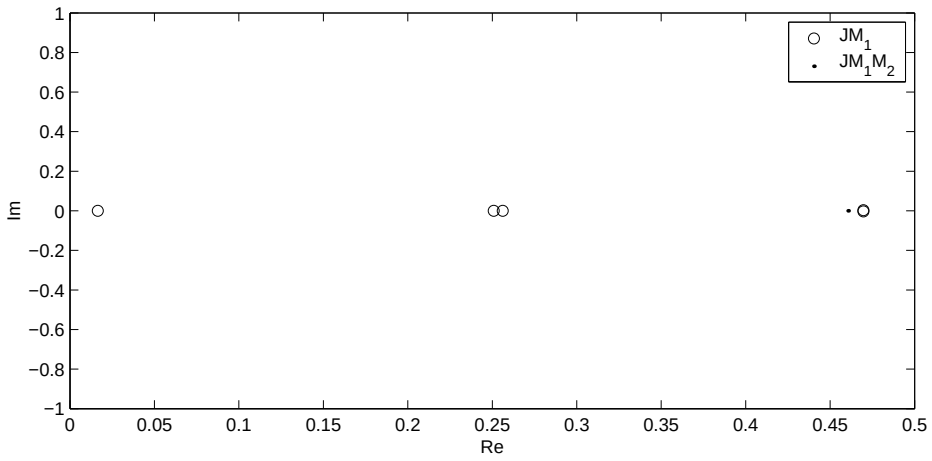


Figure 4.26: Superimposed eigenvalue spectra to the left of 0.5 for $\mathbf{J}_{RBF}\mathbf{M}_1\mathbf{M}_2$ for $D_{yy} = 5$.

$D_{yy} = 5000$, which produces a Jacobian matrix with much worse conditioning.

Figure 4.27 shows why this is so. Whereas in the previous example the eigenvalues were all less than one, in this case the largest eigenvalues are nearly 1000 times greater. An important observation is that even in this ill-conditioned case, the eigenvalue spectrum is very similar between the two matrices $\mathbf{J}_{FE}$ and $\mathbf{J}_{RBF}$.

The new eigenvalue spectra after preconditioning with $\mathbf{M}_1$ is shown in Figure 4.28. The effect of $\mathbf{M}_1$ in scaling the largest eigenvalues is clearly apparent, with most eigenvalues again clustered around unity. Recall that $\mathbf{M}_1$ uses only values drawn from the CV-FE Jacobian, $\mathbf{J}_{FE}$, yet it is still effective at preconditioning $\mathbf{J}_{RBF}$. The eigenvalue spectrum now ranges from zero to less than two, rather than several hundred. However, Figure 4.29 shows that the magnitudes of the smallest eigenvalues have diminished. This was not unexpected: in Section 2.5.2 this very point was raised and was the reason given for applying the preconditioners in the order $\mathbf{M}_1\mathbf{M}_2$ rather than

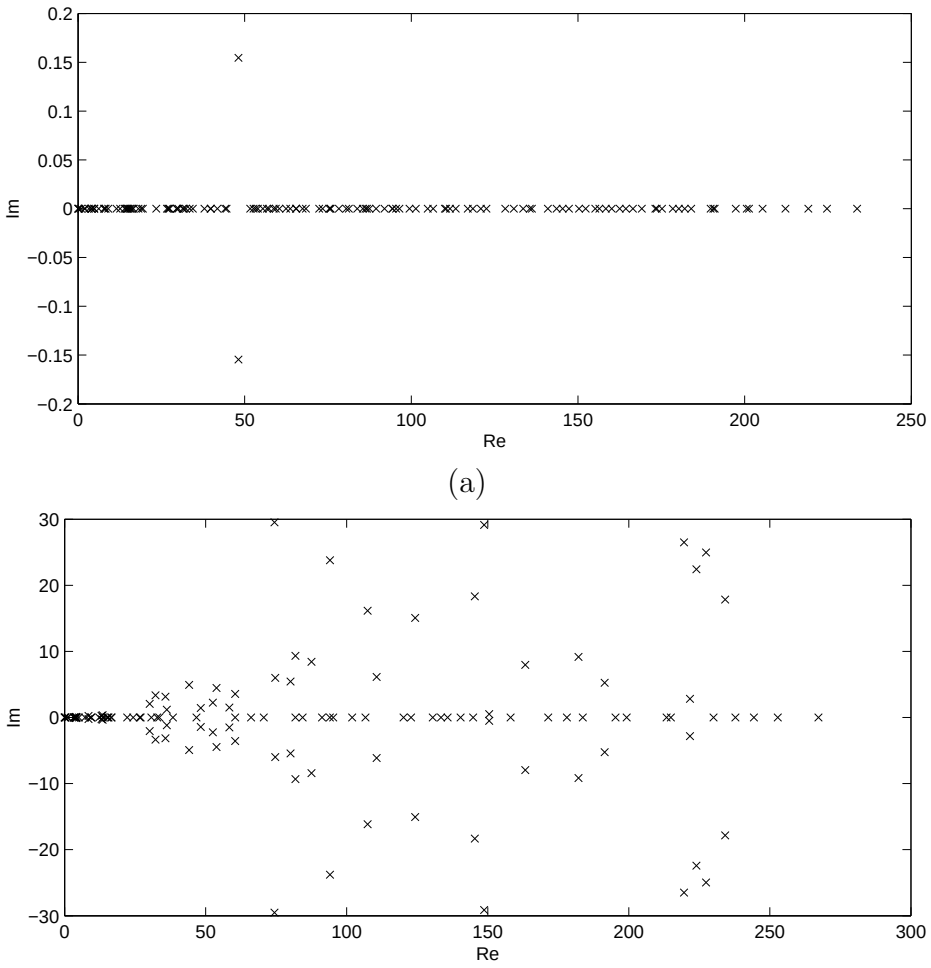
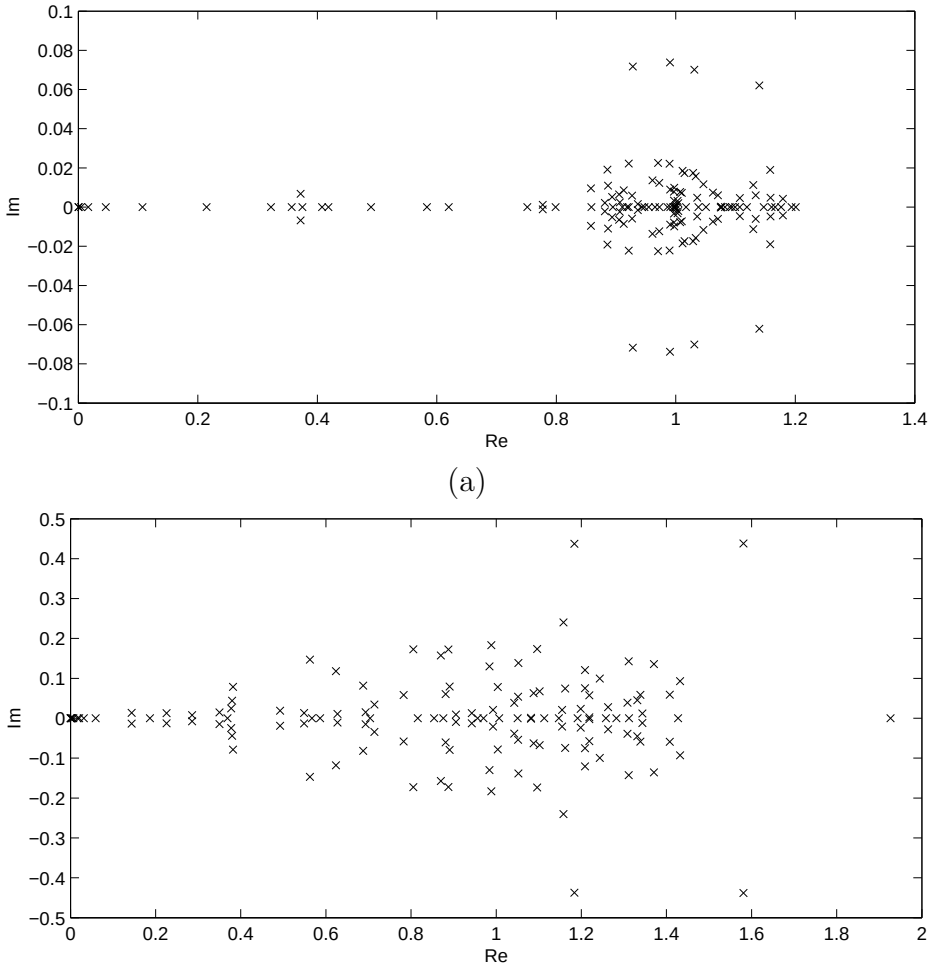


Figure 4.27: Eigenvalues of (a) $\mathbf{J}_{FE}$; (b) $\mathbf{J}_{RBF}$ for Test Problem 1 with $D_{yy} = 5000$.



(a)

(b)

Figure 4.28: Eigenvalues of (a) $\mathbf{J}_{FE}\mathbf{M}_1$; (b) $\mathbf{J}_{RBF}\mathbf{M}_1$ for Test Problem 1 with $D_{yy} = 5000$.

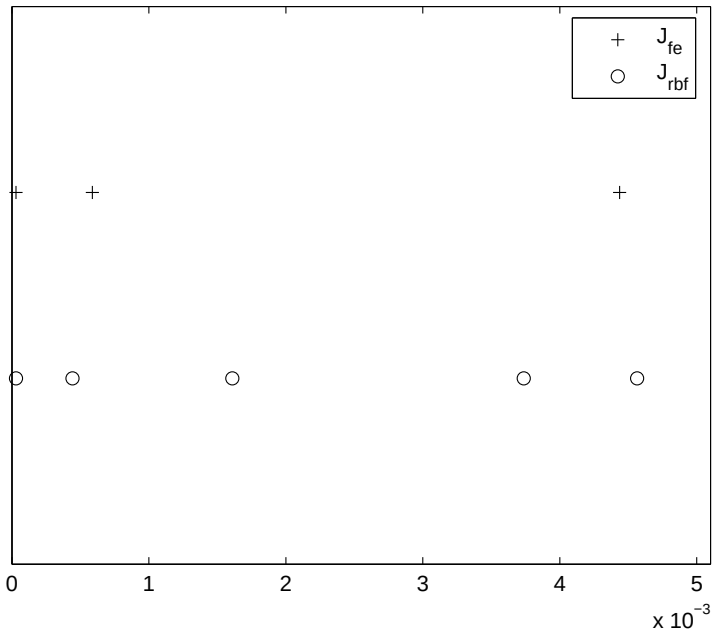


Figure 4.29: Smallest eigenvalues of $\mathbf{J}_{FE}\mathbf{M}_1$ and $\mathbf{J}_{RBF}\mathbf{M}_1$ for Test Problem 1 with $D_{yy} = 5000$.

$\mathbf{M}_2\mathbf{M}_1$. The strategy as a whole will still be successful provided that $\mathbf{M}_2$ is effective at deflating these new smallest eigenvalues.

Referring again to Figure 4.29, it appears that the deflation preconditioner $\mathbf{M}_2$, which is based on CV-FE information, could be successful at deflating the smallest two eigenvalues, but probably no more. Figure 4.30 shows the effect of applying $\mathbf{M}_2$ to $\mathbf{J}_{RBF}\mathbf{M}_1$. As was the case in the previous problem, because of the inexactness of the eigenvalue and eigenvector approximations, the effect of $\mathbf{M}_2$ has been to modify most or all of the eigenvalues, but again not to any detrimental effect.

The main effect of $\mathbf{M}_2$ can be seen in Figure 4.31, which shows that in fact the smallest three eigenvalues of $\mathbf{J}_{RBF}\mathbf{M}_1$ have been successfully deflated. There are still several small eigenvalues remaining, but these will be

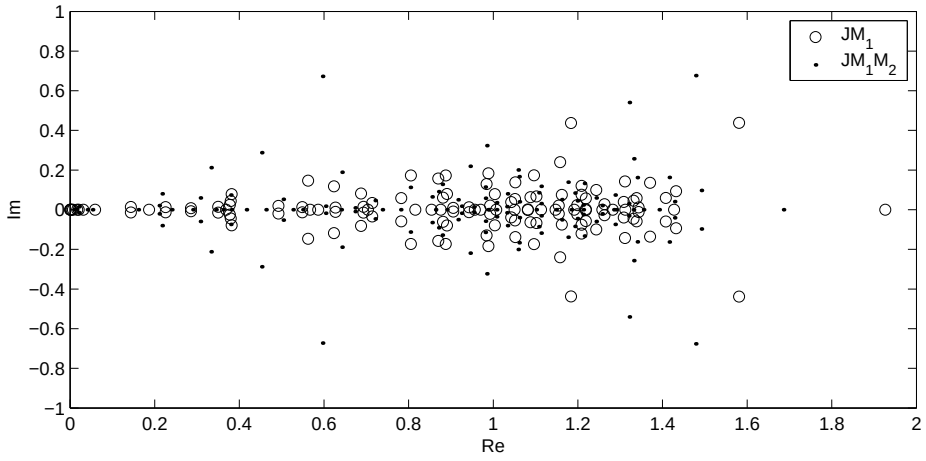


Figure 4.30: Superimposed eigenvalue spectra of $\mathbf{J}_{RBF}\mathbf{M}_1\mathbf{M}_2$ for $D_{yy} = 5000$.

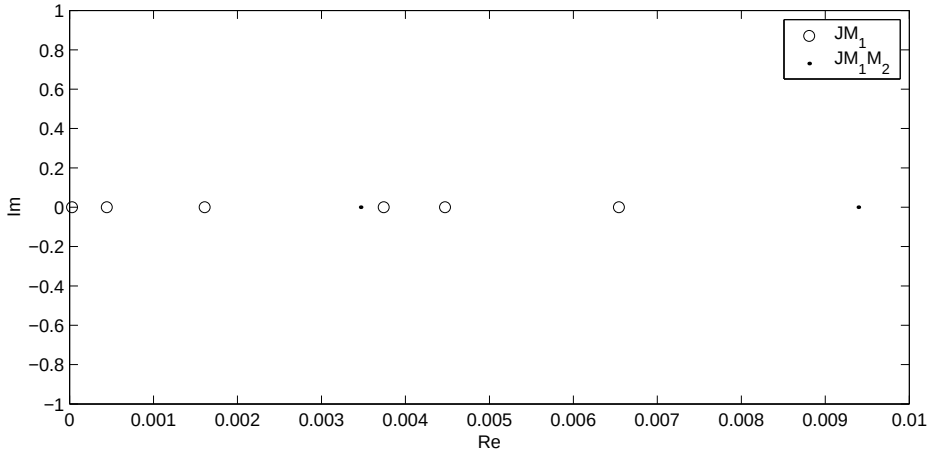


Figure 4.31: Superimposed eigenvalue spectra to the left of 0.01 for $\mathbf{J}_{RBF}\mathbf{M}_1\mathbf{M}_2$ for Test Problem 1 with $D_{yy} = 5000$.

handled within the GMRES-DR iterations of the linear solver. Alternatively, as discussed in Section 2.5.2, further deflation matrices $\mathbf{M}_3$, $\mathbf{M}_4$, etc. could be formed based on progressive information generated during the GMRES-DR iterations, but this strategy was not implemented for this thesis.

The main conclusion to be drawn from this section is that the cheaper CV-FE Jacobian may be used successfully to precondition the subsequent CV-RBF iterations. The spectral properties were found to be quite similar for the two Jacobian matrices, which is consistent with the fact that they are two representations of the same physical process. Additionally the method of shape functions can be thought of as a special case of the more general method of radial basis functions (see Section 2.3.4), so there is justification in considering the CV-FE Jacobian matrix as a simplification of the full CV-RBF Jacobian. The particular combination of preconditioners tested in this thesis has been found to be successful, but it is anticipated that other methods based on CV-FE information could also be successful.

4.4.2 Nonlinear function evaluations

This section demonstrates how the strategies outlined in Section 2.4.3 contribute to reducing the cost of the two-stage nonlinear solution process. All of the figures presented in this section are plots of the Newton-Krylov residual norms against the number of evaluations of $\mathbf{F}$ (2.14), using either CV-FE (3.37) or CV-RBF (3.38). Concentration is centred mostly on Test Problem 3 as it is a nonlinear problem, while the other two test problems are linear.

Table 4.14 lists the values of the Newton-Krylov parameters used for the tests. The number of eigenvectors used in deflation, k, was selected following similar experiments carried out in [13, 59]. All other parameter values follow the recommendations made in [50].

Table 4.14: Linear and nonlinear parameter values for numerical experiments.

Parameter	Description	Value
η_0	Initial forcing term	0.9
$\eta_{\max}$	Safeguard value for forcing term	0.9
γ	Coefficient in forcing term	0.9
α	Small parameter in line search	10^{-4}
β	Exponent in forcing term	2
$\theta_{\min}$	Safeguard lower bound for line search	0.1
$\theta_{\max}$	Safeguard upper bound for line search	0.5
maxarm	Maximum step size reductions	10
Linear:		
m	Number of iterations per cycle	40
k	Number of eigenvalues used in deflation	5

Recall from Section 2.4.3 that in a Newton-Krylov method, a forcing term may be used to minimise oversolving in computing the next Newton step $\boldsymbol{\delta\varphi}^{(n)}$. By plotting both the nonlinear residual $\|\mathbf{F}\|_2$ and the corresponding linear residuals $\|\mathbf{F} + \mathbf{J}\ \boldsymbol{\delta\varphi}\|_2$ (see (2.58)), the effectiveness of this approach can be judged. Oversolving will be evident whenever the linear residual is reduced further than the nonlinear residual.

Figure 4.32 is a plot of the residual norms against the number of function evaluations for solving Test Problem 3 using the two-stage solution strategy of CV-FE followed by CV-RBF, with an initial solution of $10xy(1-x)(1-y)$. The solid lines represent the linear residuals, while the dotted lines represent the nonlinear residuals. There are a number of important features to observe in this figure. First, observe that the two stages of the process are clearly separated. There is an initial reduction in residual norm associated with using CV-FE, followed by a sudden increase, followed by a second reduction associated with using CV-RBF.

This increase in the residual norm when moving from stage one to stage two is a reminder that even though the nonlinear residual may be small,

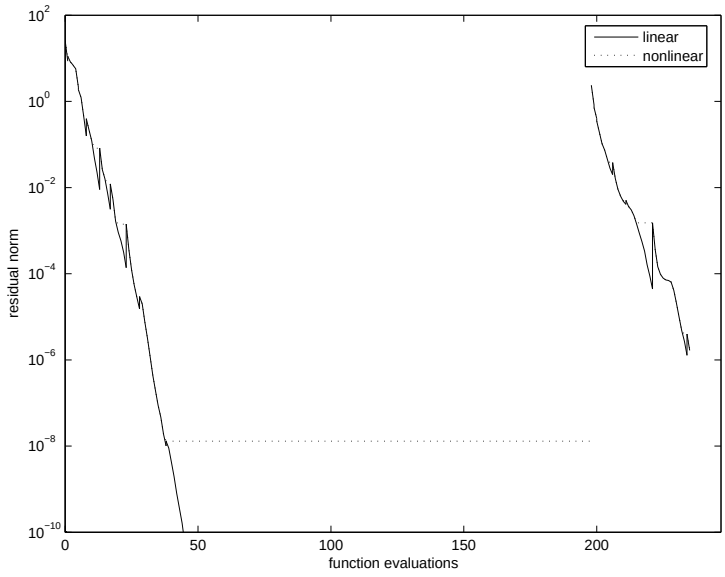


Figure 4.32: Linear and nonlinear residuals versus function evaluations for Test Problem 3 using two-stage solution strategy.

the accuracy of the actual finite volume solution can still be poor. It is the accuracy of the underlying discretisation that determines whether or not the solution of (2.13) will be a good representation of the physical solution. Recall that in this two-stage solution strategy, the CV-FE solution is used as the initial estimate for the CV-RBF iterations. Even though the CV-FE solution satisfies $\|\mathbf{F}_{FE}\| <$ tolerance, it is not nearly accurate enough to satisfy $\|\mathbf{F}_{RBF}\| <$ tolerance. Therefore, stage two commences with a sudden increase in the residual norm.

Also evident from Figure 4.32 is that there has not been any significant amount of oversolving committed, with the exception of the last nonlinear iteration in stage one, where the linear residual norm is reduced well beyond the actual tolerance of the method, and even beyond the limits of the graph, while the nonlinear residual stalls. In fact, this particular act of oversolving is deliberate. The reason for this is not a desire for higher solution accuracy, but

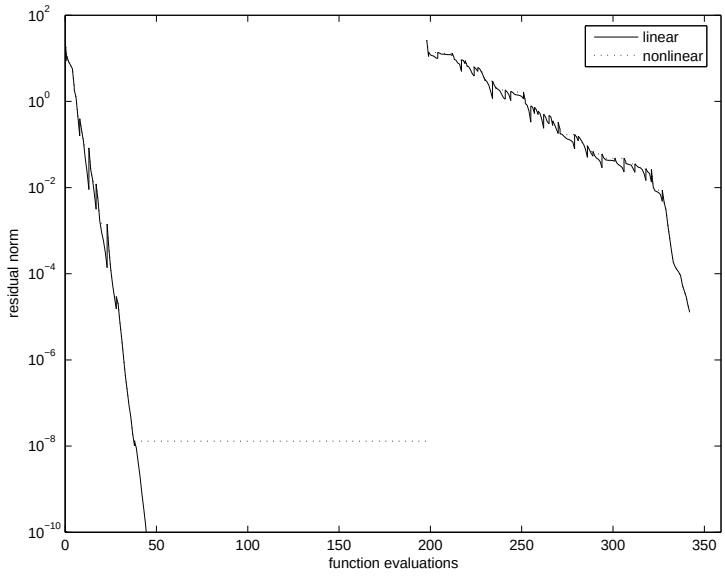


Figure 4.33: Linear and nonlinear residuals versus function evaluations for Test Problem 3 using restarting solution strategy.

so that the approximate eigenvectors generated in the GMRES-DR iterations are of high accuracy. These vectors then get incorporated into the deflation preconditioner $\mathbf{M}_2$ of Section 2.5.2 and applied during the CV-RBF iterations. Since the CV-FE iterations are much cheaper than the CV-RBF iterations, this extra time spent in stage one is beneficial even if it saves only a few CV-RBF iterations in stage two.

In producing Figure 4.32, the strategy of using the most recent stage one solution as the initial estimate for stage two has been employed. This is what is considered in this thesis to be the usual two-stage strategy. An alternative strategy would be to simply restart the method after stage one, using the original initial solution to commence stage two. Figure 4.33 shows the results of using this method. In this case more than twice as many stage two iterations are required to achieve convergence. Thus, there is significant benefit to be had from commencing stage two with the most recent stage one

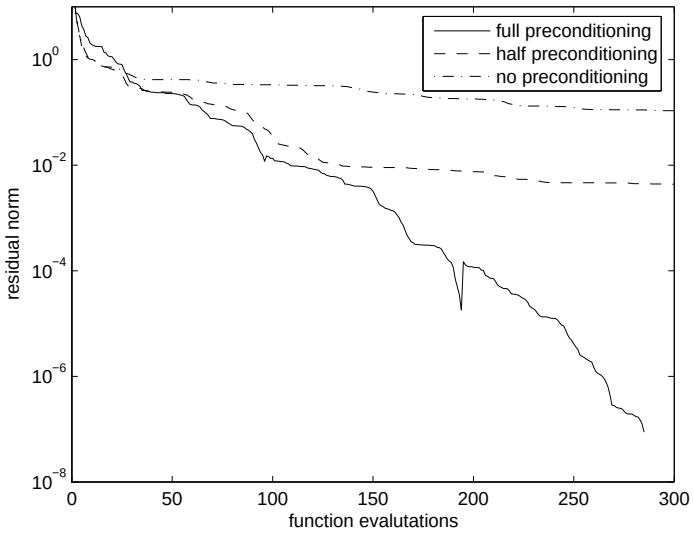


Figure 4.34: Linear residual versus function evaluations for Test Problem 1 with $D_{yy} = 5000$ using CV-RBF with various preconditioning strategies.

solution, rather than from the original initial estimate.

This section concludes with an investigation of how the convergence rates for all three test problems are affected by preconditioning. Each of Figures 4.34, 4.35 and 4.36 plots the (linear) CV-RBF residual norms against the number of function evaluations for three different choices of preconditioning strategy: full preconditioning, representing the preconditioner $\mathbf{M}_1\mathbf{M}_2$; half preconditioning, representing just the preconditioner $\mathbf{M}_1$; and no preconditioning.

Figure 4.34 shows, for Test Problem 1 with $D_{yy} = 5000$ how full preconditioning is necessary to achieve convergence. Using half preconditioning initially produces a convergence rate comparable to full preconditioning, but quickly results in stalling. Using no preconditioning at all results in the method stalling even sooner.

For Test Problem 2 with $v_x = 100$, whose results are shown in Figure

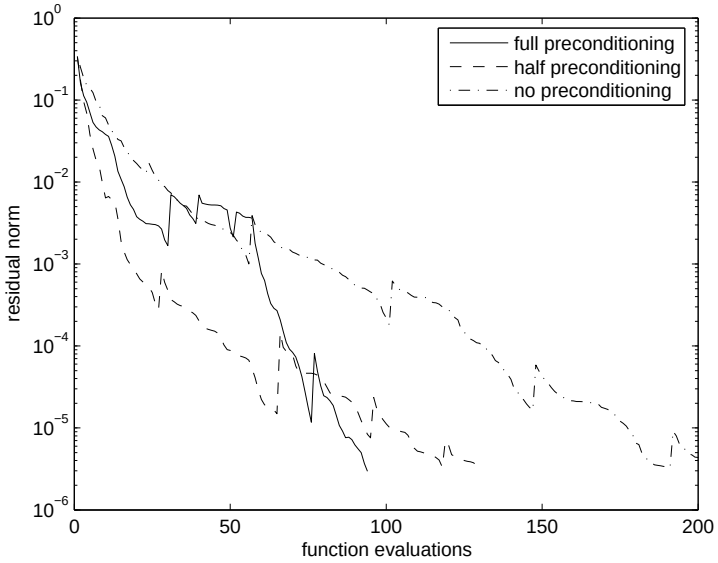


Figure 4.35: Linear residual versus function evaluations for Test Problem 2 with $v_x = 100$ using CV-RBF with various preconditioning strategies.

4.35, convergence is achieved even without preconditioning, but using the preconditioning techniques outlined in this thesis means fewer iterations are required for convergence. Interestingly, the strategy of using half preconditioning ($\mathbf{M}_1$ only) initially gives a faster convergence rate than using full preconditioning. However, it is soon caught by the full preconditioning method whose final convergence rate is much higher.

In Figure 4.36, which corresponds to Test Problem 3, the method once again converges even without preconditioning. Nevertheless, applying the preconditioner $\mathbf{M}_1$ results in convergence in only half the number of iterations, and applying $\mathbf{M}_1\mathbf{M}_2$ saves another 20% on top of that.

4.4.3 The effect of the RBF parameter c^2

In this section the effect the multiquadric parameter c^2 on the accuracy of the computed solution is investigated. Recall from Section 2.3.3 and Table

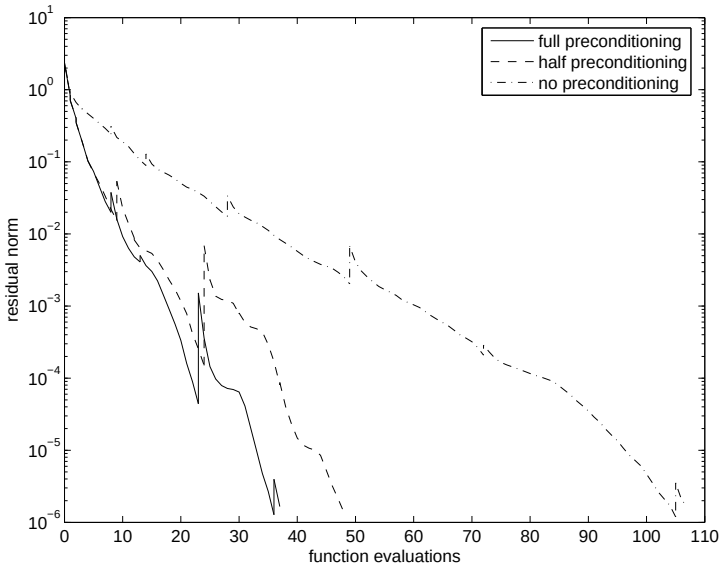


Figure 4.36: Linear residual versus function evaluations for Test Problem 3 using CV-RBF with various preconditioning strategies.

2.2 that the multiquadric has a parameter c^2 associated with it. The value of this parameter is known to effect both the accuracy and the conditioning of the interpolation [18, 53].

In [72] an algorithm is presented for approximating the optimum value of c^2 so that the error in the interpolation is minimised. The expense in applying this method is nontrivial however, and was deemed too costly to be of use in this finite volume framework. In any case, it was found during numerical experimentation that the conditioning of the RBF coefficient matrix was more relevant in determining an appropriate value of c^2 than any issues related to accuracy.

