Inaugural – Dissertation

zur

Erlangung der Doktorwürde

der

Naturwissenschaftlich–Mathematischen Gesamtfakultät

der

Ruprecht–Karls–Universität

Heidelberg

vorgelegt von

Diplom–Mathematiker Guido Kanschat

aus Krefeld

Tag der mündlichen Prüfung: 10. VI. 1996

Parallel and Adaptive Galerkin Methods for Radiative Transfer Problems

Gutachter: Prof. Dr. Rolf Rannacher
Prof. Dr. Rainer Wehrse

150 Das höchstenergische Licht, wie das der Sonne, des Phosphors in Lebensluft verbrennend, ist blendend und farblos. So kommt auch das Licht der Fixsterne meistens farblos zu uns. Dieses Licht aber durch ein auch nur wenig trübes Mittel gesehen, erscheint uns gelb. Nimmt die Trübe eines solchen Mittels zu oder wird seine Tiefe vermehrt, so sehen wir das Licht nach und nach eine gelbrote Farbe annehmen, die sich endlich bis zum Rubinroten steigert.

J. W. v. Goethe, Zur Farbenlehre

Contents

List of Figures

List of Tables

Notation

$B(\lambda, T)$	Planck's function (p. 15)
$\chi = \kappa + \sigma$	extinction coefficient (p. 18)
DG(0)	discontinuous Galerkin method of order zero (p. 26)
$\eta_X(h, u_h)$	a posteriori error estimate
$? = \partial\Omega$	boundary of the space domain Ω
$?_-$	inflow boundary (p. 18)
h	mesh parameter (p. 22)
$\hbar$	Planck's constant (p. 15)
$K \in \mathcal{T}$	triangulation cell (p. 22)
κ	absorption coefficient (p. 15)
λ	radiation wave–length (p. 15)
$\|.\|_\delta$	streamline–diffusion norm (p. 33)
$\|.\|_\Gamma$, $\|.\|_{\vartheta\Gamma}$	Norm of boundary values (p. 33)
ω_{kl}	scattering matrix weights (p. 54)
$P(\vartheta, \tilde{\vartheta})$	scattering phase function (p. 16)
$R(\tilde{\lambda}, \tilde{\vartheta}, \lambda, \vartheta)$	redistribution function (p. 15)
σ	scattering coefficient (p. 15)
Σ, Σ_ϑ	scattering operators (p. 17)
S^n	embedded unit sphere of $\mathbb{R}^{n+1}$ (p. 15)
T_ϑ	transport operator in direction ϑ (p. 17)
T	transport operator (p. 17)
$\mathcal{T}$	triangulation (p. 22)
$\vartheta \in S^n$	ordinate variable (p. 15)
$\vartheta \cdot \nabla_x$	derivative in direction of ϑ (p. 15)

Chapter 1

Introduction

This thesis is devoted to the development of an efficient algorithm to solve multi–dimensional radiative transfer problems for models including scattering. Aside from the necessity to save computation time, the amount of memory needed to store operators and data is a severe obstacle on existing computers. We therefore review the whole process from discretization over solution algorithms to implementation techniques to provide a means to solve astrophysical problems with reasonable resources of time and storage. Our main direction of improvement are the generation of more suitable grids and the implementation on supercomputers.

We investigate the radiative transfer equation in the form

$$\vartheta \cdot \nabla_x u - (\kappa + \sigma)u = \sigma \int_{S^2} P(\tilde{\vartheta}, \vartheta) u(x, \tilde{\vartheta})\, d\tilde{\vartheta} + \kappa B(\lambda, T(x)),$$

where κ, σ, P and B are positive functions described in detail in Chapter 2. The circum–stellar dust cloud of Figure 1.1 on the following page serves as a model problem for the development of our algorithm. A star is centered in a cloud of scattering and absorbing material. Around the star there is a hole where the dust has evaporated. The diameters of the star, the hole and the cloud behave typically like 1:10:100. Matter density is usually high at the inner edge of the cloud and diminishes to the outer parts. This figure shows the main features of astrophysical radiative transfer problems: huge differences in length scales and rapidly changing parameters. A solution method for these problems must be able to handle these difficulties automatically by

Figure 1.1: Dust enshrouded star

---

resolving the inhomogeneous parts to a sufficient accuracy without spending too much work in regions of smooth data and solution.

Opposed to the common idea of discretizing the derivative operator on the space domain $\Omega \subset \mathbb{R}^3$ and the integral operator in the ordinate domain, the unit sphere S^2, independently, we show a possibility to *combine* these into one Petrov–Galerkin approximation on $\Omega \times S^2$. Clearly, this is a crucial step towards a posteriori error control in the sense of Johnson et al. (cf. [12]). Additionally, it leads to a mixed finite element method enabling more general meshes than the standard tensor product splitting into $L^2(\Omega) \otimes L^2(S^2)$. This scheme includes the opportunity of using efficient parallelization techniques and obtaining highly accurate solutions. Furthermore, we replace the finite difference and quadrature schemes used by Auer, Wehrse and many others (cf. e. g. [2,3,35,36,38]) on Ω and S^2 by state of the art methods.

Instead of the widely used finite difference or finite element upwind schemes, which are of first order only, we apply the streamline diffusion finite element method to the spatial discretization. This is a stabilization of the standard finite element method developed specially for convection dominated equations by Hughes et al. in [22–24]. Thus it is best suited for the case of low scattering and shows convergence of second order on all our meshes and with

standard linear trial functions. To cover the scattering dominated case, we use a weighted form of this method. Since the streamline diffusion finite element method is a Petrov–Galerkin scheme, it provides the base for systematic a posteriori error analysis.

Subdividing the unit sphere of $\mathbb{R}^3$, we replace the usual longitude–latitude mesh by a more regular triangulation derived from regular polyhedra. This eliminates artifacts in the discrete solution due to the polar singularities of planar charts of S^2, so–called *ray effects*. Since it is more similar to an equidistribution, it reduces the number of required ordinates considerably. We show that application of the discrete Galerkin method with piecewise constant trial functions does not introduce more complexity due to multiple integral evaluations in computing finite element matrices.

Due to the high dimension of the computational domain the common approach of equidistant grids or tensor products of one–dimensional meshes leads to systems of untreatable size. Therefore, based on a posteriori error estimates, we derive an adaptive grid generation technique to avoid the main obstacle in solving radiative transfer problems, namely the huge amount of data due to high dimension of the computational domain and localized sharp edges in the resulting intensity distribution. Here adaptivity should be seen from two points of view: first, it provides reliability by limiting the actual error from above using error estimates. Compared to standard a priori convergence analysis, we can guarantee the solution value to be in a computed interval. We apply a new technique, that not only allows estimation in the energy norm like that proposed by Verfürth in [41], but is suitable for any error functional. The estimate is derived by analyzing a dual problem adapted to the specific error norm. As far as we know, this is the first time this technique is applied to radiative transfer and it shows that reliable solutions are actually computable even for such complex models. Second, it saves computing power by determining where and how far to enlarge the finite element space and where accuracy may be reduced. This may be achieved by increasing the polynomial degree of shape functions or subdividing grid cells. Since the analysis of the radiative transfer equation does not guarantee higher regularity of solutions, we choose the latter way. If the computation of the global behavior of the solution is required, grid refinement is controlled by a criterion, which combines similarity to the error estimate with minimal computational overhead. This method can generate well adapted grids in

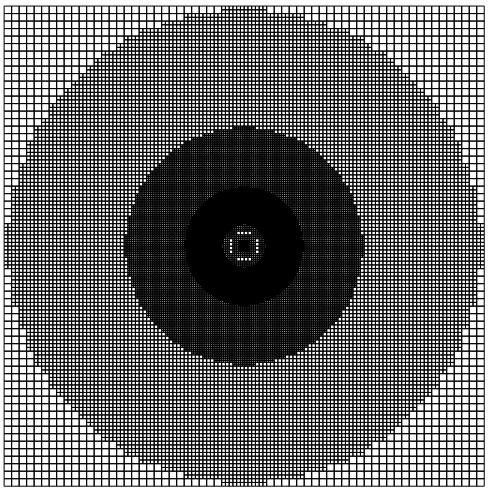


Figure 1.2: Adaptive mesh for configuration of Figure 1.1.

---

reasonable time. To compute single values extracted from the solution, we apply a new technique involving the solution to a dual problem. This method provides us with a mathematically strict and sharp error estimate suitable to control mesh refinement. We use *local refinement* to generate the adaptive grids. This allows us to avoid storing the matrices due to a convenient scaling property. As an important feature, this grid generation technique ensures the streamline diffusion method to be of optimal order even on extremely locally refined meshes. A typical mesh generated for the dust cloud configuration is displayed in Figure 1.2.

We determine the solution of the discrete linear algebraic system by a combination of non–symmetric Krylov space methods with adaptive multi–grid schemes. Krylov algorithms provide a nearly optimal inversion of the scattering operator in two steps. The discrete transport operator "pollutes" the eigenvalues of the scattering matrix, resulting in a convergence rate rapidly deteriorating on grid refinement. Our multi–grid preconditioning technique counters the pollution leading to convergence independent of refinement. Together with an efficient local smoothing iteration, this yields a solution technique of optimal complexity.

Since radiative transfer on a three–dimensional domain is at least a five–

dimensional problem, the amount of storage required is enormous even for adaptive methods with optimal grids. Therefore we implemented a parallel version of our solution algorithm to exploit the extra resources provided by modern multi–computers. Since well–resolved problems take some hours of computing time on a workstation and the monochromatic equation has to be solved several hundreds of times for one model, parallelization provides a way of reducing this time to a reasonable amount. The structure of the radiative transfer equation at a first glance offers several ways for parallelization, but it will be shown that they are of different value. For the well suited ordinate parallelization we will show satisfactory efficiency results on parallel computers.

Approaches to programming now in use in numerical mathematics are not suited for the development of scientific computing software. The problems and solution methods for "real world" simulation are much too complex and need the application of *software engineering* techniques. We develop a new approach to finite element codes using *object oriented programming*. It offers a means of verifying programs and producing reliable software on the level of simple functions as well as on the application level. This way, the handling of the usual hard tasks like adaptive grids and parallelization takes place in a well defined formalized environment, making prediction of correctness more reliable.

We begin this thesis by presenting the physical problem to be solved in Chapter 2. Then we give an overview over analytical results for the radiative transfer equation. The following chapter is devoted to variant discretization methods of the integral and differential operators and compares methods previously used with our discretization. In Chapter 4 we derive error estimates for Galerkin discretizations of the radiative transfer equation and propose various adaptive refinement strategies. The numerical solution of the discrete system is discussed in Chapter 5. We proceed with the presentation of parallelization methods for these solution algorithms. The Seventh Chapter is devoted to the software development techniques applied to implement the proposed algorithms. We conclude with two real applications from astrophysics and show the enhancements achieved by our methods.

Chapter 2

Physical Problem and Mathematical Model

We consider physical settings of a class with the following properties: In a domain Ω there is a thin gas, with negligible interaction between particles of this gas. Instead of this interaction, there are collisions of the particles with the atoms of a fixed matter distribution. The dynamics of this gas is described by Boltzmann's equation. Since the gas reacts with external matter only, the — generally quadratic — collision term of the equation reduces to a linear functional. The resulting equation for the gas density ϱ in phase space $\Omega \times \mathbb{R}^3$ has the form

$$v \cdot \nabla \varrho(x,v) + \sigma_t \varrho(x,v) = \int_{\mathbb{R}^3} R(v,v')\,\varrho(x,v')\,dv' + f \tag{2.1}$$

Remark 2.1 The approximation made here is opposite to that leading to Navier–Stokes–equations. There the quadratic interaction term forces all particles in a small space region to move into the same direction, resulting in a relation $v = v(x)$.

While for usual gases, the approximation leading to equation (2.1) is valid only for uninterestingly small densities, there are some kinds of particles, for which it is suitable even for moderate densities: neutrons, neutrinos and photons. While in the first case it is a good approximation even for neutron densities occurring in nuclear reactors, it is of very high accuracy for photons. This fact is due to the principle of wave superposition resulting from the

linearity of Maxwell's equations. The only exception is electro–magnetic radiation of very high energy, where photon–photon scattering occurs due to quantum effects. These can be neglected, since we are only interested in optical and infrared simulation.

Remark 2.2 Since we are particularly interested in electro-magnetic radiation, we will only discuss the photon transport problem further. Most results can be adapted easily to the neutron transport case.

2.1 Radiative Transfer

Obviously, using the velocity v makes no sense for photons, since they always travel with light speed $c \approx 3 \cdot 10^8 \frac{\mathrm{m}}{\mathrm{sec}}$. It is replaced by the photon momentum $2\pi\hbar/\lambda$ — $\hbar$ denoting Planck's constant and λ the wave–length — thus splitting the momentum space into directions in $\vartheta \in S^2$ and absolute values represented by λ. After scaling ($\hbar = 1$ and $c = 1$) and application of Kirchhoff's law, (2.1) may be written as *radiative transfer equation*

$$\begin{aligned} \vartheta \cdot \nabla_x u(x,\vartheta,\lambda) - \Big(\kappa(x,\vartheta,\lambda) + \sigma(x,\vartheta,\lambda)\Big) u(x,\vartheta,\lambda) \\ = \int\limits_{\mathbb{R}^+} \int\limits_{S^2} R(\tilde{\lambda},\tilde{\vartheta},\lambda,\vartheta) u(x,\tilde{\vartheta},\tilde{\lambda})\, d\tilde{\vartheta}\, d\tilde{\lambda} + \kappa(x,\vartheta,\lambda) B(\lambda, T(x)) \end{aligned} \tag{2.2}$$

Here the variables are chosen according to the macroscopic quantities described e. g. in [39]. κ and σ are the absorption and scattering coefficients respectively. Finally, B is Planck's radiation function for black bodies defined by

$$B(\lambda, T) = \frac{4\pi c\hbar}{\lambda^5} \frac{1}{e^{2\pi c\hbar/\lambda T} - 1} \tag{2.3}$$

We scale all lengths by a typical length for the model, e. g. the diameter of the whole domain. Then, all coefficients are pure numbers and we can define u and B dimensionless too (cf. e. g. [29]).

Often, the frequency coupling in the redistribution function is negligible. This leads to the monochromatic radiative transfer equation

$$\boxed{\vartheta \cdot \nabla_x u - (\kappa + \sigma) u = \sigma \int\limits_{S^2} P(\tilde{\vartheta}, \vartheta) u(x, \tilde{\vartheta})\, d\tilde{\vartheta} + \kappa B(\lambda, T(x))} \tag{2.4}$$

It is this form, we are mostly concerned with in our work. Apart from its own physical relevance for visual light it serves as example and as solution step for more complex models.

The physical meaning of the coefficient functions introduces the following properties:

$$\kappa \geqslant 0 \tag{2.5}$$
$$\sigma \geqslant 0 \tag{2.6}$$
$$B > 0. \tag{2.7}$$

The integral kernel P fulfills

$$\iint\limits_{S^2} P(\tilde{\vartheta}, \vartheta)\, d\tilde{\vartheta}\, d\vartheta = 1 \tag{2.8}$$
$$P(\vartheta, \tilde{\vartheta}) = P(-\tilde{\vartheta}, -\vartheta) \tag{2.9}$$
$$P(\vartheta, \tilde{\vartheta}) = P(\tilde{\vartheta}, \vartheta). \tag{2.10}$$

Remark 2.3 While condition (2.8) can be fulfilled by an appropriate choice of σ, (2.9) and (2.10) are in many important cases ensured by

$$P(\tilde{\vartheta}, \vartheta) = f(\alpha) \qquad \text{with} \qquad \alpha = \sphericalangle(\vartheta, \tilde{\vartheta}),$$

namely

$$f(\alpha) = \begin{cases} \frac{1}{\mu(S^2)} & \text{isotropic scattering} \\ \frac{3}{16\pi}\left(1 + \cos^2 \alpha\right) & \text{Thomson and Rayleigh scattering.} \end{cases}$$

2.1.1 Energy Transfer by Radiation

The evaluation of observer data from accretion discs and circum–stellar clouds requires a more complex radiative transfer model. Here, the heating of absorbing matter by the radiation field is taken into account, leading

to the following system of equations:

$$\begin{aligned}\vartheta \cdot \nabla_x u(x,\vartheta,\lambda) = & \\ & - \Big(\kappa(x,\vartheta,\lambda) + \sigma(x,\vartheta,\lambda)\Big) u(x,\vartheta,\lambda) \\ & + \int\limits_{\mathbb{R}^+} \int\limits_{S^2} R(\tilde{\lambda},\tilde{\vartheta},\lambda,\vartheta) u(x,\tilde{\vartheta},\tilde{\lambda})\, d\tilde{\vartheta}\, d\tilde{\lambda} \\ & + \kappa(x,\vartheta,\lambda,T) B(\lambda,T(x))\end{aligned} \tag{2.11}$$

$$\int\limits_{\mathbb{R}^+} \kappa(x,\vartheta,\lambda) B(\lambda,T(x))\, d\lambda = \int\limits_{\mathbb{R}^+} \int\limits_{S^2} \kappa(x,\vartheta,\lambda) u(x,\vartheta,\lambda)\, d\vartheta\, d\lambda \tag{2.12}$$

$$\forall (x,\vartheta,\lambda) \in \Omega \times S^2 \times \mathbb{R}^+.$$

This model is a consistent description of the energy distribution in a radiation heated stationary gas or dust cloud. One of its applications is the modeling of infrared radiation from those clouds. The occurrence of this radiation is essentially due to the energy balance equation (2.12).

The solvability of system (2.11–2.12) is ensured by analytical results in [20]. For the intended operator splitting approach, each iteration step will need the solution of several equations of type (2.4). Therefore, the efficient solution of (2.4) is a crucial step.

If a consistent radiation hydrodynamic model is desired, (2.11–2.12) has to be completed by the equation of balance of momentum coupling the first moments of the Boltzmann equations for photons and matter.

2.2 Analytical Results

In the following we concentrate only on the monochromatic radiative transfer equation (2.4). First, we want to introduce some operators to simplify notation. Let

$$T_\vartheta = \vartheta \cdot \nabla_x \qquad T = \bigotimes_\vartheta T_\vartheta$$

$$\Sigma_\vartheta = \sigma \int P(\tilde{\vartheta},\vartheta)\, d\tilde{\vartheta} \qquad \Sigma = \bigotimes_\vartheta \Sigma_\vartheta.$$

With these abbreviations, (2.4) reads

$$Tu + \chi u - \Sigma u = \kappa f \tag{2.13}$$

The equation given in the interior of the domain, , we have to prescribe boundary conditions before we can attempt to solve the radiative transfer equation. With $\mathfrak{n}_\Gamma(x)$ the outer normal to the boundary ? of Ω at x, we define two classes of boundary conditions: the inflow condition

$$u(x,\vartheta) = g(x,\vartheta) \qquad \text{on } ?_-, \tag{2.14}$$

where

$$?_- = \Big\{(x,\vartheta) \mid \mathfrak{n}_\Gamma(x)\cdot\vartheta < 0\Big\} \tag{2.15}$$

and reflecting boundary conditions

$$u(x,\vartheta) = \int\limits_{\mathfrak{n}_\Gamma(x)\cdot\tilde{\vartheta}>0} P_\Gamma(\tilde{\vartheta},\vartheta)u(x,\tilde{\vartheta})\,d\tilde{\vartheta} \qquad \forall\ \mathfrak{n}_\Gamma(x)\cdot\vartheta < 0. \tag{2.16}$$

Linear combinations of these conditions are possible to model translucent boundaries.

2.2.1 Investigation of Coefficients

The behavior of solutions to (2.13) depends on the parameters κ and strongly on σ. The numerical solution should be able to reflect these different solution properties and we shortly discuss the influence of the coefficients.

Considering the astrophysical problems we are primarily interested in, these coefficients vary over the space domain between 0 and 10^4. The simplest case clearly is $\kappa \gg 1$ and $\kappa \gg \sigma$: Here (2.13) converges to a singularly perturbed version of $u = f$. In the following analysis we therefore assume $\kappa \approx 0$, having in mind, that taking larger κ always simplifies the solution.

We examine the two extremes $\sigma = 0$ and $\sigma \to \infty$. In the first case, the radiative transfer equation decouples, leaving a set of pure convection equations to be solved independently.

For large values of σ, there exists an analysis done by Borysiewicz et al. in [8, 9]. We give a short summary of their results, which they obtain by using a least squares formulation of (2.4).

Lemma 2.1 *Let assumptions* (2.5) *to* (2.10) *be fulfilled and let* $\kappa > 0$. *Then equation* (2.13) *has a unique solution* u *with*

$$\|u\| \leqslant c \tag{2.17}$$

$$u = \sum_0^\infty \frac{1}{\sigma^\nu} u_\nu \tag{2.18}$$

where c *is independent of* σ *and* u_ν *solves*

$$\begin{aligned} \left\langle \frac{1}{\sigma} \nabla u_0, \nabla \varphi \right\rangle + \left\langle \kappa u_0, \varphi \right\rangle &= 0 \\ \left\langle \Sigma u_1, \varphi \right\rangle &= \left\langle \kappa f, \varphi \right\rangle - a\,(u_0, \varphi) \\ \left\langle \Sigma u_{\nu+1}, \varphi \right\rangle &= -a\,(u_\nu, \varphi) \quad \nu = 1, 2, \ldots \end{aligned} \tag{2.19}$$

with

$$a\,(u, v) = \left\langle (\chi - \Sigma)^{-1} \vartheta \cdot \nabla_x u, \vartheta \cdot \nabla_x v \right\rangle + \left\langle \chi u, v \right\rangle.$$

This result shows, that for large values of σ the solution of radiative transfer problems becomes independent of the ordinate variable and is well approximated by solutions to Helmholtz's equation.