In Figure 4.37 the error in the CV-RBF solution for Test Problem 1, measured using (4.16), is plotted against the value of c^2 used in the interpolations. The figure suggests that a value of c^2 around 3.0 offers the best accuracy, but that any value greater than or equal to 2.0 provides a similar level of ac-

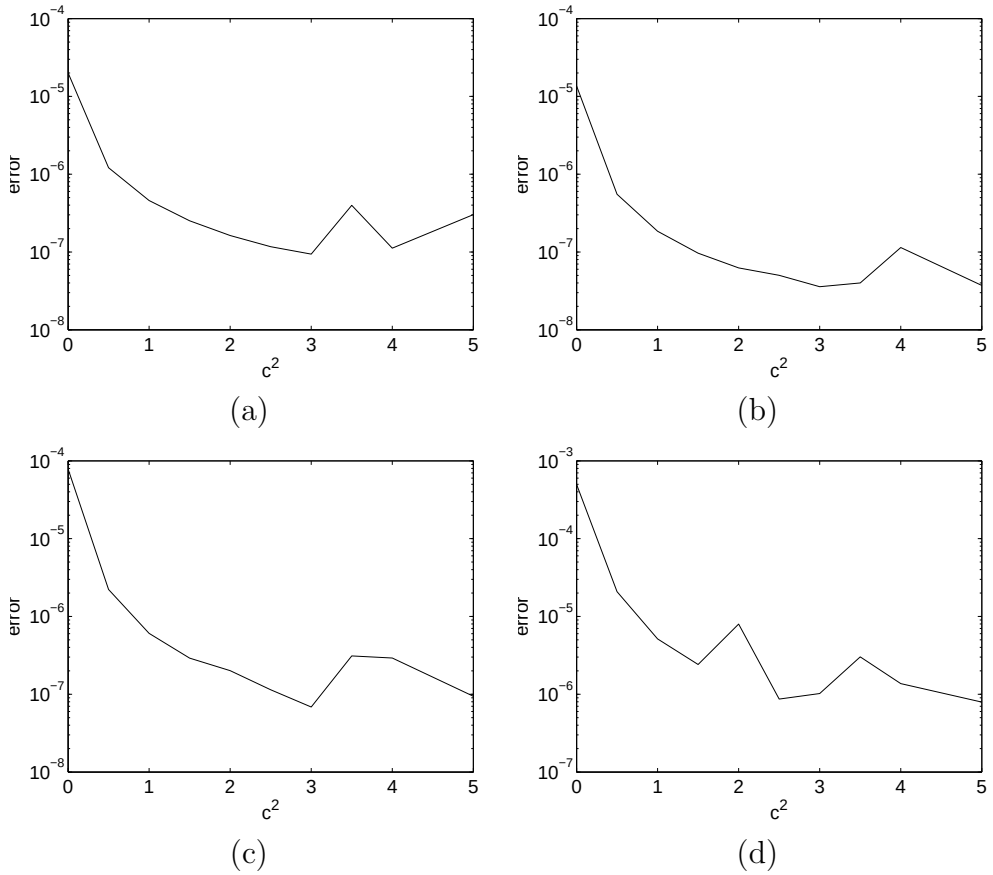


Figure 4.37: Error in solution versus c^2 for Test Problem 1 where (a) $D_{yy} = 5$; (b) $D_{yy} = 50$; (c) $D_{yy} = 500$; (d) $D_{yy} = 5000$.

curacy. The figure also suggests that if the plots were continued further, the general trend would be for the solution error to decrease as c^2 was increased. Thus it may appear that a much larger value of c^2 would offer improved accuracy.

Figure 4.38 illustrates the problem with this notion. The plots show, for Test Problem 1, the number of function evaluations required to achieve convergence in the nonlinear solution process, against the value used for c^2. It explains why the previous plots were truncated even though the trend suggested that better accuracy could be achieved by further increasing c^2.

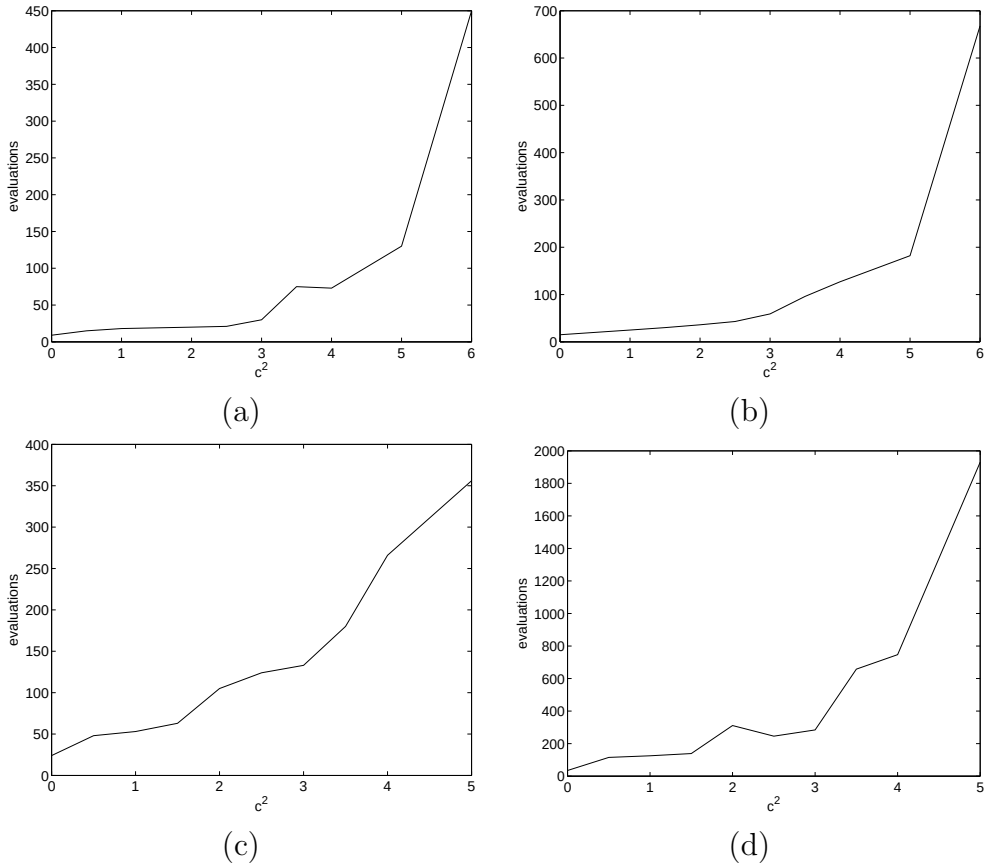


Figure 4.38: Number of function evaluations required for convergence versus c^2 for Test Problem 1 where (a) $D_{yy} = 5$; (b) $D_{yy} = 50$; (c) $D_{yy} = 500$; (d) $D_{yy} = 5000$.

The reality is that the number of iterations required to achieve convergence balloons when c^2 is increased too far, and in fact beyond the values shown in this figure convergence of the nonlinear process was not even achieved.

This rapid increase in the number of nonlinear iterations required for convergence is associated with an increasing number of RBF coefficient matrices becoming numerically singular (so ill-conditioned that one or more of their singular values are less than machine precision). Although the truncated singular value decomposition outlined in Section 2.3.3 can handle this

situation, the ill-conditioning filters through to the nonlinear system (2.13), and eventually the problem is too difficult to solve using the Jacobian-Free Newton-Krylov method outlined in Section 2.4.

Figure 4.39 shows the corresponding error plots for Test Problems 2 and 3. Once again, they suggest that larger values of c^2 offer higher accuracy than smaller values. Note that taking $c^2 = 0$ really corresponds to using a linear radial basis function (Table 2.2), and offers the worst accuracy. For these problems, using a genuine multiquadric ($c^2 > 0$) is essential to achieve good accuracy, and overall the accuracy improves gradually as c^2 is increased. Figure 4.40 shows that this observation is again tempered by the fact that larger values of c^2 lead to much greater numbers of iterations required for convergence.

Thus, choosing an appropriate value for the parameter c^2 is a balancing act. While larger values tend to result in higher accuracy finite volume solutions, they also tend to require more nonlinear iterations to achieve convergence. Conversely, smaller values can give fast convergence, but at the cost of reduced accuracy. The choice of $c^2 = 3$ for the test problems in this section was based on the premise that a single value be used for all three problems, providing near-optimal accuracy while not requiring excessively many nonlinear iterations.

In practice, the number of nonlinear iterations is likely to be a more dominant factor than the accuracy in choosing an appropriate value of c^2. The graphs in Figures 4.38 and 4.40 show that the number of nonlinear iterations required for convergence quickly gets out of hand as c^2 is increased. By contrast, the corresponding increase in accuracy is only minor. On the whole, for these test problems the value $c^2 = 3$ is the point at which convergence issues begin to dominate, and thus the prospect of using a larger value of c^2

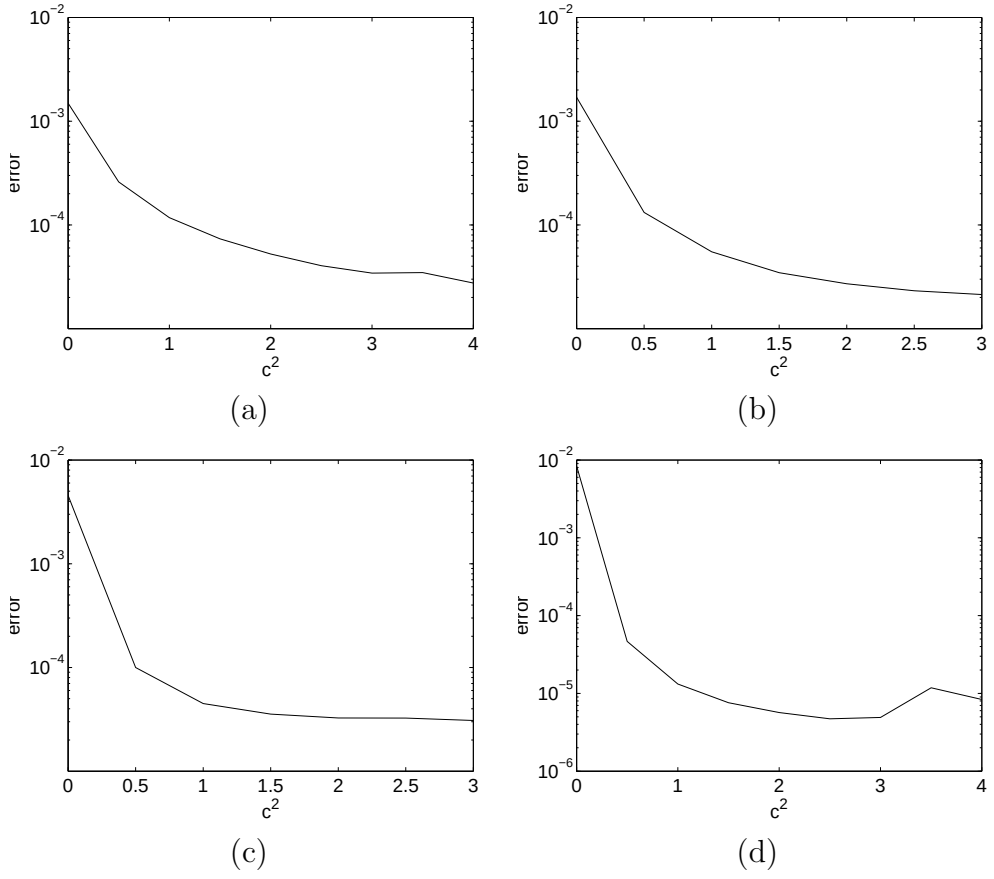


Figure 4.39: Error in solution versus c^2 for Test Problem 2 with (a) $v_x = 1$; (b) $v_x = 10$; (c) $v_x = 100$; (d) Test Problem 3.

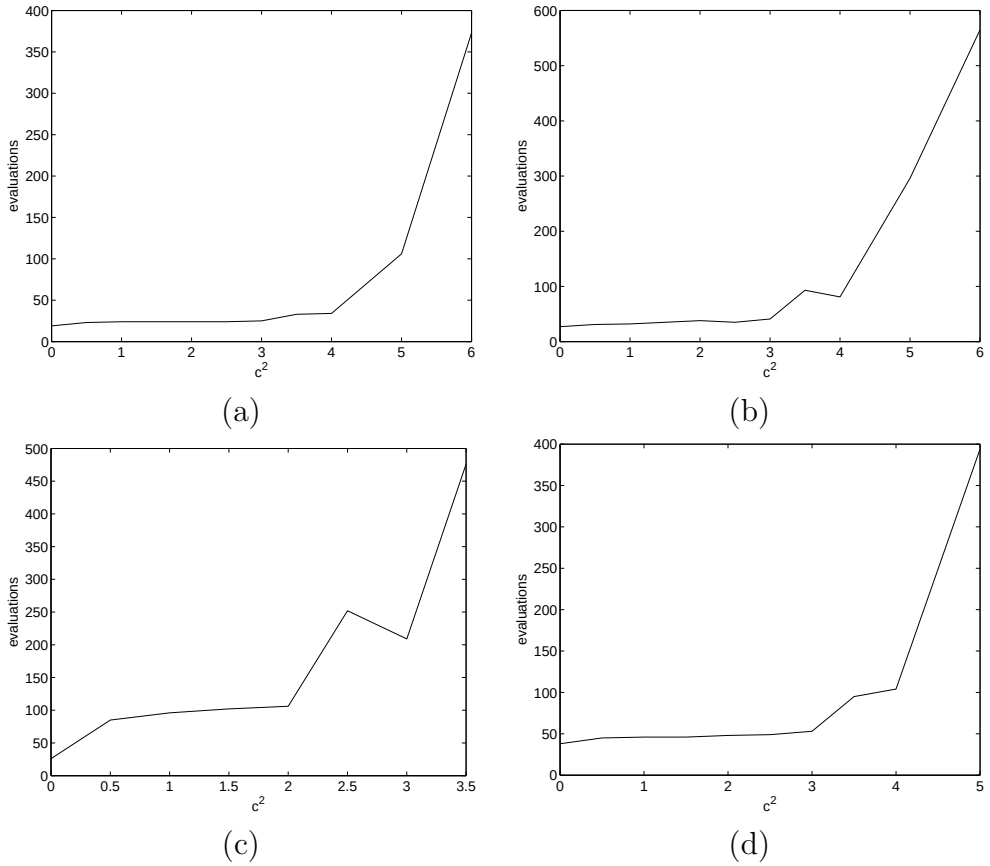


Figure 4.40: Number of function evaluations before convergence versus c^2 for Test Problem 2 with (a) $v_x = 1$; (b) $v_x = 10$; (c) $v_x = 100$; (d) Test Problem 3.

is unattractive.

Finally, it should be noted that there is some inconsistency throughout the literature on the position of the parameter c^2 in the multiquadric. The original "Hardy's multiquadric" was defined with the positioning as such [35]:

$$\phi(r) = \sqrt{c^2 + r^2} \tag{4.20}$$

When discussed in the more general context of radial basis functions, some authors define it as

$$\phi(r) = \sqrt{1 + c^2 r^2} \tag{4.21}$$

so that it is consistent with the Gaussian, and other infinitely smooth RBFs in the placement of the parameter. In this thesis the original definition, as presented in [70] has been used. The reader should note however that when the alternative definition is used, it is *small* values of c^2 that lead to ill-conditioning, as reported in [53] for example.

4.5 Conclusions

In this chapter the effectiveness of the two-dimensional schemes proposed in the previous chapter was investigated by applying them to three different test problems. A family of unstructured, triangular meshes was used for conducting these tests.

In Section 4.3.1, the analytic solutions were compared to the numerical solutions using the CV-FE method and the CV-RBF method, both visually and by measuring the relative errors over the entire mesh. For all three test problems the CV-RBF method gave rise to improved accuracy, up to three orders of magnitude. Furthermore, the contour plots produced by the

RBF solutions were indistinguishable from the analytic solution, even when high anisotropy or high advection was introduced. In particular, in Section 4.3.1 where Test Problem 2 was solved with cell Peclet numbers of 10000 and 100000, the CV-RBF solution did not exhibit the oscillations that were present in the CV-FE solution (Figure 4.13 and 4.14). Using upwinding with CV-FE also corrected these oscillations (Figure 4.15), however in this case the solution was more diffuse in profile than the analytic solution. In Section 4.3.1, tests conducted using shape functions paired with Gaussian quadrature showed that there was no significant advantage in using Gaussian quadrature over the midpoint rule in this case, justifying the use of the midpoint rule in the CV-FE method on efficiency grounds.

The results in Section 4.3.2 showed how the error in both the CV-FE method and the CV-RBF method was reduced as the mesh was refined. Plots of the solution errors against the average control volume face lengths were used to estimate the order of accuracy of the methods. It was found that the CV-RBF method with 6 nodes per interpolation for the most part performed no better than CV-FE in this regard, although the accuracy it offered was still consistently higher. The exception was for the more difficult diffusion problems, where CV-FE was unable to obtain second order accuracy, while the CV-RBF method with $n = 6$ maintained second order accuracy.

The CV-RBF methods with 12 and 25 nodes consistently offered improved orders of accuracy over CV-FE based on the measurements given in this chapter. Orders of between $\mathcal{O}(h^3)$ and $\mathcal{O}(h^5)$ were recorded for the 25 node case.

The method of preconditioning, incorporating elements of both CV-FE and CV-RBF was found in Section 4.4.1 to be effective in dealing with both small and large eigenvalues in the Jacobian spectrum. It was demonstrated in Section 4.4.2 how applying this preconditioner consistently resulted in a

reduction in the number of nonlinear function evaluations required to achieve convergence, and in some cases allowed convergence to be achieved where the method would otherwise have stalled.

Also in Section 4.4.2 the two-stage nonlinear solution strategy was found to be effective at reducing the number of CV-RBF iterations required for convergence compared to using a full restarting of the method. The deliberate oversolving of the CV-FE iterations in stage one was found to be a worthwhile strategy, with the extra time spent during this stage resulting in an effective preconditioner while not adding significantly to the overall time for solution. The use of an inexact Newton method with forcing term was found to be largely effective at eliminating oversolving during stage two.

Finally, in Section 4.4.3 the effect of the multiquadric RBF parameter c^2 was investigated. It was found that large values of c^2 are to be avoided, as they tend to result in greatly increased numbers of nonlinear iterations required to achieve convergence. Conversely, very small values of c^2 are also undesirable, as they lead to relatively inaccurate solutions. The choice of $c^2 = 3$ for these test problems tends to yield solutions of relatively high accuracy, without significant penalty in terms of nonlinear iterations.

Chapter 5

Three-dimensional considerations

In this chapter the details of implementing the finite volume method in three dimensions are presented. The structure of this chapter closely follows that of Chapter 3, which dealt with two-dimensional issues.

5.1 Meshing

In this section the properties of three-dimensional finite element meshes are discussed, complementing the discussion that given for two-dimensional meshes in Section 3.1.

Three-dimensional finite element mesh generators generally produce meshes consisting of either tetrahedral or hexahedral elements. Some more advanced mesh generators are able to produce mixed meshes, combining these element types. In this thesis, all three-dimensional meshes are tetrahedral.

The comparison of structured and unstructured meshes presented in Section 3.1 applies equally well to three dimensions. The advantage of using

an unstructured mesh is the flexibility to deal with irregular domains, and the ability to refine the mesh in regions that require closer attention. The disadvantage, especially in three dimensions, is in the complexity of the software needed to generate, work with, and visualise results over unstructured meshes. For this thesis, the mesh generator *Gmsh* [31] was used to generate the unstructured tetrahedral meshes.

5.2 Forming control volumes

Finite volume discretisation relies on constructing control volumes around each node in the mesh. In this section, the method used in the code written for this thesis, for taking the nodes and elements defined in a three-dimensional, finite element mesh and constructing control volumes is presented.

The process of forming control volumes is more complicated in three dimensions than in two, both to implement and to visualise. The process can perhaps best be described by considering how to form sub-control volumes. Recall from Section 3.2 that sub-control volumes are the contributions of elements towards forming control volumes around nodes. A single control volume is composed of several sub-control volumes, each of which is contributed by one of the elements sharing that control volume's central node. It is also useful to consider the elements themselves as being composed of sub-control volumes. In two dimensions, every triangular element is composed of exactly three sub-control volumes. In three dimensions, every tetrahedral element is composed of exactly four sub-control volumes.

Figure 5.1 shows how a single sub-control volume of a tetrahedral element is constructed in three dimensions. The process is an extension of method

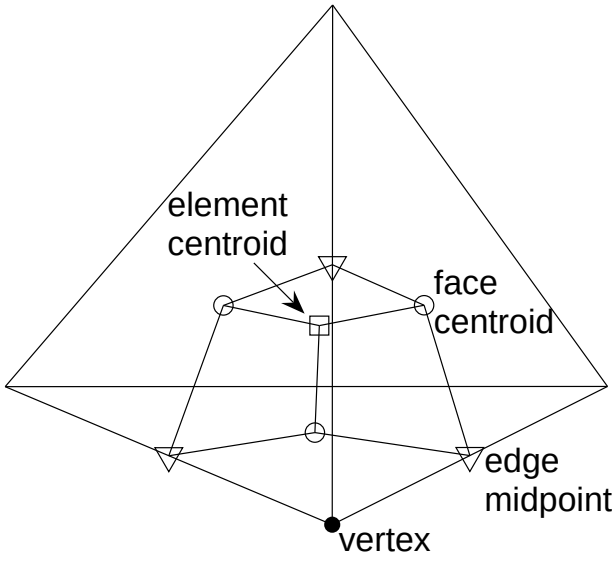
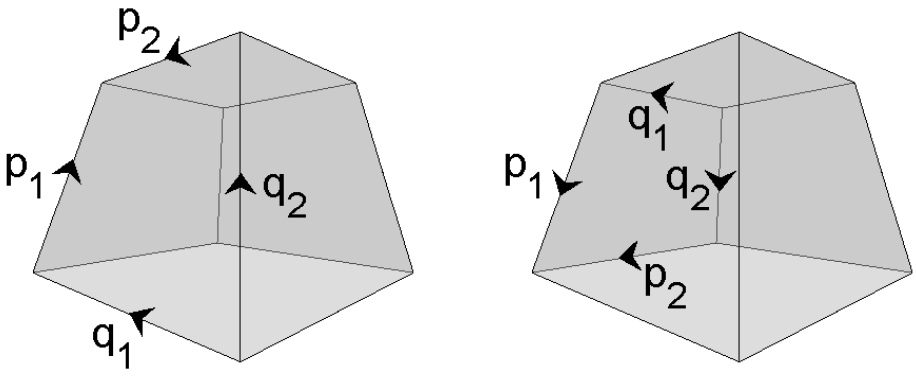


Figure 5.1: A sub-control volume with enclosing element.

(a) described in Section 3.2 (the three-dimensional analogue of method (b) is not pursued here.) It can be observed from Figure 5.1 that each sub-control volume is hexahedral in shape, with three "inner" faces and three "outer" faces. The inner faces, those that are internal to non-boundary control volumes, are sections of the element's own faces, bounded by lines connecting the element vertex, the edge midpoints and the face centroids. The outer faces, those that separate adjoining control volumes, are bounded by lines connecting the element centroid, the face centroids and the edge midpoints.

Each sub-control volume face is quadrilateral in shape. The convention of using the $\mathbf{p}$ and $\mathbf{q}$ vectors, as was done in two dimensions (recall Figure 3.4) is continued in this section. For inner faces, the $\mathbf{q}$ vectors point away from the vertex while the $\mathbf{p}$ vectors point to the face centroid. For outer faces, the $\mathbf{q}$ vectors point away from the element centroid while the $\mathbf{p}$ vectors point to the edge midpoint. This is illustrated in Figure 5.2. The numbering is such



(a) (b)

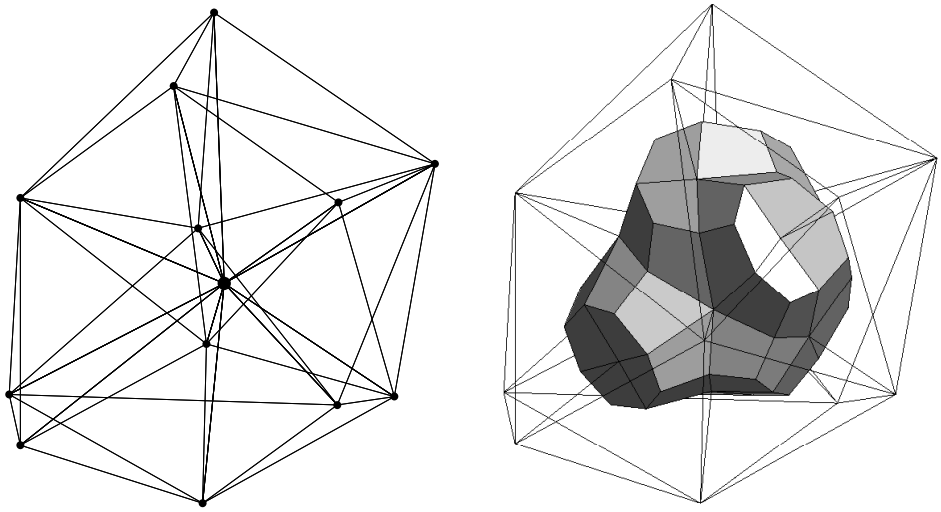
Figure 5.2: A sub-control volume face: (a) inner; (b) outer.

that $\mathbf{p}_1 \times \mathbf{p}_2$ is an outer normal to the sub-control volume face. Note that this means the numbering will be reversed for the neighbouring sub-control volume that shares the face. Formula (3.6) for computing the area of these faces is repeated here for convenience:

$$\text{Area} = \frac{1}{2} \|(\mathbf{q}_1 + \mathbf{p}_1) \times (\mathbf{p}_2 - \mathbf{p}_1)\|. \tag{5.1}$$

In three dimensions, the resulting control volumes tend to be much more complex than in two dimensions. For example, in a two-dimensional, triangular mesh, a typical control volume consists of either 12 or 6 faces, depending on whether method (a) or method (b) was used in its construction. In a three-dimensional, tetrahedral mesh, a typical control volume will consist of 30 or more faces. The reason for this extra complexity is twofold. First, sub-control volumes in three dimensions contribute three faces to the control volume, compared with two in two dimensions. Second, many more elements may share a given node in three dimensions.

Figure 5.3(a) shows how twelve different elements all share a particular node in a three-dimensional, tetrahedral mesh. With each of these elements

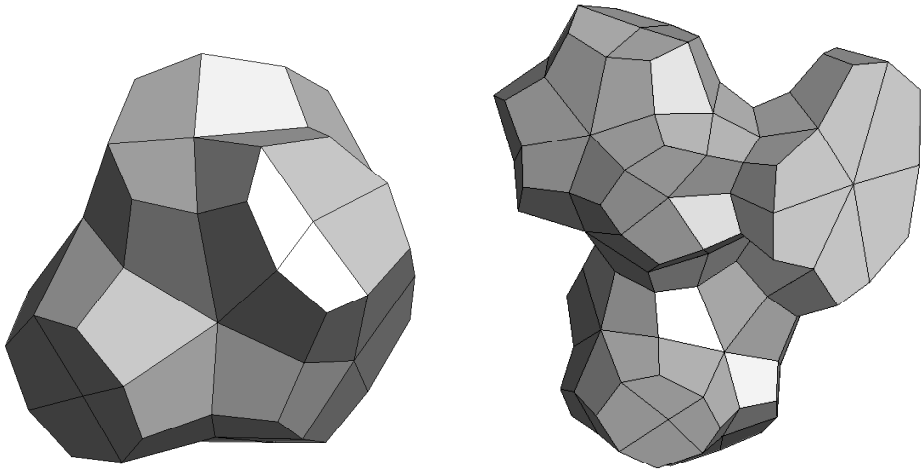


(a) (b)

Figure 5.3: (a) Mesh structure and (b) control volume for a given node.

contributing a sub-control volume consisting of three faces, the resulting control volume illustrated in Figure 5.3(b) consists of 36 faces altogether.

Figure 5.4(a) is a closer look at this single control volume, with the surrounding mesh no longer visible. The complexity of this figure makes it much more difficult in three dimensions to visualise how neighbouring control volumes fit together. Figure 5.4(b) shows how three neighbouring control volumes, one a boundary control volume, fit together. Every one of a control volume's faces either lies on a boundary, or is shared with exactly one other control volume in the mesh. The finite volume method requires computing fluxes though every one of these faces, and thereby computing the net flow through every control volume in the mesh.



(a) (b)

Figure 5.4: (a) Single control volume; (b) Three adjoining control volumes.

5.3 Conventions

Recall again from Section 2.3.6 that in order to ensure that the underlying finite volume conservation laws hold at the discrete level, fluxes computed through control volume faces must be consistent from the point of view of both adjoining control volumes. In this section the implementation aspects of ensuring this consistency in three dimensions are discussed, along with other numbering conventions.