2.2.2 Regularity

In the case of zero scattering, we have only regularity of the corresponding convection equations. That means that we gain one order of derivative in the transport direction, but the solution is only as regular as the data in crosswind direction.

It has been shown (cf. e. g. Führer and Rannacher in [17]) that the intensity u integrated over S^2 solves the weakly singular Fredholm integral equation

$$\begin{aligned} v - \Xi v &= r \\ \big(\Xi v\big)(x) &= \int_\Omega \exp\left(- \int_0^{|y-x|} \chi(x + t \tfrac{y-x}{|y-x|}\, dt \right) \frac{v(y)\sigma(y)}{|y-x|^2}\, dy \end{aligned} \tag{2.20}$$

Pitkäranta investigated the regularity of solutions to equations like (2.20) in [31, 32]. His main result is the following:

Lemma 2.2 *Let v be solution to equation* (2.20)*. Then $v \notin H^{3/2}(\Omega)$ regardless of the regularity of the data.*

This result is due to inconsistencies of the model near the boundary. In the scattering dominated case, we may use the expansion of lemma 2.1 to obtain higher regularity in the interior of Ω.

Chapter 3

Discretization

Boltzmann equations in theoretical physics are often solved by Monte–Carlo–methods. These schemes converge fast in the case of long mean free path lengths (σ small), but they have two disadvantages. Their convergence is very slow for optically thick ($\sigma \gg 1$) media and error control is only asymptotic with unknown constants. Therefore we apply a discretization method to cover a wider range of applications.

Due to the complex structure of the radiative transfer equation, there are numerous possibilities for its discretization. They can be divided into two main classes:

1. separate discretizations for the integral and differential operator and
2. complete discretizations for the whole equation.

The first class uses well known techniques for each part of the equation. The obvious advantage is the re-usability of well–proved methods by combining them into a tensor product discretization. We present some methods used in our work. On the other hand, convergence must be proved via a semi–discretization analysis (cf. [26]).

For the integral as well as the differential part of the radiative transfer equation we distinguish between Galerkin and non–Galerkin discretization methods. While the latter are widely used in physics and engineering, they do

not allow optimal error estimates obtained for Galerkin methods by exploiting additional information on the error, so called Galerkin orthogonality.

For the discretization of the integral part we show equivalence between a standard collocation and a discontinuous finite element scheme. This enables the application of superconvergence results to the collocation method as well as easy implementation of the finite element method.

In the literature many methods have been proposed to discretize the spatial transport operator (for a review cf. Führer [14,15]). First, there is the method of short characteristics proposed by Olson et al. [30]. Standard Galerkin methods for convection problems are proven to be stable only in L^2 and show oscillating behavior for discontinuous solutions. Therefore,the so called finite element upwind scheme was used by Turek et al. in [36–38]. Since standard upwind schemes are only of first order accuracy, we decided to apply the streamline diffusion method, which is of order 3/2 to 2 depending on the mesh structure (cf. [44]).

The approach of directly discretizing the whole radiative transfer equation offers the possibility of a theoretical justification and has to be considered for any approach with error control. Its implementation is — for special cases — equivalent to a semi–discretization and thus does not require extra effort in evaluating integrals. Additionally, theoretical analysis avoids semi–discrete problems and symmetry conditions for the angular discretization as required by Asadzadeh (cf. [1]).

3.1 Basic Finite Element Results

Before we cite the most important convergence results for the finite element method, we give a short overview over our notation. Given a domain $\Omega \in \mathbb{R}^n$, let $\mathcal{T}$ be a subdivision of Ω consisting of cells $K \in \mathcal{T}$. These triangulations are regular in the sense of Ciarlet [10] with the exception, that we allow one hanging node on the edge of a cell. The mesh parameter h is a piecewise constant function on Ω given by the diameter of the cell around each point. For globally refined grids, h may also denote the maximum diameter of all cells.

The cells are defined by the mapping of a reference cell K_0 into the domain Ω, in particular

- linear mapping of the unit simplex S_0^d in $\mathbb{R}^2$ and $\mathbb{R}^3$,
- bilinear mapping of the unit square Q_0^2 onto a quadrangle,
- trilinear mapping of the unit cube Q_0^3 onto a hexahedron with possibly curved surfaces and
- bilinear mapping of the unit prism $S_0^2 \times [-1, 1]$ onto an arbitrary prism in $\mathbb{R}^3$.

On such a grid we define spaces of piecewise polynomial finite element functions φ_h continuous at the cell edges. These functions are images of linear, bilinear and trilinear polynomials on the reference cell K_0 under a transformation of the same type. In the following we will call them to be in e. g. $Q_1(K)$.

Convergence of the finite element method is usually proved in two steps. First, we show, that the error can be bounded in terms of approximation properties of the finite element space V_h.

Lemma 3.1 *Let $a(u, v)$ be a continuous elliptic bilinear form on the function space V with*

$$|a(u,v)| \leqslant L\|u\|\|v\| \qquad \text{and} \qquad a(u,u) \geqslant \nu\|u\|^2$$

for all $u, v \in V$. Let u and u_h be solutions of the equations

$$\begin{aligned} a(u,v) &= \left\langle f, v \right\rangle \quad \forall\, v \in V \\ a(u_h,v) &= \left\langle f, v \right\rangle \quad \forall\, v \in V_h. \end{aligned}$$

Then the discretization error of the finite element method is limited by

$$\|u - u_h\| \leqslant \frac{L}{\nu} \inf_{w \in V_h} \|u - w\|.$$

For a proof see [10], pp. 104 f.

The second step is to state the approximation properties of finite element spaces. Having in mind the usual piecewise polynomials on $\mathcal{T}$, the following lemma is the crucial step:

Lemma 3.2 (Bramble–Hilbert) *Let Ω be a Lipschitz bounded domain. For some $k \in \mathbb{N}_0$ and $p \in [0, \infty]$ let f be a continuous linear form on $H^{k+1,p}(\Omega)$ with*

$$f(\pi) = 0 \qquad \forall\, \pi \in P_k(\Omega).$$

Then there is a constant $C = C(\Omega)$ such that

$$|f(v)| \leqslant C\, \|f\|_{H^{k+1,p'}}\, |v|_{H^{k+1,p}} \quad \forall\, v \in H^{k+1,p}.$$

A proof of this lemma may be found in [10] p. 192.

The usefulness of this rather abstract lemma is given by

Corollary 3.3 *Lemma 3.2 applied to projections on polynomial spaces of degree $k-1$ and $\Omega = K \in \mathcal{T}$ allows to estimate the interpolation error for $u \in H^k$ by*

$$\|u - \Pi_h u\|_{L^2} \leqslant C h^k |u|_{H^k}$$

Furthermore, there are additional estimates for weaker norms

$$\|u - \Pi_h u\|_{H^\nu} \leqslant C h^{\mu-\nu} \|u\|_{H^\mu} \quad \textit{for} \quad -k \leqslant \nu \leqslant 0 \leqslant \mu \leqslant k$$

3.2 The Integral Operator

This section is devoted to the discretization of Fredholm integral equations of second kind (an overview of the numerical treatment of these problems may be found in [21]). The prototype of the equations considered is

$$\chi u - \Sigma u = f \quad \text{in } G \tag{3.1}$$

with

$$\big(\Sigma u\big)(\vartheta) = \sigma \int_G P(\tilde{\vartheta}, \vartheta) u(\tilde{\vartheta}) d\tilde{\vartheta}.$$

Due to the physical assumptions of the second chapter, $\chi \geqslant \sigma$ and conditions (2.8) to (2.10) apply to P.

3.2.1 Nyström's Method

A classical method for discretizing integral operators is the application of numeric quadrature, called Nyström's method. Any Newton–Coates– or Gauß–formula may be used. For our two–dimensional computations — i. e. one ordinate dimension — we use the iterated midpoint rule on the unit circle with m equidistributed points:

$$\begin{aligned} \vartheta_i &= \frac{2\pi}{m} i \\ w_i &= \frac{1}{m} \end{aligned} \tag{3.2}$$

This formula allows an efficient implementation and is of high accuracy:

Lemma 3.4 *Let $u \in H^k(S^1)$ be a solution to (3.1) with $G = S^1$ and right hand side $f \in H^k(S^1)$. The solution to the equation discretized by formula 3.2 shall be denoted by u_h. Then the error admits*

$$\|u - u_h\| \leqslant C h^k \|u\|_{H^k}.$$

Proof: Applying the Euler–MacLaurin summation formula we know: The order of the interpolation error for equidistributed points and periodic functions is only dependent on the regularity of the integrand. By Lemma 3.1, the error estimate follows.

For integration over S^2 the first question arising is the construction of (nearly) equidistributed quadrature points to avoid discretization artifacts. It is well known, that there are only five regular polyhedra. State of the art before our investigation was using the parameterization of S^2 over $[-\pi, \pi[\times[0, \pi[$. Regular subdivision of the parameter space results in two distinctive directions at the poles. This causes a non–physical symmetry axis in the solution and deteriorates convergence, since the cells near the poles have degenerate angles.

Our approach uses the triangles of S^2 obtained by successive subdivision of an icosahedron (see Figure 3.1 on the next page). Quadrature points are the cell centers projected on S^2 and spherical cell volumes serve as weights. Since S^2 lacks the periodicity of S^1, we cannot apply the interpolation estimate of Lemma 3.4. As will be shown later in Theorem 3.7, this discretization is of

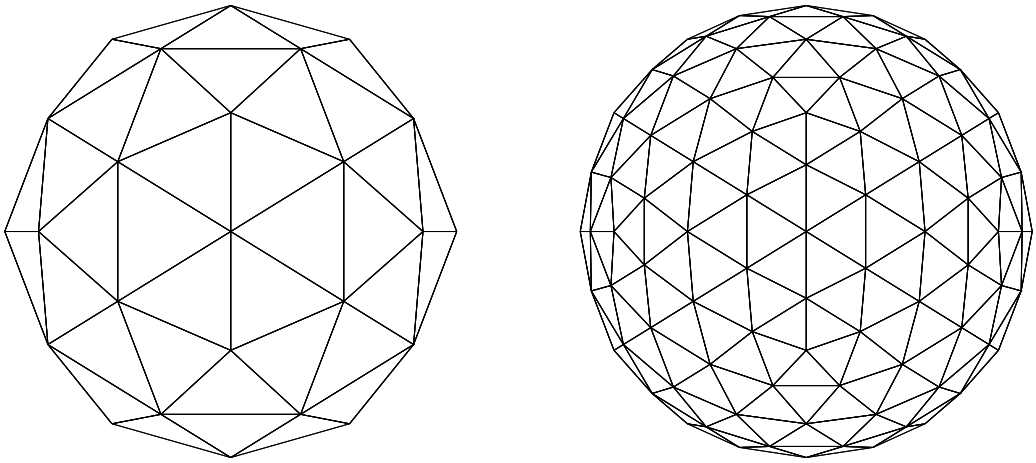


Figure 3.1: Refined icosahedron (80 and 320 cells)

second order. If we consider the number of cells lying on a great circle to be a measure for the quality of such a subdivision, we can compare our triangulation with the longitude–latitude mesh. Using for instance 40 cells on a great circle, we need a total number of 320 cells, whereas the parameterization approach needs 800.

3.2.2 Galerkin Discretization

Having in mind the development of more refined error estimates, it is advisable to look at Galerkin–discretizations for the integral operator. The integral equation (3.1) will then be used in its weak formulation: Search $u \in X = L^2(G)$ such that

$$\int_G \chi u(\vartheta)\varphi(\vartheta)d\vartheta - \iint_G \sigma P(\tilde{\vartheta},\vartheta)u(\tilde{\vartheta})\varphi(\vartheta)d\tilde{\vartheta}\,d\vartheta = \int_G f(\vartheta)\varphi(\vartheta)d\vartheta \quad \forall\,\varphi \in X. \tag{3.3}$$

Replacing X by some finite element space X_h, the computation of matrix elements requires evaluation of a double integral over each cell. For arbitrary trial functions, this integration might result in enormous computational effort. Applying discontinuous Galerkin method with piecewise constant polynomials (DG(0)–method) avoids this problem for radiative transfer due to:

Lemma 3.5 *For $f(x,\vartheta)$ independent of ϑ and smooth in x, the integration of operator $\vartheta\cdot\nabla_x$ may be replaced by*

$$\int_G \vartheta\cdot\nabla_x f(x)d\vartheta = \left(\int \vartheta\cdot\nabla_x d\vartheta\right) f(x) \quad \forall\, x\in\Omega.$$

Proof: By decomposing the derivative in direction ϑ into its components and applying, that $\partial_i f$ is constant considering the integration variable.

■

This property leads to the choice of our finite element space

$$X_h = \left\{ v \;\middle|\; v|_K = c_K, \forall K \in \mathcal{T} \right\}, \tag{3.4}$$

where $\mathcal{T}$ is a subdivision of S^2. We obtain $\mathcal{T}$ by projecting polyhedra as in Figure 3.1 onto the unit sphere. Independent of the mesh width h, the interior angles α in these triangulations are limited by

$$54^\circ \leqslant \alpha \leqslant 72^\circ$$

and areas of the triangles differ at most by a factor of two. So this triangulation is nearly uniform and does not produce artifacts known for the longitude–latitude meshes.

We are now ready to prove convergence of our discretization.

Theorem 3.6 *The physical assumptions of Chapter 2 given, the DG(0)–discretization of* (3.1) *is of first order and the error is limited by*

$$\|u-u_h\|_{L^2} \leqslant C_i\frac{\chi}{\kappa}h\|u\|_{H^1}$$

with C_i the interpolation constant of Bramble–Hilbert–lemma.

Proof: To apply Cea's lemma we show that the operator $\chi Id-\Sigma$ is bounded and elliptic. Considering the simplified integral equation (3.1) derived from the radiative transfer equation (2.13) by neglecting transport, conditions (2.5), (2.6) and (2.8) imply

$$\begin{aligned}\nu &\geqslant \kappa\\ L &\leqslant \kappa+\sigma\end{aligned}$$

We may apply now Cea's lemma to obtain

$$\|u - u_h\| \leqslant \frac{\kappa + \sigma}{\kappa} \|u - \Pi_h u\|$$

Using Bramble–Hilbert–lemma for piecewise constant functions, we conclude the proof. ■

Since this result is not sufficient compared to numerical results, we continue with a refined convergence analysis.

3.2.3 Superconvergence

If we are only interested in the error at the centers of mass of the triangles, e. g. to prove convergence of the midpoint rule, we may apply the following superconvergence result:

Theorem 3.7 *Assume the solution u of (3.1) to be in $H^{2,\infty}(G)$. Then the error at the centers of mass ξ_i is limited by*

$$u(\xi_i) - u_h(\xi_i) \leqslant Ch^2 \|u\|_{H^{2,\infty}}$$

Proof: From

$$\Big\langle (\chi - \Sigma)u, v_h \Big\rangle = \Big\langle f, v_h \Big\rangle \qquad \Big\langle (\chi - \Sigma)u_h, v_h \Big\rangle = \Big\langle f, v_h \Big\rangle$$

for all $v_h \in X_h$, we deduce

$$\begin{aligned}
\chi \Big\langle u - u_h, v_h \Big\rangle &= \Big\langle \Sigma(u - u_h), v_h \Big\rangle && \forall\ v_h \in X_h \\
\chi \Big\langle \Pi_h u - u_h, v \Big\rangle &= \Big\langle \Pi_h \Sigma(u - u_h), v \Big\rangle && \forall\ v \in X \\
\Pi_h u(\vartheta) - u_h(\vartheta) &= \frac{1}{\chi} \Big(\Pi_h \Sigma(u - u_h) \Big)(\vartheta) && \text{a.e. in } G,
\end{aligned}$$

where Π_h is the L^2–orthogonal projection from X on X_h. This leads to the error representation

$$u - u_h = u - \Pi_h u + \Pi_h u - u_h = \Big\{ u - \Pi_h u \Big\} + \Big\{ \frac{1}{\chi} \Pi_h \Sigma(u - u_h) \Big\}.$$

We conclude the proof by the following lemmas, where we show quadratic convergence of both parts of the sum.

Lemma 3.8 *Let u and u_h be solutions of the continuous and discrete integral equation* (3.1)*, respectively. Then the error $e = u - u_h$ admits the estimate*

$$\|e\|_{H^{-1}} \leqslant Ch^2\|u\|_{H^1}$$

Proof: Using the self–adjointness of Σ, we introduce the solution z of the dual problem

$$\chi z - \Sigma z = w \tag{3.5}$$

for $w \in H^1$. We get for all $w \in H^1$, using Galerkin orthogonality and the stability of the dual problem

$$\begin{aligned}\left\langle e, w\right\rangle &= \left\langle e, \chi z - \Sigma z\right\rangle = \left\langle \chi e - \Sigma e, z\right\rangle \\ &= \left\langle \chi e - \Sigma e, z - \Pi_h z\right\rangle \leqslant \|e\|_{L^2}\,\|\chi - \Sigma\|_{L(L^2,L^2)}\|z - \Pi_h z\|_{L^2} \\ &\leqslant \|e\|_{L^2}\,\|\chi - \Sigma\|_{L(L^2,L^2)}\,h\,\|z\|_{H^1} \\ &\leqslant \frac{\chi}{\kappa} C_i h\|e\|_{L^2}\|w\|_{H^1}.\end{aligned}$$

Note that we used stability of the dual problem from H^1 to H^1, which we get by differentiating the whole dual problem (3.5) and applying L^2–stability to each derivative. We conclude the proof by applying the definition of the H^{-1}–norm and the L^2–estimate of lemma 3.6:

$$\|e\|_{H^{-1}} = \max_{w\in H^1} \frac{\left\langle e, w\right\rangle}{\|w\|_{H^1}} \leqslant \left(\frac{\chi}{\kappa}\right)^2 C_i\prime h^2\|u\|_{H^1}.$$

Lemma 3.9 *Let u and u_h be solutions of the continuous and discrete integral equation* (3.1)*, respectively. Then the error admits the estimate*

$$\|\frac{1}{\chi}\Pi_h\Sigma(u - u_h)\|_{L^\infty} \leqslant Ch^2\|u\|_{H^1}$$

Proof: For sufficiently smooth phase function P the scattering operator Σ describes a continuous mapping of H^{-1} to L^∞ (cf. [21] pp. 129 ff.). We apply the previous lemma and use the obvious relation $\|\Pi_h u\|_\infty \leqslant \|u\|_\infty$ to prove the lemma.

Lemma 3.10 *Let Π_h be the L^2-projection on the space of piecewise constant finite elements, $u \in H^{2,\infty}(G)$. Then, the error at the center of mass of each grid cell obeys*

$$u(\vartheta_c) - (\Pi_h u)(\vartheta_c) \leqslant Ch^2 \|u\|_{H^{2,\infty}}.$$

Proof: Taylor expansion around ϑ_c and the vanishing of $\Pi_h f - f(\vartheta_c)$ for linear functions f yield the result.

3.3 Discretization of Transport Problems

In this section we will discuss discretization methods for the transport problem

$$\begin{aligned} \vartheta \cdot \nabla_x u + \chi u &= f \qquad \text{in} \quad \Omega \\ u &= g \qquad \text{on} \quad ?_{\underline{\ }}^{\vartheta} \end{aligned} \tag{3.6}$$

This equation models the differential part of the radiative transfer equation. It has also significance in regions with zero scattering ($\sigma = 0$), where each specific intensity satisfies (3.6).

3.3.1 Upwind Techniques

Several finite difference and finite element methods for the discretization of the transport operators have been proposed. First, there is the *short characteristic* scheme (see Figure 3.2 on the facing page) proposed by Olson et al. in [30]. It is applied in astrophysics by explicitly inverting the transport operators for each ordinate cell by cell. All coefficients and the right hand side are assumed to be piecewise constant, so the transport is solvable analytically along the line γ. In astrophysical methods, one usually applies the analytical solution on each cell, which is an exponential function. Since the start value at P_5 is interpolated — usually linearly — from its neighbors, this exact inversion is too expensive and should be replaced by an finite difference discretization. Equation 3.6 is discretized by

$$\frac{u(P_3) - u(P_5)}{h} + \chi u(P_3) = f(P_3)$$

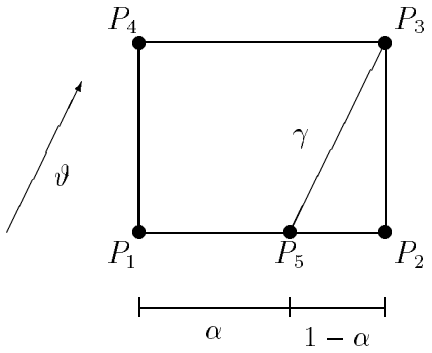


Figure 3.2: The method of short characteristics

---

which may be resolved to the update formula

$$u_3 = \frac{1}{1+\chi h}\Big(hf_3 + (1-\alpha)u_1 + \alpha u_2\Big) \tag{3.7}$$

where h is the length of γ.

This leads to a very sparse matrix with three entries per line for two–dimensional problems (five in 3D). The solution of the discrete system starts at the inflow boundary, where the function values at P_1 and P_2 are known for each cell and proceeds from layer to layer through the domain. This corresponds to the inversion of a triangular matrix if we suppose downwind ordering of mesh points.

Another possibility for transport equations is the imitation of backward differences by using sophisticated integration formulæ to compute stiffness matrices, the finite element upwind methods. These schemes usually correspond to a finite volume discretization. They share the disadvantage of insufficient error control and the finite element version often induces a rather complex matrix generation.

3.3.2 The Streamline Diffusion Method

Both transport discretization methods in the previous subsection have significant drawbacks: They are usually of first order accuracy only and they do

not possess Galerkin orthogonality to prove higher accuracy. On the other hand, standard Galerkin discretizations of higher order show oscillations for discontinuous solutions. This behavior corresponds to the fact that standard Galerkin is stable only in L^2.

The streamline diffusion finite element method is a Petrov–Galerkin scheme for (3.6), where test functions φ are replaced by $\varphi + \delta\vartheta\cdot\nabla_x\varphi$ with a suitably chosen small parameter δ. This corresponds to adding small diffusion in transport (streamline) direction only. Since this is done in a consistent way, there is no loss of accuracy.

The weak formulation of (3.6) looks then like

$$\left\langle \vartheta\cdot\nabla_x u + \chi u, \varphi + \delta\vartheta\cdot\nabla_x\varphi \right\rangle = \left\langle f, \varphi + \delta\vartheta\cdot\nabla_x\varphi \right\rangle \tag{3.8}$$

δ has to be chosen according to the size of χ and is proportional to the local mesh width to obtain the optimal balance between accuracy and stability.

We apply the streamline diffusion method to bilinear trial functions, i. e. u and φ are from the space

$$V_h = \left\{ v \in C^0(\Omega) \;\middle|\; v|_K \in Q_1(K),\ \forall K \in \mathcal{T} \right\} \tag{3.9}$$

as defined in the first section of this chapter.

Remark 3.1 The streamline diffusion method adds a weighted form of the least–squares discretization

$$\left\langle \vartheta\cdot\nabla_x u + \chi u, -\vartheta\cdot\nabla_x\varphi + \chi\varphi \right\rangle = \left\langle f, -\vartheta\cdot\nabla_x\varphi + \chi\varphi \right\rangle \tag{3.10}$$

to the standard scheme. The differential operator in equation (3.10) is of second order in the direction ϑ. Therefore, the physical boundary condition of (3.6) has to be modified carefully to preserve the physical meaning. By lemma 3.12 below follows that the streamline diffusion scheme does not have this drawback. A solution method for the radiative transfer equation with a least–squares discretization is proposed by Ressel in [33]

In the following paragraphs we list some results which led to our choice to use streamline diffusion method.

The approximation order of the streamline diffusion method for linear finite elements has been proved to be $\frac{3}{2}$ on general grids, but there is evidence

for second order convergence on nearly all computationally interesting grids. Unfortunately, a mathematical proof for second order is available only on Cartesian grids. In [44] the construction of grids with lesser convergence rate is discussed. Those grids are constructed purposely to show optimality of the theoretical results and do not occur in real calculations due to our grid generation techniques.

Instead of the L^2–norm we define the stronger problem adapted norm

$$\|u\|_\delta^2 = \|\sqrt{\delta}\vartheta\cdot\nabla_x u\|_2^2 + \|\sqrt{\kappa}u\|_2^2 + \|u\|_\Gamma^2 \tag{3.11}$$

where

$$\|u\|_\Gamma^2 = \|u\|_{\vartheta,\Gamma}^2 = \frac{1}{2}\int_\Gamma u^2(x)\,\mathfrak{n}_\Gamma(x)\cdot\vartheta\,dx$$

and show the stability estimate

Lemma 3.11 *Let $\mathcal{L}(u,v)$ be the bilinear operator defined in (3.8) with a function $\kappa \in C_0^1(\Omega)$. Then the estimate*

$$\mathcal{L}(u,u) \geqslant C\|u\|_\delta$$

holds.

Proof: Evaluation of the bilinear form and partial integration yields

$$\begin{aligned}\mathcal{L}(u,u) &= \|\sqrt{\delta}\vartheta\cdot\nabla_x u\|^2 + \|\sqrt{\kappa}u\|^2 + \left\langle \delta\vartheta\cdot\nabla_x u, u\right\rangle + \left\langle \kappa u, \delta\vartheta\cdot\nabla_x u\right\rangle \\ &\geqslant \|\sqrt{\delta}\vartheta\cdot\nabla_x u\|^2 + \|\sqrt{\kappa}u\|^2 + \|u\|_\Gamma + \|\sqrt{\kappa}u\|_\Gamma - \frac{1}{2}\left\langle \delta\vartheta\cdot\nabla_x\kappa u, u\right\rangle \\ &= \|\sqrt{\delta}\vartheta\cdot\nabla_x u\|^2 + \left\langle \left(\kappa - \tfrac{\delta\vartheta\nabla_x\kappa}{2}\right)u, u\right\rangle + \|u\|_\Gamma\end{aligned}$$

If we choose

$$\delta < \frac{2\kappa}{\vartheta\cdot\nabla_x\kappa} \tag{3.12}$$

the lemma is fulfilled with a constant $C = \min_\Omega(\frac{1}{2}, 1 - \frac{\delta\vartheta\nabla_x\kappa}{2\kappa})$ ■

Since boundedness of the operator is obvious, the previous lemma ensures convergence of the method. The weighted second order term in direction ϑ suffices to suppress oscillations in upwind direction occurring with the standard Galerkin scheme.

The following lemma has been developed in [25]:

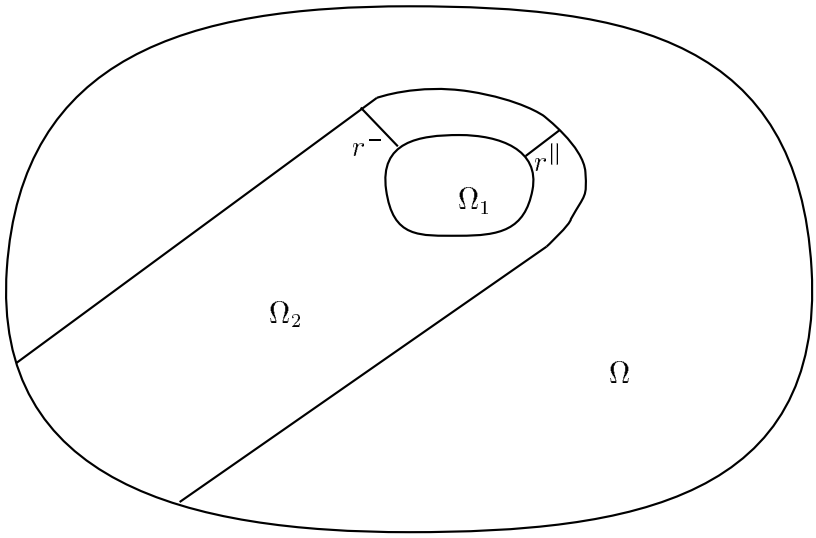


Figure 3.3: Domain of dependency

Lemma 3.12 *Let u_h solution of (3.6). For an arbitrary sub-domain $\Omega_1 \subset \Omega$ as in Figure 3.3 choose Ω_2 with $\Omega_1 \subset \Omega_2 \subset \Omega$ such that*

- *for all $x \in \Omega_2$, there holds $y = x - s \cdot \vartheta \in \Omega_2$ for $s > 0$ and $y \in \Omega$, i. e. all upstream points of x belong to Ω_2.*
- *For arbitrary points $x \in \Omega_1$ and $y \in \Omega - \Omega_2$ we split the difference vector $r = y - x$ in parts r^- and $r^{\|}$ orthogonal and parallel to ϑ respectively. These parts are to fulfill $r^{\|} > ch \log \frac{1}{h}$ and $\mid r^- \mid > c\sqrt{h} \log \frac{1}{h}$.*

With these assumptions, the estimate

$$\|u - u_h\|_{L^2(\Omega_1)} \leqslant Ch^{k+1/2}\Big(\|u\|_{H^{k+1}(\Omega_2)} + \|f\|_{L^2(\Omega)}\Big)$$

holds.

Rannacher and Zhou proved a similar estimate for the maximum norm in [45]. It follows that influence of errors at point y on the solution at point x with $|x - y| = d$ decays exponentially with $\sqrt{d}$ in crosswind and with d in upwind direction. This result corresponds to the observation, that information is transported only in streamline direction mimicking the physical model. Due

Level	Upwind	SD
2	2.23	.865
3	.829	.365
4	.395	$7.02 \cdot 10^{-2}$
5	.201	$1.42 \cdot 10^{-2}$
6	.103	$3.34 \cdot 10^{-3}$
7	$5.21 \cdot 10^{-2}$	$8.35 \cdot 10^{-4}$
8	$2.64 \cdot 10^{-2}$	$2.15 \cdot 10^{-4}$

Table 3.1: Upwind versus Streamline Diffusion (L^2–errors for the radiative transfer equation with constant coefficients)

---

to this, we can assume the same inflow and outflow boundary conditions as in the original equation (3.6).

Asymptotic results are only of questionable value for scientific calculations, since the constants are known only with uncertainty. We therefore compare the accuracy of finite difference upwind and streamline diffusion finite element method in Table 3.1. The results are obtained for the radiative transfer equation with constant coefficients on regularly refined grids of mesh size $2^{-\text{level}}$. Although streamline diffusion shows a slightly irregular convergence behavior on coarse grids, it is clearly better than upwind even on coarse meshes.

The main advantage of the streamline diffusion finite element method is its being a Petrov–Galerkin discretization. This implies Galerkin orthogonality for the approximate solution. We can therefore apply the a posteriori error control techniques to be developed in Chapter 4.

3.4 Full Galerkin Discretization

Based on the discretizations developed in the two preceding sections, we construct a discretization for the full radiative transfer equation. We obtain trial functions on $\Omega \times S^2$ by tensor products of the spaces in (3.4) and (3.9).

Base functions are easily obtained: if $\{\psi_i\}_i$ is a basis of V_h and $\{\pi_i\}_i$ a basis

of X_h, then $\{\varphi_{ij}\}_{ij}$ with

$$\varphi_{ij}(x,\vartheta) = \psi_i(x)\pi_j(\vartheta)$$

is a basis of the tensor product space

$$W_h \subset W = L^2(\Omega \times S^2).$$

According to the norm for transport problems (3.11) we define the norm

$$\|u\|_{W_h}^2 = \|\sqrt{\delta}\,\vartheta\cdot\nabla_x u\|_{\Omega\times S^2}^2 + \|\sqrt{\kappa}\,u\|_{\Omega\times S^2}^2 + \|u\|_{\Gamma\times S^2}^2 \tag{3.13}$$

for radiative transfer equation discretized with streamline diffusion, where

$$\|u\|_{\Gamma\times S^2}^2 = \frac{1}{2}\iint\limits_{S^2\,\Gamma} u^2\ \mathfrak{n}_\Gamma(x)\cdot\vartheta\,dx\,d\vartheta \tag{3.14}$$

The full discretization with streamline diffusion stabilization of the transport operator reads (where we abbreviate $\left\langle .,. \right\rangle = \left\langle .,. \right\rangle_{\Omega\times S^2}$)

$$\left\langle \vartheta\cdot\nabla_x u + \chi u - \Sigma u, \varphi + \delta\vartheta\cdot\nabla_x\varphi \right\rangle = \left\langle f, \varphi + \delta\vartheta\cdot\nabla_x\varphi \right\rangle \quad \forall\ \varphi \in W_h. \tag{3.15}$$

Lemma 3.13 *Assuming $\chi \in C_0^1(\Omega)$, discretization (3.15) is stable in the norm $\|.\|_{W_h}$ defined in (3.13).*

Proof: Like in the pure transport case, the proof relies on a partial integration of first order terms.

$$\begin{aligned}
&\left\langle \vartheta\cdot\nabla_x u + \chi u - \Sigma u, u + \delta\vartheta\cdot\nabla_x u \right\rangle \\
&\quad = \left\langle \vartheta\cdot\nabla_x u, \delta\vartheta\cdot\nabla_x u \right\rangle + \left\langle \chi u, u \right\rangle + \left\langle \chi u, \delta\vartheta\cdot\nabla_x u \right\rangle + \left\langle \vartheta\cdot\nabla_x u, u \right\rangle \\
&\qquad\qquad - \left\langle \Sigma u, u \right\rangle - \left\langle \Sigma u, \delta\vartheta\cdot\nabla_x u \right\rangle \\
&\quad = \|u\|_{W_h} - \left\langle \delta\vartheta\cdot\nabla_x\kappa u, u \right\rangle - \left(\|\sqrt{\sigma}\,u\| + \|\sqrt{\delta\sigma}\,u\|_{\Gamma\times S^2} \right) \\
&\quad \geqslant \|\sqrt{\delta}\,u\| + \|\sqrt{\chi - \sigma - \tfrac{\delta\vartheta\nabla_x\kappa}{2}}\,u\| + \|\sqrt{1-\delta\sigma}\,u\|_{\Gamma\times S^2} \\
&\qquad\qquad \geqslant C\|u\|_{W_h},
\end{aligned}$$

provided that

$$\delta < \min\Big\{\frac{2(\chi-\sigma)}{\vartheta\cdot\nabla_x\kappa}\ ,\ \frac{1}{\sigma}\Big\}. \tag{3.16}$$

Next we propose a decomposition of the radiative transfer equation into a differential and an integral part. To give more insight into its structure, we assume the phase function P to be constant. Then, the scattering operator Σ may be seen as a projection operator from W onto $V = L^2(\Omega)$

$$\Big(\Sigma u\Big)(x) = \sigma\tfrac{1}{\mu(S^2)}\int\limits_{S^2} u(x,\vartheta)\,d\vartheta$$

We introduce the auxiliary function $v = \Sigma u$ and modify our equation yielding the mixed formulation

$$\begin{aligned}\Big\langle(\vartheta\cdot\nabla_x+\chi)u,\varphi\Big\rangle - \Big\langle v,\varphi\Big\rangle \ &= \Big\langle f,\varphi\Big\rangle\\ \Big\langle -\Sigma u,\psi\Big\rangle_\Omega + \Big\langle v,\psi\Big\rangle_\Omega &= 0\\ \forall\quad \varphi\in W,\qquad \psi\in V&\end{aligned}$$

Ellipticity with respect to $\|u\|_{W_h}$ of this system is obtained from the same property of the original equation by using the fact that $v = \Sigma u$ strongly. v may therefore be eliminated leaving exactly the situation of lemma 3.13.

We now build a tensor product discretization for the domains Ω and S^2 by choosing

$$\begin{aligned} V_h &\subset V\\ W_h^T = V_h\otimes X_h &\subset W\end{aligned}$$

with V_h and $X_h \subset L^2(S^2)$ defined in (3.9) and (3.4) respectively.

The full Petrov–Galerkin formulation reads: find $(u_h, v_h) \in W_h^T \times V_h$ with

$$\begin{aligned}\Big\langle(\vartheta\cdot\nabla_x+\chi)u_h,\varphi_h\Big\rangle - \Big\langle v_h,\varphi_h\Big\rangle \ &= \Big\langle f,\varphi_h\Big\rangle\\ \Big\langle -\Sigma u_h,\psi_h\Big\rangle_\Omega + \Big\langle v_h,\psi_h\Big\rangle_\Omega &= 0\end{aligned}$$

$$\forall\ \varphi_h\in W_h^T, \psi_h\in V_h$$

This discretization shares the advantages and disadvantages of all tensor product grids: local refinement is possible, but it is always a whole factor grid that is refined. We therefore construct a more sophisticated method of local refinement.

We observe that for transport dominated problems, the regions where the intensity jumps differ for the ordinates. It is therefore desirable to have a special grid for each direction, that is

$$W_h = V_h^1 \times V_h^2 \cdots \times V_h^m.$$

This makes a special handling of the global coupling of all directions by Σ necessary. The solution is using the mixed formulation with different finite element spaces for different equations. The weak formulation reads now: find $(u_h, v_H) \in W_h \times V_H$ with

$$\begin{aligned} \Big\langle (\vartheta \cdot \nabla_x + \chi) u_h, \varphi_h \Big\rangle - \Big\langle v_H, \varphi_h \Big\rangle &= \Big\langle f, \varphi_h \Big\rangle \\ \Big\langle -\Sigma u_h, \psi_H \Big\rangle_\Omega + \Big\langle v_H, \psi_H \Big\rangle_\Omega &= 0 \end{aligned}$$

$$\forall\, \varphi_h \in W_h, \psi_H \in V_H.$$

If we ensure

$$\forall\, i \quad V_H \subset V_h^i \tag{3.17}$$

then we have the strong condition $v_H = \Pi_H \Sigma u_h$, where Π_H is the usual L^2 projection on V_H. For sake of simplicity, we investigate ellipticity in L^2 by

$$\begin{aligned} &\Big\langle (\vartheta \cdot \nabla_x + \chi) u_h, u_h \Big\rangle - \Big\langle \Pi_H \Sigma u_h, u_h \Big\rangle \\ &\qquad - \Big\langle \Sigma u_h, \Pi_H \Sigma u_h \Big\rangle + \Big\langle \Pi_H \Sigma u_h, \Pi_H \Sigma u_h \Big\rangle \\ =&\Big\langle (\vartheta \cdot \nabla_x + \chi) u_h, u_h \Big\rangle - \Big\langle \Pi_H P \Sigma u_h, u_h \Big\rangle \\ =&\Big\langle (\vartheta \cdot \nabla_x + \chi) u_h, u_h \Big\rangle - \Big\langle \Sigma u_h, \Pi_H u_h \Big\rangle \\ =&\Big\langle (\vartheta \cdot \nabla_x + \chi) u_h, u_h \Big\rangle - \Big\langle \Sigma u_h, u_h \Big\rangle + \Big\langle \Sigma u_h, u_h - \Pi_H u_h \Big\rangle, \end{aligned}$$

which proves

Lemma 3.14 *Let a discretization of the radiative transfer equation of the form* (3.17) *fulfill condition* (3.17). *Then the ellipticity estimate*

$$\left\langle (\vartheta \cdot \nabla_x + \chi) u_h, u_h \right\rangle - \left\langle \Pi_H \Sigma u_h, u_h \right\rangle \geqslant \|\vartheta \cdot \nabla_x + \chi\| \|u_h\|^2 - \|\Sigma\| \|u_h\| \|u_h - \Pi_H u_h\| \quad (3.18)$$

holds.

Remark 3.2 It is clear that using the methods of lemma 3.13, we can prove stability in the stronger norm $\|.\|_{W_h}$.

Remark 3.3 This discretization technique is especially useful in the transport dominated and moderately scattering case. In highly scattering regions, the intensities for different ordinates tend to be the same, thus making the tensor product method adequate.

Remark 3.4 The stability estimate (3.18) only involves computed quantities, so the choice of the common mesh function H may be adaptively based on this inequality.

Chapter 4

Adaptivity

Numerical computation of solutions to partial differential equations often requires a huge amount of memory and computation time. For economic reasons — considering radiative transfer problems even for the ability to compute at reasonable accuracy — it is inevitable to reduce the size of the discrete system to be solved. A well–known method to reach this goal is the exploitation of symmetry to reduce the dimension of a problem, but many problems need computation of the full three–dimensional model. Here, adaptive methods provide a flexible means to reduce computation costs and — combined with a posteriori error estimates — produce reliable and accurate solutions. This way, even radiative transfer problems can be solved on a workstation (at least in the two–dimensional case). Three–dimensional radiative transfer needs sophisticated grid adaption even on supercomputers to make problems computable.

The aim of simulation is solving the physical equation, such that there is a guaranteed error estimate for the quantity of interest denoted by the functional $\mathcal{E}(u)$. It may be written in the abstract form

$$\left|\mathcal{E}(u - u_h)\right| \leqslant \mathrm{TOL}. \tag{4.1}$$

Since the error $u - u_h$ is not known it has to be replaced by an estimate $\eta_{\mathcal{E}}(h, u_h)$.

The solution should be obtained with as few as possible resources, i. e. on a nearly optimal grid in the finite element context. Using the function $h(x)$

denoting the local mesh width, this task reads as the following optimization process:

$$\begin{aligned} h(x) &= \max \quad \forall\, x \in \Omega \\ \eta_{\mathcal{E}}(h, u_h) &\leqslant \mathrm{TOL}. \end{aligned} \tag{4.2}$$

The iteration process applied to determine the optimal mesh function h must converge rapidly, since each step requires the solution of the differential equation. In the first section, we present some optimization strategies for this problem.

A real mesh can only approximate this optimal grid function h, since the mesh generation process establishes additional constraints to the function h. So, optimality of a mesh has to be seen with respect to the grids obtainable in the algorithm. In the second section, we present different mesh generation processes and refinement criteria based on error estimates.

In the third section of this chapter we discuss estimates for different norms and their theoretical derivation.

4.1 Adaptive Algorithms

The iterative process for solving a partial differential equation essentially reads like the algorithm in Figure 4.1. In the first step, we solve the problem $Au = f$ on a starting grid (lines 9–11). Then, in each adaptive step, we adapt the grid according to the approximate solution u (5–7) and transfer u to the new mesh (8) to be used as start value of the iterative solution (11). If the estimate e is lower than a given tolerance (12–13) we stop the loop and do the postprocessing.