A tetrahedral element consists of four nodes, four faces and four sub-control volumes. A simple numbering scheme to ensure one can easily associate nodes, faces and sub-control volumes with one another is shown in Figure 5.5. Each node, face and sub-control volume is numbered from 1 to 4, with node 1 belonging to sub-control volume 1 and lying opposite face 1, and so on for nodes 2, 3 and 4. Note that for readability, only face 1 and control volume 1 are labelled in Figure 5.5. Once again, this is a *local* numbering scheme particular to a given element. *Globally*, every node, face and

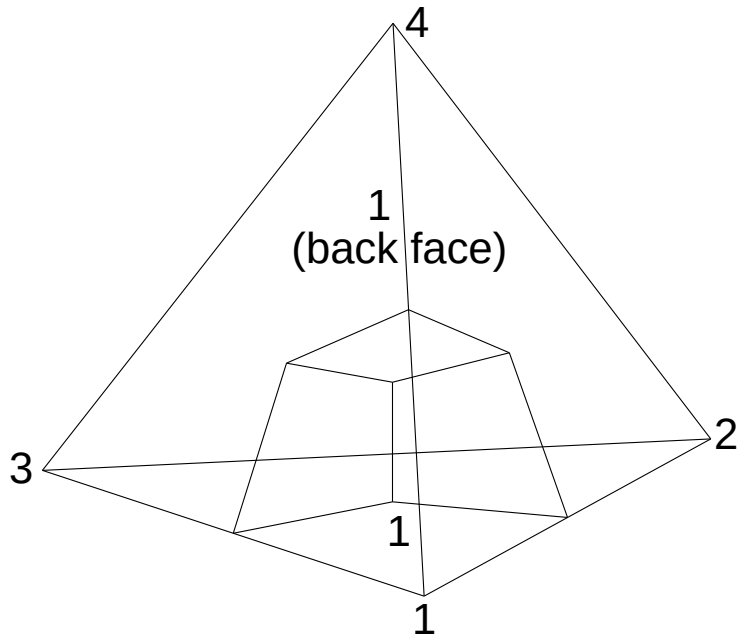


Figure 5.5: Numbering convention for nodes, faces and sub-control volumes.

sub-control volume in the mesh has a unique identifier, with no particular relationship between them.

In three dimensions, there is no easy convention to ensure that each control volume face is processed only once. Recall that in two dimensions, using an anticlockwise ordering (see Figure 3.6), ensures that each sub-control volume is responsible for the flux through a single control volume face. In three dimensions there can be no such convention, because there are a total of six control volume faces within every element, but only four sub-control volumes. The responsibility for these six control volume faces must therefore be carefully distributed amongst the sub-control volumes to ensure that each face is processed exactly once. Two possible conventions are

- sub-control volumes 1,2 and 3 are assigned two faces each, sub-control volume 4 is assigned none.

- sub-control volume 1 is assigned three faces, sub-control volumes 2,3 and 4 are assigned one each.

For this thesis the second convention is used. Boundary faces are handled the same way as in two dimensions: each sub-control volume must check if any of its inner faces lies on a boundary, and if so, compute the fluxes through it as well. In practice, this check is performed just once, at the time when the control volumes are being constructed. At this time, the inner faces are assigned to boundary control volumes, along with whatever outer faces have been assigned using the convention just discussed.

5.4 Integration

In Section 2.2, the approximation of the advective-diffusive flux and source components of (2.10) using numerical integration were discussed. The details pertaining to these integrations in three dimensions are now discussed.

Consider again the control volume illustrated in Figure 5.4(a). As was done in equation (2.16), the integral around the control volume's surface may be broken up into integrals over each of its faces.

$$\iint_{\sigma_i} (\mathbf{D}\boldsymbol{\nabla}\varphi \cdot \hat{\mathbf{n}} - \mathbf{v}\varphi)\, \mathrm{d}\sigma = \sum_{\tau=1}^{Nf_i} \iint_{\sigma_{i\tau}} (\mathbf{D}\boldsymbol{\nabla}\varphi \cdot \hat{\mathbf{n}} - \mathbf{v}\varphi)\, \mathrm{d}\sigma. \tag{5.2}$$

5.4.1 Midpoint rule

These surface integrals may one again be computed using the midpoint rule

$$\iint_{\sigma_{i\tau}} (\mathbf{D}\boldsymbol{\nabla}\varphi \cdot \hat{\mathbf{n}} - \mathbf{v}\varphi)\, \mathrm{d}\sigma \approx A(\sigma_{i\tau}) \left[\mathbf{D}\boldsymbol{\nabla}\varphi \cdot \hat{\mathbf{n}} - \mathbf{v}\varphi\right]_{\mathbf{m}_{i\tau}} \tag{5.3}$$

where $A(\sigma_{i\tau})$ is the area of face $\sigma_{i\tau}$, computed using (5.1). This yields the approximation

$$\iint_{\sigma_i} (\mathbf{D}\boldsymbol{\nabla}\varphi - \mathbf{v}\varphi) \cdot \hat{\mathbf{n}} \, \mathrm{d}\sigma \approx \sum_{\tau=1}^{Nf_i} A(\sigma_{i\tau}) \left[(\mathbf{D}\boldsymbol{\nabla}\varphi - \mathbf{v}\varphi) \cdot \hat{\mathbf{n}} \right]_{\mathbf{m}_{i\tau}} . \tag{5.4}$$

Approximating the source component of (2.10) using (2.19) requires computing the volume ΔV_i. This is achieved by summing the volumes of each of the sub-control volumes of V_i. Each sub-control volume is a hexahedron whose volume may itself be computed by decomposing it into six pyramids which share an arbitrary vertex $\mathbf{R}$ located within the hexahedron. Figure 5.6 shows one such example. The formula for the volume of this pyramid is given in [16]:

$$\text{Volume} = \frac{1}{12}(\mathbf{r}_1 + \mathbf{r}_2) \cdot [(\mathbf{q}_1 + \mathbf{p}_1) \times (\mathbf{p}_2 - \mathbf{p}_1)] \tag{5.5}$$

5.4.2 Gaussian quadrature

Gaussian quadrature may also be used to compute the advective-diffusive flux across a control volume face, but this requires two-dimensional Gaussian quadrature. In Section 3.4.2, this method was used to compute two-dimensional "volume" integrals. Now, in three dimensions, it is used to compute surface integrals. Formula (3.22) still applies, and once again the

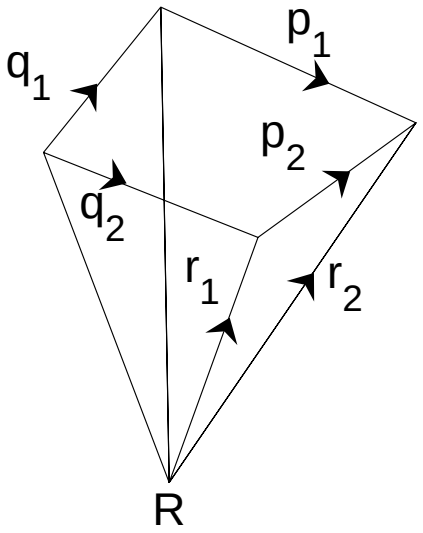


Figure 5.6: One of six quadrilateral-based pyramids comprising a sub-control volume. All share the arbitrary vertex **R**.

parameterisation

$$\begin{aligned}\mathbf{x}(\xi,\eta) = &\frac{1}{4}(1-\xi)(1-\eta)\mathbf{R}_1 + \frac{1}{4}(1+\xi)(1-\eta)\mathbf{R}_2 \\ &+ \frac{1}{4}(1+\xi)(1+\eta)\mathbf{R}_3 + \frac{1}{4}(1-\xi)(1+\eta)\mathbf{R}_4, \\ &-1 \le \xi \le 1, \quad -1 \le \eta \le 1\end{aligned} \tag{5.6}$$

is used, with the vertices $\mathbf{R}_i$ as shown in Figure 3.4(b), and the Jacobian of the transformation given by

$$\left|\frac{\partial \mathbf{x}}{\partial(\xi,\eta)}\right| = \left\|\frac{\partial \mathbf{x}}{\partial \xi} \times \frac{\partial \mathbf{x}}{\partial \eta}\right\|. \tag{5.7}$$

Applying four-point Gaussian quadrature in two dimensions, with the use of Table 2.1, yields

$$\begin{aligned}\iint_{\sigma_{i\tau}} (\mathbf{D}\boldsymbol{\nabla}\varphi\cdot\hat{\mathbf{n}} - \mathbf{v}\varphi)\,\mathrm{d}\sigma \approx\ & [\mathbf{D}\boldsymbol{\nabla}\varphi\cdot\hat{\mathbf{n}} - \mathbf{v}\varphi]_{\mathbf{x}(u_1,u_1)} \left\|\frac{\partial\mathbf{x}}{\partial\xi}\ \times\ \frac{\partial\mathbf{x}}{\partial\eta}\right\|_{(u_1,u_1)} \\ &+\ [\mathbf{D}\boldsymbol{\nabla}\varphi\cdot\hat{\mathbf{n}} - \mathbf{v}\varphi]_{\mathbf{x}(u_1,u_2)} \left\|\frac{\partial\mathbf{x}}{\partial\xi}\ \times\ \frac{\partial\mathbf{x}}{\partial\eta}\right\|_{(u_1,u_2)} \\ &+\ [\mathbf{D}\boldsymbol{\nabla}\varphi\cdot\hat{\mathbf{n}} - \mathbf{v}\varphi]_{\mathbf{x}(u_2,u_1)} \left\|\frac{\partial\mathbf{x}}{\partial\xi}\ \times\ \frac{\partial\mathbf{x}}{\partial\eta}\right\|_{(u_2,u_1)} \\ &+\ [\mathbf{D}\boldsymbol{\nabla}\varphi\cdot\hat{\mathbf{n}} - \mathbf{v}\varphi]_{\mathbf{x}(u_2,u_2)} \left\|\frac{\partial\mathbf{x}}{\partial\xi}\ \times\ \frac{\partial\mathbf{x}}{\partial\eta}\right\|_{(u_2,u_2)}.\end{aligned} \tag{5.8}$$

The source component of (2.10) is computed using three-dimensional Gaussian quadrature, which requires a total of eight function evaluations per sub-control volume. However this is mitigated by the fact that it is a once-off expense per sub-control volume, as the source is not considered dependent on φ. (If in fact the source was considered dependent on φ, then the method of Gaussian quadrature would still be applicable, but its expense would now be roughly equivalent to the expense for the advective-diffusive fluxes: eight points per sub-control volume versus four points per control volume face and multiple faces per sub-control volume).

The extension of (2.21) to three dimensions is given in [10]:

$$\int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} g(u,v,w)\,\mathrm{d}u\,\mathrm{d}v\mathrm{d}w \approx \sum_{j=1}^{p}\sum_{k=1}^{p}\sum_{l=1}^{p} w_j w_k w_l\, g(u_j,u_k,u_l) \tag{5.9}$$

with weights w_i and abscissas u_i still given by Table 2.1. Applying (5.9) to (2.20) requires mapping each sub-control volume to $[-1,1]\times[-1,1]\times[-1,1]$. This can be achieved with the following parameterisation involving the sub-

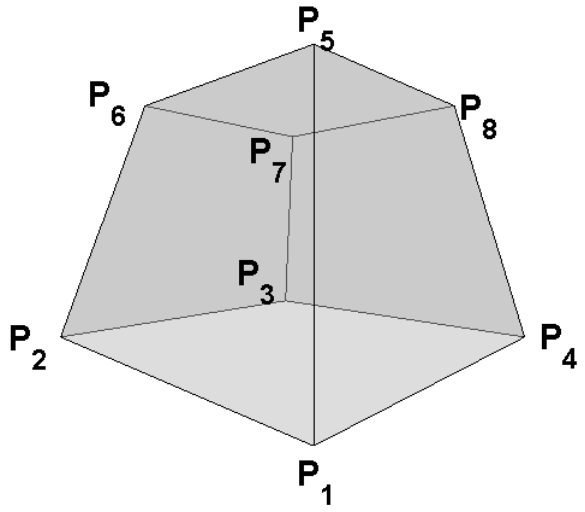


Figure 5.7: Sub-control volume vertex labels.

control volume vertices $\mathbf{P}_i$ (see Figure 5.7):

$$\begin{aligned}\mathbf{x}(\xi,\eta,\zeta) = &\frac{1}{8}(1-\xi)(1-\eta)(1-\zeta)\mathbf{P}_1 + \frac{1}{8}(1+\xi)(1-\eta)(1-\zeta)\mathbf{P}_2 \\ &+ \frac{1}{8}(1-\xi)(1+\eta)(1-\zeta)\mathbf{P}_3 + \frac{1}{8}(1+\xi)(1+\eta)(1-\zeta)\mathbf{P}_4 \\ &+ \frac{1}{8}(1-\xi)(1-\eta)(1+\zeta)\mathbf{P}_5 + \frac{1}{8}(1+\xi)(1-\eta)(1+\zeta)\mathbf{P}_6 \\ &+ \frac{1}{8}(1-\xi)(1+\eta)(1+\zeta)\mathbf{P}_7 + \frac{1}{8}(1+\xi)(1+\eta)(1+\zeta)\mathbf{P}_8, \\ &-1 \le \xi \le 1, \quad -1 \le \eta \le 1, \quad -1 \le \zeta \le 1\,,\end{aligned} \tag{5.10}$$

with the Jacobian of the transformation given by

$$\left|\frac{\partial \mathbf{x}}{\partial(\xi,\eta,\zeta)}\right| = \frac{\partial \mathbf{x}}{\partial \xi} \cdot \frac{\partial \mathbf{x}}{\partial \eta} \times \frac{\partial \mathbf{x}}{\partial \zeta}\,. \tag{5.11}$$

Applying (5.10) and (5.11) to (2.20) yields

$$\iiint_{V_i} S \,\mathrm{d}V = \int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} S(\mathbf{x}(\xi,\eta,\zeta)) \,\frac{\partial \mathbf{x}}{\partial \xi} \cdot \frac{\partial \mathbf{x}}{\partial \eta} \times \frac{\partial \mathbf{x}}{\partial \zeta} \;\mathrm{d}\xi \,\mathrm{d}\eta \,\mathrm{d}\xi \tag{5.12}$$

$$\approx \sum_{j=1}^{2}\sum_{k=1}^{2}\sum_{l=1}^{2} w_j w_k w_l \, S(\mathbf{x}(u_j,u_k,u_l)) \left[\frac{\partial \mathbf{x}}{\partial \xi} \cdot \frac{\partial \mathbf{x}}{\partial \eta} \times \frac{\partial \mathbf{x}}{\partial \zeta}\right]_{(u_j,u_k,u_l)}. \tag{5.13}$$

5.5 Interpolation

5.5.1 Shape functions

In three dimensions, the method of shape functions produces a linear interpolant $s_i(x,y,z)$ over each tetrahedral element. If the vertices of the element are (x_1,y_1,z_1), (x_2,y_2,z_2), (x_3,y_3,z_3) and (x_4,y_4,z_4), the shape function interpolant is given by

$$s_i(x,y,z) = \sum_{j=1}^{4} N_j(x,y,z)\varphi_j \tag{5.14}$$

where the shape functions $N_j(x,y,z)$ are given by:

$$\begin{aligned} N_1(x,y,z) = \frac{1}{\Delta}\Big\{ & x_2y_3z_4 - x_2z_3y_4 - y_2x_3z_4 + y_2z_3x_4 + x_3z_2y_4 - z_2y_3x_4 \\ & - (y_2z_3 - y_2z_4 - z_2y_3 + z_2y_4 + y_3z_4 - z_3y_4)x \\ & + (x_2z_3 - x_2z_4 - x_3z_2 + x_3z_4 + z_2x_4 - z_3x_4)y \\ & - (x_2y_3 - x_2y_4 - y_2x_3 + y_2x_4 + x_3y_4 - y_3x_4)z\Big\}, \end{aligned}$$

$$N_2(x,y,z) = \frac{1}{\Delta}\Big\{-x_1y_3z_4 + x_1y_4z_3 + x_3y_1z_4 - x_3y_4z_1 - x_4y_1z_3 + x_4y_3z_1$$
$$+ (y_1z_3 - y_1z_4 - y_3z_1 + y_3z_4 + y_4z_1 - y_4z_3)x$$
$$- (x_1z_3 - x_1z_4 - x_3z_1 + x_3z_4 + x_4z_1 - x_4z_3)y$$
$$+ (x_1y_3 - x_1y_4 - x_3y_1 + x_3y_4 + x_4y_1 - x_4y_3)z\Big\},$$

$$N_3(x,y,z) = \frac{1}{\Delta}\Big\{x_1y_2z_4 - x_1y_4z_2 - x_2y_1z_4 + x_2y_4z_1 + x_4y_1z_2 - x_4y_2z_1$$
$$- (y_1z_2 - y_1z_4 - y_2z_1 + y_2z_4 + y_4z_1 - y_4z_2)x$$
$$+ (x_1z_2 - x_1z_4 - x_2z_1 + x_2z_4 + x_4z_1 - x_4z_2)y$$
$$- (x_1y_2 - x_1y_4 - x_2y_1 + x_2y_4 + x_4y_1 - x_4y_2)z\Big\},$$

$$N_4(x,y,z) = \frac{1}{\Delta}\Big\{-x_1y_2z_3 + x_1y_3z_2 + x_2y_1z_3 - x_2y_3z_1 - x_3y_1z_2 + x_3y_2z_1$$
$$+ (y_1z_2 - y_1z_3 - y_2z_1 + y_2z_3 + y_3z_1 - y_3z_2)x$$
$$- (x_1z2 - x_1z_3 - x_2z_1 + x_2z_3 + x_3z_1 - x_3z_2)y$$
$$+ (x_1y_2 - x_1y_3 - x_2y_1 + x_2y_3 + x_3y_1 - x_3y_2)z\Big\},$$

and

$$\Delta = \begin{vmatrix} 1 & x_1 & y_1 & z_1 \\ 1 & x_2 & y_2 & z_2 \\ 1 & x_3 & y_3 & z_3 \\ 1 & x_4 & y_4 & z_4 \end{vmatrix} \tag{5.15}$$

is six times the tetrahedral element's volume.

With s_i given by (5.14), $\boldsymbol{\nabla} s_i$ is thus computed as

$$\boldsymbol{\nabla} s_i(x,y,z) = \sum_{j=1}^{4} \boldsymbol{\nabla} N_j(x,y,z)\varphi_j \tag{5.16}$$

which is constant throughout the element.

5.5.2 Radial basis functions

The method of radial basis functions in three dimensions takes a set of scattered points $\{(x_j, y_j, z_j), j = 1, \ldots, n\}$, along with the corresponding function values $\{\varphi_j, j = 1, \ldots, n\}$, and fits an interpolating function $s_i(x, y, z)$ given by

$$s_i(x,y,z) = \sum_{j=1}^{n} \lambda_j \phi(r_j) + c_0 + c_1 x + c_2 y + c_3 z \tag{5.17}$$

with the conditions

$$s_i(x_j, y_j, z_j) = \varphi_j, \quad j = 1, 2, \ldots, n \tag{5.18}$$

and

$$\sum_{j=1}^{n} \lambda_j = \sum_{j=1}^{n} \lambda_j x_j = \sum_{j=1}^{n} \lambda_j y_j = \sum_{j=1}^{n} \lambda_j z_j = 0 \tag{5.19}$$

where

$$r_j = \sqrt{(x - x_j)^2 + (y - y_j)^2 + (z - z_j)^2}\,. \tag{5.20}$$

The comments from Section 3.5.2 on the effect of the parameter c^2 on the multiquadric continue to apply in three dimensions: increasing c^2 flattens the multiquadric, and increases the condition number of the coefficient matrix $\boldsymbol{\Lambda}$ (2.36).

5.6 Finite volume discretisation

With the methods for integration and interpolation in place, the definition of the nonlinear component functions of the finite volume discretisation (2.12) in three dimensions can be completed. In the case of CV-FE, f_i is given by

$$f_i(\boldsymbol{\varphi}) = \sum_{\tau=1}^{Nf_i} A(\sigma_{i\tau}) \left[\mathbf{D}(s_i(\mathbf{m}_{i\tau}))\boldsymbol{\nabla} s_i(\mathbf{m}_{i\tau}) \cdot \hat{\mathbf{n}} - \mathbf{v}(s_i(\mathbf{m}_{i\tau}))s_i(\mathbf{m}_{i\tau})\right] + S(\mathbf{x}_i)\Delta V_i \tag{5.21}$$

where s_i is the shape function interpolant (5.14), $A(\sigma_{i\tau})$ is the area of control volume face $\sigma_{i\tau}$ computed using (5.1) and ΔV_i is the volume of control volume V_i computed by summing the volumes of the six sub-pyramids with volumes given by (5.5). In case of CV-RBF, f_i is given by

$$\begin{aligned} f_i(\boldsymbol{\varphi}) = &\sum_{\tau=1}^{Nf_i} \sum_{j=1}^{2} \sum_{k=1}^{2} \left\| \frac{\partial \mathbf{x}}{\partial \xi} \times \frac{\partial \mathbf{x}}{\partial \eta} \right\|_{(u_j,u_k)} \\ &\times \left[\mathbf{D}(s_i(\mathbf{x}(u_j,u_k)))\boldsymbol{\nabla} s_i(\mathbf{x}(u_j,u_k)) \cdot \hat{\mathbf{n}} - \mathbf{v}(s_i(\mathbf{x}(u_j,u_k)))s_i(\mathbf{x}(u_j,u_k))\right] \\ &+ \sum_{j=1}^{2} \sum_{k=1}^{2} \sum_{l=1}^{2} w_j w_k w_l \, S(\mathbf{x}(u_j,u_k,u_l)) \left[\frac{\partial \mathbf{x}}{\partial \xi} \cdot \frac{\partial \mathbf{x}}{\partial \eta} \times \frac{\partial \mathbf{x}}{\partial \zeta}\right]_{(u_j,u_k,u_l)} \end{aligned} \tag{5.22}$$

where s_i represents the RBF interpolant (5.17).

In the next chapter, the accuracy and the efficiency of these different finite volume discretisation strategies are compared, by using them to solve a number of test problems. The effectiveness of the Jacobian-free Newton-Krylov method and associated preconditioning techniques discussed in Chapter 2 is also put to the test, along with the adaptive switching strategy and preconditioner parallelisation.

Chapter 6

Numerical experiments in three dimensions

In this chapter the effectiveness of the three-dimensional schemes proposed in the previous chapter are investigated, by applying them to three different test problems. The first two sections of this chapter outline the test problems used, and the meshes used in solving them. The final two sections present the results of these tests, concentrating on the different aspects of accuracy and efficiency. Within these sections, the results are divided into subsections concerning specific aspects of the results, including time and memory requirements, the effect of parallelisation, and the effectiveness of the adaptive solution strategy. In each subsection, results from solving one or more of the test problems are presented that illustrate the outcomes of the tests.

In some cases the results mirror those in two dimensions, while in others there is more to be said in the three-dimensional setting. In particular, matters of efficiency are a prime concern in three dimensions, so these will be given much more attention than they were in Chapter 4.

6.1 Test problems

Once again, in order that solid conclusions about the effectiveness of these methods can be drawn, the test problems used in these numerical experiments have known analytic solutions. All are based on the steady-state advection-diffusion problem

$$\boldsymbol{\nabla} \cdot (\mathbf{D}\boldsymbol{\nabla}\varphi - \mathbf{v}\varphi) + S = 0 \tag{6.1}$$

on the domain $(0,1) \times (0,1) \times (0,1)$. For simplicity in discussing and visualising the solutions, the problem is interpreted as a heat transfer problem, as it was in two dimensions. Different choices for $\mathbf{D}$, $\mathbf{v}$ and S, along with different combinations of boundary conditions, give rise to the various test problems. The thermal diffusivity $\mathbf{D}$ is taken to have the form

$$\mathbf{D} = \begin{pmatrix} D_{xx} & 0 & 0 \\ 0 & D_{yy} & 0 \\ 0 & 0 & D_{zz} \end{pmatrix} \tag{6.2}$$

where D_{xx}, D_{yy} and D_{zz} are the thermal diffusivities in the x, y and z directions respectively, and are potentially dependent on φ.

6.1.1 Test Problem 1

The first test problem is the three-dimensional extension of Test Problem 1 in Section 4.1.1. It is the linear, steady state heat diffusion problem obtained by taking a constant $\mathbf{D}$ and setting $\mathbf{v} = \mathbf{0}$ along with $S = g_0$ (constant) in (6.1). The boundaries $x = 0$, $y = 0$ and $z = 0$ are insulated, while the boundaries $x = 1$, $y = 1$ and $z = 1$ are subject to Newtonian cooling with external temperature φ_∞ and heat transfer coefficient h. This leads to the

Table 6.1: Physical parameters for Test Problem 1.

Parameter	Description	Value
D_{xx}	Thermal diffusivity in x direction	$5\ m^2 s^{-1}$
D_{yy}	Thermal diffusivity in y direction	$\{5, 50, 5000\}\ m^2 s^{-1}$
D_{zz}	Thermal diffusivity in z direction	$\{5, 500, 5000\}\ m^2 s^{-1}$
g_0	Source	$10\ K m^{-2} s^{-1}$
h	Heat transfer coefficient	$2\ W m^{-2} K^{-1}$
φ_∞	External temperature	$20\ K$

following boundary value problem:

$$\begin{aligned}
D_{xx}\frac{\partial^2\varphi}{\partial x^2} + D_{yy}\frac{\partial^2\varphi}{\partial y^2} + D_{zz}\frac{\partial^2\varphi}{\partial z^2} + g_0 &= 0 && \text{on } (0,1)\times(0,1)\times(0,1)\\
\frac{\partial\varphi}{\partial x} &= 0 && \text{at } x = 0\\
D_{xx}\frac{\partial\varphi}{\partial x} &= h(\varphi_\infty - \varphi) && \text{at } x = 1\\
\frac{\partial\varphi}{\partial y} &= 0 && \text{at } y = 0\\
D_{yy}\frac{\partial\varphi}{\partial y} &= h(\varphi_\infty - \varphi) && \text{at } y = 1\\
\frac{\partial\varphi}{\partial z} &= 0 && \text{at } z = 0\\
D_{zz}\frac{\partial\varphi}{\partial z} &= h(\varphi_\infty - \varphi) && \text{at } z = 1
\end{aligned} \tag{6.3}$$

The parameter values used for this problem are listed in Table 6.1. For some tests, the parameters D_{yy} and D_{zz} were varied to provide increasing challenges to the numerical solution methods.