There are two criteria for the efficiency of this adaptive algorithm:

1. optimality of the final grid and
2. speed of convergence.

If the multi–grid method is used to solve the discrete linear system in each adaptive step, the weight is put very much on the first criterion. Indeed,

```
1   Triangulation  tr := start_triangulation
2   Vector  u := 0;
3   for step :=0 to maxsteps
4         if (step ≠ 0)
5               adapt(tr, u);
6               tr.coarse();
7               tr.refine();
8               tr.interpolate(u);
9         Vector  b := tr.rhs(f);
10        Matrix  A := tr.matrix();
11        u := A^-1 b;
12        double  e := tr.estimate(u);
13        if (e < TOL) break;
14  tr.postprocess(u);
```

Figure 4.1: Adaptive Iteration

Becker has shown in [4], that a tightly coupled adaptive multi–grid method is much faster than solving to a given tolerance and refining alternatingly.

As we are restricted to use iterative methods based on orthogonal sequences of vectors, it is just the other way round. Each refinement step means a restart of the iteration, since only the start vector may be interpolated from the previous grid. Due to the change in the operator, the orthogonality of other vectors is lost.

So we have to look for an refinement criterion, which is a good compromise between grid optimality and convergence rate, putting stress on the latter.

Given a local error indicator η_K approximating the error contribution of cell K, the following criteria seem possible:

1. Refine if $\eta_K \cdot \lambda_K > \gamma \mathrm{TOL}$ with $\lambda_K = \#K$ the overall number of cells and γ a constant slightly below unity.

2. Refine if $\eta_K > \alpha \max_K \eta_K$ with $\alpha \in]0, 1[$.

3. (numerus clausus) Sort cells on η_K and refine the first ν ones.

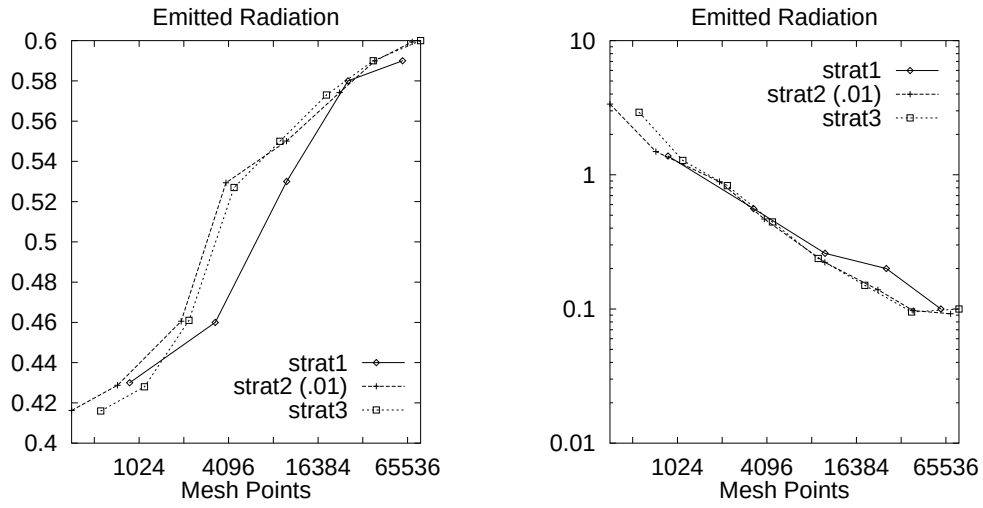


Figure 4.2: Comparison of refinement strategies

Criterion 1 compares each local error estimate with the global tolerance. The weights are chosen such that local refinement stops automatically if the global criterion is reached. Indeed, we can replace the check of the global error by this local criterion. This is true, if the parameter λ_K is updated each time a cell is refined. Optimality of the resulting grids is shown in a joint work with Becker and Suttmeier (cf. [6]).The parameter γ ensures, that convergence is not asymptotically to TOL, but that the estimate reaches TOL in a few steps.

Refinement criteria 2 and 3 try to equidistribute the error over the domain. Since both methods are monotonous and we have a priori bounds for the estimator, the global estimate converges until lower than TOL. Here the global error has to be checked independently.

Criteria 1 and 2 may be easily generalized to obtain double refinement for cells with especially large local error indicators and coarsening of cells with a small indicator, which is important for time dependent problems.

Method 3 is especially valuable, if a computation "as accurate as possible" is desired. Then, the parameter ν has to be determined by the remaining memory resources.

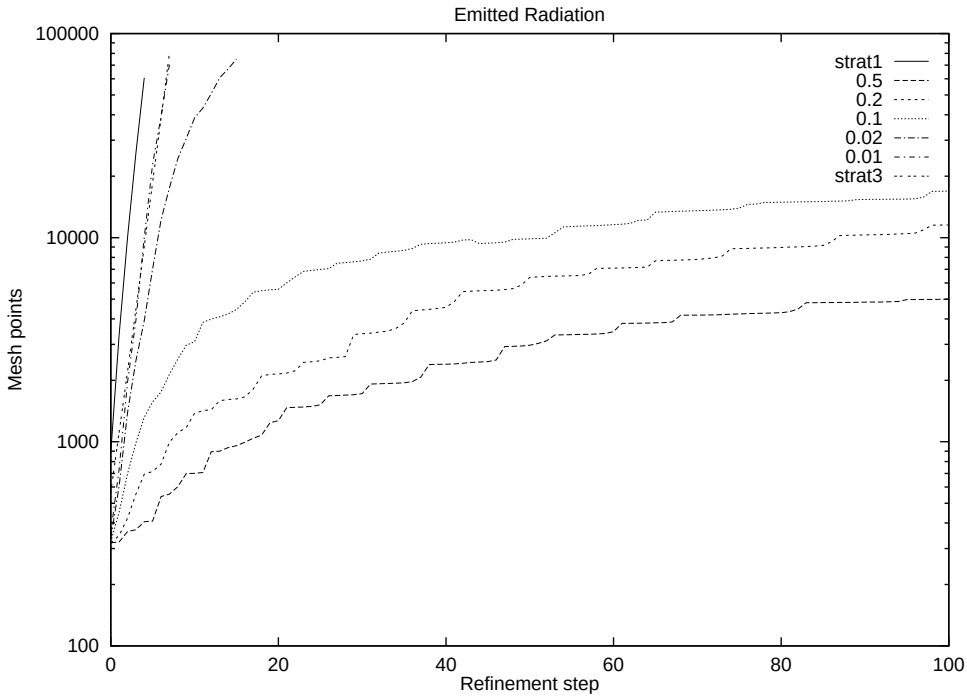


Figure 4.3: Parameter dependence of the second strategy

We compare these refinement strategies applied to the dust cloud example in Chapter 8. Parameter values are $\gamma = 0.8$ and $\nu = 1/3\#K$. The results are shown in Figures 4.2 and 4.3. The choice of the parameter α in Criterion 2 is important as may be seen in Figure 4.3. While e. g. Verfürth in [41] proposes $\alpha \approx 0.5$ for Poisson's equation, a sufficient refinement rate is not achieved even for $\alpha \in \{0.1, 0.2\}$. Parameter values of $\alpha \in \{0.01, 0.02\}$ in our case yield the necessary convergence speed. We observe in Figure 4.2, that all three methods generate grids of nearly the same efficiency.

The convergence of the second strategy depends strongly on the parameter α and the structure of the problem. The dust cloud model problem has very localized features and the error contributions are limited to a very small portion of the domain. The result are very small refinement regions, often just one cell if $\alpha > 0.1$. The necessarily fine tuning of α makes this algorithm inadequate in the if we want to cover a wide range of applications. With the last strategy, we can easily control the growth rate of meshes and therefore the memory consumption and convergence speed by varying the parameter

ν. Since the action of ν is obvious, this allows experimenting with different convergence rates, especially, if memory is insufficient to reach the desired accuracy. In this case, we can approximate the best possible value with the second strategy. Provided computing resources are sufficient and an optimal error estimate exists, the first strategy is obviously the best to reach the prescribed tolerance, since it is very fast and it does not need parameter tuning.

4.2 Grid Generation

An important topic in designing adaptive algorithms is the generation of computational meshes. We can interpret the output of an error estimator as a mesh function $h : \Omega \to \mathbb{R}$, describing the desired local mesh width. According to this notion, different grid generators have been developed:

1. Execute the following three steps to compute a new mesh:
   (a) Randomly distribute points in the domain Ω with local density $h^d(x)$.
   (b) Connect these points using a Delaunay algorithm to get a simplicial triangulation.
   (c) Apply a smoothing method to avoid degenerate simplices.
2. The "advancing front" generator begins with a triangulation of the boundary. It consecutively constructs layers of cells protruding to the interior and thus filling the whole domain. This method is known to cause severe topological problems, if the domain is not convex or the boundary surface has rapidly changing curvature.
3. The mesh generator we use constructs the final grid by successive local refinement of a very coarse starting triangulation. Curved boundaries are approximated by pulling division points of those boundary edges (faces in 3D) on the desired curve (surface). This generator is the only one, which generates the structures needed by multi–level methods. It is this last property, which made us decide for successive refinement.

distributed points	**advancing front**	**local refinement**
Grid regularity		
good, depends on the smoothing algorithm and is therefore a trade-off to the approximation of $h(x)$	good if topologic problems of overlapping cells are solved	perfect for straight boundaries, tends to degenerate cells in critical boundary regions
Approximation of mesh function $h(x)$		
best approximation	good approximation	mediocre approximation, since the mesh width can only jump by factors of two between adjacent cells.
Multi-grid		
grid hierarchies have to be constructed a posteriori	same as left entry	grids are constructed as hierarchy, so multi-grid is inherent
Transfer between adaptive grids		
Needs point search algorithms and interpolation in cells. This often causes loss of accuracy.	same as left entry	uses multi-grid prolongation and restriction operators, therefore is highly accurate

Table 4.1: Comparison of grid generation methods

The generation of non–uniform grids by using tensor products of arbitrary one–dimensional meshes is a strategy widely used. We do not consider its application, since the grids obtained are restricted too much to approximate the mesh function h. Additionally, these method often produces cells with an aspect ration of more than 10^4. The approximation properties of those cells are rather bad, such that they deteriorate convergence of the whole grid.

4.3 Error Estimates

In this section we develop a posteriori error estimates for the Galerkin discretizations described in Chapter 3. The technique presented follows essentially the method proposed by Johnson et al. in [12] and applies the enhancements proposed by Becker/Rannacher in [7]. Consider a solution u to the radiative transfer equation and u_h to its discretization. Let $e = u - u_h$ be the error function. In computing a physically relevant problem, usually a desired accuracy is prescribed in the form

$$\left|\mathcal{E}(e)\right| \leqslant \mathrm{TOL}. \tag{4.3}$$

Here $\mathcal{E}(.)$ is the functional describing the value to be computed by the simulation, e. g. the L^2-norm, the value at just one point or a boundary integral as in the example of Figure 8.1 on page 84. According to this functional we choose $r_{\mathcal{E}}$ such that

$$\mathcal{E}(e) = \left\langle e, r_{\mathcal{E}} \right\rangle. \tag{4.4}$$

Examples for $r_{\mathcal{E}}$ are

$$\mathcal{E}(e) = \|e\|_{L^2} \qquad r_{\mathcal{E}} = e \tag{4.5}$$

$$\mathcal{E}(e) = \tfrac{1}{\mu(M)} \int_M e \qquad r_{\mathcal{E}} = \chi(M) \tag{4.6}$$

$$\mathcal{E}(e) = e(x_0) \qquad r_{\mathcal{E}} = \delta_{x_0}. \tag{4.7}$$

Here $\chi(M)$ denotes the characteristic function of the set M defined as a distribution by

$$\int_\Omega \chi(M)\varphi \, dx = \frac{1}{\mu(M)} \int_M \varphi \, d\mu(x) \qquad \forall\, \varphi \in C^\infty(\Omega)$$

and $\delta_{x_0} = \chi(\{x_0\})$ the Dirac functional (μ denotes a suitably chosen measure on M).

Introducing the dual $\mathcal{L}^*$ of the radiative transfer operator and the solution z of the dual problem

$$\mathcal{L}^* z = r_{\mathcal{E}}. \tag{4.8}$$

we replace the error norm in (4.3) in terms z and the residual $\mathcal{R}(u_h)$ by

$$\begin{aligned}\Big\langle e, r_{\mathcal{E}} \Big\rangle &= \Big\langle \mathcal{L}e, z \Big\rangle \\ &= \Big\langle \mathcal{L}e, z - z_h \Big\rangle \qquad (4.9)\\ &= \Big\langle f - \mathcal{L}u_h, z - z_h \Big\rangle \\ &= \Big\langle \mathcal{R}(u_h), z - z_h \Big\rangle, \qquad (4.10)\end{aligned}$$

In equation (4.9), we used a characteristic feature of finite element methods, the Galerkin orthogonality

$$\Big\langle \mathcal{L}u - \mathcal{L}u_h, w_h \Big\rangle = 0 \qquad \forall\, w_h \in V_h. \qquad (4.11)$$

We now apply the formalism developed so far to the radiative transfer equation and state

Lemma 4.1 *Let u be solution to the radiative transfer equation (2.13) and u_h to a Galerkin discretization. Then, the error $\mathcal{E}(u - u_h)$ admits the representation*

$$\mathcal{E}(u - u_h) = \Big\langle Tu_h + \chi u_h - \Sigma u_h - \kappa f, z - z_h \Big\rangle \qquad (4.12)$$

where z is solution to the dual problem

$$\begin{aligned}-Tz + \chi z - \Sigma z &= r_{\mathcal{E}} \qquad \text{in } \Omega \qquad (4.13)\\ z &= 0 \qquad \text{on } ?_+. \qquad (4.14)\end{aligned}$$

Proof: Since (4.12) is just application of (4.10) to radiative transfer equation we have to prove the correctness of the dual problem:

Clearly, the operator χId is self adjoint. The action of T on a function $u \in C^1(\Omega \times S^2)$ results from the sum of operators T_ϑ with

$$T_\vartheta u(x, \tilde{\vartheta}) = \delta(\tilde{\vartheta} - \vartheta)\; \vartheta \cdot \nabla_x u(x, \tilde{\vartheta}).$$

By partial integration we obtain $T_\vartheta^* = -T_\vartheta$ by

$$\begin{aligned}&\iint\limits_{S^2\,\Omega} \delta(\tilde{\vartheta} - \vartheta)\; \vartheta \cdot \nabla_x u(x, \tilde{\vartheta})\; v(x, \tilde{\vartheta})\; dx\; d\tilde{\vartheta} \\ &\qquad = -\int\limits_{\Omega} u(x, \vartheta) \vartheta \cdot \nabla_x v(x, \vartheta) \\ &\qquad\qquad = -\iint\limits_{S^2\,\Omega} \delta(\tilde{\vartheta} - \vartheta)\; u(x, \tilde{\vartheta})\; \vartheta \cdot \nabla_x v(x, \tilde{\vartheta})\; dx\; d\tilde{\vartheta}\end{aligned}$$

The self adjointness of the scattering operator Σ follows directly from symmetry condition (2.10) by applying Fubini's theorem to

$$\iint_{\Omega}\int_{S^2} \sigma \int_{S^2} P(\vartheta,\tilde{\vartheta})u(x,\tilde{\vartheta})v(x,\vartheta)\; d\tilde{\vartheta}\, d\vartheta\, dx$$

Remark 4.1 In the proof of lemma 4.1 we assumed zero inflow boundary conditions. In the case of non–trivial boundary conditions, we save the duality relations of operators by prescribing these values in the weak form. This corresponds to adding a term $\int_{\Gamma_-}(u-g)v\; dx$ to the weak formulation of the radiative transfer equation.

4.3.1 Mean Quadratic Error

In this subsection we analyze the mean quadratic error in the space variable. We consider u to be the solution of a semi–discrete radiative transfer equation with a fixed number of ordinates.

The appropriate error functional for L^2–error estimates is $r_{\mathcal{E}} = e$. Clearly, we cannot solve the dual problem with right hand side e, since e is the unknown quantity. We therefore use a stability estimate of the form (denoting by the space $L^2(L^2)$ functions on S^2 with values in $L^2(\Omega)$).

$$\|z\|_Y \leqslant C_s \|e\|_{L^2(L^2)}. \tag{4.15}$$

Remark 4.2 This estimate involves stability of the *continuous* dual problem and does not use properties of the discretization space.

Remark 4.3 Since stability estimate (4.15) is used to limit the interpolation $z - z_h$ in (4.10), the norm $\|.\|_Y$ should be as strong as possible to obtain an estimate of the form

$$\|z - z_h\|_{L^2(L^2)} \leqslant h^{\alpha}\|z\|_Y$$

with $\alpha > 0$.

The error is observed to decay of order h^2 and the residual decays of first order. To obtain optimal bounds, we would like to have an error estimate of the form

$$\|e\| \leqslant C_s C_i \|h\mathcal{R}(u_h)\| \tag{4.16}$$

with the interpolation constant C_i depending only on the trial functions.

This estimate requires the space Y to be $L^2(H^1)$, but it holds

Remark 4.4 The operator on the left hand side of (4.13) allows control only of $\|z\|_{\Omega\times S^2}$ and $\|\vartheta\cdot\nabla_x z\|_{\Omega\times S^2}$, but not of the gradient ∇z.

We therefore apply a well–known remedy from numerics of hyperbolic equations, the method of artificial diffusion (see e. g. [12, 16, 27]). Equation (2.4) is augmented by a small diffusion term, yielding

$$\varepsilon\Delta u_\varepsilon + Tu_\varepsilon + \chi u_\varepsilon - \Sigma u_\varepsilon = \kappa B \tag{4.17}$$

where $\varepsilon \approx h^2|\mathcal{R}(u_h)|$. The error $\|u-u_h\|$ is the sum of the error $\|u-u_\varepsilon\|$ between the artificial diffusion equation and the radiative transfer equation and the discretization error $\|u_\varepsilon - u_h\|$. Since the error due to artificial diffusion is of order h^3, it may be neglected and the overall error is dominated by the discretization error.

Theorem 4.2 *The modified problem* (4.17) *admits the error estimate*

$$\|u_\varepsilon - u_h\| \leqslant C_i C_s \left\{ \|\min\{1, \tfrac{h^2}{\varepsilon}\}\mathcal{R}(u_h)\| + \|D_h^2(u_h)\| \right\} \tag{4.18}$$

with a stability constant C_s depending only on the domain and the coefficients of the continuous dual problem. The approximate second derivative $D_h^2(u_h)$ is defined by

$$\|D_h^2(u_h)\| = \left(\sum_E \left| h^{-3/2} \frac{\partial u_l - \partial u_r}{\mathfrak{n}} \right|^2 \right)^{1/2} \tag{4.19}$$

Proof: The dual problem corresponding to (4.17) is

$$\varepsilon\Delta z - Tz - \delta T^2 z + \chi z - \Sigma z = e$$

with right hand side $e = u_\varepsilon - u_h$. Applying the techniques described above we obtain

$$\begin{aligned} \|e\| &= \left\langle \varepsilon\nabla e, \nabla z \right\rangle + \left\langle (T+\chi-\Sigma)e, z \right\rangle \\ &= \left\langle \hat{\mathcal{R}}(u_h), z \right\rangle \leqslant \hat{\|} \frac{C_s h^2}{\varepsilon} \hat{\mathcal{R}}(u_h)\| \|\nabla^2 z\| \end{aligned}$$

where $\hat{\mathcal{R}}(.)$ is the residual of the modified radiative transfer equation. The weighting factor in front of the residual is taken from Lemma 4.3 below. Obviously, the dual problem is stable in L^2 and we obtain the additional estimate

$$\left\langle \hat{\mathcal{R}}(u_h), z \right\rangle \leqslant \tilde{C}_s \|\mathcal{R}(u_h)\| \, \|z\|_{L^2}. \tag{4.21}$$

Since the error admits both estimates, the statement of the theorem is true with $C_s = \max(\hat{C}_s, \tilde{C}_s)$. ■

Lemma 4.3 *The solution to the equation*

$$-\varepsilon \Delta z - Tz - \delta T^2 z + \chi z - \Sigma z = e \tag{4.22}$$

admits the stability estimate

$$\|z\|_{L^2(H^2)} \leqslant \frac{C_s}{\varepsilon} \|e\|_{L^2(L^2)}. \tag{4.23}$$

Proof: Equation (4.22) obviously admits the estimates

$$\|z\|_{L^2(L^2)} \leqslant C \|e\|_{L^2(L^2)} \qquad\qquad \|z\|_{L^2(H^1)} \leqslant \frac{C}{\varepsilon} \|e\|_{L^2(L^2)}.$$

We can absorb the lower order terms to the right hand side, since $\chi - \Sigma$ is positive semi–definite on $L^2(L^2)$, yielding

$$-\varepsilon \Delta z - \delta T^2 z = g$$

The differential operator on the left hand side has no coupling over the ordinate space. Therefore, we can apply standard H^2–estimates for elliptic problems like Theorems 8.8 and 8.10 in Gilbarg / Trudinger [18]. Integrating over S^2 results in estimate (4.23).

In the proofs of Theorem 4.2 and of Lemma 4.3 we have assumed zero boundary conditions. They can be easily extended to inflow boundary conditions using the techniques described in [12].

Numerical tests have shown, that the error estimate of Theorem 4.2 is far from being optimal, since the stability of the dual problem is too weak. In the next subsection we follow a new approach avoiding stability by computing an approximation for the solution to the dual problem with right hand side independent of e.

4.3.2 Boundary Integral Error

Now we consider a situation like in Figure 8.1 on page 84. The value desired from the computation is given by the functional

$$\mathcal{E}(u) = E(u) \equiv \int\limits_{\Gamma_+(\vartheta_{\mathrm{Obs}})} u(x, \vartheta_{\mathrm{Obs}})\ \mathfrak{n}_\Gamma \cdot \vartheta_{\mathrm{Obs}}\, dx \tag{4.24}$$

The appropriate right hand side of the dual problem reads

$$r_E = \delta(\vartheta - \vartheta_{\mathrm{Obs}})\ \chi(?_+(\vartheta_{\mathrm{Obs}})) \tag{4.25}$$

Now we can continue at (4.10) to get an efficient a posteriori estimate (see also [7]) for this special value:

$$\begin{aligned}\Big\langle e, r_E \Big\rangle &\leqslant C_{\mathrm{Sec}} \Big\langle \mathcal{R}(u_h), z^E - z_h^E \Big\rangle \\ &= C_{\mathrm{Sec}} \sum_K \Big\langle \mathcal{R}(u_h), z^E - \Pi_h z^E \Big\rangle_K \\ &\leqslant C_{\mathrm{Sec}} C_i \sum_K \|\mathcal{R}(u_h)\|_K\, \|\nabla^2 z^E\|_K,\end{aligned}$$

with the solution z^E of the dual problem not depending on u_h. In practical calculations, it is hard or even impossible to calculate z^E and we replace it by the approximate dual solution z_h^E and add a security constant C_{Sec} to account for the error of z_h^E. This constant results from an L^∞–estimate for z_h^E and should be in the range of 1–1/4 ($C_{\mathrm{Sec}} = 1/4$ means that we allow over–refining once due to the lack of accuracy of z_h^E). We use the second derivative of z_h^E defined in equation (4.19). These considerations result in

Lemma 4.4 *The discretization error $E(u - u_h)$ is limited by*

$$|E(e)| \leqslant \eta_E = \sum_{K \in \mathcal{T}} \eta_E(K), \tag{4.26}$$

where $\eta_E(K)$ denotes the local error indicator

$$\eta_E(K) = C_{Sec} C_i \|\mathcal{R}(u_h)\|_K\, \|D_h^2 z^E\|_K. \tag{4.27}$$

Chapter 5

Numerical Solution

This chapter will discuss methods to solve the linear system of equations resulting from the discretization methods of Chapter 3. A suitable solution algorithm has to show good convergence properties for transport dominated, scattering dominated and mixed problems, since astrophysical applications usually show both kinds of behavior in different parts of the domain. Before we look at iterative methods in section 5.2, we investigate the structure and eigenvalue distribution of the discrete system.

After shortly discussing the drawbacks of stationary iterations like the Λ–iteration common in astrophysics, we follow the way of Turek [36] and present two fast Krylov space schemes for our equation in subsection 5.2.1. A very important role plays the preconditioning method. In subsection 5.2.2 we will compare three different preconditioners for the streamline diffusion finite element method based on the upwind discretization of Chapter 3, standard Gauß–Seidel and multi–level splitting.

Finally, we devote a section to the discussion of implementation questions. Due to the exorbitant memory requirements of radiative transfer computations this must be an important issue in this thesis. We show, that a problem adapted matrix representation may considerably reduce memory consumption.

5.1 The Discrete System

We give a short description of the matrices resulting from the various discretizations introduced in Chapter 3.

The discrete system has the form

$$Ax = b \tag{5.1}$$

where $x, b \in X = \mathbb{R}^n \otimes \mathbb{R}^m$ and $A : X \to X$.

From the operator form (2.13) of the radiative transfer equation, we derive the representation

$$A = T_h + M_h(\chi) - S_h \tag{5.2}$$

with a suitable discretization $M_h(\chi) \approx \chi Id_x$ of the multiplication with a function $\chi(x)$ depending on the space variable.

For our tensor product discretizations, these operators may be split up further:

$$\begin{aligned} T_h &= \operatorname{diag}(\mathfrak{T}_1, \ldots, \mathfrak{T}_m) \\ M_h(\chi) &= \operatorname{diag}(\mathfrak{M}_1(\chi), \ldots, \mathfrak{M}_m(\chi)) \\ S_h &= \begin{pmatrix} \omega_{11}\mathfrak{M}_1(\sigma) & \cdots & \omega_{1m}\mathfrak{M}_1(\sigma) \\ \vdots & & \vdots \\ \omega_{m1}\mathfrak{M}_m(\sigma) & \cdots & \omega_{mm}\mathfrak{M}_m(\sigma) \end{pmatrix}. \end{aligned} \tag{5.3}$$

The finite difference upwind scheme leads to matrices $\mathfrak{T}_i$ defined by (3.7) and $\mathfrak{M}_i(\sigma)$ the diagonal matrix with $m_{ii} = \sigma(x_i)$.

With the finite element streamline diffusion tensor product discretization, the entries of these matrices are defined by

$$\mathfrak{T}_i^{jk} = \left\langle \varphi_j + \delta\vartheta_i \cdot \nabla_x \varphi_j, \mathcal{T}\varphi_k \right\rangle \tag{5.4}$$

$$\mathfrak{M}_i^{jk}(\chi) = \left\langle \varphi_j + \delta\vartheta_i \cdot \nabla_x \varphi_j, \chi\varphi_k \right\rangle \tag{5.5}$$

$$\mathfrak{M}_i^{jk}(\sigma) = \left\langle \varphi_j + \delta\vartheta_i \cdot \nabla_x \varphi_j, \sigma\varphi_k \right\rangle. \tag{5.6}$$

Due to the equivalence result of Chapter 3, this structure is valid for the full Galerkin discretization too.

We now investigate the condition number and eigenvalue structure of these matrices.

Lemma 5.1 *Assuming $\sigma \gg 1$ and $\kappa = \alpha\sigma$ with some parameter $\alpha > 0$, the condition number of the discrete operator (5.2) is given by*

$$c_{Cond} \approx \frac{1+\alpha}{\alpha}\sigma^2 \tag{5.7}$$

Proof: Due to the assumption $\sigma \gg 1$ we may omit the transport operator in our considerations.

For the model case of constant phase function P, discretization (3.2) of equation (2.4) yields a scattering matrix

$$S = (\kappa+\sigma) - \Sigma_m = \begin{pmatrix} \kappa + (1-\frac{1}{m})\sigma & -\frac{1}{m}\sigma & \dots & -\frac{1}{m}\sigma \\ -\frac{1}{m}\sigma & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ -\frac{1}{m}\sigma & \dots & -\frac{1}{m}\sigma & \kappa + (1-\frac{1}{m})\sigma \end{pmatrix} \tag{5.8}$$

This matrix has two eigenvalues, κ with the eigenvector $(1,\dots,1)^T$ and the $(m-1)$–fold eigenvalue $(\kappa+\sigma)$.

Therefore we have

$$\|S\| = (1+\alpha)\sigma$$
$$\|S^{-1}\| = \frac{1}{\alpha\sigma}$$

The proof is concluded by applying the definition $c_{\mathrm{Cond}} = \|S\|/\|S^{-1}\|$. ■

A graphical representation of the eigenvalue structure is given in Figure 5.1 on the following page. Part a) shows the eigenvalues of the integral operator. These are smeared out by convolution with the mass matrix and addition of the transport part in b). Preconditioning confines them back to a smaller region in c).

5.2 Iterative Methods

The standard algorithm used in astrophysics for some years is the so called Λ–iteration (see Figure 5.2 on the next page). In numerics of integral equations, it is known as Picard–iteration too. Considering the whole discrete system,

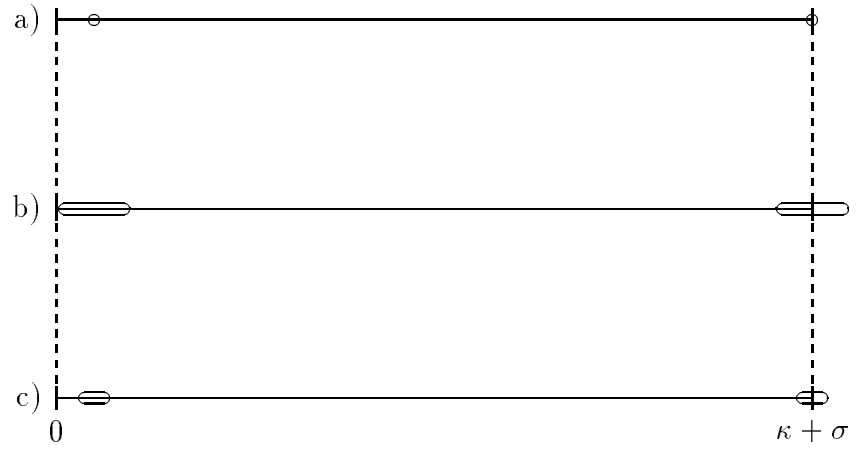


Figure 5.1: Eigenvalue distribution of the discrete system

$u^{(0)} := u_0$
for $i := 1$ to $steps$
    for k :=1 to m
        $v_k^{(i-1)} = \sum_{l=1}^{N} \omega_{kl} u_l^{(i-1)}$
    for k :=1 to N
        $u_k^{(i)} := T_k^{-1} v_k^{(i-1)} + F_k$

Figure 5.2: The Λ–Iteration

---

it is a Richardson method with nearly block–Jacobi–preconditioning. Using a full Jacobi preconditioner is a first step to better convergence rates as described by Turek in [36].

Since the transport operator T is inverted explicitly, these methods converge very fast for transport dominated problems. Exploiting the triangular matrix structure of upwind–discretizations, the inversion of T is indeed very cheap (one matrix–vector–multiplication).

Remark 5.1 Considering the streamline diffusion method the accurate inversion of T is not possible. An iterative procedure, solving to very high accuracy would slow down the solution process essentially (cf. the results by Führer in [15]). Here, we use an approximate inverse obtained by one multigrid step or a Gauß–Seidel step. The latter is clearly sensible to increase of the condition number due to refinement.

Unfortunately, in the interesting case of scattering dominance this method — as like as other stationary iterations — breaks down, since the condition number of the iteration matrix grows high.

Since the convergence rate of preconditioned Richardson iteration methods is only depending on the condition number, these are not suited for the scattering dominated case.

5.2.1 Krylov Space Methods

The eigenvalue distribution (5.8) shown in Figure 5.1 on the facing page proposes application of a Krylov space method. These methods minimize the iteration error over the affine space

$$X_\nu = x_0 + \mathcal{K}_\nu \qquad \text{with} \qquad \mathcal{K}_\nu = \operatorname{span}\Big\{A^\iota(b - Ax_0)\Big\}_{\iota=0,\ldots,\nu-1}$$

and admit the following error estimate:

Lemma 5.2 *The iteration error of the ν'th step is estimated by*

$$\|u^\nu - u\| \leqslant C\eta(\breve{A}, \nu) \tag{5.9}$$

with

$$\eta(\breve{A}, \nu) = \min_{\substack{p\in P_n \\ p(0)=1}} \max_{\lambda\in\sigma(\breve{A})} |p(\lambda)|$$

where $\breve{A}$ is the preconditioned system matrix.

Proof: Confer [28], page 33.

Since the eigenvalues have a large gap between the lower and the higher cluster, small polynomial degrees suffice to reduce the estimate. Considering only the scattering operator, the cg- or GMRES–algorithm converges in two steps (choose $p \in P_2$ as $p(x) = (\kappa - x)(\kappa + \sigma - x)$).

Indeed, the Bi-cgstab algorithm of van der Vorst (cf. [42]) proved rather promising in Turek's paper [36]. It usually reduces the errors much faster than stationary methods, but it shows a very irregular convergence history. We compare it to GMRES, the only method for non-symmetric systems,

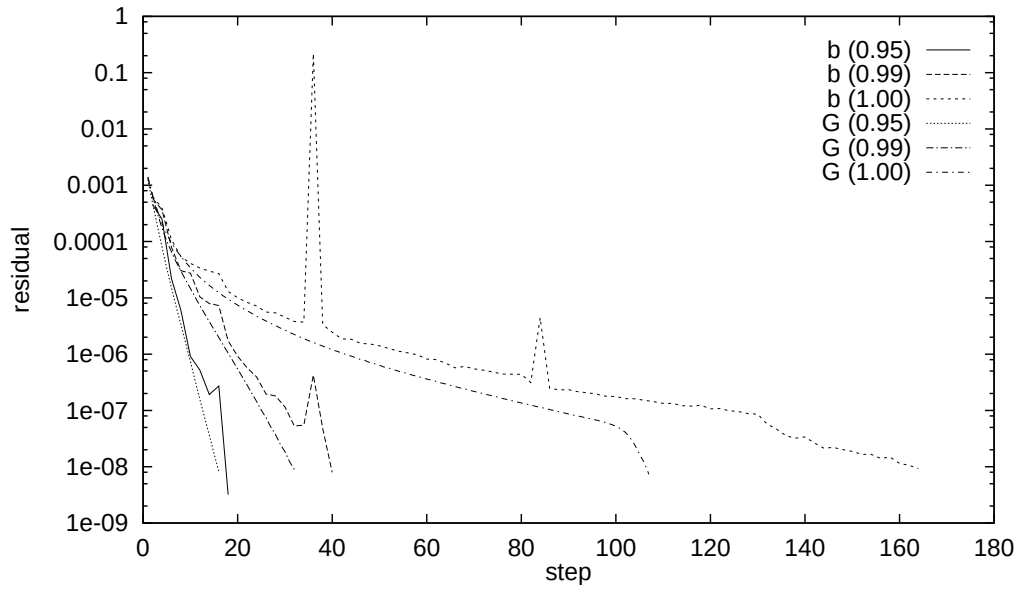


Figure 5.3: Comparison of Bi-cgstab and GMRES for different scattering parameters

---

which really minimizes the error in each step (thus, the only one for which the estimate (5.9) holds). The results are shown in Figure 5.3 for the Eddington problem of section 8.1. We display the norm of the residual over the number of matrix–vector–multiplications, since Bi-cgstab uses two of them per step. We observe, that GMRES converges faster for moderate ($\sigma = 0.95\chi$) scattering values as well as the equilibrium case $\sigma = \chi$. On the other hand the memory requirements of GMRES make us decide for Bi-cgstab, since GMRES needs one auxiliary vector for each iteration step, whereas Bi-cgstab needs a fixed number of eight additional vectors.

5.2.2 Preconditioning

The appropriate choice of a preconditioner is crucial for these methods to converge. The block–Jacobi–method mentioned above is a natural choice. It is exactly A^{-1} in the absence of scattering. For high values of σ, the Lanczos process filters out the low eigenmodes of the integral operator, while the preconditioner reduces those of the differential operator.

	$\chi = 0$			$\chi = 10, \sigma = 9$			$\chi = 1000, \sigma = 999$		
Level	Upw.	Gauß	MG	Upw.	Gauß	MG	Upw.	Gauß	MG
1	7	5	1	7	5	3	9	8	14
2	9	8	2	12	8	5	42	51	38
3	10	10	2	11	9	6	111	48	40
4	13	14	2	11	9	7	184	37	42
5	23	18	3	13	12	7	154+31	58	44
6	34	28	6	18	17	7	158+13	73	44
7	53	58	9	32	36	6	152+8	77	40

Table 5.1: Bi-cgstab Iteration steps regular refinement and constant coefficients (multiple numbers show breakdowns in Bi-cgstab)

	$\chi_{\max} = 64, \omega = .999$		$\chi_{\max} = 6400, \omega = .99$	
Level	Gauß	MG	Gauß	MG
4	0.11	0.025	0.58	0.29
5	0.19	0.085	0.66	0.35
6	0.35	0.112	0.68	0.35
7	0.59	0.249	0.65	0.35
8	$\geqslant 1$	0.394	0.64	0.39

Table 5.2: Contraction numbers for the dust cloud on regular grids

---

Exact inversion of T can not be applied to the streamline diffusion method. Therefore, we examine various preconditioning schemes for T.

First, we apply the upwind discretization (3.7) as a preconditioner to the streamline diffusion transport operator. A priori, we observe, that its approximation is of first order only. Therefore, we expect deterioration of convergence on finer meshes. This effect can be seen in Table 5.1. Note, that there are even breakdowns of the Bi-cgstab algorithm in the last column. Furthermore, this preconditioning method failed entirely, when we applied it to the dust cloud example and to three–dimensional problems.

Gauß–Seidel preconditioning is known to be a good method for convection equations. Sorting the points in downwind direction it converges faster than the first scheme. But it shows a strong dependency on the mesh width h too: for transport dominated problems the number of iterations is proportional to $1/h$, as shown in Table 5.1.

	$\chi_{\max} = 64, \omega = .99$			$\chi_{\max} = 640, \omega = .999$		
Step	Pts.	Gauß	MG	Pts.	Gauß	MG
1	289	0.101	0.034	289	0.550	0.429
2	565	0.592	0.069	570	0.985	0.564
3	1110	0.807	0.113	1143	0.994	0.543
4	2209	0.987	0.125	2280	$\geqslant 1$	0.717
5	4439	$\geqslant 1$	0.178	4622		0.645
6	9007		0.223	9243		0.676
7	18554		0.297	18462		0.667
8	38448		0.313	36905		0.694

Table 5.3: Contraction numbers on adaptive grids for a dust cloud

---

To avoid this behavior it is common practice for differential equations to apply a multi-grid technique. Since the Krylov space methods used rely on bi–orthogonal sequences of vectors, we have to choose between solving very accurate ($\|r\| \lessapprox 10^{-15}$) or doing a fixed number of steps. The second variant is much faster and shows the desired results (cf. Tables 5.1 to 5.3): The number of iteration steps does not grow anymore after a certain mesh size is reached (the slow down in the left column of Table 5.1 is due to the better resolution of sharp edges in the solution).

We compare the different preconditioning methods on regularly refined grids with constant coefficients in Table 5.1. The non–multi–grid schemes are considerably slower and upwind causes even breakdowns in the Bi-cgstab algorithm in the last column. Tables 5.2 and 5.3 show results for Gauß–Seidel and multi–grid preconditioners solving the dust cloud problem. Although in many cases, multi–grid convergence is not as good as in the elliptic case, Gauß–Seidel preconditioning is not stable enough to be sufficient. In particular, it diverges very early on adaptive grids.

Although multi–grid requires more computational effort than Gauß–Seidel, it is the only preconditioning method, which allows the solution of a wide spectrum of problems. Since in our parallel implementation preconditioning is executed simultaneously, the deficiency of computational cost is reduced on supercomputers.

5.3 Matrix Implementation

Considering the huge amount of unknowns involved in radiative transfer computations due to the three– to six–dimensional domain, the matrix storing technique is of great importance. We should also bear in mind, that in adaptive algorithms the solution of the linear system has a time consuming counterpart: assembly of the system matrix. We thus look for a method to reduce generation time and storage size.

The usual way applied to systems of differential equations, e. g. the Navier–Stokes–Equations, is the storage of a usual sparse matrix with block entries (4×4). Alternatively, often 16 sparse matrices are stored. Since our blocks have size $m \times m$ with m between 10 and 1000 this is not acceptable.

For our discretization we use the following

Lemma 5.3 *Let K_0 be a parallelogram of dimension d and the matrix A_0 the discretization of an arbitrary differential operators of order ω on K_0.*

For a cell K_ν resulting from the ν–fold regular refinement K_0 into similar cells, the according matrix is obtained by the scaling

$$A_\nu = 2^{\nu-\omega} A_0 \tag{5.10}$$

Proof: Immediate by observing that derivatives are $\mathcal{O}(2^\nu)$ and the integration volume is $\mathcal{O}(2^{-\omega})$. ■

Remark 5.2 A similar result is true for triangular cells, where single derivatives have to be replaced by their negative for the center child triangle. In the vicinity of curved boundaries, this lemma is not applicable, since division points on boundary edges have to be moved onto the boundary curve and the proof relies on the refinement into *similar* cells.

Application of Lemma 5.3 allows generation of element matrices without expensive integration. Since we have only artificial boundaries in our radiative transfer problems, even the restriction regarding boundary curves does not apply.

Recalling that the system matrix A of a finite element discretization is obtained by

$$A = \sum_K A_K, \tag{5.11}$$

we use this representation to execute the application of A to a vector. Here and in the following, the element matrices A_K are supposed to be of the same size as A thus referring to global node numbers. Instead of building A according to (5.11) and multiplying $v = Au$, we compute $v_K = A_K u$ and sum up afterwards. The loss of efficiency traded in by this method is given by

$$\frac{\#\text{entries per line in } A_K \times \#\text{cells per node}}{\#\text{entries per line in } A}, \tag{5.12}$$

which is $\frac{16}{9}$ and $\frac{64}{27}$ for quadrilateral and hexahedral meshes respectively.

This way we avoid the compilation of a global system matrix. Application of the matrix–scaling lemma 5.3 during matrix–vector–multiplication

$$v = \sum_K 2^{\nu_K - \omega} A_0 \; u$$

finally reduces the memory requirements for matrices nearly to zero. We have to accept a slowdown of matrix–vector–multiplication compared to a global matrix, but may reduce the memory needed for the whole program by a factor of four applying these techniques. This is especially important using parallel computers with small local memory.