Using the method of eigenfunction expansion, an analytic solution to

problem (6.3) is possible [37]. The spectral representations in this case are

$$\delta(x-\xi) = \sum_{n=1}^{\infty} \frac{1}{N_x(\mu_n,x)} X(\mu_n,x)X(\mu_n,\xi) \tag{6.4}$$

$$\delta(y-\eta) = \sum_{m=1}^{\infty} \frac{1}{N_y(\lambda_m,y)} Y(\lambda_m,y)Y(\lambda_m,\eta) \tag{6.5}$$

$$\delta(z-\zeta) = \sum_{k=1}^{\infty} \frac{1}{N_z(\gamma_k,z)} Z(\gamma_k,z)Z(\gamma_k,\zeta) \tag{6.6}$$

where

$$X(\mu_n,x) = \mu_n \cos(\mu_n x)\,, \tag{6.7}$$

$$N_x(\mu) = \frac{\mu^2}{2}\left(1+\frac{hD_{xx}}{h+\mu {D_{xx}}^2}\right) \tag{6.8}$$

and the μ_n are the positive roots of

$$\tan(\mu_n) = \frac{h}{\mu_n D_{xx}} \tag{6.9}$$

with similar expressions for Y, Z, N_y, N_z, λ_m and γ_k. Thus the solution is given by

$$\varphi(x,y,z) = \sum_{n=1}^{\infty}\sum_{m=1}^{\infty}\sum_{k=1}^{\infty} \frac{\alpha(n,m,k)X(\mu_n,x)Y(\lambda_m,y)Z(\gamma_m,z)}{(D_{xx}\mu_n^2 + D_{yy}\lambda_m^2 + D_{zz}\gamma_k^2)N_x(\mu_n)N_y(\lambda_m)N_z(\gamma_k)}\,,$$
$$0\le x\le 1, \quad 0\le y\le 1, \quad 0\le z\le 1 \tag{6.10}$$

where

$$\alpha(n,m,k) = \int_0^1\int_0^1\int_0^1 X(\mu_n,\xi)Y(\lambda_m,\eta)Z(\gamma_k,\zeta)g_0\ \mathrm{d}\zeta\mathrm{d}\eta\ \mathrm{d}\xi\,.$$

Visualisations of this solution for the combinations of D_{xx}, D_{yy} and D_{zz}

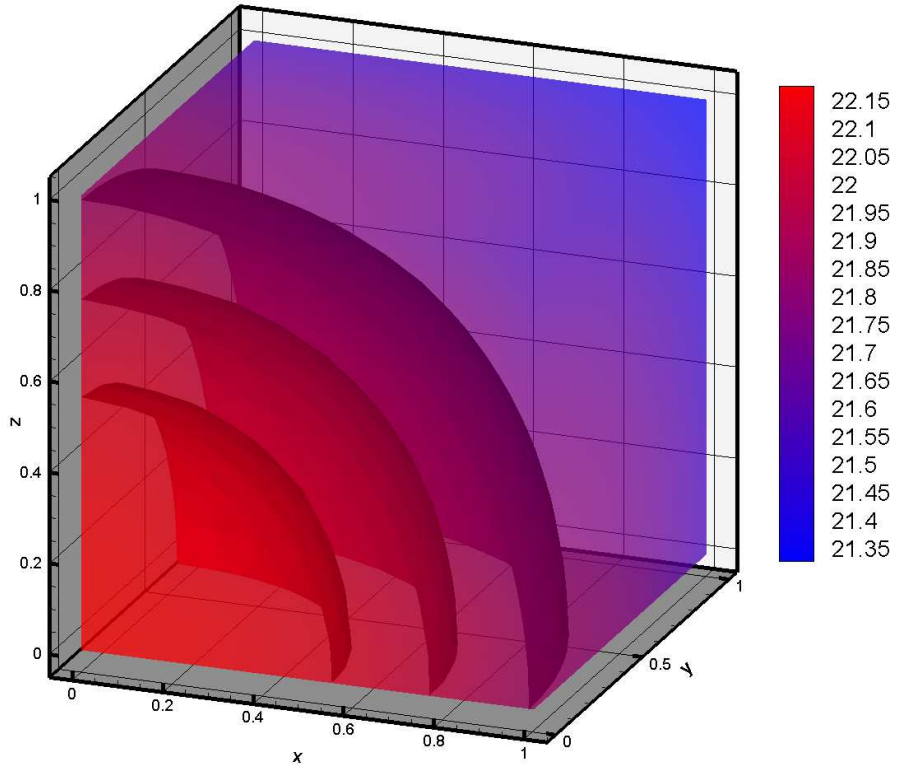


Figure 6.1: Analytic solution of Test Problem 1 with $\mathbf{D} = \text{diag}(5, 5, 5)$.

given in Table 6.1 are shown in Figures 6.1 to 6.6. Each solution is illustrated in two different ways. Figures 6.1, 6.3 and 6.5 are three-dimensional volume visualisations, with several isosurfaces (surfaces of constant temperature) drawn to help illustrate the general behaviour of the solution. Figures 6.2, 6.4 and 6.6 are contour slices parallel to the xy plane that give a "top down" view of the solution.

Figures 6.1 and 6.2 illustrate the solution when $\mathbf{D} = \text{diag}(5, 5, 5)$. With the inner boundaries insulated, the temperature is maximum at the origin.

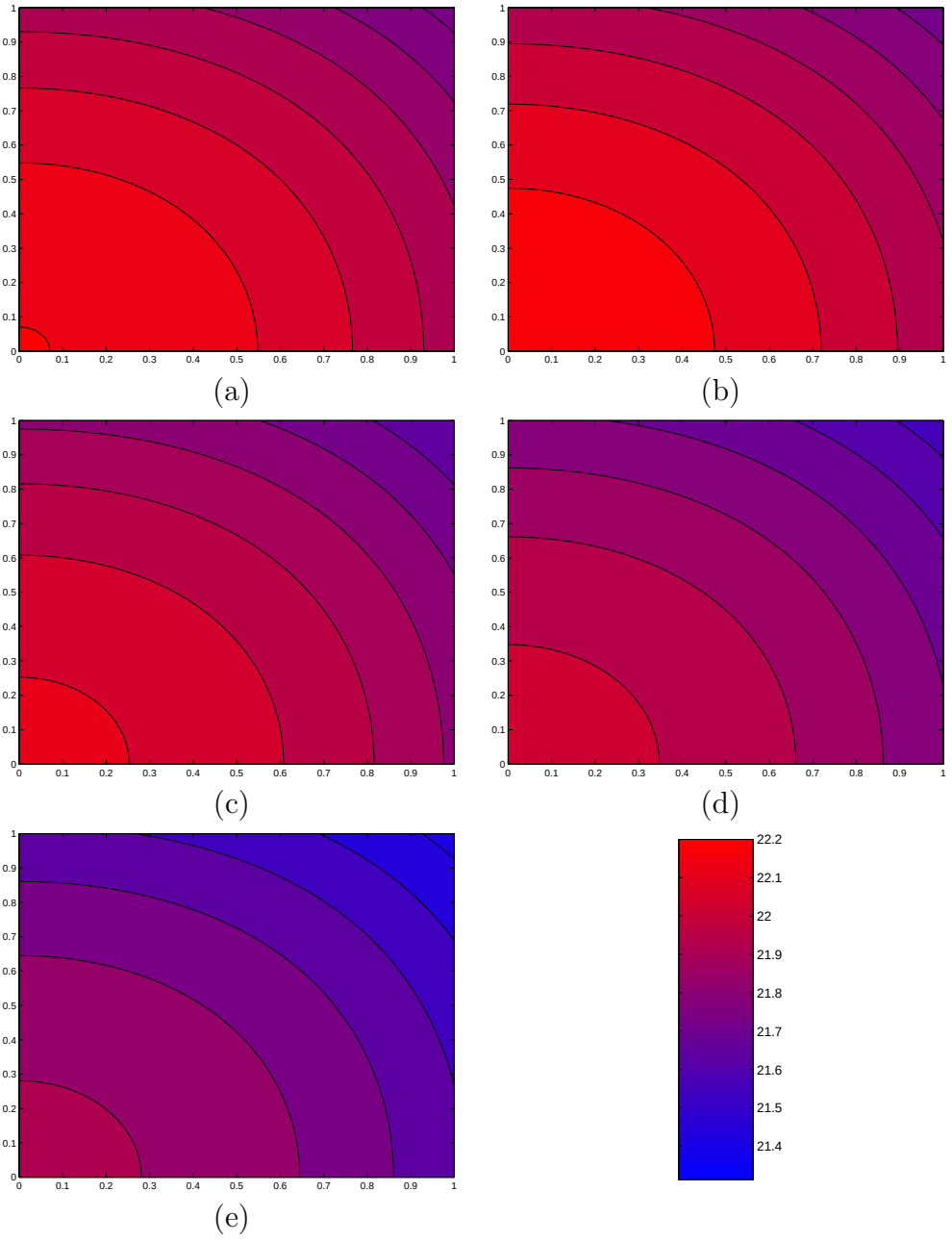


Figure 6.2: Analytic solution of Test Problem 1 with $\mathbf{D} = \text{diag}(5, 5, 5)$. Contours at (a) $z = 0.1$; (b) $z = 0.3$; (c) $z = 0.5$; (d) $z = 0.7$; (e) $z = 0.9$.

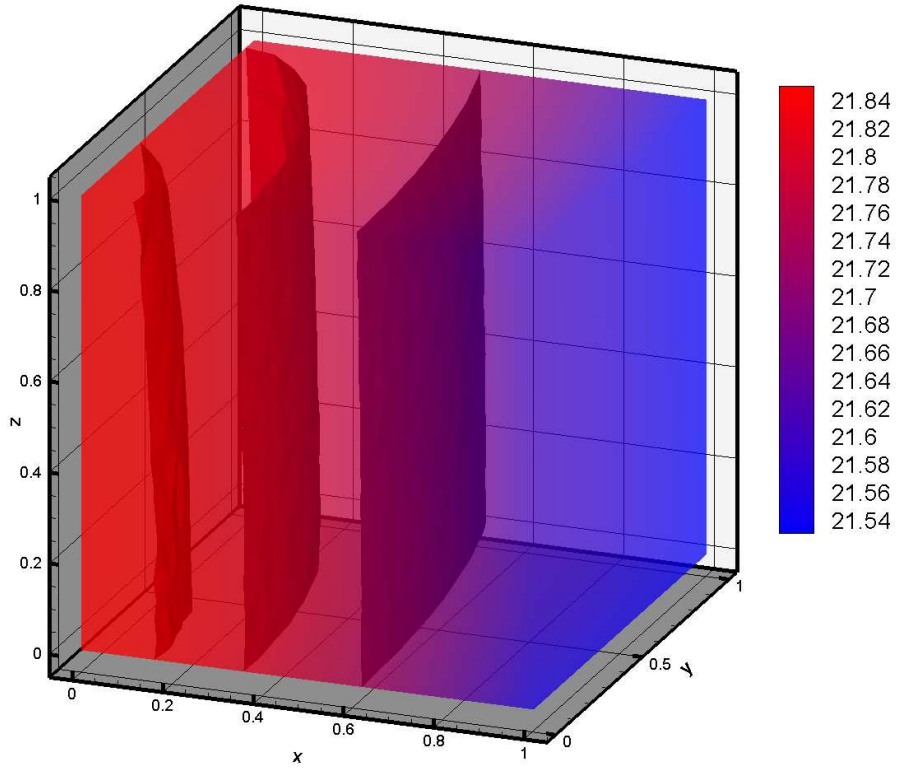


Figure 6.3: Analytic solution of Test Problem 1 with $\mathbf{D} = \text{diag}(5, 50, 500)$.

The equal diffusivities mean that the heat diffuses equally in all directions, giving the symmetric profile observed in Figure 6.1. The isosurfaces are therefore spherical in shape, centred at the origin. The heat conduction on the outer boundaries means that the temperature decreases away from the origin, and is minimal at the point $(1, 1, 1)$.

The symmetry in the solution is lost when D_{yy} and D_{zz} are increased to 50 and 500 respectively. Now, as Figure 6.3 shows, the problem is becoming more reminiscent of a two-dimensional problem. The much greater diffusion

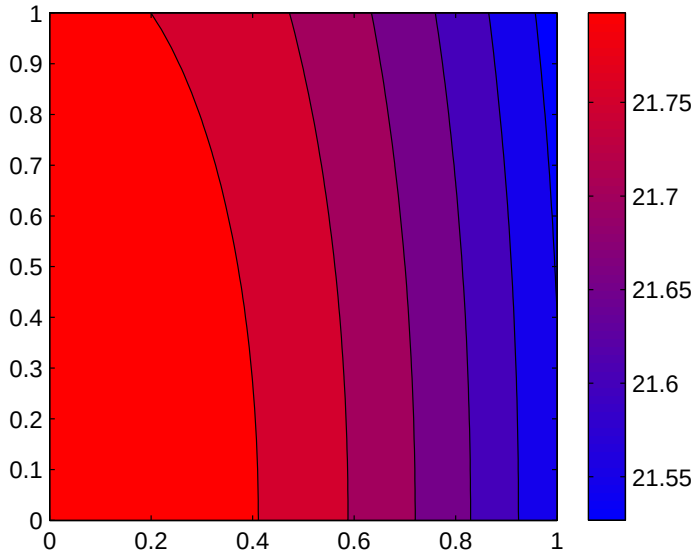


Figure 6.4: Analytic solution of Test Problem 1 with $\mathbf{D} = \text{diag}(5, 50, 500)$. Contour at $z = 0.5$.

in the z direction means that there is no significant variation of temperature in this direction. Indeed, only one contour plot (at $z = 0.5$) is necessary in Figure 6.4 to show the solution from top down, as contours at other values of z are virtually identical. There is still some curvature in the isosurfaces, with the diffusion in the y direction not completely dominating that in the x direction.

The third example sees both D_{yy} and D_{zz} increased to the value 5000. The corresponding solution is depicted in Figures 6.5 and 6.6. With this combination of parameters, the diffusion in the y and z directions are so dominant that there is virtually no variation in temperature in either direction. The solution now behaves almost as if it were a one-dimensional problem, with only the variation of temperature in the x direction significant. A single contour plot at $z = 0.5$ is again sufficient to characterise the

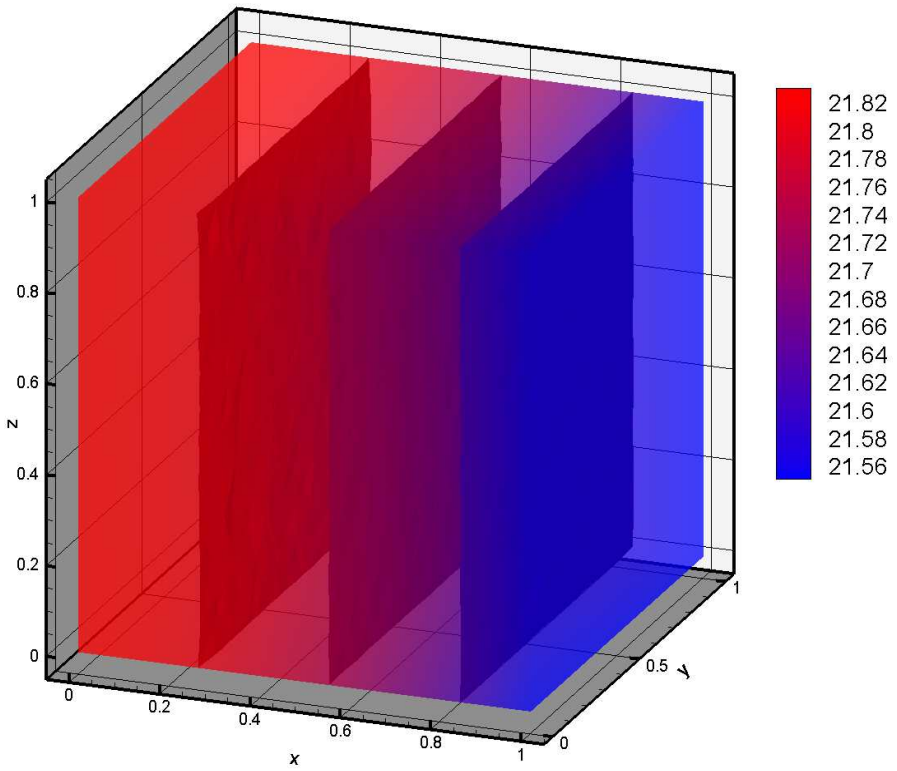


Figure 6.5: Analytic solution of Test Problem 1 with $\mathbf{D} = \text{diag}(5, 5000, 5000)$.

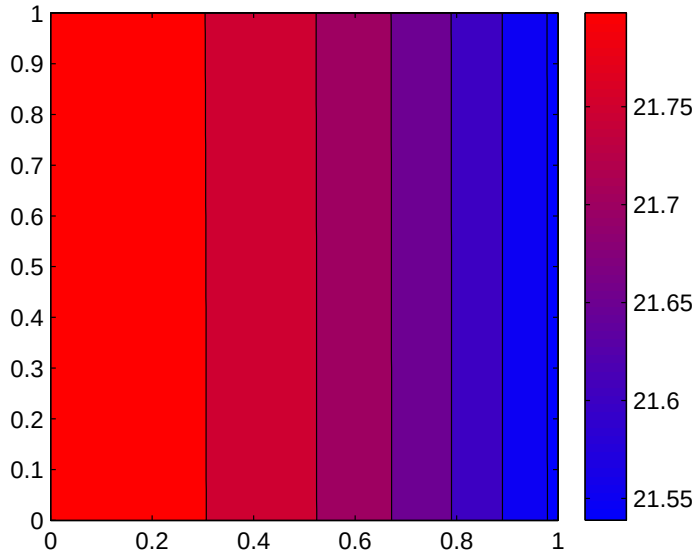


Figure 6.6: Analytic solution of Test Problem 1 with $\mathbf{D} = \text{diag}(5, 5000, 5000)$. Contour at $z = 0.5$.

solution's behaviour.

6.1.2 Test Problem 2

The second test problem uses nonlinear diffusivity $D_{xx} = D_{yy} = D_{zz} = \varphi^{1.3}$, along with $\mathbf{v} = \mathbf{0}$ in (6.1). The source term is constructed by substituting the imposed solution

$$\begin{aligned} \varphi(x, y, z) &= 10xyz(1-x)(1-y)(1-z)e^{-(x^2+y^2+z^2)}, \\ &\qquad 0 \le x \le 1, \quad 0 \le y \le 1, \quad 0 \le z \le 1 \end{aligned} \tag{6.11}$$

into (6.1). On all boundaries, the Dirichlet condition $\varphi = 0$ is prescribed.

The physical nature of this solution is that of a hot interior, with the maximum temperature at around $(0.4, 0.4, 0.4)$, cooling further away from

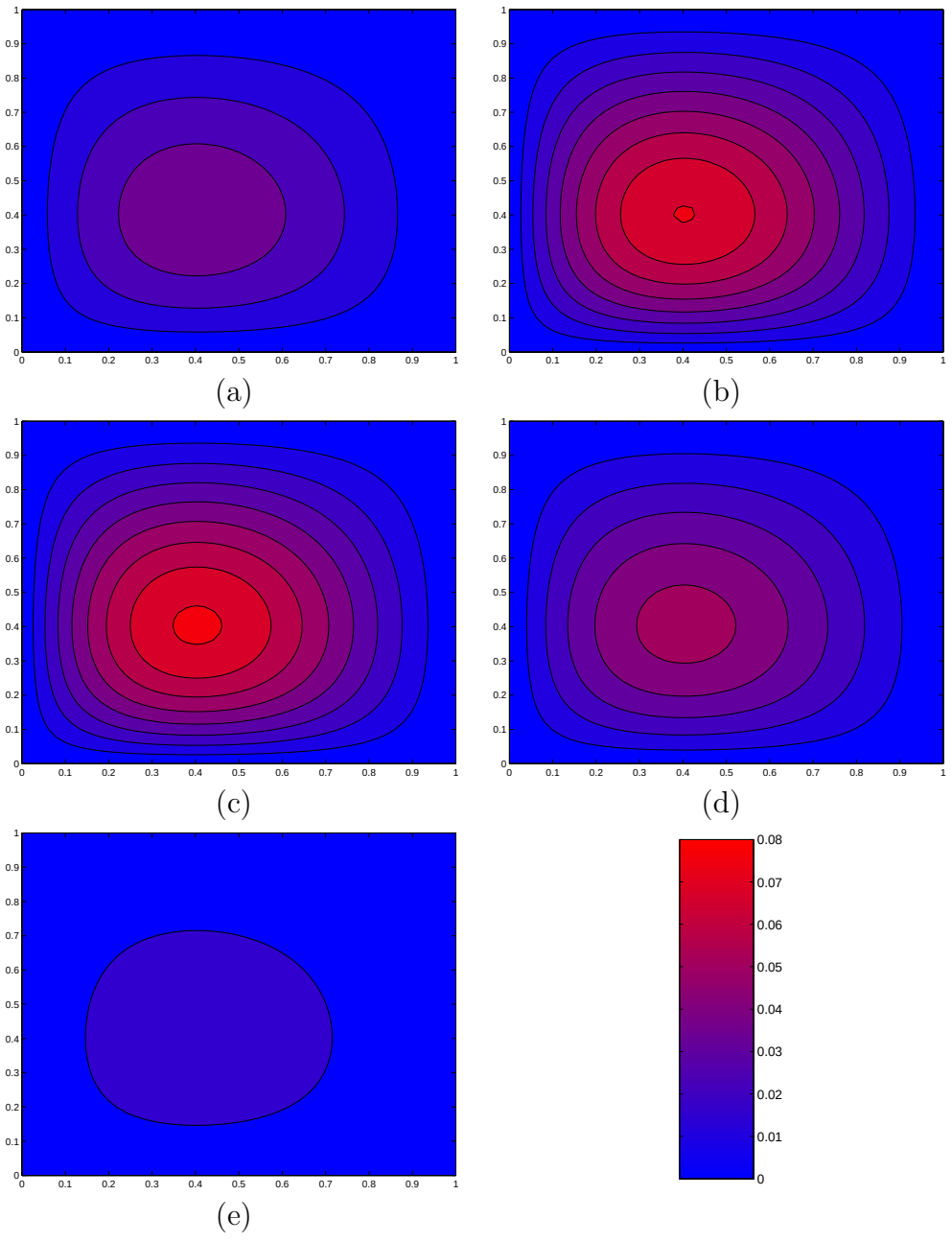


Figure 6.7: Analytic solution of Test Problem 2. Contours at (a) $z = 0.1$; (b) $z = 0.3$; (c) $z = 0.5$; (d) $z = 0.7$; (e) $z = 0.9$.

this point and zero on the boundaries. Isosurfaces emanating from the hottest point are ellipsoid-like at first, becoming squarish as they move closer to the boundary. This behaviour is evident in Figure 6.7 which depicts the solution (6.11) as a set of five contours in the xy plane.

6.1.3 Test Problem 3

Test Problem 3 is a nonlinear advection-diffusion problem also based on (6.11), this time with $\mathbf{D} = \mathbf{I}$ and $\mathbf{v} = 10(\varphi, \varphi, \varphi)^T$. The source is once again constructed so that the function

$$\begin{aligned} \varphi(x,y,z) &= 10xyz(1-x)(1-y)(1-z)e^{-(x^2+y^2+z^2)}, \\ & \qquad 0 \le x \le 1, \quad 0 \le y \le 1, \quad 0 \le z \le 1 \end{aligned} \tag{6.12}$$

depicted in Figure 6.7 is the analytic solution. The Dirichlet condition $\varphi = 0$ is again prescribed on all boundaries.

6.2 Meshes

Six examples of the unstructured, tetrahedral meshes used for these test problems are shown in Figure 6.8. They were generated using the mesh generator *Gmsh* [31] over a unit cube, with a uniform edge length requested throughout the mesh. Visualisations were produced using *DistMesh* [69]. As always when using an unstructured mesh generator, the resultant meshes have some variability of edge lengths. Table 6.2 lists the relevant properties of these six meshes, including minimum, maximum and average edge lengths.

Comparing Table 6.2 to Table 4.3 shows one reason why matters of efficiency are so much more of a concern in three dimensions than they are in

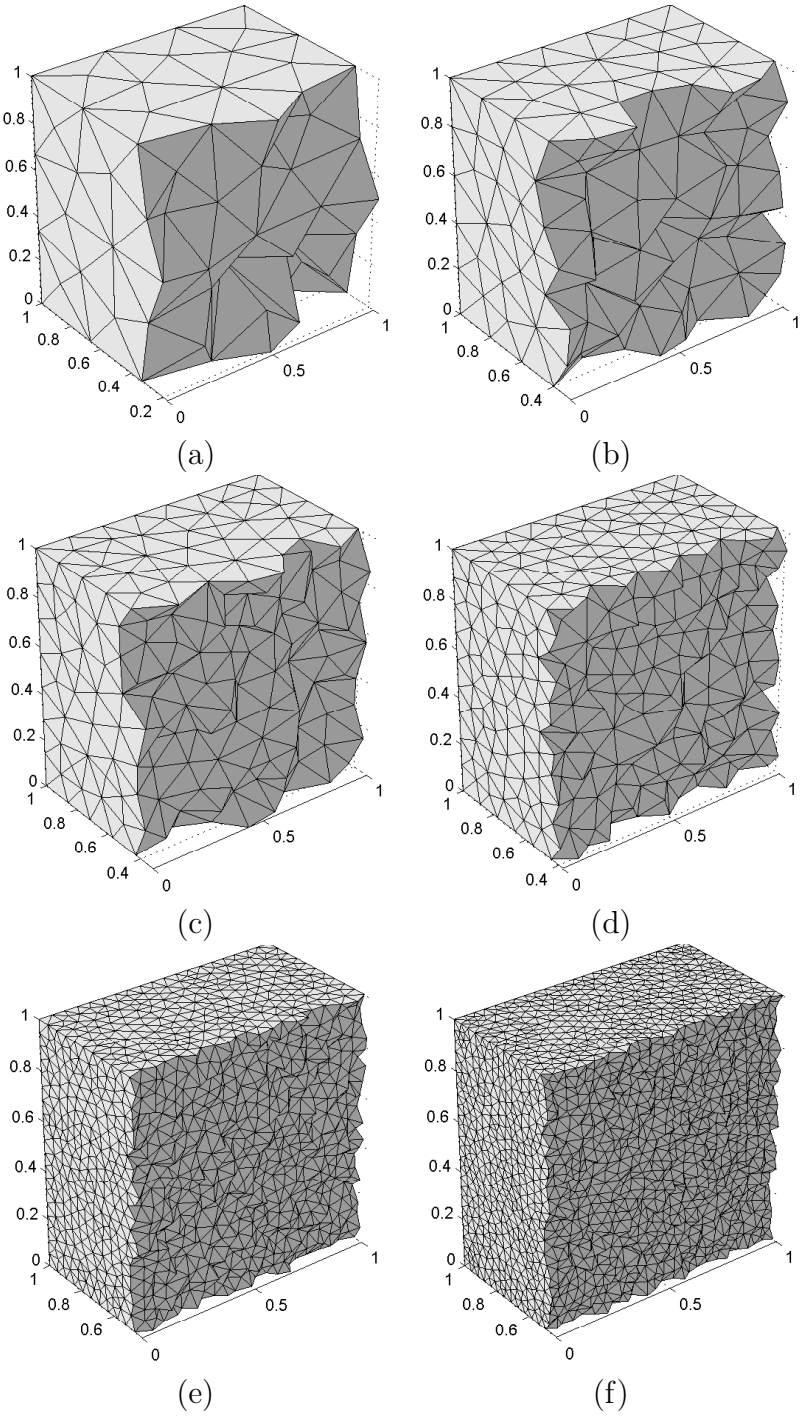


Figure 6.8: Unstructured meshes with average edge length (a) 0.246; (b) 0.186; (c) 0.138; (d) 0.111; (e) 0.056; (f) 0.045.

Table 6.2: Properties of meshes illustrated in Figure 6.8.

Mesh	Nodes	Elements	Edge lengths		
			Minimum	Maximum	Average
(a)	178	647	0.090	0.410	0.246
(b)	387	1542	0.098	0.311	0.186
(c)	825	3654	0.063	0.226	0.138
(d)	1567	7081	0.048	0.208	0.111
(e)	10385	53889	0.023	0.101	0.056
(f)	19367	103008	0.018	0.088	0.045

two. In two dimensions the unstructured triangular mesh with average edge length 0.054 comprised just 450 nodes. In three dimensions, for around the same average edge length, the number of nodes required is 10385. In two dimensions for example, it would be feasible to have stored the full Jacobian matrix for this problem: a 450×450 double precision floating point matrix requires just over 1.5MB of storage. For the corresponding problem in three dimensions, the memory required to store a full Jacobian matrix is more than 800MB. This gives some idea of the kinds of space and efficiency issues that must be dealt with when working in three dimensions.

6.3 Results concerning accuracy

In this section, the accuracy of the new CV-RBF method is compared to that of the CV-FE method. Results from solving the three test problems of Section 6.1 are shown, and the results compared to the analytic solutions, both graphically and by analysing computed solution errors.

6.3.1 Results for a single mesh

For the results in this section, the unstructured mesh comprising 1567 nodes and 7081 elements, illustrated in Figure 6.8(d) was used to solve each of the

Table 6.3: RBF parameters for solving Test Problems 1, 2 and 3.

Parameter	Description	Value
ϕ	Radial basis function	Multiquadric
c^2	Multiquadric parameter	1.0
n	Number of nodes per interpolation	20

three test problems. In computing the solution errors, the following formula was used:

$$\begin{aligned} \text{error} &= \frac{\|\boldsymbol{\varphi}^{(e)} - \boldsymbol{\varphi}^{(a)}\|_2}{\|\boldsymbol{\varphi}^{(e)}\|_2} \\ &= \frac{\sqrt{\sum_{i=1}^{N}(\varphi_i^{(e)} - \varphi_i^{(a)})^2}}{\sqrt{\sum_{i=1}^{N}(\varphi_i^{(e)})^2}} \end{aligned} \tag{6.13}$$

where superscript (e) symbolises the (exact) analytic solution and superscript (a) the (approximate) numerical solution.

For the CV-RBF method the RBF parameters used are listed in Table 6.3. The reason for selecting the multiquadric as the basis function has been discussed in Section 2.3.3. With the knowledge gained from the two-dimensional experimentation, but also mindful of the increased problem sizes in three dimensions, the value of $n = 20$ nodes per interpolation is used.

An appropriate value for the parameter c^2 was again obtained through preliminary numerical investigation. Recall from Section 4.4.3 that in two dimensions, the effect of increasing this parameter was primarily to increase the number of function evaluations required for convergence. In Section 6.4.2 the results of the same investigation conducted in three dimensions are presented. They show that $c^2 = 1$ produces acceptable accuracy without penalising the efficiency of the method. Therefore the value $c^2 = 1$ is used throughout these numerical experiments in three dimensions.

Table 6.4: Errors using CV-FE and using CV-RBF for Test Problem 1.

D	diag(5, 5, 5)	diag(5, 50, 500)	diag(5, 5000, 5000)
CV-FE	4.40E-05	1.34E-04	9.33E-04
CV-RBF	1.30E-06	2.32E-06	5.81E-05

Test Problem 1

The results of solving Test Problem 1 for the values of D_{xx}, D_{yy} and D_{zz} given in Table 6.1 are shown in Table 6.4. The results show that in each case the CV-RBF method offers an improvement in accuracy of between one and two orders of magnitude over the CV-FE method. Similar to the observations made in two dimensions, as the parameters D_{yy} and D_{zz} are increased, the accuracy of the numerical solution decreases, irrespective of which method is used to compute it. This is consistent with the notion that these problems were of increasing numerical difficulty.

Figures 6.9 and 6.10 show the temperature contours for solving Test Problem 1 with $\mathbf{D} = \text{diag}(5, 5, 5)$ using CV-FE and CV-RBF respectively. From Figure 6.9 it is evident that the CV-FE solution matches the analytic solution for the most part. The one major discrepancy is in Figure 6.9(a), where the numerical solution contour nearest the origin (where the temperature is greatest) is slightly out of place.

Figure 6.10 shows that the CV-RBF solution for this problem has no contours out of place, and is visually indistinguishable from the analytic solution. This is consistent with the results of Table 6.4 that show that the CV-RBF solution is more than an order of magnitude more accurate than the CV-FE solution for this problem.

Altering $\mathbf{D}$ to $\text{diag}(5, 50, 500)$ produces the solutions shown in Figures 6.11 and 6.12. Now that the diffusion is much stronger in the z direction, the analytic solution appears identical when sliced parallel to the xy plane (recall

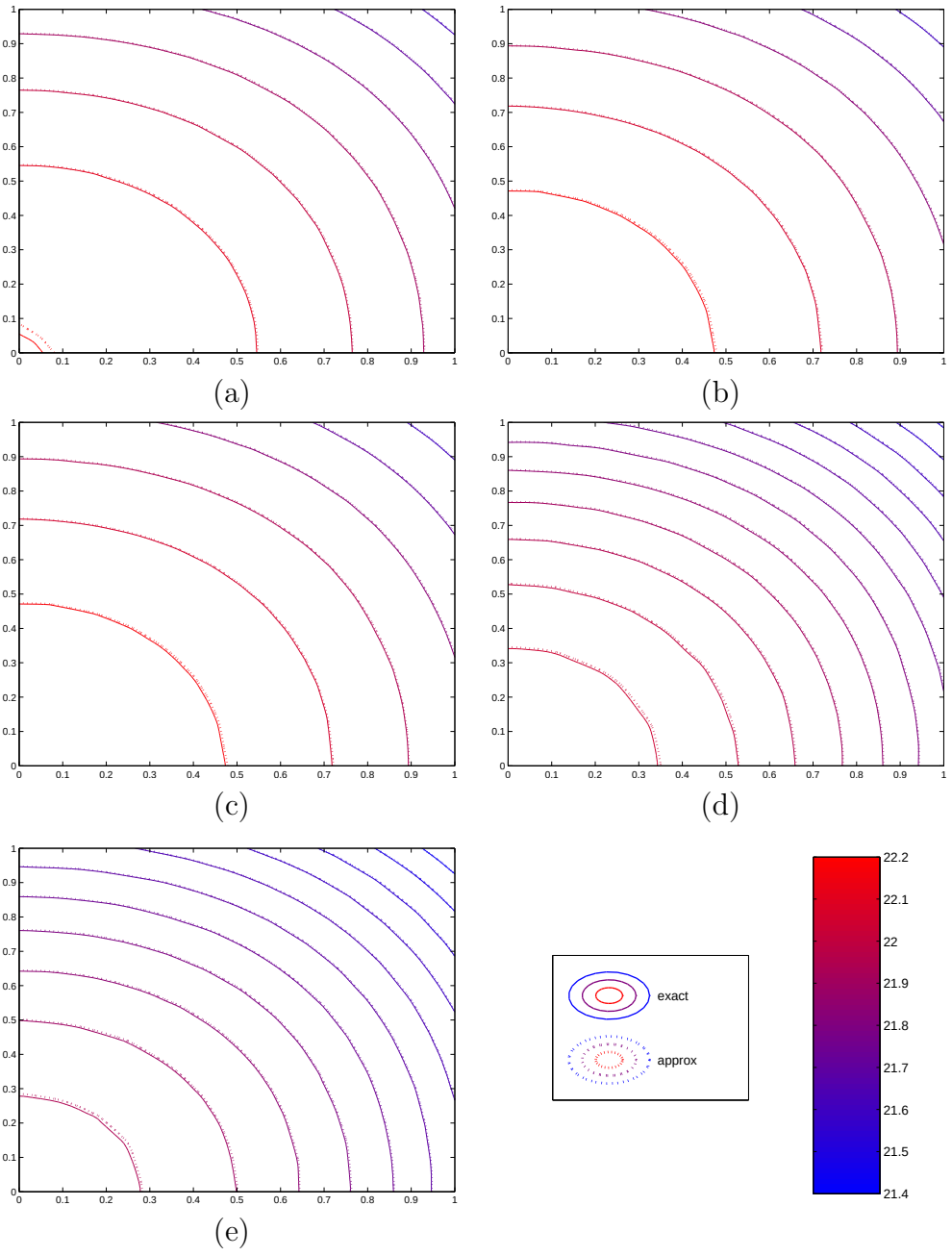


Figure 6.9: Temperature contours for Test Problem 1 with $\mathbf{D} = \text{diag}(5, 5, 5)$ using CV-FE. Contours at (a) $z = 0.1$; (b) $z = 0.3$; (c) $z = 0.5$; (d) $z = 0.7$; (e) $z = 0.9$. Colour bar shows temperature.

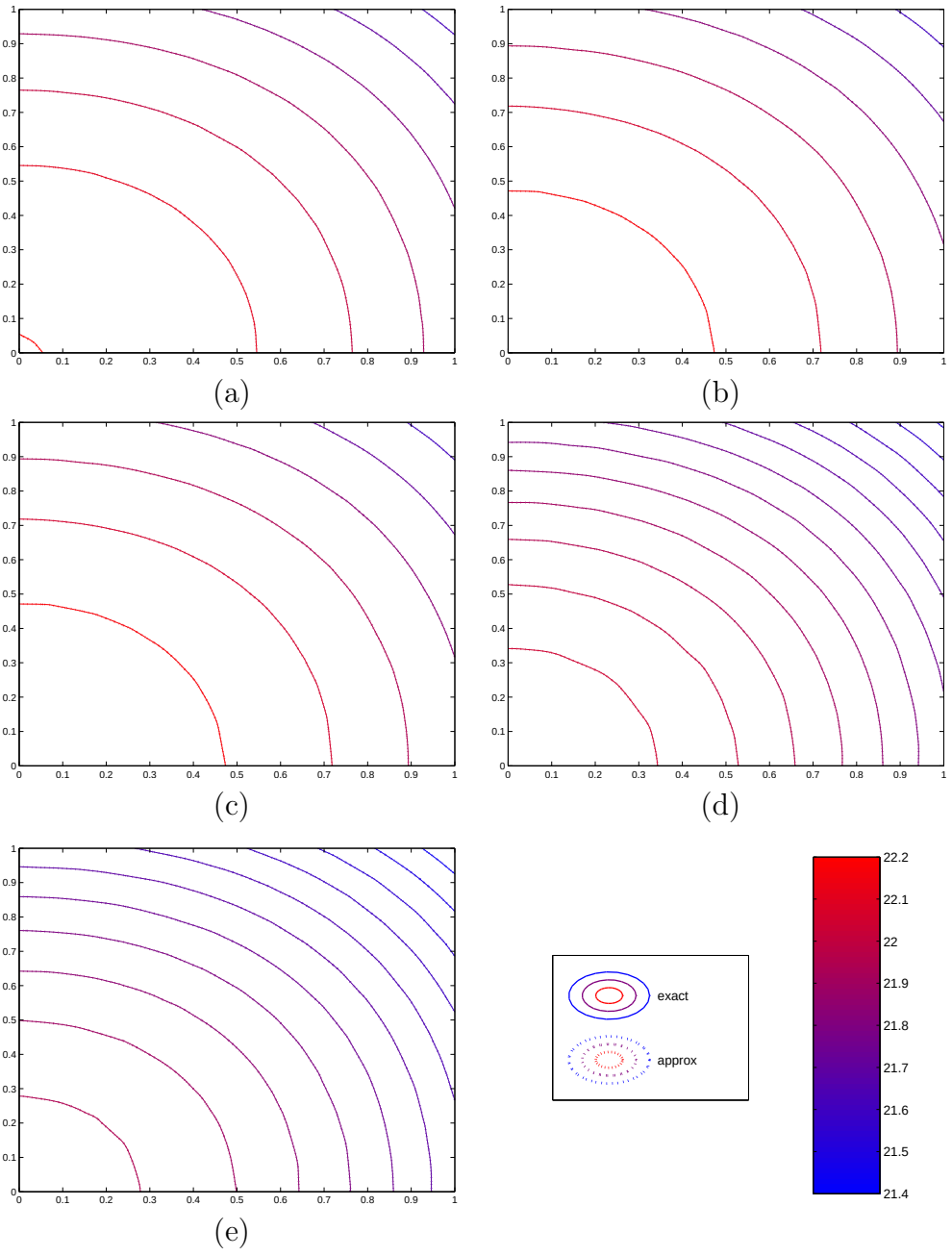


Figure 6.10: Temperature contours for Test Problem 1 with $\mathbf{D} = \text{diag}(5, 5, 5)$ using CV-RBF. Contours at (a) $z = 0.1$; (b) $z = 0.3$; (c) $z = 0.5$; (d) $z = 0.7$; (e) $z = 0.9$. Colour bar shows temperature.

Figure 6.4). Figure 6.11 shows that the CV-FE solution does not behave this way. There is variation in the five numerical solutions (a)-(e). In particular, Figures 6.11(b), (c) and (d) exhibit a spurious contour near the origin. In addition, the remaining contours are visibly out of position, although their shape is at least representative of the correct physical behaviour.

Figure 6.12 shows that the CV-RBF solution does not exhibit any of these problems. There are no contours out of position, there are no spurious contours, and all solutions (a)-(e) appear identical.

Setting the parameters D_{yy} and D_{zz} to 5000 results in the CV-FE solution depicted in Figure 6.13. For this problem there is no noticeable variation in z for either the exact or the numerical solution, so a single contour is sufficient to illustrate the behaviour. Figure 6.13 shows that the CV-FE solution has failed to reproduce the correct temperature variations in the x direction. This failure is explained by the large anisotropic ratio present in this problem: the diffusivities in the y and z directions are one thousand times greater than the diffusivity in the x direction. With the y and z diffusivities so dominant, the method has failed to adequately capture the much subtler diffusion in the x direction.

In Figure 6.14 the CV-RBF solution, although not indistinguishable from the analytic solution, is still much improved over the CV-FE solution. The more accurate discretisation has, by and large, captured the correct temperature progression in the x direction. The approximate solution contours are now just slightly offset from their correct positions, and there are no spurious or missing contours.

These results can be summarised by noting that the CV-RBF method produced a solution of superior accuracy to the CV-FE method in each case. Particularly for the cases where there was a high anisotropic ratio, the CV-

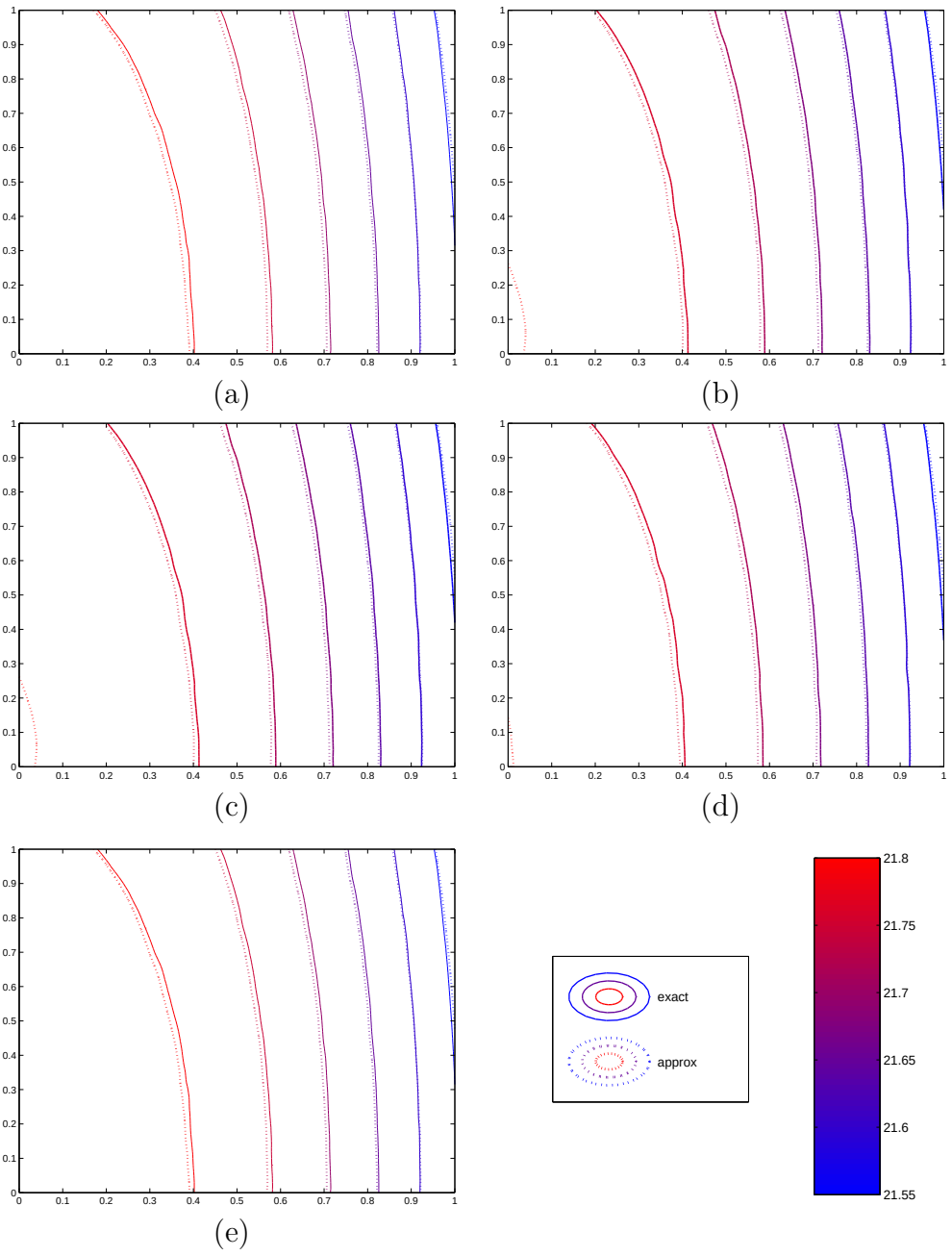


Figure 6.11: Temperature contours for Test Problem 1 with $\mathbf{D} = \text{diag}(5, 50, 500)$ using CV-FE. Contours at (a) $z = 0.1$; (b) $z = 0.3$; (c) $z = 0.5$; (d) $z = 0.7$; (e) $z = 0.9$. Colour bar shows temperature.

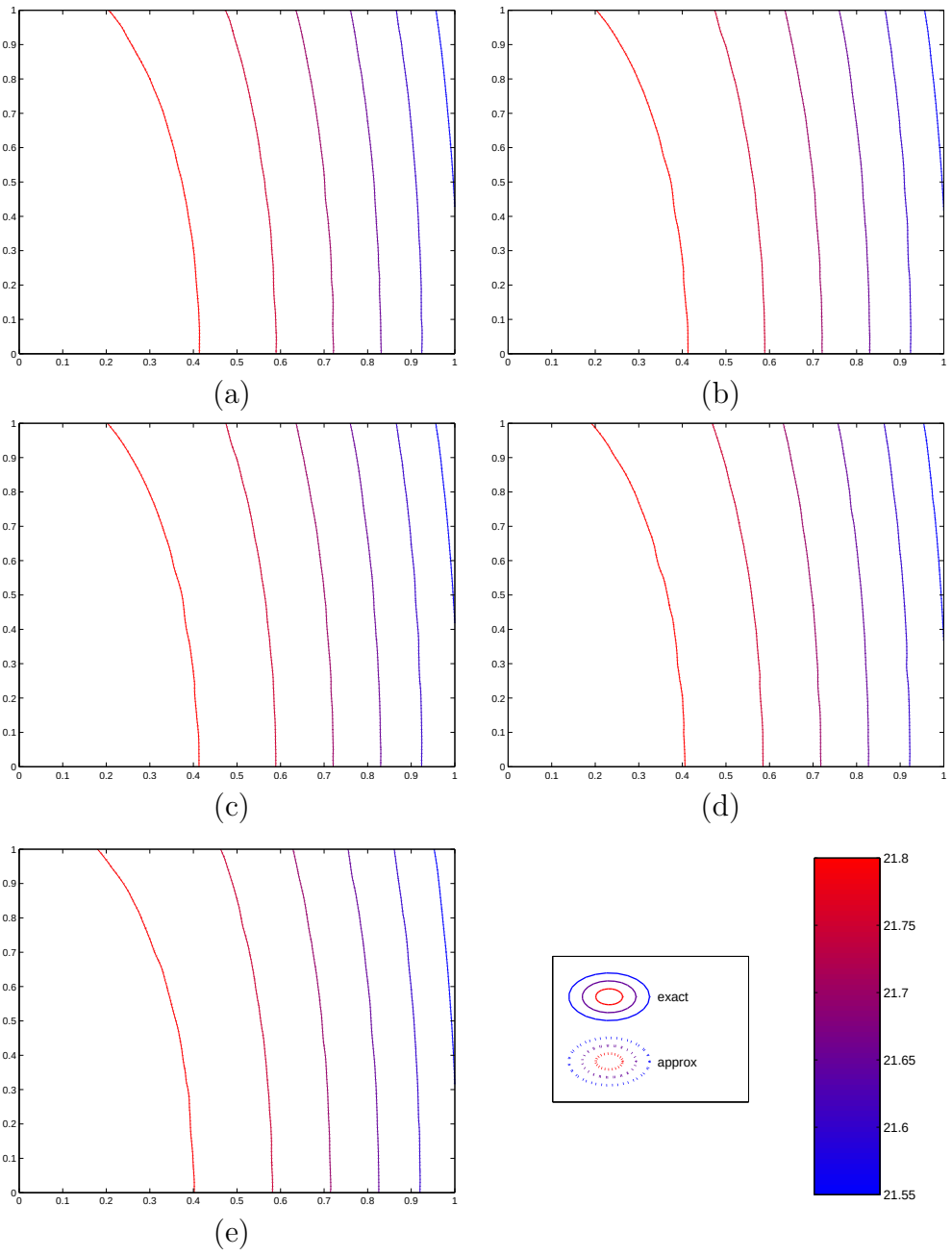


Figure 6.12: Temperature contours for Test Problem 1 with $\mathbf{D} = \text{diag}(5, 50, 500)$ using CV-RBF. Contours at (a) $z = 0.1$; (b) $z = 0.3$; (c) $z = 0.5$; (d) $z = 0.7$; (e) $z = 0.9$. Colour bar shows temperature.

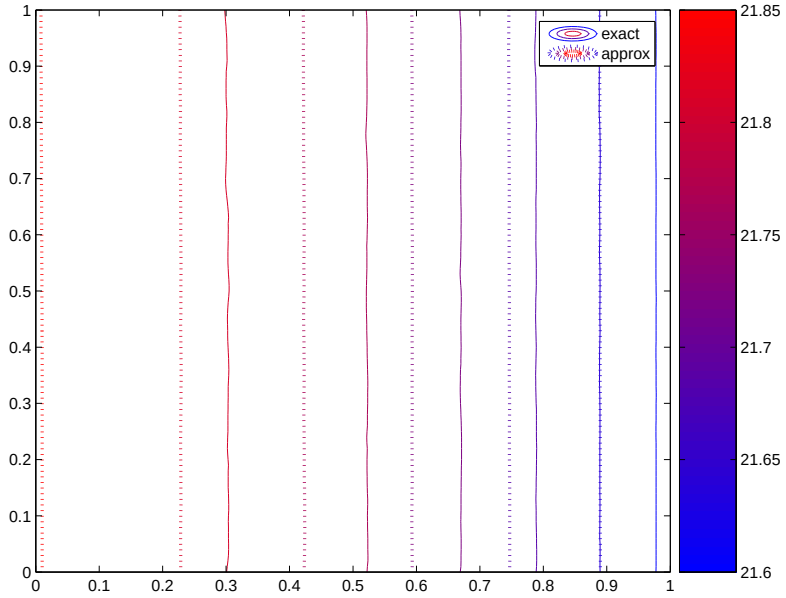


Figure 6.13: Solution contour for Test Problem 1 with $\mathbf{D} = \text{diag}(5, 5000, 5000)$ using CV-FE. Contour at $z = 0.5$.

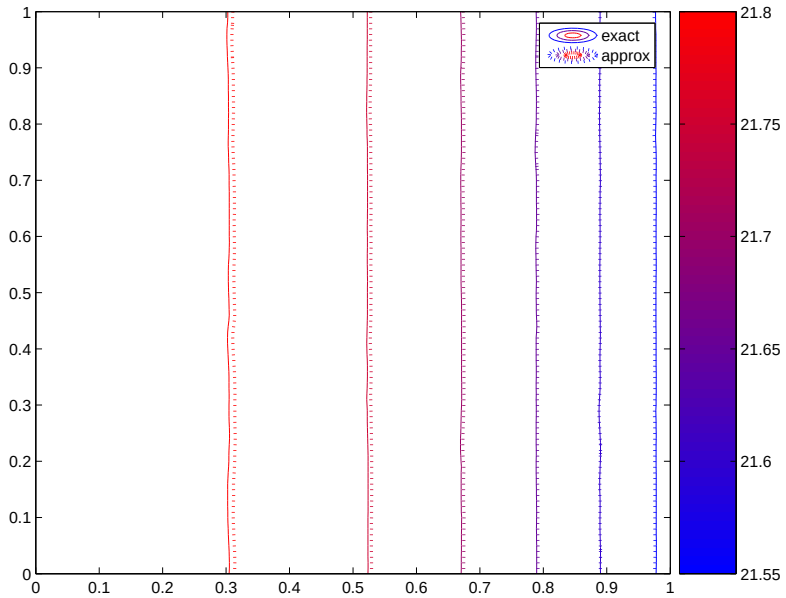


Figure 6.14: Solution contour for Test Problem 1 with $\mathbf{D} = \text{diag}(5, 5000, 5000)$ using CV-RBF. Contour at $z = 0.5$.

Table 6.5: Errors using CV-FE and CV-RBF for Test Problems 2 and 3.

	Test Problem 2	Test Problem 3
CV-FE	2.90E-02	1.54E-02
CV-RBF	1.53E-03	7.80E-04

RBF method was more successful in capturing this behaviour.

Test Problems 2 and 3

The results of solving Test Problems 2 and 3 (both of which have the same solution) are given in Table 6.5. Once again, the CV-RBF method has given rise to an improvement in the accuracy of the solution, compared to CV-FE. For both the diffusion and the advection problem, the improvement in accuracy is between one and two orders of magnitude.

In Figure 6.15 the temperature contours obtained when solving Test Problem 2 using CV-FE are shown. From these contour plots, it can be seen that the main weakness of this numerical solution is its failure to adequately capture the correct behaviour in the interior, where the temperature is highest and the diffusion is strongest. This is best seen in Figure 6.15(c) where the numerical solution has its innermost contour well outside the correct location indicated by the analytic solution. Away from this point, the numerical solution contours are more representative of the analytic solution behaviour, although they are still far from a perfect visual match.

In Figure 6.16, the corresponding solution using CV-RBF is almost a perfect visual match for the analytic solution. The only exception is the innermost contour of the slice shown in Figure 6.16(c), where the approximate solution contour is positioned just slightly further out than it should be. Overall this solution is a much better representation of the correct solution behaviour, particularly in the interior. Once again note that the misshapen-

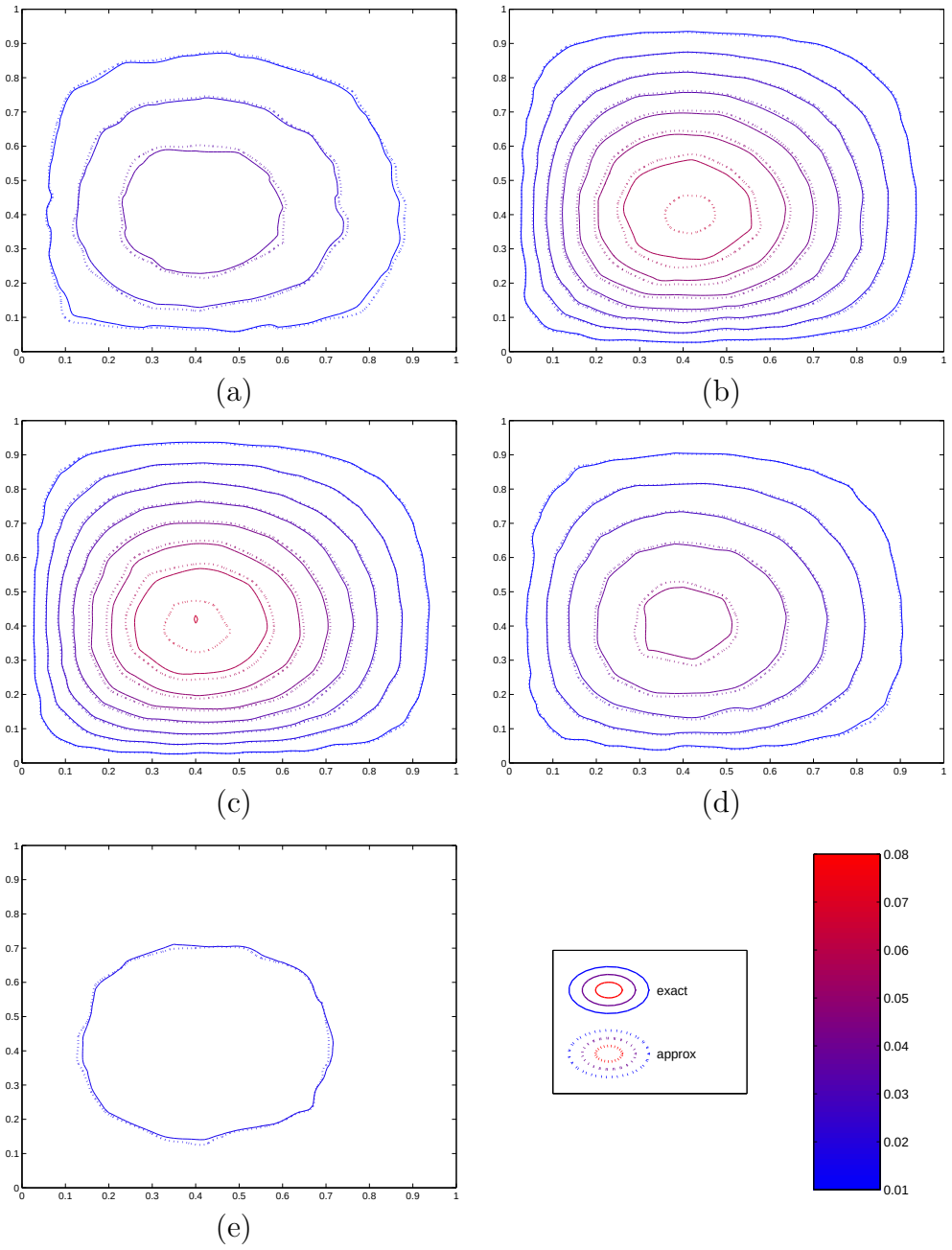


Figure 6.15: Temperature contours for Test Problem 2 using CV-FE. Contours at (a) $z = 0.1$; (b) $z = 0.3$; (c) $z = 0.5$; (d) $z = 0.7$; (e) $z = 0.9$. Colour bar shows temperature.