Chapter 6

Parallelization

In this chapter we will discuss different ways of parallelization resulting from the structure of the radiative transfer equation. A recent work of Väth (cf. [40]) dealt with this problem, but due to the architecture used, a SIMD machine with about 8000 processors, the results are not applicable to main stream parallel machines of MIMD type like those at the IWR[1] in Heidelberg. Eriksson et al. found another approach for the time dependent neutron transport problem in [13]. The efficiency of their method relies on the combination of discontinuous Galerkin methods and the time stepping scheme, so it is not applicable to the stationary problem we investigate.

As the bilinear form in the radiative transfer equation consists of the sum of the two operators T and Σ acting on space and ordinate domain respectively, we may also choose between two parallelization strategies. In the first section we refer to the problems, a spatial splitting of the whole domain causes. The second section describes in detail the ordinate parallelization we apply. We consider its effect on the linear solvers of Chapter 5. The efficient way of implementing parallel algorithms using distributed objects in C++ is presented in the third section. We conclude this chapter by deriving theoretical estimates for the efficiency and compare them to results obtained on the parallel computers at the IWR.

[1] Interdisziplinäres Zentrum für wissenschaftliches Rechnen — interdisciplinary center for scientific computing

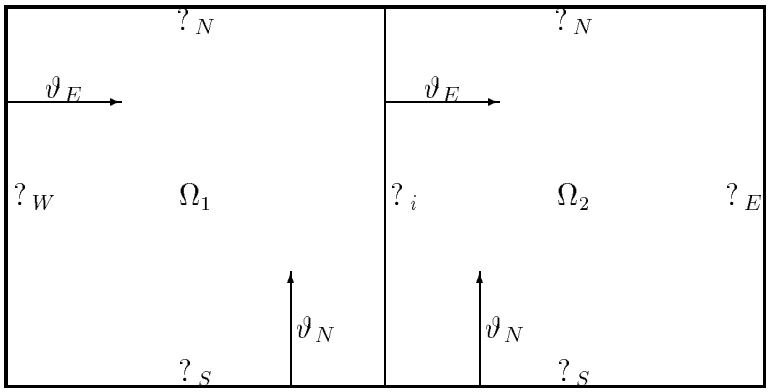


Figure 6.1: Domain decomposition for transport problems

6.1 Domain Splitting Strategies

Convection dominated problems differ in one specific point from elliptic problems: There is a distinct direction of information flow. This has to be considered in the development of parallelization strategies. While a domain decomposition for Poisson's equation should minimize the length of interior edges, this does not yield an efficient method for convection.

Consider a preconditioning step on the simple domain in Figure 6.1 and regard transport direction ϑ_N. Each processor starts inverting the transport operator at $?_S$ and pushes information towards $?_N$. Since we use continuous finite element spaces, there should be some information interchange across $?_i$. This can be done by iteratively averaging inner boundary values at $?_i$.

Following lemma 3.12 the error induced at $?_i$ decays exponentially to the interior of Ω_1 and Ω_2. Therefore we expect good convergence.

Looking now at direction ϑ_E we are confronted with a changed situation. While processor 1 starts at the inflow boundary $?_W$ and produces a rather accurate solution in the first step, processor 2 starts with random boundary conditions. It will produce useful results in the second step, where processor 1 reproduces the result of the first step.

We conclude that parallelization strategies for transport equations should not divide the domain across the transport direction.

The solution of the radiative transfer equation consists of a bundle of transport inversions for different directions. The construction of an efficient domain decomposition method would — following the above argument — require a direction dependent splitting of the domain Ω. Since this produces immense implementation problems, we decided to look for another way of parallelization.

6.2 Ordinate parallelization

The second strategy distributes the ordinate space S^2 of the radiative transfer equation. Since we use discontinuous shape functions for the ordinate variable and there is no local coupling due to the integral operator, this results in a true non–overlapping parallelization. Not even boundary points have to be averaged over different nodes.

Clearly, it has disadvantages too: as the integral is a global operator, ordinate parallelization involves global communication.

Considering the somewhat extreme case of just one ordinate per node, we explain its effect on the two steps of the Λ–iteration (Figure 5.2).

1. Evaluation of the integral sums involves — since ω_{kl} in (5.3) depends on the source and the destination ordinate — a communication sweep, where each processor gets the data of each other one. This can be achieved in m steps parallel compared to m^2 steps for a sequential code.

2. Inversion of the transport operators is a parallel task without any communication.

The application of more sophisticated methods like Bi-cgstab and GMRES bases on these operations too, augmented by some vector scalings and additions (full parallel) and scalar products (involves global collection).

When we decide whether to use ordinate parallelization, we have to answer an important question: does it offer a means for highly parallel computing or can we just occupy a moderate number of processors?

π_1	π_2	$\pi_3 \ldots \pi_{m-1}$	π_m
$u_1, \ldots, u_{l_1}$	$u_{f_2}, \ldots, u_{l_2}$	$\ldots$	$u_{f_m}, \ldots, u_{l_m}$
$v_1, \ldots, v_{l_1}$	$v_{f_2}, \ldots, v_{l_2}$	$\ldots$	$v_{f_m}, \ldots, v_{l_m}$

Figure 6.2: Two objects of a distributed vector class on m processors

---

The answer depends on the number of angles required to approximate the solution. In two–dimensional test computations, about 30–60 ordinates showed to be necessary for constant phase function. Extrapolating to the three–dimensional problem, 1280 ordinates mean 40 points on a great circle of S^2. So even putting several angles on one node, this leaves work for some hundred processors. Thinking of the rapidly changing phase function of Mie–scattering, there should be enough work for systems installed in the next years.

6.3 Distributed objects

An important question arising from the development of parallel code is the encapsulation of the parallelism. Since computer clusters have very different programming interfaces and runtime characteristics (cf. section 6.4), changes between platforms should affect as few as possible portions of the code. The usage of higher level standard libraries often reduces efficiency, since machine characteristics cannot be used. Here, the object oriented programming concept of information hiding may be very helpful.

Consider e. g. the situation of Figure 6.2. The underlying vector class has the following features:

- Each vector data spans over computing nodes π_1 to π_m.
- Each node π holds the consecutive vector components $u_{f_\pi}, \ldots, u_{l_\pi}$.
- There is a mapping between global vector indices i and local indices (π, ι) defined by $i = f_\pi + \iota$. Assuring $f_\pi = l_{\pi-1} + 1$ this mapping is

bijective.

We see that addition and scalar multiplication of these vectors is purely local, since only corresponding indices of both vectors are involved. A global operation is e. g. the scalar product $u \cdot v$. We implement this product by evaluating the local scalar products and summing up by data exchange.

From the abstraction level of the operations named above, this parallel vector behaves like a usual one. It can be fed into an iterative solver just like any other vector (using a suitable parallel matrix).

On different systems, the data collection of the scalar product should be implemented according to machine characteristics, e. g. using a binary tree, hypercube or other topology. Due to information hiding (the solver does not know about the implementation of a scalar product, it just expects a certain result), the changes in the code are very small and local to vector routines.

The application of these concepts to our vector classes allow a safe parallelization even of the adaptive algorithms described in Chapter 4. Verification of the code is always possible on sequential and parallel computers, which is important to localize errors due to the parallel implementation.

6.4 Parallel systems

Since the first steps in distributed computing, numerous parallelization paradigms have been developed. The most important are:

SIMD **S**ingle **I**nstruction **M**ultiple **D**ata, all processors are synchronized on instruction level and perform the same code on different data. Typical systems as the MasPar or CM-2 contain some thousands of very simple processors.

MIMD **M**ultiple **I**nstruction **M**ultiple **D**ata, each processor can perform independent tasks. This is e. g. the idea of workstation clusters running PVM.

SPMD **S**ingle **P**rogram **M**ultiple **D**ata, a mixture of SIMD and MIMD, where the same program is automatically loaded on each computing

node. Examples are the Parsytec machines SC-T805 and PPC-GC with PARIX environment.

Since MIMD and SPMD have similar concepts and may be mutually emulated, we will refer to both as MIMD.

A second difference between parallel systems is the organization of memory and the resulting method of data exchange:

Shared memory: all processors have access to the same address space of the machine. Data exchange is done by just reading memory written by another processor. Due to bus access conflicts, this model allows only moderately parallel systems (up to about 32 processors).

Distributed memory: each processor has its own address space. Data have to be transmitted explicitly from one node to another (*message passing*).

Fortunately, the development seems to converge a bit in the last years. The SIMD paradigm has nearly vanished due to its disadvantages from the programming and construction point of view. All highly parallel machines use message passing systems, although there is a trend to multi–processor nodes.

Regarding the parallel operating systems, there is a standardization too. After systems like PARIX (for Parsytec computers) and PVM (for Workstation clusters) have been developed independently, system providers now try to evolve standards for message passing functions like MPI.

But even considering just MIMD message passing systems there is a wide variety of platforms showing totally different behavior. The overall execution time of a parallel program is determined essentially by three factors. Clearly, in numerical applications the *floating point performance* (measured in FLOPS, floating point operations per second) of the processor is of high importance. Algorithms exchanging a big number of small data packages are sensitive to the *communication startup time* (sec). Programs passing huge blocks of data rely on a high *communication bandwidth* (bytes/sec).

```
read_data;
create_grid;
initialize(matrix, vectors);
solve(matrix, vectors);
write_data;
```

Figure 6.3: A simple solution program

It is more useful to use values normalized to double precision arithmetic to investigate the efficiency of a program:

$$t_s = \text{startup time} \cdot \text{FLOPS}$$
$$c_b = \frac{\text{bandwidth}}{8\text{FLOPS}}.$$

Usually, computer producers provide for some values of these quantities, called peak performance. Due to non–optimal compilers, problem structure and operating system overhead, these values are hardly reached. We consider the actual values, occurring from our application.

At the IWR in Heidelberg we have access to two parallel systems: the older INMOS Transputer T805 based SuperCluster (SC-T805) with 128 nodes and the GigaCluster consisting of 96 nodes with two PowerPC 601 processors each (PPC-GC). While communication and computing power are well balanced on the SC-T805 with $c_b = 0.8$, this ratio is $c_b = 0.2$ on the PPC-GC.

6.5 Efficiency Considerations

Before analyzing the differences in execution time, we develop a model to interpret these data. Runtime results are then given for the whole program on different platforms and for the iterative solver.

6.5.1 Time Complexity Model

First we consider the non–adaptive program shown in Figure 6.3. For our ordinate parallelization all operations with exception of the solver are truly

parallel, i. e. avoid communication. The program runs on all nodes without synchronization, until initialization is complete. Any delay occurring with respect to the sequential version has to be due to operating system overhead and file data access. Since we made sure that file data is small, this should be negligible (see Table 6.1). The time on the Parsytec machine begins to grow at 16 processors, since the loading of the program itself consumes much time due to the low performance of data exchange between host and parallel computer.

We now focus on the linear solvers. As for the single level iterations, they consist of the following classes of operations:

1. Application of the radiative transfer operator
2. Preconditioning
3. Scaled vector additions
4. scalar products

In our implementation, classes 2 and 3 are truly parallel, communication only occurring in 1 and 4. To compute scalar products, each nodes collects the product for its part of the vector and then communicates just *one* number. Clearly, on suitable machines this data exchange is negligible for vector lengths of some 70.000 entries.

That leaves us with the analysis of matrix–vector–multiplication. Considering the matrix structure of (5.2) on page 54, the parallel version of this operation consists of two parts:

1. the multiplication of $\mathfrak{T}_k + \mathfrak{M}_k(\chi) - \mathfrak{M}_k(\sigma)$ with the local vector components u_k and
2. the addition of $\{u_j\}_{j \neq k}$ of all other components and multiplication of the sum with $\mathfrak{M}_k(\sigma)$.

Whereas the first part is inherently parallel, the sum in the second is the only part of the program, which causes communication. We show the sequential algorithm and an optimized parallel version in Figure 6.4 on the next page.

```
1   for k := 0 to m
2       v_k := 0;
3       for j := 0 to m
4           v_k := v_k + ω_kj u_j;
5       w_k := 𝔐_k(σ) v_k;
```

```
1   h_out = v_p;
2   v_p = 0;
3   for k := 1 to m
4       start_send(p + 1, h_out);
5       start_receive(p − 1, h_in);
6       v_p := v_p + h_out;
7       wait_comm();
8       h_out := h_in;
9   v_p := v_p + h_out;
10  w_p := 𝔐_p(σ) v_p;
```

Figure 6.4: Sequential and parallel matrix–vector–multiplication

system	# procs	sec
PPC-GC	4	26
PPC-GC	8	26
PPC-GC	16	30
PPC-GC	32	32
SPARC 10/51	1	26
RS6000 PPC	1	23

Table 6.1: Initialization times for 70.000 nodes (2D) in seconds

---

For this method, the processors are located in a ring topology, i. e. $p \in \mathbb{Z}_\pi$, where π is the number of processors. This way, each node has two neighbors ($p+1$ and $p-1$) to communicate with. By doing the communication (lines 4, 5 and 7 on the right) and vector addition (line 6) in parallel, the efficiency of the algorithm is bounded from below by c_b if $c_b < 1$ and is about 100% if $c_b \geqslant 1$, since nearly the same operations have to be done, but only m^2/ν times instead of m^2 times in the sequential version.

6.5.2 Results

First, we compare initialization times — generation of triangulation, operator and right hand side — of Table 6.1. These should be constant, since the amount of work is proportional to the number of processors and everything

ordinates	$1 \cdot p$	$2 \cdot p$	$4 \cdot p$	$8 \cdot p$	$16 \cdot p$	$32 \cdot p$
processors						
2	1.97	3.02	5.12	9.33	17.75	34.60
4	2.04	3.09	5.19	9.40	17.83	34.69
8	2.17	3.23	5.32	9.53	17.97	34.82
16	2.46	3.51	5.61	9.87	18.23	35.08
32	3.02	4.08	6.20	10.18	18.81	35.67
64	4.17	5.22	7.31	11.52	19.96	36.82

Table 6.2: Time for Bi-cgstab on the SC-T805 in seconds (1 step, 280 points)

ordinates	$1 \cdot p$	$2 \cdot p$	$4 \cdot p$	$8 \cdot p$
processors				
2	6.6	8.5	13.2	23.6
4	7.1	9.3	14.1	24.6
8	9.3	11.8	16.4	28.0
16	13.0	15.1	20.1	30.8
32	20.0	22.9	28.3	38.9
64	34.6	37.5	42.4	57.9

Table 6.3: Time for Bi-cgstab on the PPC-GC in seconds (1 step, 70.000 points)

---

is done in parallel. The slight growth of this time on the PPC-GC is due to the slow loading of the code itself onto the parallel machine.

In Tables 6.2 and 6.3 we compare the execution time for one Bi-cgstab–step. On the Transputer system, we could only store about 300 space points due to local memory restrictions. In each column, the problem size is scaled with the number of processors. Looking at the table for the Transputer system, we see, that the execution time is nearly independent of the number of processes, if there are at least four ordinates on each node. This corresponds to nearly 100% efficiency. If the load of each processor is smaller, efficiency decays. Using one ordinate per node, there is only a slowdown of two using 32 times as many processors corresponding to an efficiency of 50%. These excellent results can be achieved, since communication bandwidth and computing

power are balanced on this system, i. e. $c_b \approx 0.8$.

On the PPC-GC floating point operations are about twenty times faster than on the SC-T805. Since communication velocity has increased only by a factor of four, the considerations of the last subsection predict a drop of efficiency. Indeed, in the first column, the elapsed time grows by a factor of five (20% efficiency) and even for eight ordinates per node, there is a growth of execution time by a factor of two corresponding again to an efficiency of 50%. According to the time complexity, the lower bound of the efficiency is about 20%, but obviously, the preconditioning time suffices to balance it in the last column.

Since the relatively slow communication of the PPC-GC is peculiar among modern parallel computers, the performance of our algorithm can be considered sufficient. Even on this computer, the solution of the applications in Chapter 8 is accelerated to an amount, where experimenting with parameters is possible in acceptable time.

Chapter 7

Software Development

Scientific computing software to solve "real life" problems tends to become more and more complex. The necessary validation of program code becomes a hard task by this development. A proof of correctness is possible only for simple data structures and small programs. The classic approach to more reliable software used in numerical computations is *modular programming*. Applying this paradigm, a complex algorithm is decomposed into small simple parts which can be tested independently. But, as mentioned above, not only methods are complex, but data structures too. *Object oriented programming* now allows the modularization of algorithms and data by combining both of these aspects into the same structure. This leads to the notion of *classes* and *objects*.

The tight coupling of data and methods operating on them in object oriented programming, forces design to become an important step in software development. In previous numerical codes, this has been regarded only with the aim of reducing memory usage and computation time.

In a complex piece of software, design of classes has to optimize a combination of four development aims:

1. computing speed,
2. memory requirements,
3. verifiability of code and
4. flexibility.

While the first two points have been investigated very thoroughly in the last 30 years — see e. g. BLAS routines — points three and four are real white patches on the map. Additionally to the known trade–off between speed and memory consumption points three and four introduce a much more complicated balance. From the economic point of view we have to supplement run–time efficiency by development efficiency. In particular, software designed to develop new algorithms must obey this point, since implementation time usually exceeds run–time by orders of magnitude.

The main idea to improve flexibility and correctness of a program lies in the restriction of data access. In usual FORTRAN or C code a great deal of data is handled by common blocks and global variables, respectively. These allow unrestricted read and write access to their members, so the programmer can not be sure where these structures are changed or corrupted. This verification problem is augmented by the fixed structure of implementation: a change in the data structure necessitates a change in all functions using it.

Considering application to partial differential equations, there are several parts of the algorithms which may be separated to a high degree. There is a base level of classes describing the adaptive grid generation and handling of multi–grid structures. This part of software is invariant for a huge class of finite element problems. A second level on top of the first defines basic numerical operations like application of an operator to a discrete function. Here, the dependence on the structure of the physical problem is very strong. A third part of code provides standard numerical solvers as cg and Bi-cgstab for linear systems and Newton's method for nonlinear problems. This level should be implemented in an abstract way to allow usage for different applications.

We would like to illustrate these concepts with the implementation of DEAL (cf. [5]), a C++ class library for finite elements developed by the author, Franz–Theo Suttmeier and Roland Becker, as well as its application to radiative transfer problems.

7.1 Grid Handling

Let us first regard the abstract concept of a triangulation of a domain Ω as a hierarchy of grid cells with a certain topology. Its basic functionality is

```
class Triangulation
{
    void      read(File);
    void      refine(int levels);
    void      adaptive_refine(double tolerance);
    Cell*     first_cell();
    Cell*     next_cell();
    Vertex*   first_vertex();
    Vertex*   next_vertex();
};
```

Figure 7.1: The Triangulation class

shown in Figure 7.1, where the triangulation is reduced to a set of cells and vertices with some additional mesh generation functions. This mesh should be able to consist of triangles and quadrangles as well as tetrahedra, prisms and hexahedra in two and three dimensions respectively. This means that we have to extract basic information from all these geometric objects to identify them as a *cell*. The cell information necessary for adaptive refinement was first collected in [34] in a very abstract way. This led to the definition of a cell essentially following Figure 7.2 on the facing page.

The function *refine* of Triangulation works by traversing all cells and forcing each cell to refine itself. Since a correct triangulation is characterized by consistent values of vertices, neighbors and father/child information, *refine* and *coarse* function of Cell ensure conservation of these data. Certainly, a cell can hardly supply any of these functions, since topology information is specified only for more concrete objects. Therefore, they are declared as an abstract interface and are implemented in derived classes for e. g. triangles and quadrilateral cells, where we know the number of neighbors is three or four respectively.

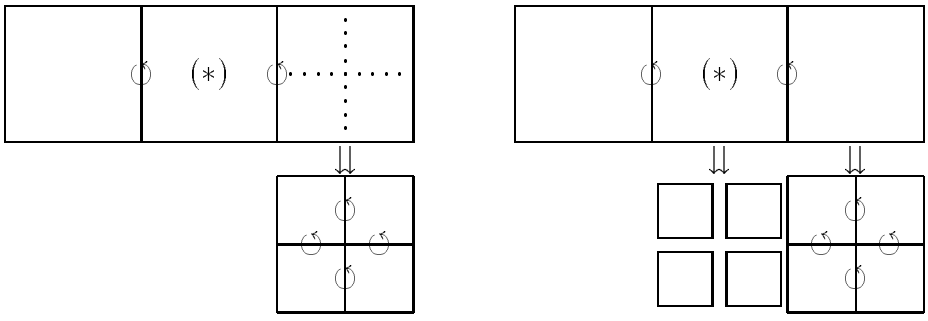
The functional interface to these values enables high flexibility regarding implementation. Considering e. g. the children there are two possibilities of obtaining the corresponding pointers. They may be stored in a cell producing memory consumption, but making fast access possible. Alternatively, they can be reconstructed from the list of cells stored in the triangulation. According to the algorithms used as well as preference for memory or speed

```
class Cell
{
    int         number_of_vertices();
    int         number_of_neighbors();

    Vertex*     vertex(int nr);
    Cell*       neighbor(int nr);
    Cell*       father();
    Cell*       child(int nr);

    int         level();
    int         index();

    void        refine();
    void        coarse();
    double      refinement_criterion();
};
```

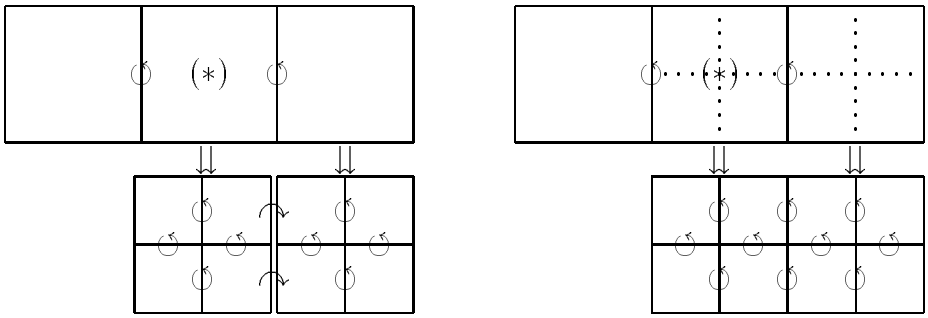
Figure 7.2: The Cell class (topology and refinement)

optimization, these techniques should be chosen appropriately. The interface of Figure 7.2 allows the necessary internal change of representation on a purely local basis, since implementation details are hidden for other functions. The correctness of working code using this function *child* is not influenced by internal changes, if only the result of *child* is correct.