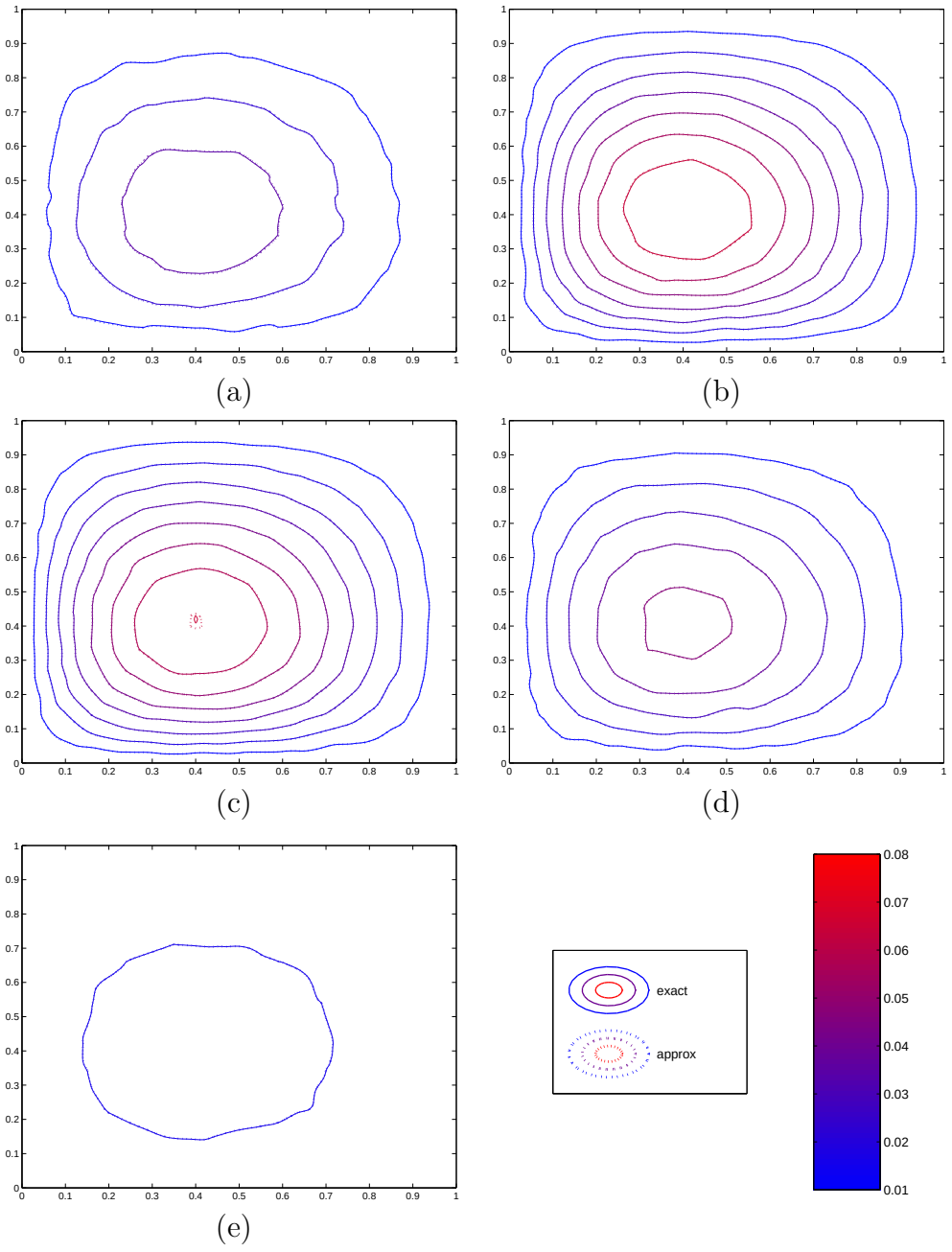


Figure 6.16: Temperature contours for Test Problem 2 using CV-RBF. Contours at (a) $z = 0.1$; (b) $z = 0.3$; (c) $z = 0.5$; (d) $z = 0.7$; (e) $z = 0.9$. Colour bar shows temperature.

ness of the contours in all solutions is an artefact of the visualisation process, produced when the results over the tetrahedral mesh are mapped onto a regular grid by the visualisation software.

Figures 6.17 and 6.18 illustrate the results of solving Test Problem 3, which has the same solution as Test Problem 2, but includes a nonlinear advection term. The CV-FE solution in Figure 6.17 is seen to suffer from the same problems as in the previous example. In the interior, where the advection process now dominates, the approximate solution contours are significantly out of position. In Figure 6.18 this incorrect behaviour is again largely corrected by using CV-RBF. The only contour visibly out of position is the innermost contour of the slice at $z = 0.5$ (Figure 6.18(c)).

In summary, whether the problem was constructed using nonlinear diffusion or nonlinear advection, the CV-RBF method was better equipped to capture the correct solution behaviour, particularly in regions of high temperature and high advection or diffusion.

Shape functions with Gaussian quadrature

Analogous to what was done in Section 4.3.1, in this section the CV-RBF method is applied with $n = 4$ nodes per interpolation to solve Test Problems 1, 2 and 3. In this way, the accuracy of the shape function interpolation/Gaussian quadrature pairing is tested, to see if the CV-FE method is being unfairly penalised by not employing Gaussian quadrature as its method of interpolation.

Tables 6.6 and 6.7 show the results of these tests, along with the results for CV-FE from Tables 6.4 and 6.5 for comparison purposes. As was observed in two dimensions, in no case is the accuracy of the CV-FE method significantly improved by the use of Gaussian quadrature.

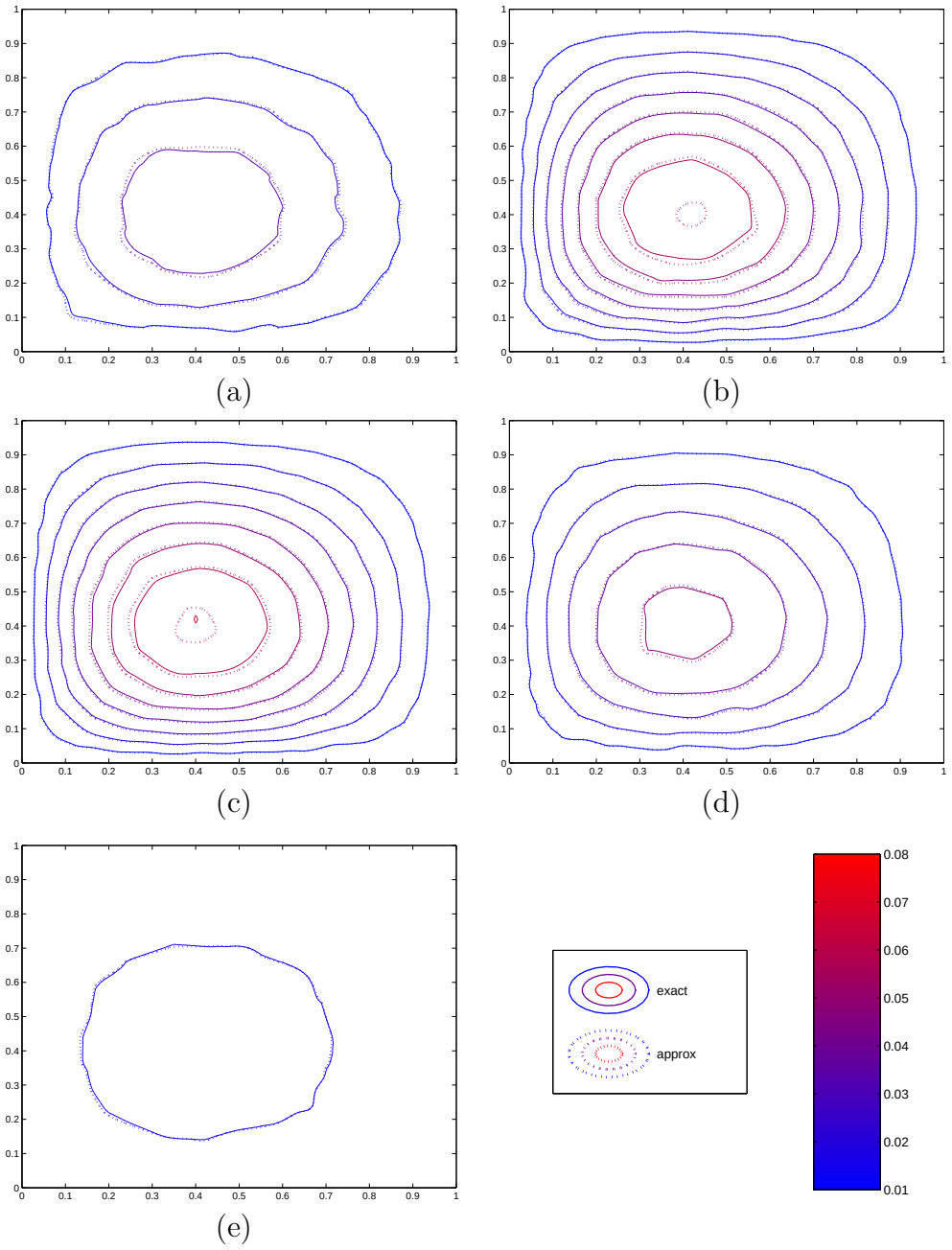


Figure 6.17: Temperature contours for Test Problem 3 using CV-FE. Contours at (a) $z = 0.1$; (b) $z = 0.3$; (c) $z = 0.5$; (d) $z = 0.7$; (e) $z = 0.9$. Colour bar shows temperature.

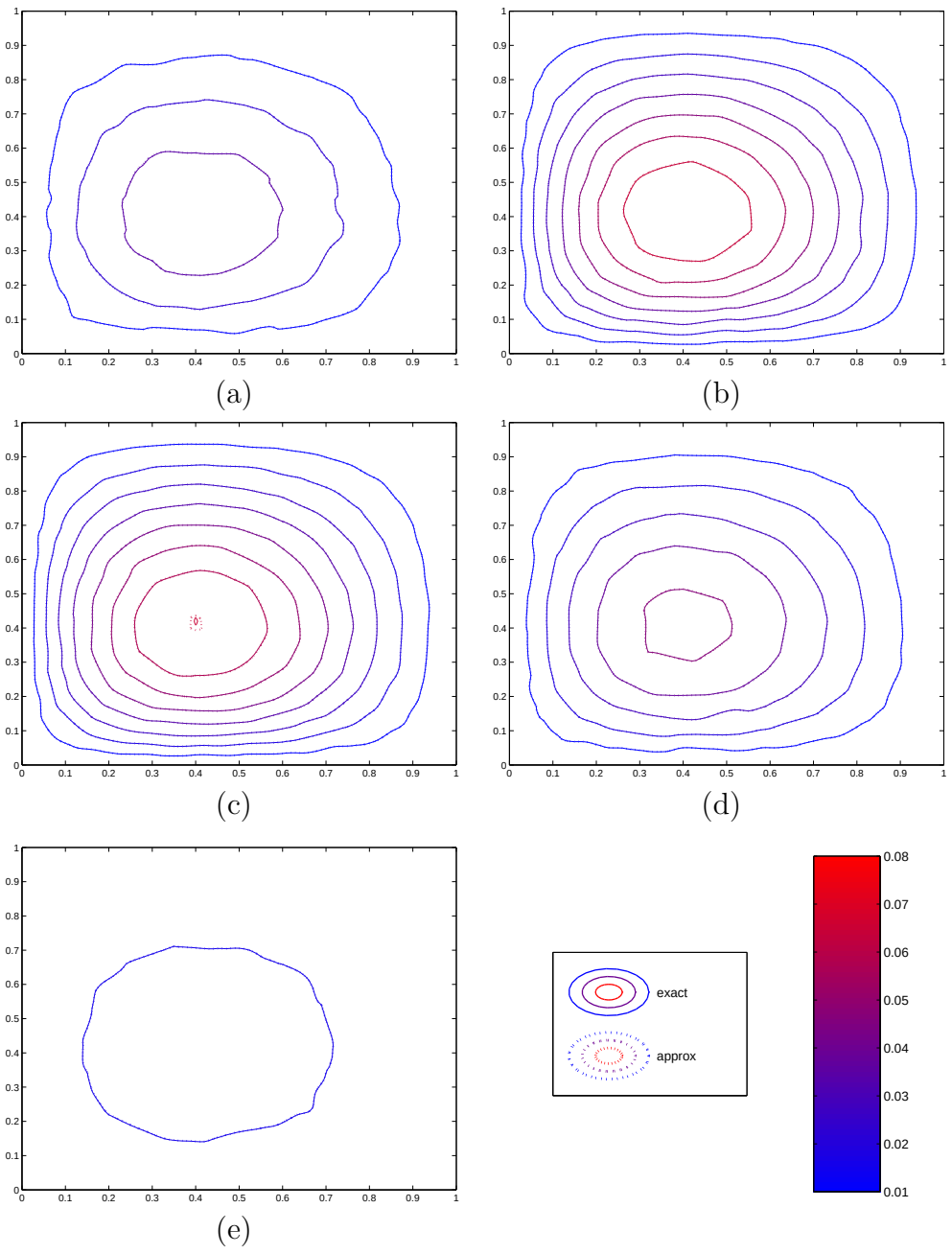


Figure 6.18: Temperature contours for Test Problem 3 using CV-RBF. Contours at (a) $z = 0.1$; (b) $z = 0.3$; (c) $z = 0.5$; (d) $z = 0.7$; (e) $z = 0.9$. Colour bar shows temperature.

Table 6.6: Errors using CV-FE and using CV-RBF ($n = 4$) for Test Problem 1.

D	diag(5, 5, 5)	diag(5, 50, 500)	diag(5, 5000, 5000)
CV-FE	4.40E-05	1.34E-04	9.33E-04
CV-RBF(n=4)	4.35E-05	1.33E-04	9.32E-04

Table 6.7: Errors using CV-FE and CV-RBF ($n = 4$) for Test Problems 2 and 3.

	Test Problem 2	Test Problem 3
CV-FE	2.90E-02	1.54E-02
CV-RBF(n=4)	2.09E-02	9.12E-03

6.3.2 Results over several meshes

In this section, orders of accuracy of the CV-FE and CV-RBF methods are estimated. As was the case in Section 4.3.2, plots of the error in the solution against the average control volume face edge length are used to observe how the error decreases as the mesh is refined.

Once again, since unstructured meshes are being used, some degree of variation in the results will be present. In Section 4.3.2 the general trend of the error reduction was nevertheless clear, and was found to be consistent across all three test problems. This section is an investigation of whether the same is true in three dimensions.

The same error measurement as was used in Section 4.3.2 is also applied here:

$$\begin{aligned} \text{error} &= \|\boldsymbol{\varphi}^{(e)} - \boldsymbol{\varphi}^{(a)}\|_\infty \\ &= \max|\varphi_i^{(e)} - \varphi_i^{(a)}|, \quad i = 1 \ldots N \end{aligned} \tag{6.14}$$

where superscript (e) symbolises the (exact) analytic solution and superscript (a) the (approximate) numerical solution.

Table 6.8: Estimated orders of CV-FE and CV-RBF methods for solving Test Problem 1 ($\mathbf{D} = \text{diag}(5, 5, 5)$), Test Problem 2 and Test Problem 3.

	n		
	4	8	20
Test Problem 1	2.3	1.8	2.8
Test Problem 2	2.0	2.4	3.6
Test Problem 3	1.7	2.0	2.7

Three discretisation strategies were used in the process of solving Test Problems 1, 2 and 3. The first was the usual CV-FE method using four nodes per interpolation. The other two strategies were the CV-RBF method using 8 and 20 nodes per interpolation respectively. The order, p, of the method was then estimated by fitting the curve

$$\text{error} = \text{constant} \times h^p \tag{6.15}$$

using logarithmic regression, where h is the average face length of the control volume exhibiting the maximum error in (6.14). The value of p computed by the regression model was used as the estimate of the method's order of accuracy, and the results are summarised in Table 6.8.

In Section 4.3.2 it was found that using $n = 6$ points per RBF interpolation did not necessarily produce an estimated order of accuracy that was superior to using shape functions with $n = 3$. In two dimensions, $n = 3$ corresponds to using the element's own vertices in the interpolation (shape functions), while $n = 6$ corresponds to using the element's vertices as well as its nearest neighbours' vertices.

In three dimensions the corresponding values are $n = 4$ for shape functions, and $n = 8$ for the element's vertices plus its nearest neighbours' vertices. Using these values, similar results are observed in three dimensions as were observed in two. There is no appreciable difference in the orders of the

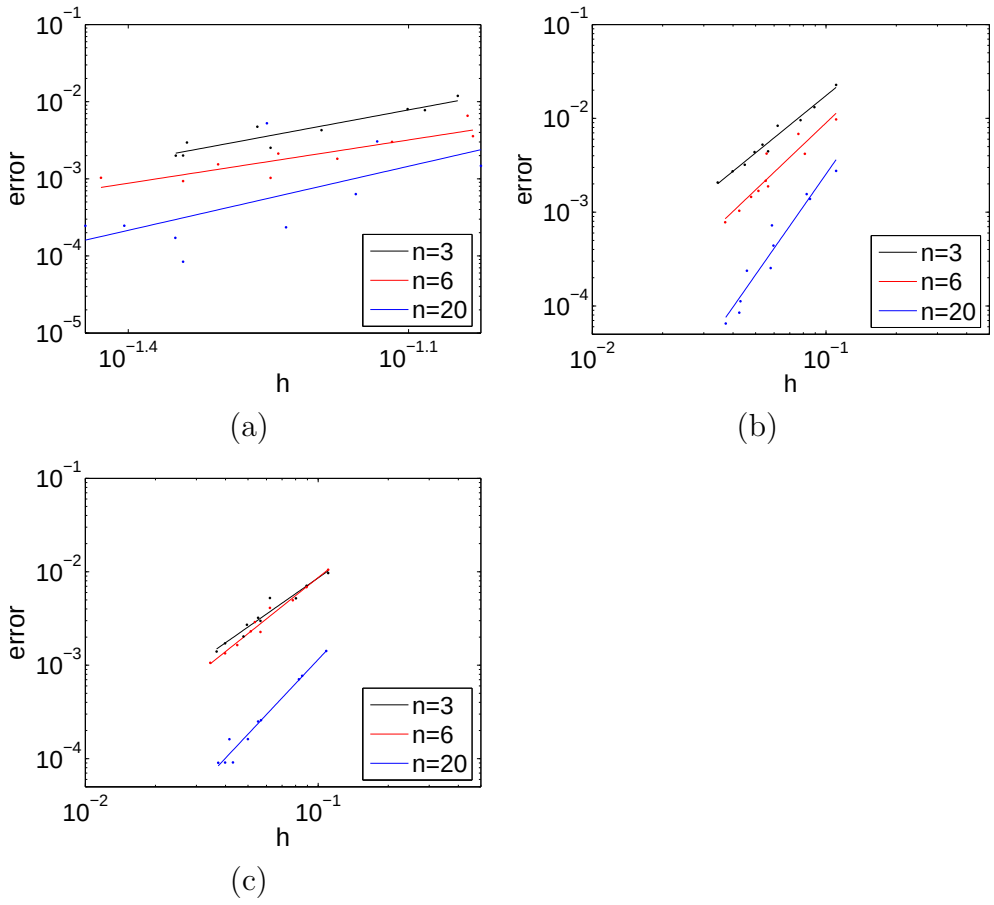


Figure 6.19: Error versus mesh refinement for (a) Test Problem 1 with $\mathbf{D} = \text{diag}(5, 5, 5)$; (b) Test Problem 2; (c) Test Problem 3. Least squares line also shown.

methods, with both appearing to offer close to second order accuracy. Using $n = 20$ nodes per interpolation however does result in a significant increase in the estimated order of the method, with close to third order accuracy observed across the three test problems. As was the case in two dimensions, the results vary somewhat, but this could be caused by the variation in the unstructured mesh edge lengths. In any case, the general trend is clear: using more nodes in each RBF interpolation results in a higher observed order of the CV-RBF method.

In Figure 6.19 the plots of the error in the solution against the average mesh edge length are shown. The results are immediately identifiable with those that were observed in two dimensions: there is a characteristic reduction in the error associated with the refinement of the mesh. Furthermore this behaviour is consistent across all three test problems. The increased accuracy offered by the CV-RBF method is evident, as is the increased order in the case where $n = 20$.

6.4 Results concerning efficiency

6.4.1 Preconditioning

Jacobian structure

In this section the Jacobian structure resulting from using CV-RBF for the finite volume discretisation is compared to that of using CV-FE. Figure 6.20 shows the sparsity patterns for the Jacobian matrices generated on the unstructured mesh illustrated in Figure 6.8(d) for four different choices of n, the number of nodes used in each element's interpolation.

From Table 6.2, unstructured mesh (d) has a total of 1567 nodes, so

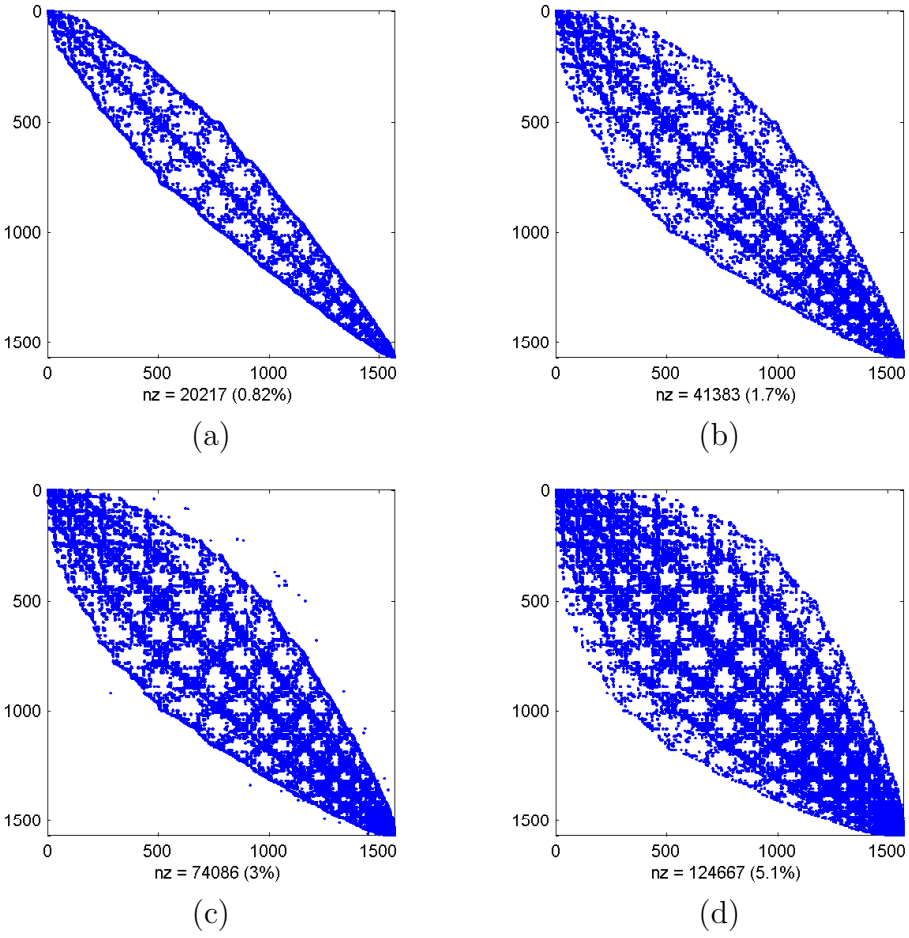


Figure 6.20: Jacobian matrices for mesh (d) of Figure 6.8 when using (a) $n = 4$; (b) $n = 8$; (c) $n = 20$; (d) $n = 40$.

each of these Jacobian matrices are of dimension 1567×1567. The impact that the number of nodes used in each element's interpolation has on the sparsity of these matrices can be seen in the figure. With $n = 4$ nodes used per interpolation (shape functions) the matrix is less than 1% dense, with a relatively tight bandwidth. Increasing n to 8 essentially doubles the Jacobian density, and significantly increases the bandwidth. Further increasing n to 20 again increases the density by roughly a factor of two, although this is not at the expense of greatly increasing the bandwidth. Instead, more of the bandwidth is filled-in with nonzero entries. The final value of $n = 40$ adds significantly to both the density and bandwidth, although even now the matrix is only 5.1% dense.

These results can be contrasted with those observed in two dimensions, where using only 25 nodes per interpolation produced a matrix density of almost 20% (Figure 4.20(d)). In three dimensions, using 40 nodes per interpolation has only resulted in a matrix density of just over 5%. However it would be incorrect to conclude that in three dimensions one can therefore use 40 or even more nodes per interpolation without penalty. As was discussed in Section 6.2, the meshes in three dimensions tend to be much larger than in two for the same average edge length. With so many more elements to process, it is important for the efficiency of the method that the number of nodes used per interpolation is carefully chosen to provide good accuracy without excessive cost. The choice of $n = 20$ was found in the previous section to provide improved accuracy and orders of convergence compared to CV-FE, so this value continues to be used throughout this section.

Jacobian spectra

The method of preconditioning outlined in Section 2.5.2 and illustrated previously in Section 4.4.1 is independent of the problem dimension. Furthermore, Figure 6.20 demonstrates that the structure of the Jacobian matrices in three dimensions is consistent with those encountered in two dimensions. Therefore the same detailed analysis on the effect of preconditioning that was conducted in two dimensions is not repeated here. Instead, one simple example is presented that shows how the observations made in two dimensions continue to apply in three.

Figure 6.21 is a plot of the eigenvalues of the Jacobian matrix for Test Problem 1 with $\mathbf{D} = \text{diag}(5, 5000, 5000)$ when using CV-FE. All eigenvalues are real, and range up to almost 2500. The corresponding plot when using CV-RBF is given in Figure 6.22. The pattern is consistent with that observed in two dimensions, namely that the eigenvalues of $\mathbf{J}_{RBF}$ are mostly complex, but with small imaginary parts and comparable magnitudes to those of $\mathbf{J}_{FE}$.

In Figures 6.23 and 6.24 the resulting spectra after applying the approximate inverse preconditioner $\mathbf{M}_1$ of Section 2.5.2 are shown. Once again, the result has been to cluster the majority of eigenvalues around unity, with the maximum eigenvalue now less than 2 for both $\mathbf{J}_{FE}$ and $\mathbf{J}_{RBF}$. This reinforces the notion that using the CV-FE Jacobian as the basis for preconditioning the CV-RBF iterations can be successful.

Finally, in Figure 6.25 the effect of the deflation preconditioner $\mathbf{M}_2$ of Section 2.5.2 on $\mathbf{J}_{RBF}$ is shown. Using estimates of the smallest five eigenvalues of $\mathbf{J}_{FE}\mathbf{M}_1$, the deflation preconditioner $\mathbf{M}_2$ is constructed, and applied to $\mathbf{J}_{RBF}\mathbf{M}_1$. The effect is to successfully deflate four of the five smallest eigenvalues, while leaving the fifth (and largest) undeflated. Once again the information drawn from the CV-FE Jacobian has been found to be effective

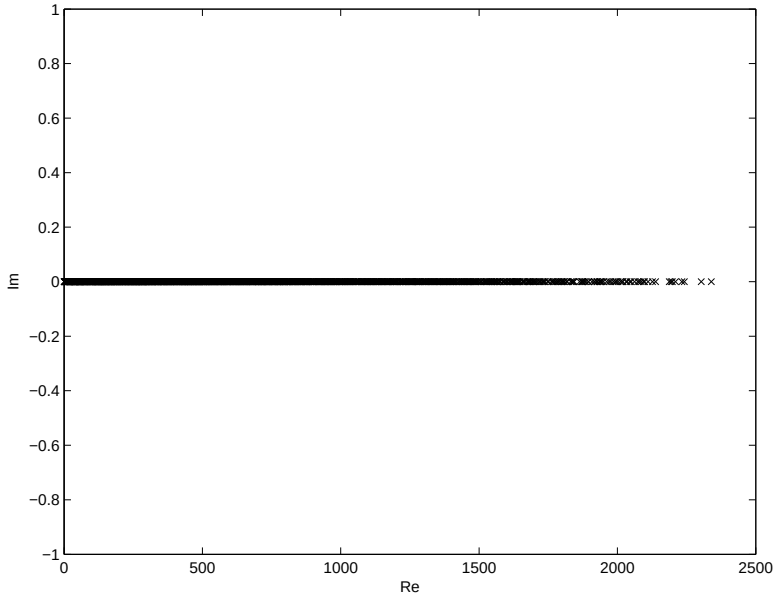


Figure 6.21: Eigenvalues of $\mathbf{J}_{FE}$ for Test Problem 1 with $\mathbf{D} = \text{diag}(5, 5000, 5000)$.

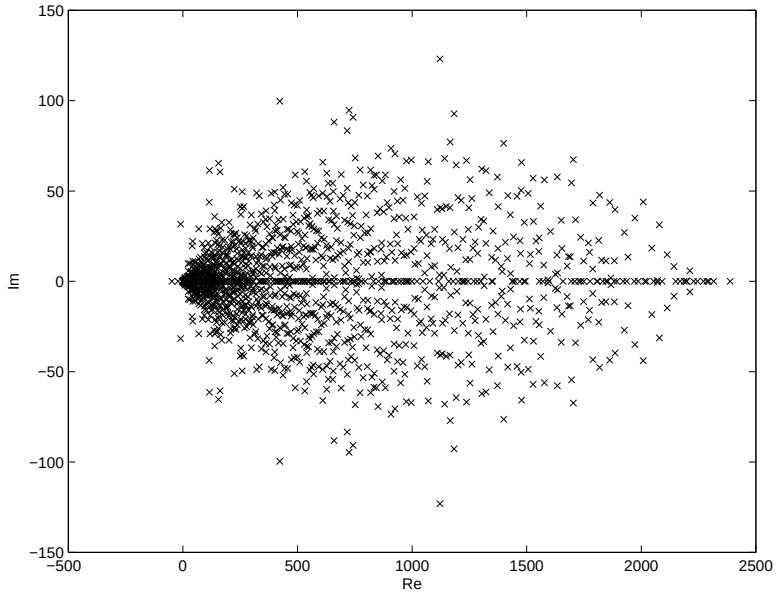


Figure 6.22: Eigenvalues of $\mathbf{J}_{RBF}$ for Test Problem 1 with $\mathbf{D} = \text{diag}(5, 5000, 5000)$.

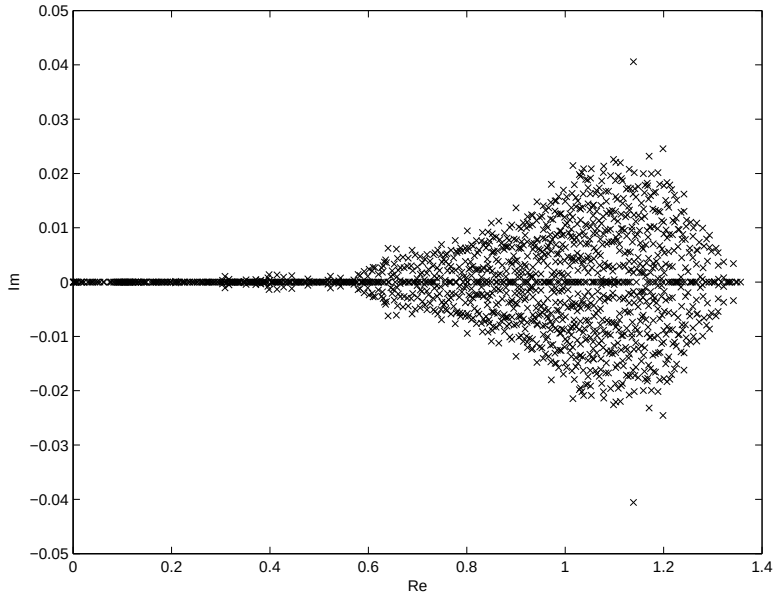


Figure 6.23: Eigenvalues of $\mathbf{J}_{FE}\mathbf{M}_1$ for Test Problem 1 with $\mathbf{D} = \text{diag}(5, 5000, 5000)$.

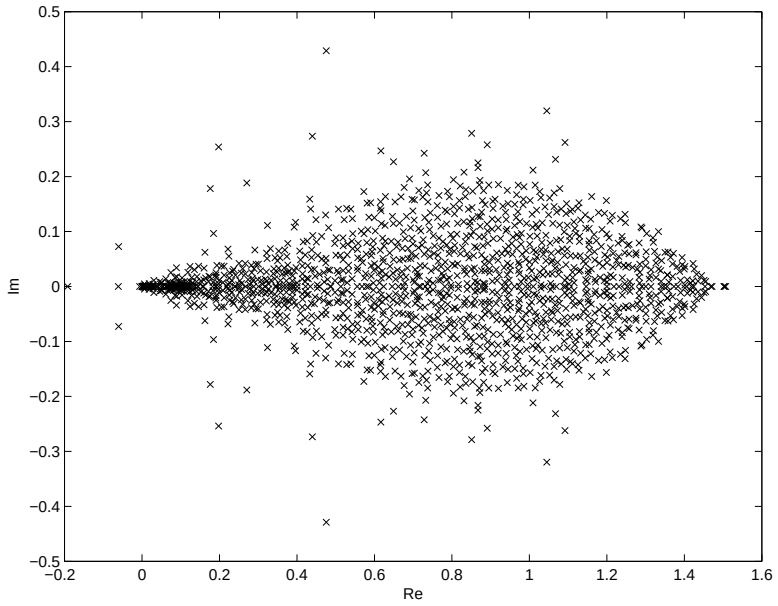


Figure 6.24: Eigenvalues of $\mathbf{J}_{RBF}\mathbf{M}_1$ for Test Problem 1 with $\mathbf{D} = \text{diag}(5, 5000, 5000)$.

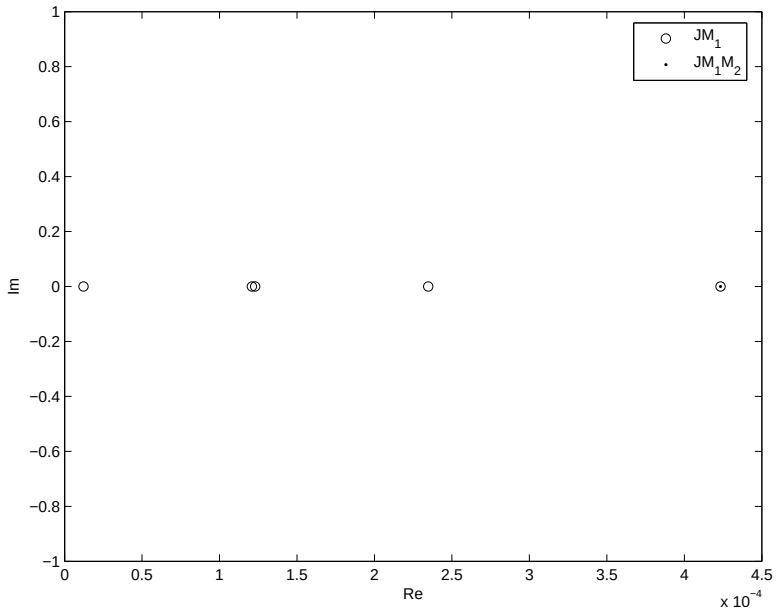


Figure 6.25: Smallest eigenvalues of $\mathbf{J}_{RBF}\mathbf{M}_1$ and $\mathbf{J}_{RBF}\mathbf{M}_1\mathbf{M}_2$ for Test Problem 1 with $\mathbf{D} = \text{diag}(5, 5000, 5000)$.

in preconditioning the CV-RBF problem.

Parallelisation

Recall from Section 2.6 that the preconditioner $\mathbf{M}_1$ of Section 2.5.2 can be parallelised by distributing the N least squares problems (2.85) over multiple processors. This parallelisability is extremely important when dealing with matrices arising from large, three-dimensional problems. For example, mesh (e) of Figure 6.8 gives rise to a 10385×10385 sparse Jacobian matrix. Generating the preconditioner $\mathbf{M}_1$ for this matrix takes more than three minutes when run on a single processor.

In this section the effectiveness of the parallel preconditioner implementation at reducing this generation time is investigated. To do so, the same preconditioner $\mathbf{M}_1$ is constructed over the same mesh, on a parallel computer

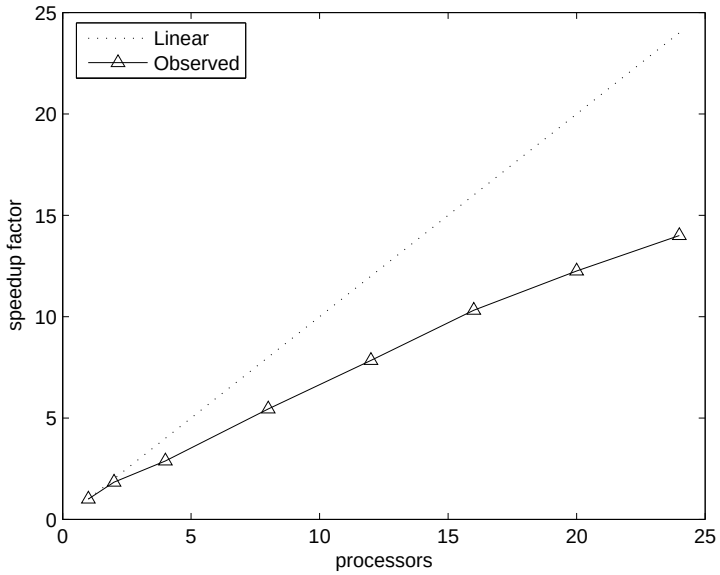


Figure 6.26: Parallel speedup for generating preconditioner $\mathbf{M}_1$ over mesh (e) of Figure 6.8.

Table 6.9: Parallel timings for generating preconditioner $\mathbf{M}_1$ over mesh (e) of Figure 6.8.

Processors	1	2	4	8	12	16	20	24
Time (s)	196	107	68	38	25	19	16	14

using increasing numbers of processors. The machine used for these tests is an SGI Origin 3000 with 600MHz R14000 processors and 90 GB of shared memory, with the OpenMP API [8] used to implement the parallelism.

The resulting timings for runs with different numbers of processors are given in Table 6.9. In Figure 6.26, the speedup factors (time for 1 processor divided by time for n processors) are plotted, along with the theoretical, ideal, linear speedup.

Ideally, when running an algorithm in parallel, doubling the number of processors would halve the execution time. From Figure 6.26, it is clear that

this ideal linear speedup is not achieved in this case. For example, increasing from one processor to two results in a reduction in time of approximately 55%, for a speedup factor of 1.8. Doubling again to four processors reduces this time by a further 64%, for an overall speedup factor of 2.8, and so on.

The reason that perfect linear speedup is not achieved is related to the overhead involved in spawning the separate processes. For this particular algorithm, it is the need for each process to dynamically allocate enough memory to store and solve the small least squares problems (2.85) that accounts for most of the overhead. In fact, it was found experimentally that allocating the least possible memory for the work array required by the underlying LAPACK [1] routine `dgels` was the optimum strategy for this problem, even though it results in a sub-optimal time to compute each least squares solution. When using the LAPACK documentation's recommended work array size, the small time savings in solving each least squares problem were swamped by the increased time for each process to allocate the required memory.

Nevertheless, the parallel speedup achieved for this problem is perfectly acceptable: a speedup factor of almost 15 times was achieved when using 24 processors in total. This sort of speedup factor enables the method to be used even on very large problems. For example, under these conditions a problem that would take a whole day to precondition when run in serial would take just over an hour and a half when run in parallel on 24 processors.

6.4.2 The effect of the RBF parameter c^2

In Section 4.4.3 it was found that in two dimensions the optimum choice of c^2 was dictated primarily by efficiency concerns. Large values of c^2 were associated with large numbers of nonlinear iterations required for the nonlinear

solution process to converge. The accuracy of the final solution depended to an extent on the value of c^2, but any value of c^2 that was simply not too small was sufficient to obtain near-optimum accuracy.

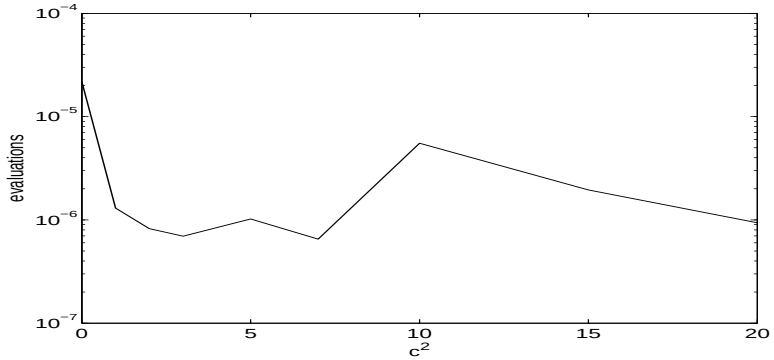
In Figures 6.27 and 6.28 the results of the corresponding tests in three dimensions are shown. Figure 6.27 shows how, particularly for Test Problems 2 and 3, the error in the final solution is not greatly affected by the value of c^2, provided it is not too small. Even for Test Problem 1, the "default" value of $c^2 = 1$ offers close to the best accuracy.

Figure 6.28 again highlights that the efficiency of the method is the primary concern dictating the choice of c^2. Large values of this parameter are associated with increasingly high numbers of nonlinear iterations required to achieve convergence. Particularly in three dimensions, where every additional function evaluation is costly, it is therefore advisable to choose the value of c^2 to be small. This explains the approach taken for all numerical experiments in this chapter, where the value of c^2 was taken to be 1 throughout.

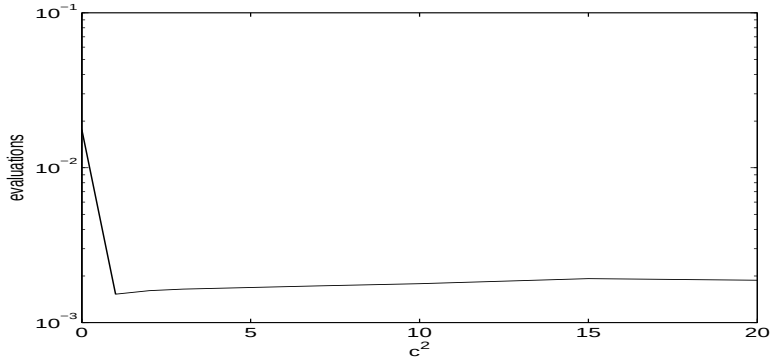
Note finally that there is no theoretical difficulty in using a different value of c^2 for every single interpolation in the mesh. Practically though, this would require some method of choosing the optimum value for each interpolation, possibly through trialling different values for each element or using a strategy like that proposed in [72]. Performing this process for every element in the mesh seems likely to be too costly for practical purposes. Therefore, throughout this thesis the same value of c^2 has been used throughout the mesh.

6.4.3 Comparative resource requirements

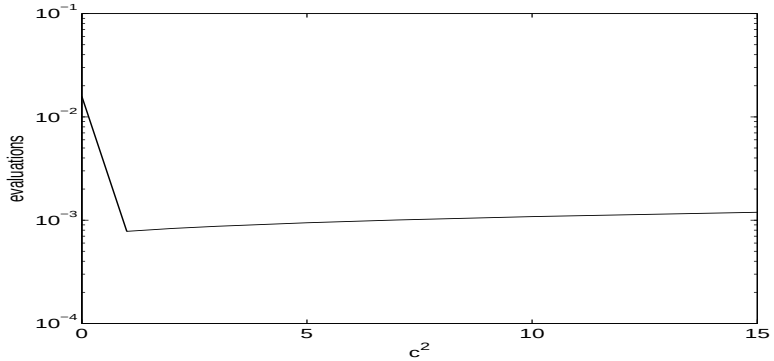
In previous sections it has been demonstrated how the accuracy achieved by the CV-RBF method is superior to that achieved by the classical CV-FE



(a)

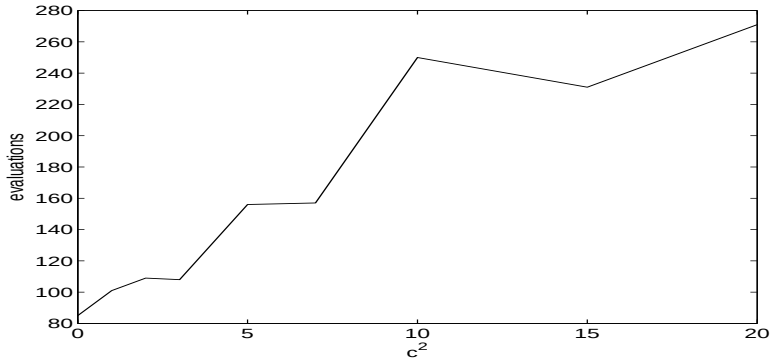


(b)

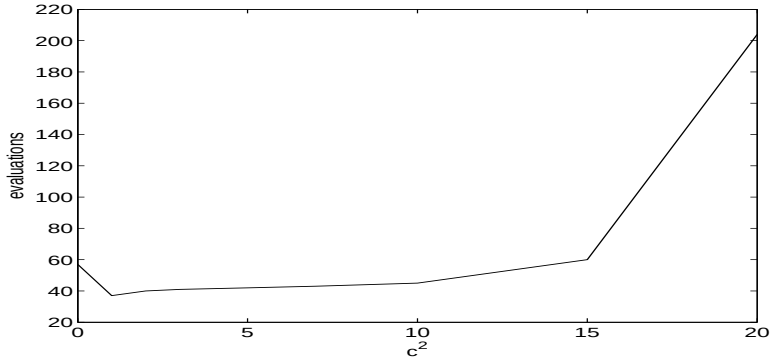


(c)

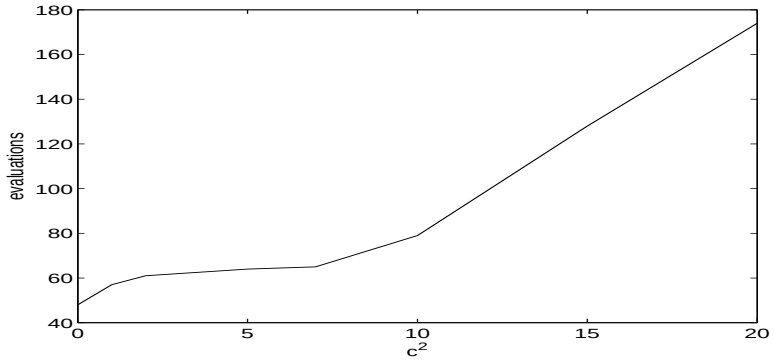
Figure 6.27: Error in solution versus c^2 for (a) Test Problem 1 with $\mathbf{D} = \text{diag}(5, 5, 5)$; (b) Test Problem 2; (c) Test Problem 3.



(a)



(b)



(c)

Figure 6.28: Number of function evaluations before convergence versus c^2 for (a) Test Problem 1 with $\mathbf{D} = \text{diag}(5, 5, 5)$; (b) Test Problem 2; (c) Test Problem 3.

method. In producing this additional accuracy, the resource requirements of CV-RBF are more demanding than they are for CV-FE on the same mesh. Particularly in three dimensions, these additional resource requirements must be taken into account when deciding if the method is practical.

An important question then, is whether the extra accuracy offered by CV-RBF compensates for the extra resources required to compute it. In this section, the efficiency of the CV-RBF is investigated, by comparing its resource requirements to those of CV-FE over a refined mesh, such that the accuracy achieved by the two methods is equivalent.

In order to make these comparisons, the resource requirements are monitored over the following three phases of the preprocessing and solution process:

1. generating the required mesh.

2. optimising the quality of the mesh.

3. solving the problem using the finite volume method.

The first two phases, involving mesh generation and optimisation, have largely been ignored up to this point of the experimentation, as they are not the focus of this thesis. However, the generation and in particular the optimisation of an unstructured, tetrahedral mesh is by no means a trivial task and the time required for these phases, particularly when the desired mesh is very fine, should not be overlooked. Therefore the time taken for both of these phases is included when comparing the total time taken for CV-FE on a fine mesh to that of CV-RBF on a coarser mesh.

Along with the time taken to solve a problem, another important factor that cannot be overlooked for any method of solution is the amount of

memory it requires. All computers have limited amounts of physical memory, so fast solution methods with steep memory requirements are not always preferable to slower methods that require less physical memory. The memory requirements are therefore also monitored, along with the time requirements, for all three phases in the solution process.

The tests are based on Test Problem 3 and were conducted on an Intel Pentium 4 PC with a 2.8GHz processor and 1GB of memory. In Section 6.3.1 it was found that solving this problem using CV-RBF resulted in a solution that was accurate to within 7.80E-04, while CV-FE produced an accuracy of 1.54E-02. The mesh used for these tests was the 1567 node, 7081 element mesh illustrated in Figure 6.8(d). Generating this mesh using the mesh generator *Gmsh* took 3 seconds, and optimising it for quality an additional 12 seconds with a maximum memory requirement of 20 MB. Running the full CV-RBF solver on this problem took a total of 54 seconds and required 36MB of memory.

Successively refining this mesh, and solving using CV-FE, eventually yields a solution of comparable accuracy to that achieved by CV-RBF (when compared using the error measurement (6.13)). The mesh that achieves this accuracy is illustrated in Figure 6.8(f). It is a mesh comprising 19367 nodes and 103008 elements that took 48 seconds to generate, and 152 seconds to optimise for quality, with a maximum memory requirement of 94 MB. Running the CV-FE code on this mesh resulted in a total running time of 228 seconds and a memory usage of 324MB.

These results are summarised in Tables 6.10 and 6.11. The results show that in every respect measured it takes more resources to generate a solution of equivalent accuracy using CV-FE than it does using CV-RBF. Whether it is the time and memory required to generate and refine the mesh or to apply

Table 6.10: Mesh sizes for CV-RBF and CV-FE to achieve comparable accuracy for Test Problem 3.

Method	Nodes	Elements
CV-RBF	1567	7081
CV-FE	19367	103008

Table 6.11: Resource requirements for CV-RBF and CV-FE (comparable accuracy) for Test Problem 3.

Phase	Method	Time (s)	Memory (MB)
Mesh generation	CV-RBF	3	16
	CV-FE	48	70
Mesh optimisation	CV-RBF	12	20
	CV-FE	152	94
Finite volume method	CV-RBF	54	36
	CV-FE	228	324

the finite volume method, using CV-RBF to generate the solution can be up to 10 times more efficient than CV-FE.

6.4.4 Adaptive switching

In this section the effectiveness of the adaptive switching strategy discussed in Section 2.4.4 is investigated. The idea of this method is to adaptively determine which elements in the mesh do not require the full accuracy offered by radial basis function interpolation, and thereby possibly save on computational effort without sacrificing the accuracy of the method.

Throughout this thesis several ways in which the method of shape functions can be usefully employed as part of an overall RBF-based finite volume scheme have been presented. One example is the use of CV-FE as the first stage of the nonlinear solution process, which offers a cheap way of producing a solution of reasonable accuracy with which to commence the CV-RBF iterations. Another example is the use of CV-FE Jacobian information in

constructing a preconditioner for the CV-RBF iterations. Doing so allows the CV-RBF iterations to be carried out in a completely Jacobian-free manner, while still providing an effective means of preconditioning.

In this section, a third way in which CV-FE can be used as part of the overall method is investigated, this time by helping to identify which elements in the mesh can make do with using shape functions as local interpolants, and which elements must use RBFs to achieve the desired accuracy.

The adaptive switching process is an extension of the two stage nonlinear solution strategy that has been tested to this point. The only change to the method is the inclusion of Algorithm 4 (see Section 2.4.4) just after the switch has been made from CV-FE to CV-RBF. The idea of the method is to monitor how the absolute residual components of $\mathbf{F}$ change as a result of switching from CV-FE to CV-RBF. If a residual component is largely unchanged at a given node then this is an indication that shape functions may be sufficient there. If the residual component increases significantly in magnitude at a given node then this indicates that RBF interpolant produces substantially different results from the shape function interpolant, and should therefore be retained.

The reader is reminded of the two parameters introduced in Section 2.4.4 for the adaptive switching algorithm. The first parameter, n_{init}, is the number of nodes used in the initial RBF interpolations, computed during the switch from CV-FE to CV-RBF. Previously, these interpolations would immediately be put to use in the stage two CV-RBF iterations. In the adaptive strategy, Algorithm 4 is applied instead, to determine precisely where RBFs are actually required to increase the accuracy.

For elements deemed to require RBFs, the second parameter, n_{final}, dictates how many nodes will be used for each RBF interpolation. The moti-

vation for using two values of n was discussed in Section 2.4.4: it allows for using a small value of n_{init} to initially determine for which elements RBFs are required, before using the large value of n_{final} only for these elements' interpolations to generate the high accuracy desired.

Tables 6.13, 6.14 and 6.15 list the results of solving Test Problem 3 using this adaptive switching strategy with $n_{\text{init}} = 10$, $n_{\text{init}} = 15$ and $n_{\text{init}} = 20$ respectively. Within each table, values of 10, 15, 20 and 25 have been used for n_{final}. For comparison, Table 6.12 lists the results for the non-adaptive strategy, when using $n = 10$ and $n = 20$.

The tables list the following measurements related to the accuracy and efficiency of the methods:

Switch time: The total time taken for the switch between nonlinear stages one and two. This includes the construction of the initial RBF interpolants using n_{init} nodes per element, the time for Algorithm 4, and the subsequent recalculation of RBF interpolants using n_{final} nodes per element (where applicable).

Solve time: The time taken from after the switch has been made, until the solution has converged. This is the time taken for the stage two nonlinear iterations. Note that the time for the stage one CV-FE iterations is not reflected in the recorded times as it does not vary between methods and is not significant by comparison.

Avg. nodes: The number of nodes used per element-wise interpolation, averaged over the entire mesh. This is equal to: [(number of elements using shape functions) $\times$ 4 + (number of elements using RBFs) $\times$ n_{final}] / (number of elements in mesh).

Error: The error in the solution, as measured by (6.13).

Each set of measurements just described has been taken for three different choices of threshold value. Recall from Section 2.4.4 that the threshold value plays a role in determining whether a node is classified as requiring shape functions or RBFs. If its absolute residual component is less than the threshold, it is classified as requiring shape functions, otherwise it is classified as requiring RBFs.

To compute the threshold values, the interval between the maximum and minimum absolute components of the residual vector was divided into two zones, based on pre-determined ratios. These ratios are listed in Tables 6.13, 6.14 and 6.15 as percentages. A ratio such as 25/75 means that the bottom 25% of the range (measured logarithmically) was allocated as the shape function zone, while the top 75% was the RBF zone. Every residual component falls within one of the two zones, and is classified accordingly.

For example, if the maximum residual component is 1.0e-3 and the minimum is 1.0e-7, than the threshold value when using the 25/75 ratio would be 1.0e-4. Nodes with residual components less than this threshold would be classified as requiring only shape functions, while nodes with residual components greater than this would be classified as requiring RBFs. Note that this does not necessarily mean that 75% of the elements in the mesh will be classified as requiring RBFs. The actual distribution depends on the value of each residual component, and also how the nodes are connected throughout the mesh.

The first observation made from Tables 6.13, 6.14 and 6.15 is that there appears to be little point in using a threshold value corresponding to 25/75 or even 50/50. With these values, there is almost no difference in any row of any table between the average number of nodes per interpolation and the value n_{final}. For example, with $n_{\text{init}} = 10$ (Table 6.13) and $n_{\text{final}} = 20$, the

Table 6.12: Results of non-adaptive strategy on Test Problem 3.

n		
10	switch time	14.5
	solve time	15.3
	avg. nodes	10
	error	1.98E-02
20	switch time	18.8
	solve time	32.5
	avg. nodes	20
	error	7.80E-04

Table 6.13: Results of adaptive strategy on Test Problem 3 with $n_{\text{init}} = 10$.

n_{final}	Threshold:	25/75	50/50	75/25
10	switch time	15.1	15	14.8
	solve time	15.1	15	15.1
	avg. nodes	10	9.9	7.6
	error	1.98E-02	1.97E-02	1.73E-02
15	switch time	18.7	18.1	16.9
	solve time	20.1	21	18
	avg. nodes	15	14.8	10.7
	error	8.52E-03	8.55E-03	8.55E-03
20	switch time	19.8	19.8	18
	solve time	32	31.8	25
	avg. nodes	20	19.7	13.7
	error	7.82E-04	9.05E-04	2.93E-03
25	switch time	21.8	21.8	19.2
	solve time	38.8	38.7	30.8
	avg. nodes	25	24.6	16.7
	error	7.01E-04	8.44E-04	2.69E-03

Table 6.14: Results of adaptive strategy on Test Problem 3 with $n_{\text{init}} = 15$.

n_{final}	Threshold:	25/75	50/50	75/25
10	switch time	17.8	17.9	17.2
	solve time	15.2	15.2	15.3
	avg. nodes	10	9.9	7.9
	error	1.98E-02	1.98E-02	1.76E-02
15	switch time	16.1	16.2	16
	solve time	20.2	21.6	18.3
	avg. nodes	15	14.9	11.2
	error	8.52E-03	8.52E-03	8.26E-03
20	switch time	21.7	21.3	19.2
	solve time	32.1	31.9	26
	avg. nodes	20	19.8	14.5
	error	7.80E-04	8.31E-04	2.36E-03
25	switch time	22.9	23	20.6
	solve time	38.7	38.7	31.8
	avg. nodes	25	24.7	17.8
	error	6.97E-04	7.58E-04	2.23E-03

Table 6.15: Results of adaptive strategy on Test Problem 3 with $n_{\text{init}} = 20$.

n_{final}	Threshold:	25/75	50/50	75/25
10	switch time	21	20.8	20.2
	solve time	15.4	15.3	14.4
	avg. nodes	10	9.8	6.7
	error	1.98E-02	1.97E-02	1.54E-02
15	switch time	21.8	21.8	20.5
	solve time	21.9	21.2	17.4
	avg. nodes	14.9	14.7	8.9
	error	8.52E-03	8.52E-03	7.97E-03
20	switch time	19.3	19.7	19.3
	solve time	32.7	31.6	21.9
	avg. nodes	19.9	19.6	11.1
	error	8.01E-04	9.34E-04	3.49E-03
25	switch time	23.9	25.8	22.2
	solve time	38.5	38.1	25.4
	avg. nodes	24.9	24.4	13.3
	error	7.45E-04	9.23E-04	3.31E-03

average number of nodes per element when using a 25/75 threshold is in fact 20 to one decimal place. When using a threshold of 50/50 it is still 19.7. In other words, there are simply not enough elements being switched to shape functions to justify the method. This results in the average number of nodes used per interpolation being almost the same as if RBFs had been used throughout.