We ensure for each function, that all objects involved in its operation — not only explicitly modified objects — are in an admissible state after the function returns control to the calling context. Take e. g. cell refinement as shown in Figure 7.3 on the following page. We start with a valid locally refined triangulation (a). The cells are linked by the mutual neighborship relation $\circlearrowleft$ and the "is child of" relation $\downdownarrows$. We display two consecutive layers of refinement on top of each other. After splitting the middle cell $(*)$ into four children, the topology is corrupted (b). Setting topology information for the children (c), the cell we operated on is valid, but the operation has destroyed the triangulation structure: there is only an unidirectional neighborship relation $\curvearrowright$ from the children of $(*)$ to their neighbors. Finally, the topology information of neighboring cells is updated too and the topology is valid again (d). Compared to other strategies, this process not only avoids



(a) Starting grid with 1 refined cell (b) Children generated



(c) Topology of children (d) Refined grid

Figure 7.3: Refinement of a cell

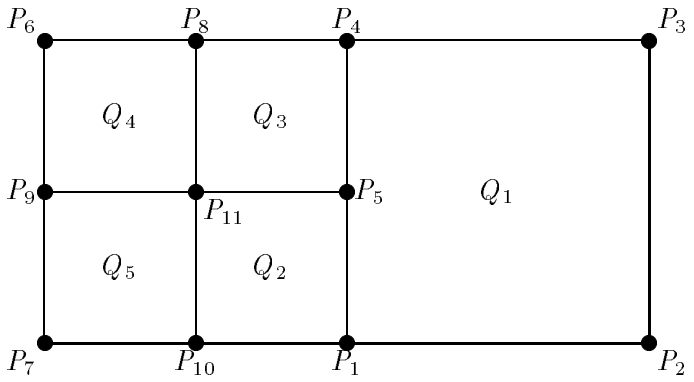


Figure 7.4: A hanging node

expensive postprocessing, but enables us to stop refinement after processing an arbitrary cell (which is actually done in adaptive algorithms). While the states (b) and (c) are "virtual" grids never to be seen in an application, (a) and (d) represent states, where further operations may be inserted.

7.1.1 Refinement Edges

Refining a grid locally, there are edges between different levels of refinement and the problem of "hanging nodes" arises. Using special interface cells not only disturbs the topology of the grid hierarchy, but causes difficulties implementing accurate grid transfers. We decided to choose numerical treatment of these nodes. For theoretical treatment of our methods, we use the formalism of "hierarchical bases" developed in [43].

For sake of simplicity, we will refer only to bilinear elements in the generic situation of Figure 7.4. It is obvious from the construction, that the method applies to all conforming shape functions.

Cell Q_1 needs five shape functions, which are denoted in the hierarchical basis notation

- the 4 usual bilinear shape functions $\varphi_1 \ldots \varphi_4$ on a quadrilateral cell and
- 1 additional function φ_5 satisfying

- the Lagrange interpolation condition $\varphi_5(P_i) = \delta_{5i}$ for all points of Q_1,
- $\varphi_5 = 0$ on the non–divided edges of Q_1 and
- φ_5 is linear on the edges $Q_1 \cap Q_2$ and $Q_1 \cap Q_3$

We remark, that we do not imply any knowledge of the behavior of φ_5 in the interior of Q_1. It should be chosen to offer good interpolating properties. Outside Q_1, on Q_2 and Q_3, φ_5 has the shape of a standard bilinear finite element function.

While the hierarchical basis approach is natural for Q_1, from the point of view of cells Q_2 and Q_3, the shape functions in P_1, P_4 and P_5 should be in nodal representation. We need a transformation between the hierarchical and nodal base functions $\{\varphi_i\}$ and $\{\psi_i\}$. For the construction, we evaluate a function $u_h = \sum u_i^H \varphi_i = \sum u_i^N \psi_i$ in the mesh points:

$$\begin{array}{rcccl} u(P_1) & = & u_1^N & = & u_1^H \\ u(P_4) & = & u_4^N & = & u_4^H \\ u(P_5) & = & u_5^N & = & u_5^H + \frac{1}{2}u_1^H + \frac{1}{2}u_4^H \end{array} \tag{7.1}$$

This results in the matrix C of basis exchange (only considering the interesting points)

$$C : u^H \mapsto u^N \tag{7.2}$$

$$\begin{pmatrix} u_1^N \\ u_4^N \\ u_5^N \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \frac{1}{2} & \frac{1}{2} & 1 \end{pmatrix} \begin{pmatrix} u_1^H \\ u_4^H \\ u_5^H \end{pmatrix} \tag{7.3}$$

At this point we have to decide, whether there should be a degree of freedom at point P_5. First, we consider the case of adding this degree of freedom. Interpolation estimates for Q_2 and Q_3 stay the same, while they are not worse (compared to the standard case) for Q_1. As we use the complete matrix in nodal representation, the element matrices of Q_2 and Q_3 have the correct form. We have to modify the element matrix of cell Q_1. We build it as usual

$$a_{ij}^H = a\Big(\varphi_i, \varphi_j\Big)_{Q_1} \qquad i,j = 1,\ldots,5$$

The implementation of this algorithm is simplified by the following remark:

Remark 7.1 The hierarchical basis matrix with additional hanging nodes consists of two parts. A sub-matrix corresponding to the usual nodes coincides with the standard element matrix without hanging nodes. For each hanging node, it is added one row and column due to the extra base function.

To get the nodal representation required to build the global matrix, we apply the basis transfer

$$A^N = CA^HC^T \tag{7.4}$$

Alternatively, the node P_5 could be omitted. This has algorithmic advantages in the use of element matrices. In particular, R. Becker has developed a highly efficient multi–grid algorithm based on this technique in [4].

In this case, we have to deal with the problem of node P_5 occurring in the matrices of Q_2 and Q_3 but not in Q_1. It is of algorithmic advantage to let point P_5 be part of the mesh, so we search a treatment on matrix level. The hierarchical representation of $u(P_5)$ is a natural choice. The desired behavior is achieved by setting $u_5^H = 0$ and thus omitting base function φ_5. This way of cancelling node values at P_5 clearly involves no modifications of the right hand side, as would be necessary in nodal representation. Accordingly, we have to apply the inverse base transformation as in (7.4), where C is deprived of the third row. (7.3):

$$\begin{pmatrix} u_1^H \\ u_4^H \end{pmatrix} = \begin{pmatrix} 1 & 0 & \frac{1}{2} \\ 0 & 1 & \frac{1}{2} \end{pmatrix} \begin{pmatrix} u_1^N \\ u_4^N \\ u_5^N \end{pmatrix}$$

This conforms to first averaging u_1^N and u_4^N to u_5^H, then applying the operator A and finally distributing the value of u_5^H to the neighboring points. The corresponding changes of base have to be applied to the right hand side generated by finite element integration:

$$\begin{aligned} f_1^H &= f_1^N + f_5 \\ f_4^H &= f_4^N + f_5. \end{aligned}$$

Method	rich	cg	Bi-cgstab	cr	GMRES	QMR
$v = Au$		*	*	*	*	*
$d = b - Au$	*	*	*	*	*	*
$v = \tilde{A}^{-1}u$	*	*	*	*	*	*
$v = A^T u$						*
$v = \tilde{A}^{-T}u$						*
$\alpha = u \cdot v$		*	*	*	*	*
$v = \alpha u$		*	*	*	*	*
$v = \alpha u + \gamma w$			*			*
$v = v + u$	*					*
$v = v + \alpha u$		*			*	
$v = \alpha v + \beta u$		*		*		*
$v = \alpha v + \beta u + \gamma w$			*			

Table 7.1: Common iterative solver interface

7.2 Linear Solvers

In section 5.3 we pointed out the importance of a sophisticated, problem adapted matrix implementation. Furthermore, special programming techniques like parallelization require the usage of vectors with non–standard behavior. On the other hand, different problems require different linear solution methods. Especially considering non–symmetric linear systems, there have been developed several methods. Since — with exception of GMRES — there is no sufficient theoretical justification for these methods, they have to be chosen by trying for a special problem.

Since the implementation of new iterative solvers like GMRES or QMR methods with look–ahead is rather complex, we consider it an important feature of a finite element programming library to provide a means for easily testing iterative methods. The methods should be provided by the library and use problem dependent matrix–vector and vector–vector operations like those described in sections 5.3 and 6.3. To allow a high degree of optimization, we propose the rather fat interface shown in Table 7.1. From this table it is clear, that the solvers have different requirements to matrices and vectors. We implement the iterative methods as function templates, taking matrix and vector types as template arguments. Using an abstract matrix class

would not be a good idea since the user would have to provide for even the unneeded functions of the interface. Additionally, the vector type would not be free and there would be a lot of insecure casting operations or run time checking, which slows down development of new codes considerably.

Chapter 8

Applications

Development of astrophysical models is especially difficult since experimental possibilities are very restricted. There are no ways to change parameters on an existing system and it is also impossible to observe an object from different directions if it is not rotating. So we are restricted to the information we obtain by electro–magnetic emission of distant objects in the direction of earth, in particular visual light, infrared and radio waves. A typical setting for these problems is shown in Figure 8.1. Here, the observed system is

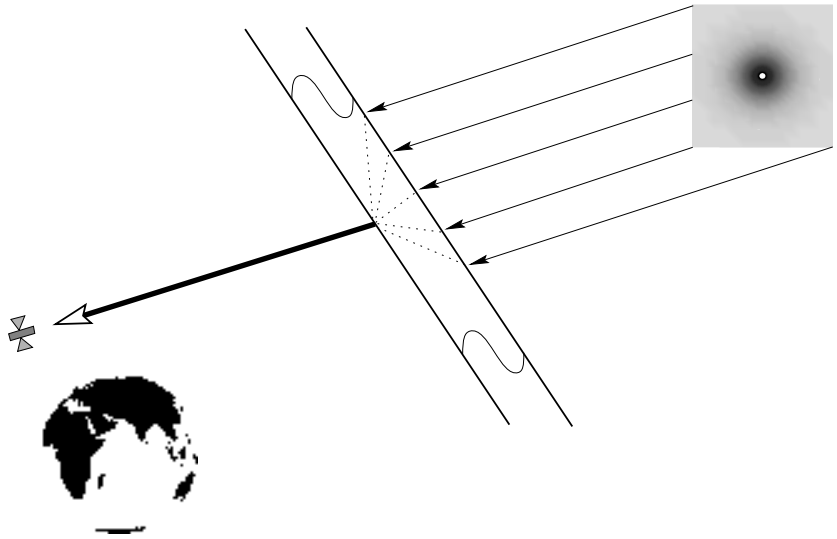


Figure 8.1: A typical observer situation

supposed to be very far from earth. Even with powerful telescopes angular resolution of the object is not possible, it appears as a point source. The only information we get is time and wavelength dependency of electro–magnetic radiation.

Verification of existing models of distant objects is only possible by numerical simulation and comparison with measured data. In the second section, we do such calculations for the problem of a circumstellar dust cloud. On the other hand, simulation allows to change model parameters to get more insight into physical processes. This effect is the aim of our simulation in the first section on Eddington luminosity. Even in a simple geometric situation, a higher dimensional simulation reveals effects not observable in the one–dimensional case.

8.1 Eddington Luminosity

Up to now, models based on one–dimensional calculations predict a maximum luminosity of a star, the so called Eddington luminosity, which cannot be exceeded. A higher radiation flux would tear the star apart, since radiative forces would exceed gravitation. Actually, there are objects observed, e. g. novæ, which seem to emit much more radiation, than predicted by this theory.

The crucial ingredient to allow these one–dimensional calculations is horizontal homogeneity. Now there are considerations, that by omitting this homogeneity a much higher luminosity is possible. On the other hand, if explosions occur, they could be local and do not destroy the star.

We made comparative simulations to investigate how the maximum radiative pressure gradient and emitted radiation depend on inhomogeneities of the material. The setting for these calculations is shown in Figure 8.2 on the following page. We consider a smooth layer spreading horizontally with an oscillating opacity

$$\chi(x,y) = \cos(\pi y) * \left(\frac{\chi_{\max} + \chi_{\min}}{2} + \frac{\chi_{\max} - \chi_{\min}}{2} \cos(\pi x) \right).$$

This layer is illuminated by upward radiation from the interior of the star. The results of these calculations consist of

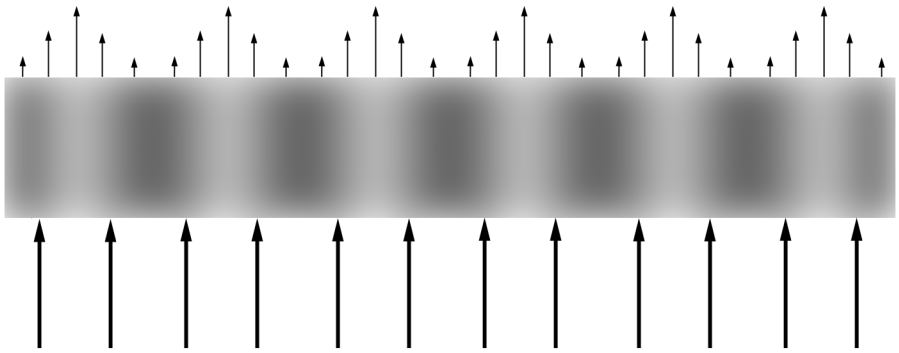


Figure 8.2: Model configuration to investigate Eddington luminosity

		scatter = 99.99%			scatter = 90%		
$\chi_{\min}$	$\chi_{\max}$	$F_{\max}$	E_N	$E_N/F_{\max}$	$F_{\max}$	E_N	$E_N/F_{\max}$
99	101	0.00018	0.0009	25	0.71	0	0
50	150	0.00017	0.005	28	0.80	0	0
9	191	0.00015	0.019	100	0.91	10^{-5}	10^{-5}
1	199	0.00022	0.086	391	0.93	0.036	0.39

Table 8.1: Radiative force and emitted radiation

---

1. the radiative force field $F_R(x) = \int_{S^2} \vartheta\, I(x,\vartheta)\, d\vartheta$ with its maximum value

2. and the mean radiation emitted to the top $E_N = \int ?_N I(x,\vartheta_N)\, dx$.

This example is of moderate computing complexity, since all coefficients are smooth and even the solution does not have strong jumps. Therefore, it is a suitable application to verify the discretization and solution algorithms of our program without introducing additional difficulties due to localized features.

We show the relation between maximum radiative force in the interior of the domain and emitted radiation in Table 8.1. Values for different variations of the extinction and for two different albedos are given. We see, that in the case of high albedo, the quotient of E_N and $F_{\max}$ grows about a factor of 15 compared to the inhomogeneous problem. If scattering is smaller, this growth is even more dramatic. In the cases of a homogeneous matter distribution,

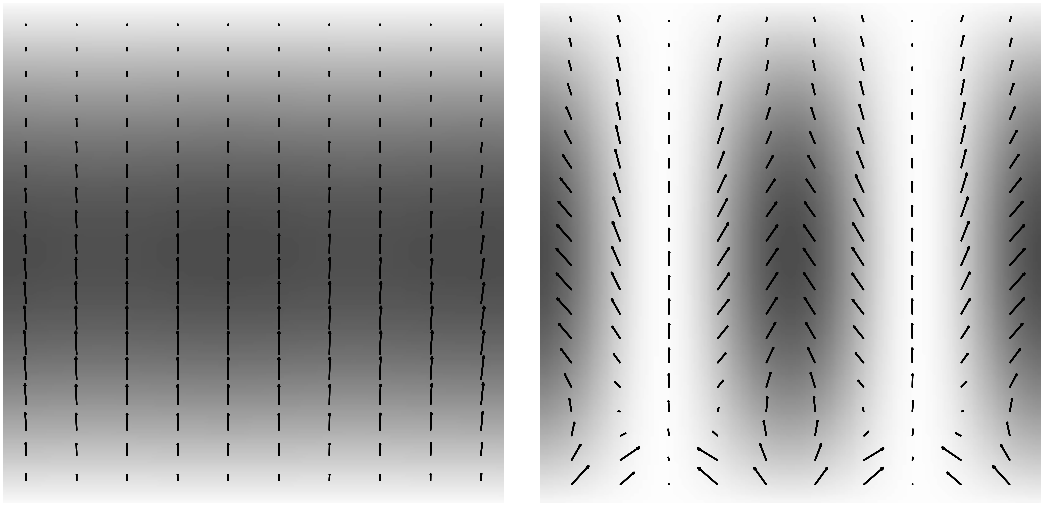


Figure 8.3: Radiative pressure field for small and large variation of χ.

---

the simulation resulted in a luminosity of zero and only in the last line, there is a significantly positive value of E_N.

Figure 8.3 shows the radiative pressure field for the two extremal configurations $\chi_{\min} = 99$ and $\chi_{\min} = 1$. The background color shows the opacity χ with darker grey for higher opacity. While the field is always directed upwards in the case of small variation, it has additional structure in the other case. It is an interesting result, that radiative forces are directed towards regions of higher density. This feature could result in a destabilization of the layer structure such that a homogeneous layer is not a stable configuration. Our simulations should be combined with a model of hydrodynamics to understand the development of such a layer.

8.2 Dust Enshrouded Stars

Many stars, especially young ones are surrounded by dense clouds consisting of dust and gases. These clouds are heated by radiation from the star and emit themselves light of larger wavelengths due to Planck's formula (2.3). The temperature field in the cloud is given by the energy equation in (2.11).

The geometry of the cloud is enforced by hydro-dynamical processes. Around

a central star, there is a region, where due to the high temperature no dust can exist. This hole is about ten radii of the star. Outside this area, there is dust of a high density decaying with the square of the distance to the star.

The aim of this work was to accelerate the solution of the monochromatic radiative transfer equation, to allow the solution of the full system in a further step. We reach this aim by the adaptive and parallel algorithms described in Chapters 4 and 6, respectively. The refinement history in Figure 8.4 on the next page shows four meshes generated during the adaptive refinement process.

With our new adaptive approach we can predict the discretization error of our method. For the first time, this estimate is sharp in the sense that the true error is overestimated by a factor below ten on sufficiently fine grids. We achieve this by using local weights (local in space and ordinates) obtained from the approximation of the second derivative of the dual solution as described in Chapter 4. For a graphical representation of the dual solution to the intensity emitted in direction WSW confer to Figure 8.5 on page 90. We show the dual solution for different ordinates, the ordinate of interest (direction WSW) and the opposite one in the first row as well as two intermediate ordinates W and S in the second. Note, that the dual solution is one order of magnitude larger for the WSW ordinate itself than for all others. The smoothness of the dual solution causes that the estimate obtained by formula (4.27) on page 52 is reliable with a safety constant $C_{\mathrm{Sec}} \approx 1.5$. A mesh history obtained by this estimate and the numerus clausus criterion always doubling the number of cells is shown in 8.4 on the next page. Although the estimate is dominated by the residual, we observe stronger refinement in the WSW direction due to the dual solution part.