The 75/25 threshold is the one choice that does actually result in a significant reduction in the average number of nodes used per interpolation. For the same example as just given, using the 75/25 threshold produces an average of 13.7 nodes per element, which is a 31% reduction compared to the full RBF method.

Unfortunately, the tables also show that this reduction in the average number of nodes per interpolation is accompanied by a reduction in accuracy. Using the same example, the errors when using thresholds 25/75, 50/50 and 75/25 are 7.82E-04, 9.05E-04 and 2.93E-03 respectively. The first two of these are comparable to the error when using the full RBF method with 20 nodes (7.80E-04 from Table 6.12), but the third represents a more substantial increase in the error.

Another illustrative example is in Table 6.15 with $n_{\text{init}} = n_{\text{final}} = 20$. In this case the error with threshold 75/25 is 3.49E-03, which is again more than the error in the full method when using 20 nodes. It is, however, significantly better than the full method when using 10 nodes. From Table 6.12 the error in the full method using $n = 10$ is 1.98E-02. Thus using $n_{\text{init}} = n_{\text{final}} = 20$, the adaptive method comfortably improves upon this accuracy, while only using 11.1 nodes per interpolation on average.

As far as timings are concerned, the tables show that the time consumed in switching from stage one to stage two is not greatly increased when using

the adaptive method. From Table 6.12 the switch time when using 20 nodes is 18.8 seconds. The maximum switch times for a 75/25 threshold in Tables 6.13 and 6.14 are 19.2 and 20.6 seconds respectively, while in Table 6.15 the maximum switch time is still only 22.2 seconds. It also seems from these results that the strategy of using a smaller value of n_{init} is only slightly beneficial at reducing the time to switch.

It can also be seen from Tables 6.13, 6.14 and 6.15 that with a 75/25 threshold, the adaptive method converged faster than the full 20 node method (which, from Table 6.12, took 32.5 seconds). This includes methods that used $n_{\text{final}} = 25$, although disappointingly even these did not approach the accuracy offered by the full 20 node method. For methods using $n_{\text{final}} = 15$ and $n_{\text{final}} = 20$, solve times of 15-25 seconds were typical, a definite improvement over the full method. Using $n_{\text{final}} = 10$ did not significantly further reduce the solve times, and is therefore not recommended.

In summary, it is believed that there is merit to this adaptive strategy. The approach of using the nonlinear residual components to identify elements not in need of the full RBF accuracy can be successful. Using a threshold corresponding to 75% shape, 25% RBF did significantly reduce the average number of nodes used in interpolations across the mesh. The extra processing time in making these classifications was not found to be a significant factor in the overall running time of the method. The time spent in stage two of the nonlinear process was certainly reduced by using shape functions wherever possible. The main downside of the approach was that it never completely produced the accuracy obtained by the full CV-RBF method. This was caused, at least in part, by the difficulty in mapping the node-wise measure of effectiveness (using the residual components) to an element-wise indicator of which interpolation method to use.

6.5 Conclusions

In this chapter the effectiveness of the three-dimensional schemes proposed in the previous chapter was investigated by applying them to three different test problems. A family of unstructured, tetrahedral meshes was used for conducting these tests.

In the first set of tests, the analytic solutions were compared to the numerical solutions using CV-FE and CV-RBF, both visually and by measuring their relative errors over the entire mesh. For all problems, both linear and nonlinear, CV-RBF gave rise to improved accuracy over CV-FE, up to almost two orders of magnitude. Furthermore, the contour plots produced by the CV-RBF solutions were mostly indistinguishable from the analytic solution, with the only exception being when very high anisotropy was present. Even in these cases, the solutions were much more representative of the correct physical behaviour than those computed using CV-FE.

The next tests showed how the error produced by CV-FE and CV-RBF was reduced as the mesh was refined. Plots of the errors against the average element edge lengths were used to estimate the order of accuracy of the methods. It was found that the CV-RBF method with 8 nodes per interpolation performed no better than CV-FE in this regard, as both were measured as second order accurate. However, the CV-RBF methods with 20 nodes did offer an improved order of accuracy, which was estimated to be third order.

The method of preconditioning, incorporating elements of both CV-FE and CV-RBF was found to be effective in dealing with both small and large eigenvalues in the Jacobian spectrum. Timings for runs of the parallel preconditioning code using different numbers of processors showed how it can achieve a significant speedup when run in parallel. A speedup factor of almost 15 times was achieved when using 24 processors in total.

The time and memory requirements for meshing and solving Test Problem 3 were presented, showing how the resource requirements for CV-RBF are much less than for CV-FE to achieve a solution of comparable accuracy.

The adaptive strategy of switching between RBFs and shape functions during stage two of the nonlinear solution process was tested. It was found that such a strategy could reduce the computational overhead by reducing the average number of nodes used per interpolation, although the final solution never quite achieved the accuracy that the full CV-RBF method had achieved.

Chapter 7

Conclusions

The objective of this PhD research programme was to investigate the effectiveness of a finite volume method incorporating radial basis functions for simulating nonlinear transport processes. It has been found that this new method offers improved accuracy over existing control-volume finite-element discretisation techniques, permitting relatively coarse meshes to be used in obtaining solutions without a significant loss of precision. Furthermore, these solutions are obtained in less time, with a reduction in computational overheads compared with traditional shape function-based methods. A two-stage nonlinear solution strategy is beneficial, whereby shape functions are used to commence the solution process, before introducing radial basis functions in stage two to further refine the accuracy. Shape functions are also beneficial as part of an overall parallel preconditioning strategy. The next section summarises all of the relevant findings made during this research programme. The chapter concludes with some recommendations for future research.

7.1 Summary and discussions

In Chapter 1, the objectives of this research programme were outlined. These will now be examined to demonstrate how they were achieved, and to highlight the major contributions of this research.

Implement the finite volume method over unstructured two and three-dimensional meshes

The mesh generator *Gmsh* was used to generate families of unstructured triangular and tetrahedral meshes in two and three dimensions respectively. Using these meshes, methods were implemented to construct control volumes around each node in the mesh. It was found that in two dimensions, a method of forming control volumes whereby element centroids were connected directly to one another was advantageous, giving high accuracy while still requiring one less flux evaluation per control volume face than the method of connecting centroids to face midpoints. Conversely, in three dimensions the preferred method was to connect element centroids to face midpoints, owing to the increased complexity in forming three-dimensional control volumes.

Numbering and ordering conventions were established to ensure that fluxes between control volume faces were computed only once, and were consistent from the points of view of the two nodes on either side of the face. In two dimensions this was most easily accomplished by using an anticlockwise ordering convention, but in three dimensions, responsibilities for control volume faces had to be explicitly assigned to each sub-control volume.

Traditional CV-FE methods incorporating shape functions were implemented, and found to behave as reported in the literature, i.e. they were unable to fully capture the correct behaviour in cases of high anisotropy or high advection. The primary weakness of this method was identified as

being a lack of accuracy in the interpolation used in the discretisation. A set of requirements was identified that any alternative interpolation method would have to satisfy in order to be successful in the finite volume framework. In particular, it was identified that the interpolation scheme needed to be a local, rather than a global, method, preferably utilising element-wise interpolations.

Develop an accurate finite volume discretisation method for two and three dimensions

The method of radial basis functions (RBFs) was found to meet the requirements for a successful interpolation scheme within the finite volume framework. A new discretisation method, for convenience referred to here as CV-RBF, incorporating RBFs as a means of interpolation, along with Gaussian quadrature as a means of integration was developed. Methods for handling ill-conditioned RBF coefficient matrices through the use of the truncated singular value decomposition, and for efficiently computing the interpolation on a shifted set of function values were implemented. An algorithm for selecting a cloud of local nodes to use for each element's interpolation was developed. Methods for improving the accuracy of the RBF interpolation near boundaries were investigated, but found to be unnecessary owing to the use of RBFs as local, rather than global, interpolants.

A suite of numerical experiments was conducted in two and three dimensions to gauge the accuracy and efficiency of the new method. It was found that the CV-RBF method consistently achieved higher accuracy than the traditional CV-FE method, typically between two and three orders of magnitude in two dimensions, and between one and two orders in three dimensions. This was observed directly by comparing the numerical solutions with the

known, analytic solutions of the test problems. Furthermore, the resulting contour plots when compared with the analytic solutions confirmed that the CV-RBF method remained consistent with the analytic solution, while the CV-FE method often produced solutions that deviated significantly from the correct behaviour.

In particular, when large anisotropy ratios were introduced, the CV-FE solutions introduced significant errors, with spurious contours and other characteristics inconsistent with the analytic solution. This was true for both the two- and three-dimensional test problems. The CV-RBF method however was still able in most cases to produce solutions that were visually indistinguishable from the analytic solution. Furthermore, for the two-dimensional advection test problem, the CV-FE solution exhibited oscillatory behaviour for large advection rates, especially for cell Peclet numbers of ten thousand or more. The CV-RBF method however exhibited no signs of this nonphysical behaviour.

When solving the nonlinear test problems, the CV-FE method also consistently failed to reproduce the correct internal behaviour of the analytic solution, where the diffusion and advection was strongest. This was one problem that the CV-RBF method was not completely immune to, however the effect was only noticeable in three dimensions within the innermost set of contours, and to a much lesser extent than with the CV-FE solution.

Refining the mesh, and plotting the solution errors against the average control volume face edge length, showed that the CV-RBF method tended to result in higher orders of convergence than the CV-FE method. Interestingly though, this was generally only true when using more than six, or eight, nodes per interpolation in two and three dimensions respectively. With six or eight nodes, although the error of the method tended to be less when using

CV-RBF, both CV-FE and CV-RBF were measured at second order. However, with more nodes used per interpolation, both the accuracy and order of convergence with the CV-RBF method were improved, with orders measured as high as $\mathcal{O}(h^5)$ in two dimensions and $\mathcal{O}(h^3)$ in three.

By refining the mesh used for the CV-FE solution, it was eventually possible to obtain a solution of comparable accuracy to the CV-RBF solution on a coarse mesh. In doing so however, memory requirements for the method, along with the time taken for mesh generation, mesh optimisation and numerical solution increased significantly. It was found that the CV-RBF method was significantly more efficient than CV-FE when it came to generating a solution of comparable accuracy.

An adaptive strategy designed to automatically switch between using RBFs and using shape functions was investigated, with mixed results. It was found that the strategy could be successful at reducing the computational overhead, and consequently the time required to compute the solution, but the resultant accuracy was never quite able to match that of the full method.

Use an efficient Jacobian-free Newton-Krylov method with parallel preconditioning to solve the underlying nonlinear system

A Jacobian-free Newton-Krylov method was implemented to solve the resultant nonlinear system of equations. It was recognised that a two-stage nonlinear solution process was beneficial, whereby CV-FE was used initially to solve the problem, after which this solution was improved upon by a second stage employing CV-RBF. Using CV-FE in this way allowed a solution of reasonable accuracy to be used as the initial estimate for the much more expensive CV-RBF iterations.

It was also recognised that the method of shape functions is actually a

special case of the method of radial basis functions. In this way, the CV-FE discretisation process can be thought of as a simplification of the full CV-RBF discretisation process, and thus potentially useful in forming an effective preconditioner. This fact was exploited as part of the two-stage nonlinear solution process, whereby extra stage one iterations were expended in order to obtain accurate eigenvector information for use in preconditioning the GMRES-DR iterative solver.

A combination of two preconditioners was used, one based on approximate inverse norm minimisation, the other on explicit eigenvalue deflation. The approximate inverse preconditioner was assembled in parallel, with significant speedup factors achieved on a shared memory architecture. It incorporated aspects of both CV-FE and CV-RBF in its construction, and as a result was effective in preconditioning both stages of the nonlinear solution process. The deflation preconditioner was also constructed using CV-FE information, yet was consistently able to deflate small eigenvalues in the full CV-RBF Jacobian. In combination these preconditioners were effective at reducing the number of stage two nonlinear iterations required for convergence, and indeed in some cases achieving convergence where otherwise it would not have been achieved.

7.2 Recommendations for future research

Based on the outcomes of this research programme, the following recommendations are made for further research in this area:

- Incorporation of a fully implicit, Crank-Nicholson or other time-stepping scheme so that problems with temporal as well as spatial variation may be solved. Care would need to be taken that the improvements made in the spatial discretisation were not nullified by an

inaccurate temporal scheme.

- Incorporation of coupled nonlinear PDEs, with a corresponding block Newton-Krylov method. The scheme proposed in this thesis should be readily extensible to two or more coupled nonlinear PDEs, with each dependent variable subject to its own set of local RBF interpolations. The method already permits efficient shifting of interpolated function values: this could easily be extended to allow for outright switches of function values from one variable to the next.

- Application to real-world industrial or scientific problems. With the previous two extensions in place, the method would be fully equipped to deal with the types of real-world problems encountered in the field of computational fluid dynamics.

- Further investigation on advection-dominated problems and alternatives to upwinding. Given the success of the CV-RBF method for the two-dimensional test problem with high advection, it would be instructive to compare the method against other techniques such as flux limiting for advection-dominated problems.

- A more sophisticated adaptive switching strategy could be investigated to try to achieve the high accuracy of the CV-RBF method without all the associated costs. Perhaps the mapping from the node-wise indicators to the element-wise classifications could be refined, or perhaps the method needs to be more versatile in the allowable numbers of nodes per element-wise interpolation. Alternatively, the performance of the adaptive scheme could be continuously monitored throughout the process, rather than implemented as a one-off step between the nonlinear solution stages.

Bibliography

[1] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. LAPACK user's guide third edition. Available from: `http://www.netlib.org/lapack/lug/`.

[2] C. Bailey and M. Cross. A finite volume procedure to solve elastic solid mechanics problems in three dimensions on an unstructured mesh. *Int. J. Numer. Meth. Eng.*, 38:1757–1776, 1995.

[3] C. Bailey, G.A. Taylor, M. Cross, and P. Chow. Discretisation procedures for multi-physics phenomena. *J. Computational and Applied Mathematics*, 103:3–17, 1999.

[4] R. K Beatson, J. B. Cherrie, and C. T. Mouat. Fast fitting of radial basis functions: Methods based on preconditioned GMRES iteration. *Advances in Computational Mathematics*, 11:253–270, 1999.

[5] R. K. Beatson, W. A. Light, and S. Billings. Fast solution of the radial basis function interpolation equations: Domain decomposition methods. *SIAM J. Sci. Comput.*, 22(5):1717–1740, 2000.

[6] S. Bellavia and B. Morini. A globally convergent Newton-GMRES subspace method for systems of nonlinear equations. *SIAM J. Sci. Comput.*, 23(2):940–960, 2001.

[7] M. Benzi. Preconditioning techniques for large linear systems: a survey. *J. Computational Physics*, 182:418–477, 2002.

[8] OpenMP Architecture Review Board. OpenMP application program interface. Available from: `http://www.openmp.org/drupal/mp-documents/spec25.pdf`.

[9] P.N. Brown and Y. Saad. Hybrid Krylov methods for nonlinear systems of equations. *SIAM J. Sci. Statist. Comput.*, 11(3):450–481, 1990.

[10] R.L. Burden and J. D. Faires. *Numerical Analysis, seventh edition.* Brooks/Cole, 2001.

[11] K. Burrage and J. Erhel. On the performance of various adaptive preconditioned GMRES strategies. *Numerical Linear Algebra with Applications*, 5(2):101–121, 1998.

[12] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans. Reconstruction and representation of 3D objects with radial basis functions. *Computer Graphics (SIGGRAPH 2001 proceedings)*, pages 67–76, 2001.

[13] A. Chapman and Y. Saad. Deflated and augmented Krylov subspace techniques. *Numerical Linear Algebra with Applications*, 4(1):43–66, 1997.

[14] P. Chow, M. Cross, and K. Pericleous. A natural extension of the conventional finite volume method into polygonal unstructured meshes for cfd application. *Applied Mathematical Modelling*, 20:170–183, 1996.

[15] R.D. Cook. *Concepts and applications of finite element analysis*. Wiley, 2001.

[16] D. E. Davies and D. J. Salmond. Calculation of the volume of a general hexahedron for flow predictions. *AIAA Journal*, 23:954–956, 1985.

[17] G. Dhatt and G. Touzot. *The finite element method displayed, translation of: une présentation de la méthode des éléments finis*. Wiley-Inerscience, 1984.

[18] T.A. Driscoll and B. Fornberg. Interpolation in the limit of increasingly flat radial basis functions. *Computers and Mathematics with Applications*, 43(3):413–422, 2002.

[19] S. C. Eisenstat and H. F. Walker. Globally convergent inexact Newton methods. *SIAM J. Optimization*, 4(2):393–422, 1994.

[20] S. C. Eisenstat and H. F. Walker. Choosing the forcing terms in an inexact Newton method. *SIAM J. Sci. Comput.*, 17(1):16–32, 1996.

[21] J.F. Epperson. *An Introduction to Numerical Methods and Analysis*. Wiley, 2002.

[22] J. Erhel, K. Burrage, and B. Pohl. Restarted GMRES preconditioned by deflation. *Journal of Computational and Applied Mathematics*, 69:303–318, 1996.

[23] Liu F., Turner I.W., Anh V., and Su N. A two-dimensional finite volume method for transient simulation of time- and scale-dependent transport

in heterogeneous aquifer systems. *Journal of Applied Mathematics and Computing*, 11(1–2):215–241, 2003.

[24] Liu F., Turner I. W., and Anh V. An unstructured mesh finite volume method for modelling saltwater intrusion into coastal aquifers. *Korean Journal of Computational Mathematics*, 9(2):391–407, 2002.

[25] J. Fainberg and H.-J. Leister. Finite volume multigrid solver for thermo-elastic stress analysis in anisotropic materials. *Computer Methods in Applied Mechanics and Engineering*, 137:167–174, 1996.

[26] N. Fallah, C. Bailey, and M. Cross. Comparison of finite element and finite volume methods application in geometrically nonlinear stress analysis. *Applied Mathematical Modelling*, 24:439–455, 2000.

[27] W. J. Ferguson and I. W. Turner. A control volume finite element numerical simulation of the drying of spruce. *J. Computational Physics*, 125:59–70, 1996.

[28] B. Fornberg, T.A. Driscoll, G. Wright, and R. Charles. Observations on the behavior of radial basis function approximations near boundaries. *Computers & Mathematics with Applications*, 43(3–5):473–490, 2002.

[29] C. Franke and R. Schaback. Solving partial differential equations by collocation using radial basis functions. *Applied Mathematics and Computation*, 93(1):73–82, 1998.

[30] R. Franke. Scattered data interpolation: Tests of some methods. *Math. Comp.*, 38:181–200, 1982.

[31] Christophe Geuzaine and Jean-Franois Remacle. Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities. Available from: `http://www.geuz.org/gmsh/`.

[32] G. H. Golub and C. F. Van Loan. *Matrix Computations, 3rd edition*. Johns Hopkins University Press, 1996.

[33] Per Christian Hansen. The truncated SVD as a method for regularization. *BIT*, 27(4):534–553, 1987.

[34] Per Christian Hansen. Truncated singular value decomposition solutions to discrete ill-posed problems with ill-determined numerical rank. *SIAM J. Sci. Stat. Comput.*, 11(3):503–518, 1990.

[35] R.L. Hardy. Multiquadric equations of topography and other irregular surfaces. *J. Geophys. Res.*, 76:1905–1915, 1971.

[36] R.L. Hardy. Theory and applications of the multiquadric-biharmonic method. 20 years of discovery 1968–1988. *Computers Math. Applic.*, 19(8–9):105–210, 1990.

[37] J.M. Hill and J.N. Dewynne. *Heat conduction*. Oxford, 1987.

[38] Y.C. Hon, K.F. Cheung, X.Z. Mao, and E.J. Kansa. A multiquadric solution for shallow water equation. *ASCE J. Hydraulic Engineering*, 125(5):524–533, 1999.

[39] Y.C. Hon and X.Z. Mao. A radial basis function method for solving options pricing model. *Financial Engineering*, 8(1):31–50, 1999.

[40] M. Ilic and I.W. Turner. Approximately invariant subspaces. *ANZIAM*, 44 (E):C378–C399, 2003.

[41] P. A. Jayantha. *Accurate Finite Volume Methods for the Numerical Simulation of Transport in Highly Anisotropic Media*. PhD thesis, School of Mathematical Sciences, Queensland University of Technology, 2002.

[42] P. A Jayantha and Turner I. W. A comparision of gradient approximations for use in finite-volume computational models for two-dimensional diffusion equations. *Numerical Heat Transfer Part B: Fundamentals*, 40(5):367–390, 2001.

[43] P. A. Jayantha and Turner I. W. On the use of surface interpolation techniques in generalised finite volume strategies for simulating transport in highly anisotropic porous media. *ICRACM2001 conference proceedings - Journal of Computational and Applied Mathematics (North-Holland)*, 2002.

[44] P.A. Jayantha and Turner I.W. A second order finite volume technique for simulating transport in anisotropic media. *Internat. J. Numer. Methods Head Fluid Flow*, 13(1):31–56, 2003.

[45] P.A. Jayantha and I.W. Turner. Generalised finite volume strategies for simulating transport in strongly orthotropic porous media. *ANZIAM J.*, 44:C443–C463, 2003.

[46] P.A. Jayantha and I.W. Turner. A second order control-volume finite-element least-squares strategy for simulating diffusion in strongly anisotropic media. *Journal of Computational Mathematics*, 13(1):31–56, 2005.

[47] E. J. Kansa. Multiquadrics—a scattered data approximation scheme with applications to computational fluid dynamics. *Computers & Mathematics with Applications*, 19(8/9):127–161, 1990.

[48] E. J. Kansa and Y. C. Hon. Circumventing the ill-conditioning problem with multiquadric radial basis functions: Applications to elliptic partial differential equations. *Computers & Mathematics with Applications*, 39(7–8):123–137, 2000.

[49] C. T. Kelly. *Iterative Methods for Linear and Nonlinear Equations*. SIAM, 1995.

[50] C. T. Kelly. *Solving Nonlinear Equations with Newton's Method*. SIAM, 2003.

[51] D. A. Knoll and D. E. Keyes. Jacobian-free Newton-Krylov methods: A survey of approaches and applications. *J. Comp. Phys.*, 193:357–397, 2004.

[52] M.A. Kon and L. Plaskota. Neural networks, radial basis functions, and complexity. Available from: `http://citeseer.nj.nec.com/161674.html`.

[53] E. Larsson and B. Fornberg. Theoretical and computational aspects of multivariate interpolation with increasingly flat radial basis functions. *Comput. Math. Appl.*, 49:103–130, 2005.

[54] J. Li, Y. Chen, and D. Pepper. Radial basis function method for 1-D and 2-D groundwater contaminant transport modeling. *Computational Mechanics*, 32(1–2):10–15, 2003.

[55] D. Lowe. *Radial basis function networks,* in The handbook of brain theory and neural networks, pages 779–782. MIT Press, 1998.

[56] W.R. Madych and S.A. Nelson. Bounds on multivariate polynomials and exponential error estimates for multiquadric interpolation. *Journal of Approximation Theory*, 70:94–114, 1992.

[57] Zong min Wu and Robert Schaback. Local error estimates for radial basis function interpolation of scattered data. *IMA J. Numer. Anal.*, 13:13–27, 1993.

[58] R. B. Morgan. A restarted GMRES method augmented with eigenvectors. *SIAM J. Matrix Analysis and Applications*, 16(4):1154–1171, 1995.

[59] R. B. Morgan. GMRES with deflated restarting. *SIAM J. Sci. Comput.*, 24(1):20–37, 2002.

[60] T. J. Moroney. Templates for the iterative solution of linear systems using augmented Krylov subspace methods. *Honours Thesis, School of Mathematical Sciences, Queensland University of Technology*, 2002.

[61] K. W. Morton. *Numerical Solution of Convertion-Diffusion Problems.* Chapman & Hall, 1996.

[62] K. W. Morton, M. Stynes, and E.Süli. Analysis of a cell-vertex finite volume method for convection-diffusion problems. *Mathematics of computation*, 66:1389–1406, 1997.

[63] Bojan Niceno. EasyMesh: A two-dimensional quality mesh generator. Available from: `http://www-dinma.univ.trieste.it/nirftc/research/easymesh`.

[64] Carl F. Ollivier-Gooch and Michael Van Altena. A high-order accurate unstructured mesh finite-volume scheme for the advection-diffusion equation. *Journal of Computational Physics*, 181(2):729–752, 2002.

[65] S. V. Patankar. *Numerical Heat Transfer and Fluid Flow*. Hemisphere Publishing Corporation, 1980.

[66] M. Pernice and H. F. Walker. NITSOL: A Newton iterative solver for nonlinear systems. *SIAM J. Sci. Comput.*, 19(1):302–318, 1998.

[67] P. Perré and I. W. Turner. Transpore: A generic heat and mass transfer computational model for understanding and visualising the drying of porous media. *Intl. J. of Drying Technology* (Invited and Revised Paper from IDS98), 17(7):1273–1289, 1999.

[68] P. Perré and I. W. Turner. A heterogeneous wood drying computational model that accounts for material property variation across growth rings. *Chemical Engineering Journal*, 86:117–131, 2002.

[69] P.-O. Persson and G. Strang. A simple mesh generator in MATLAB. *SIAM Review*, 46(2):329–345, 2004.

[70] M. J. D. Powell. *The theory of radial basis function approximations in 1990,* in Advances in Numerical Analysis, Vol. II: Wavelets, Subdicision Algorithms and Radial Basis Functions, pages 105–210. Oxford University Press, 1990.

[71] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C*. Cambridge University Press, 1997.

[72] S. Rippa. An algorithm for selecting a good value for the parameter c in radial basis function interpolation. *Adv. Comput. Math.*, 11:193–210, 1999.

[73] Y. Saad. A flexible inner-outer preconditioned GMRES algorithm. *SIAM J. Sci. Comput.*, 14(2):461–469, 1993.

[74] Y. Saad. *Iterative Methods for Sparse Linear Systems*. PWS, 1996.

[75] Y. Saad and M. H. Shultz. GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Statist. Comput.*, 7(3):856–869, 1996.

[76] Yousef Saad and Henk A. van der Vorst. Iterative solution of linear systems in the 20th century. *J. Comput. Appl. Math.*, 123(1–2):1–33, 2000.

[77] R. Schaback. Error estimates and condition numbers for radial basis function interpolation. *Advances in Computational Mathematics 3*, pages 251–264, 1995.

[78] R. Schaback. Improved error bounds for scattered data interpolation by radial basis functions. *Math. Comp.*, 68(225):201–216, 1999.

[79] S. Song and M. Chen. Third order accurate large-particle finite volume method on unstructured triangular meshes. *SIAM J. Sci. Comput.*, 23(5):1456–1463, 2002.

[80] H. Sutter. The concurrency revolution. *C/C++ Users Journal*, 23(2), 2005.

[81] L. N. Trefethen and D. Bau. *Numerical Linear Algebra*. SIAM, 1997.

[82] S. L. Truscott. *A Heterogeneous Three-Dimensional Computational Model for Wood Drying.* PhD thesis, School of Mathematical Sciences, Queensland University of Technology, 2004.

[83] S. L. Truscott and I. W. Turner. An investigation of spatial and temporal weighting schemes for use in unstructured mesh control volume finite element methods. *ANZIAM*, 44:C759–C779, 2003.

[84] S. L. Truscott and I. W. Turner. An investigation of the accuracy of the control volume finite element method based on triangular prismatic elements for simulating diffusion in anisotropic media. *Numer. Heat Transfer, Part B: Fundamentals*, 46(3):243–268, 2004.

[85] E. Turkel. Accuracy of schemes with nonuniform meshes for compressible fluid flows. *J. Appl. Num. Math.*, 2(6):529–550, 1986.

[86] I. W. Turner and W. J. Ferguson. An unstructured mesh cell-centred control-volume method for simulating heat and mass transfer in porous media: Application to softwood drying, Part I: The isotropic model. *Applied Mathematical Modelling*, 19:654–667, 1995.

[87] I. W. Turner and W. J. Ferguson. An unstructured mesh cell-centred control-volume method for simulating heat and mass transfer in porous media: Application to softwood drying, Part II: The anisotropic model. *Applied Mathematical Modelling*, 19:668–674, 1995.

[88] I. W. Turner and P. Perré. *A synopsis of the strategies and efficient resolution techniques used for modelling and numerically simulating the drying process,* in Mathematical Modeling and Numerical Techniques in Drying Technology. Marcel Dekker, Inc., 1996.

[89] I. W. Turner and P. Perré. The use of implicit flux limiting schemes in the simulation of the drying process: A new maximum flow sensor applied to phase mobilities. *Applied Mathematical Modelling*, 25:513–540, 2001.

[90] I. W. Turner, J. R. Puiggali, and W. Jomaa. A numerical investigation of combined microwave and convective drying of a hygroscopic porous material: A study based on pine wood. *Trans. IChemE.*, 76(A):193–209, 1998.

[91] Eric W. Weisstein. Quadrilateral. *From MathWorld–A Wolfram Web Resource*, 1999. Available from: `http://mathworld.wolfram.com/Quadrilateral.html`.