In Table 8.2 we compare the refinement optimal for L^2–error control with the boundary error estimate. Using the numerus clausus refinement strategy we generate meshes of about the same sizes for both criteria. Since the constants in the L^2–estimate are not sharp enough and the estimate is asymptotically not optimal, we do not use it to estimate the error. Note that we would have to apply inverse estimates and trace theorems to estimate the boundary integral value desired. We see that the indicator applying the dual solution converges much faster to a limit of about 0.617 (obtained by extrapolation). The L^2–indicator needs about 8 times the number of grid points to reach the same accuracy as the estimate. We obtained similar results for different sets

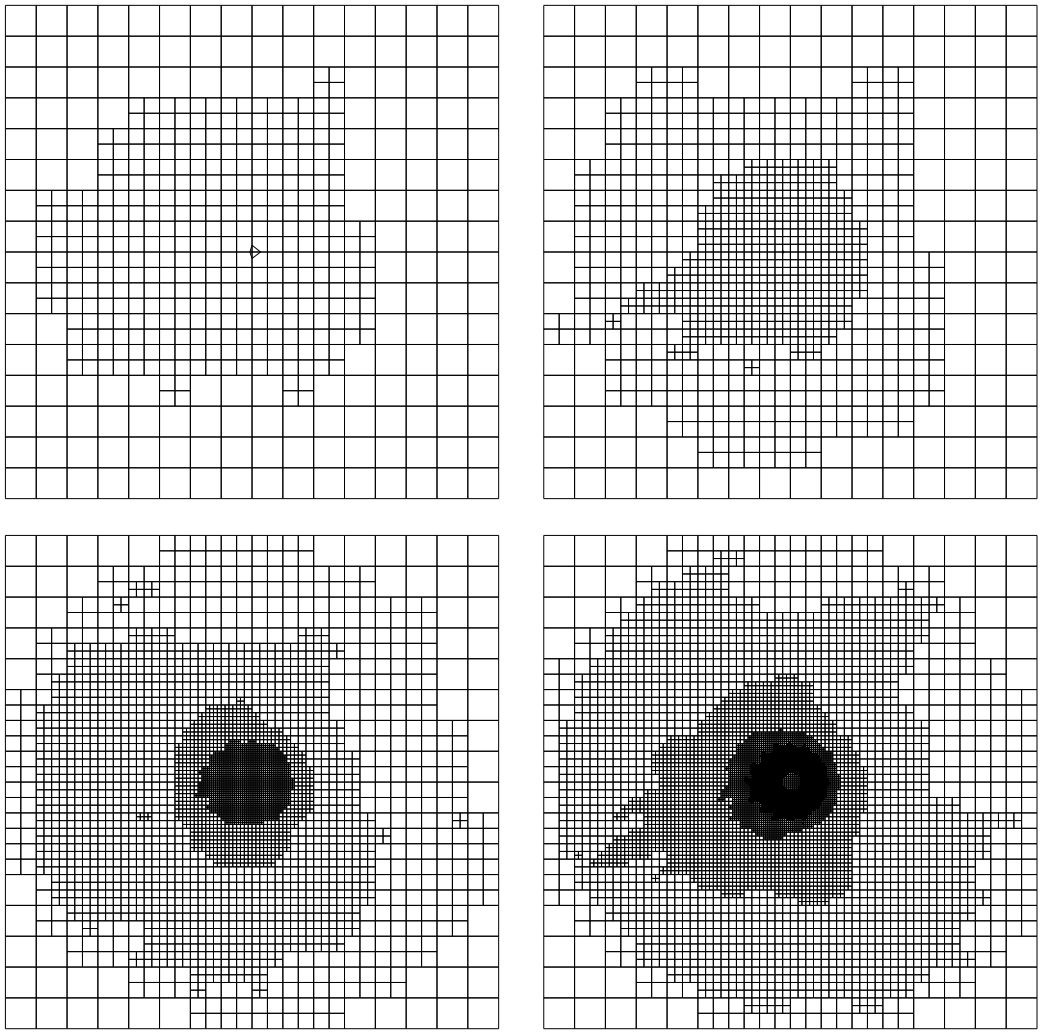


Figure 8.4: Refinement history for the dust cloud (steps 1, 2, 4 and 6)

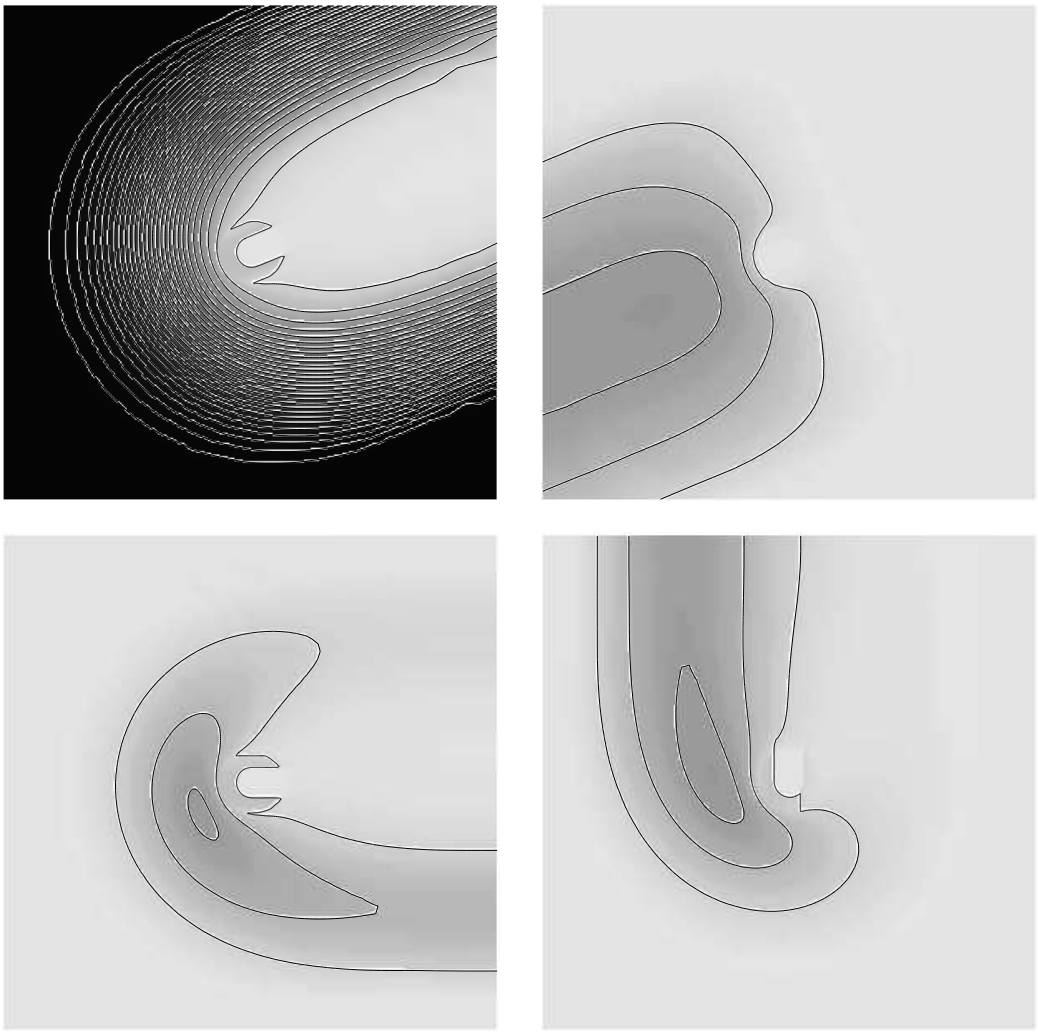


Figure 8.5: Dual solutions for the boundary integral estimate

L^2–indicator		boundary–indicator		estimates	
points	value	points	value	η	$\frac{C_{\mathrm{Sec}}\eta}{e}$
564	0.181	576	0.417	3.1695	23.77
1105	0.210	1146	0.429	1.0804	8.62
2169	0.311	2264	0.461	0.7398	7.11
4329	0.405	4506	0.508	0.2861	3.94
8582	0.460	9018	0.555	0.1375	3.33
17202	0.488	18857	0.584	0.0526	2.39
34562	0.537	39571	0.599	0.0211	1.76
68066	0.551	82494	0.608	0.0084	1.40

Table 8.2: Comparison between indicators based L^2–error and boundary integral error control

of parameters, so this comparison gives evidence that the estimates using computed dual solutions generate grids with much less cells than global indicators. This is due to the fact that the new approach we apply uses a locally weighted residual to estimate the error instead of a global stability constant. Since the weights are computed by measuring the possible contribution of local residuals to the error functional, the new estimator is much finer than the one using global stability. Additionally, the last column shows that the estimate is a sharp upper bound for the error of the boundary integral.

From the numerical point of view, the problem is very difficult to solve. The intensity inside the star is 100 in all computations, but the values at the boundary are only between unity and 10^{-7}. The advantage of the dual solution approach lies in damping out of the residual in the parts of the central region, where the intensities have minor influence on the boundary integral. The result is the ability to calculate a boundary integral value of $3.5 \cdot 10^{-7}$ up to an absolute accuracy of about 10^{-7} using 37,000 mesh points only (optical depth 21, $\sigma = 0.5\,\kappa$). We can compute the integral value of 0.00054 up to a guaranteed relative accuracy of 5% on a grid of the same size in case of a smaller optical depth of about eleven. The computation with 16 ordinates needs 2:34 hours on the PPC-GC including the complete mesh generation. This should be seen in relation to the about same amount of time necessary for coarse computations without error control using conventional methods. Those simulations are suitable to give an impression of the global qualitative behavior of the solution, but they often miscalculate the boundary

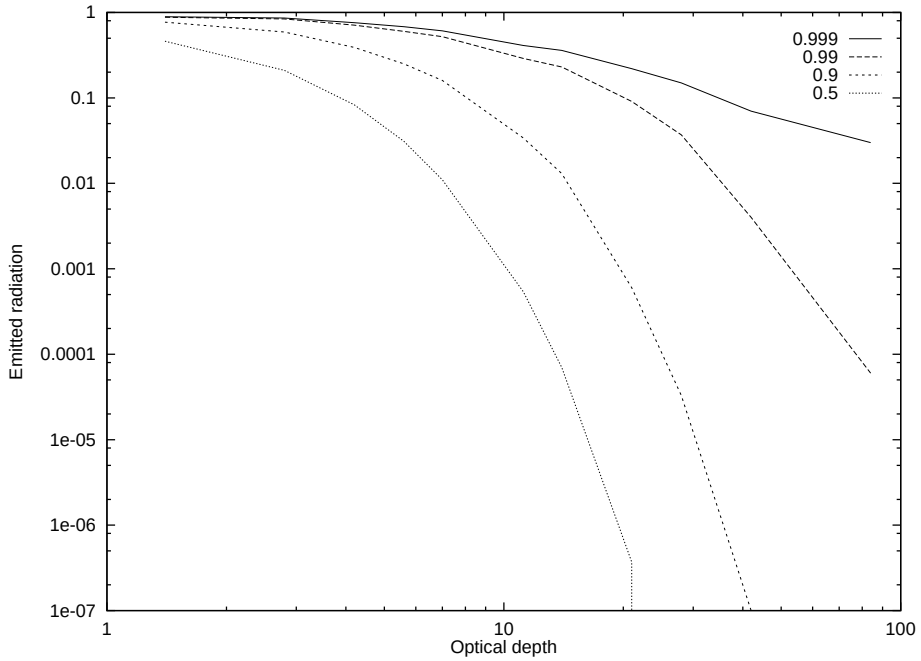


Figure 8.6: Relative emission of the dust cloud for a star radiating with intensity 1 (curves for different scattering parameters)

---

integral value by orders of magnitude.

For a wide range of the spectrum, the decoupling of wavelengths is a good approximation of the physical properties. For these data, we display the ratio between radiation of the star and the emission of the cloud for different opacities in Figure 8.6. The curves show the radiation dependent on the optical depth defined by the integral of χ over the line of sight. These curves show a strong dependence on σ if the optical depth is high, whereas scattering has small influence in the optically thin case. These results emphasize the necessity to produce accurate solutions for the scattering dominated problem. By multiplication of these computed values with the appropriate values of Planck's function, we obtain the emission profile of the dust cloud for small wavelengths (visual light).

8.3 Further Development

So far we have shown only two–dimensional applications of our methods. We computed a cloud similar to that of the first section with a full three–dimensional geometry. The results are shown on the color plate. The pictures show the light emitted by the cloud in the direction of the observer, i. e. the appearance of such a cloud in the sky. The first figure shows the geometry of the ellipsoidal cloud with three embedded stars. Pictures b and d demonstrate the results for a high opacity observed from different points of view. Figure c shows the simulation for a smaller opacity.

The techniques used in the two–dimensional case are all applicable in three dimensions, but since the memory requirements are much larger, the computational grids have to be rather coarse on existing computers (the maximum of about 100,000 cells on a 64 M-Byte node leads to about 30 cells in each coordinate direction). A domain decomposition approach would not solve this problem, since we need about 300–1000 ordinates, thus the parallel machine is fully utilized using ordinate parallelization too. Therefore, we can only simulate with low resolution to get qualitative results. The accurate solution of three–dimensional radiative transfer problems requires more powerful computers than those available at the IWR by now.

The second extension of our algorithms is the inclusion of the wave–length dependence. There are three mechanisms where the coupling of intensities for different wave–lengths occurs:

- The scattering phase function $P(\tilde{\vartheta}, \vartheta)$ is only an approximation of the redistribution function $R(\tilde{\lambda}, \tilde{\vartheta}, \lambda, \vartheta)$. This extension is straight forward, since the structure of the coupling is the same as that of scattering.
- Another transport term $\partial u/\partial \lambda$ may be introduced to model Doppler effects due to fast movement of the dust particles.
- The equation of local energy equilibrium

$$\int_{\mathbb{R}^+} \int_{S^2} \kappa(x, \vartheta, \lambda) u(x, \vartheta, \lambda)\, d\vartheta\, d\lambda = \int_{\mathbb{R}^+} \kappa(x, \vartheta, \lambda) B(\lambda, T(x))\, d\lambda$$

  causes a nonlinear coupling of wave–lengths. This equilibrium is important to model infrared radiation of dust clouds, e. g. to compare with observer data of the ISO satellite.

The last mechanism has no additional memory requirements, since the wave–length integral can be accumulated step by step avoiding storing the intensities for each wave–length. The same is true for a large class of redistribution functions. In particular the extension of our adaptive concepts to the wave–length dependent problem is interesting, since coefficients and solution can differ extremely between two nearby wave–lengths.

Bibliography

[1] M. Asadzadeh. *Convergence Analysis of some Numerical Methods for Neutron Transport and Vlasov Equations.* PhD thesis, Chalmers Tekniska Högskola, Göteborg, 1986.

[2] L. H. Auer and F. Paletou. Two–dimensional radiative transfer with partial redistribution. *A&A*, 284:675–686, 1994.

[3] B. Baschek, J. Adam, R. Plate, H. Störzer, and R. Wehrse. Line radiation from accretion discs: Spatially resolved distribution. *Adv. Space Res.*, 8(2):315–319, 1988.

[4] R. Becker. *An Adaptive Finite Element Method for the Incompressible Navier–Stokes Equations on Time–dependent Domains.* Dissertation, Universität Heidelberg, 1995.

[5] R. Becker, G. Kanschat, and F.-T. Suttmeier. DEAL — differential equations analysis library. Available via `http://gaia.iwr.uni-heidelberg.de/DEAL.html`, 1995.

[6] R. Becker, G. Kanschat, and F.-T. Suttmeier. Optimality criteria for adaptive finite element meshes. Technical report, IWR, Universität Heidelberg, 1996. to appear.

[7] R. Becker and R. Rannacher. Weighted a posteriori error control in fe methods. Preprint 96–1, IWR, Universität Heidelberg, January 1996. submitted to ENUMATH 95, Paris.

[8] M. Borysiewicz, J. Mika, and G. Spiga. Asymptotic analysis of the linear boltzmann equation. *Math. Meth. Appl. Sci.*, 3:405–423, 1981.

[9] M. Borysiewicz and R. Stankiewicz. Weak solution and approximate methods for the transport equation. *J. Math. Anal. Appl.*, 68:191–210, 1979.

[10] P. G. Ciarlet. *The Finite Element Method for Elliptic Problems*, volume 4 of *Studies in Mathematics and its Applications*. North–Holland, Amsterdam, New York, Oxford, first edition, 1978.

[11] O. J. Dittmann. *Transport polarisierter Strahlung in Medien beliebiger Geometrie*. Dissertation, Universität Heidelberg, 1995.

[12] K. Eriksson, D. Estep, P. Hansbo, and C. Johnson. *Adaptive Finite Element Methods*. North-Holland, 1996.

[13] K. Eriksson, P. Hansbo, and C. Johnson. Parallelization of numerical methods for the neutron transport equation. unpublished, 1994.

[14] C. Führer. A comparative study of finite element solvers for hyperbolic problems with applications to radiative transfer. Preprint 93–65, IWR, Universität Heidelberg, November 1993.

[15] C. Führer. Finite–Elemente–Diskretisierungen der 2d–Strahlungstransportgleichung. Diplomarbeit, Universität Heidelberg, 1993.

[16] C. Führer and G. Kanschat. Error control for radiative transfer problems. Preprint 95–31, IWR, Universität Heidelberg, July 1995.

[17] C. Führer and R. Rannacher. Error analysis for the finite element approximation of a radiative transfer model. Preprint 94–46, IWR, Universität Heidelberg, July 1994. submitted to M^2AN.

[18] D. Gilbarg and N. S. Trudinger. *Elliptic Partial Differential Equations of Second Order*. Springer, Berlin, Heidelberg, ..., second edition, 1983.

[19] J. W. von Goethe. *Zur Farbenlehre*. Cotta, Hamburg, 1810.

[20] F. Golse and B. Perthame. Generalized solutions of the radiative transfer equations in a singular case. *Commun. Math. Phys.*, 106:211–239, 1986.

[21] W. Hackbusch. *Integralgleichungen*, volume 68 of *LAMM*. Teubner, Stuttgart, 1989.

[22] T. Hughes and M. Mallet. A new finite element formulation for computational fluid dynamics iii — the generalized stgreamline operator for multidimensional advective–diffusive systems. *Comp. Meth. Appl. Mech. Engng.*, 58:305–328, 1986.

[23] T. Hughes and M. Mallet. A new finite element formulation for computational fluid dynamics iv — a discontinuity–capturing operator for multidimensional advective–diffusive systems. *Comp. Meth. Appl. Mech. Engng.*, 58:329–336, 1986.

[24] T. Hughes, M. Mallet, and A. Mizukami. A new finite element formulation for computational fluid dynamics ii — beyond supg. *Comp. Meth. Appl. Mech. Engng.*, 54:341–355, 1986.

[25] C. Johnson, U. Nävert, and J. Pitkäranta. Finite element methods for linear hyperbolic problems. *Comp. Mech. in Appl. Mech. Engng.*, 45:285–312, 1984.

[26] C. Johnson and J. Pitkäranta. Convergence of a fully discrete scheme for two–dimensional neutron transport. *SIAM J. Numer. Anal.*, 20(5):951–966, October 1983.

[27] C. Johnson and A. Szepessy. Adaptive finite element methods for conservation laws based on a–posteriori error estimates. Preprint 1992–31, Dept. of Math., Chalmers Univ., Göteborg, 1992.

[28] C. T. Kelley. *Iterative Methods for Linear and Nonlinear Equations*, volume 16 of *Frontiers in Appl. Math.* SIAM, Philadelphia, 1995.

[29] E. W. Larsen, J. E. Morel, and W. F. Miller Jr. Asymptotic solutions of numerical transport problems in optically thick, diffusive regions. *Comp. Phys.*, 69:283–324, 1987.

[30] G. L. Olson, L. H. Auer, and J. R. Buchler. A rapidly convergent iterative solution of the non–lte line radiation transfer problem. *JQSRT*, 35:431–442, 1986.

[31] J. Pitkäranta. On the differential properties of solutions to fredholm equations with weakly singular kernels. *J. Inst. Math. Appl.*, 24:109–119, 1979.

[32] J. Pitkäranta. Estimates for the derivatives of solutions to weakly singular fredholm integral equations. *SIAM J. Math. Anal.*, 11(6):952–968, November 1980.

[33] K. J. Ressel. *Least–Squares Finite–Element Solution of the Neutron Transport Equation in Diffusive Regimes.* PhD thesis, University of Colorado, Denver, 1994.

[34] W. C. Rheinboldt and C. K. Mesztenyi. On a data structure for adaptive finite element mesh refinements. *ACM Trans. Math. Software*, 6:166–187, 1980.

[35] L. G. Stenholm, H. Störzer, and R. Wehrse. An efficient method for the solution of 3-d radiative transfer problems. *J. Quant. Spectrosc. Radiat. Transfer*, 45(1):47–56, 1991.

[36] S. Turek. An efficient solution technique for the radiative transfer equation. *Imp. Comp. Sci. Eng.*, 5:201–214, 1993.

[37] S. Turek. A generalized mean intensity approach for the numerical solution of the radiative transfer equation. *Computing*, 54(1):27–38, 1995.

[38] S. Turek and R. Wehrse. Spectral appearance of dust enshrouded stars: A finite element approach to 2d radiative transfer. Preprint 94–43, IWR, Universität Heidelberg, July 1994. submitted to A & A.

[39] A. Unsöld and B. Baschek. *Der neue Kosmos.* Springer, Berlin, Heidelberg, ... , 4 edition, 1988.

[40] H. M. Väth. Three–dimensional radiative transfer on a massively parallel computer. *A&A*, 284:319–331, 1994.

[41] R. Verfürth. A posteriori error estimation and adaptive mesh–refinement techniques. *J. Comp. Appl. Math*, 50:67–83, 1994.

[42] H. van der Vorst. Bi-cgstab: A fast and smoothly converging variant of bi-cg for the solution of nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 13(2):631–644, March 1992.

[43] H. Yserentant. On the multi-level splitting of finite element spaces. *Numer. Math.*, 49:379–412, 1986.

[44] G. Zhou. How accurate is the streamline diffusion method? Preprint 95–22, IWR, Universität Heidelberg, April 1995.

[45] G. Zhou and R. Rannacher. Pointwise superconvergence of the streamline diffusion finite element method. Preprint 94–72, IWR, Universität Heidelberg, December 1994.