Astron. Astrophys. 284, 319–330 (1994)

ASTRONOMY AND ASTROPHYSICS

Three-dimensional radiative transfer on a massively parallel computer

H.M. Väth

Department of Physics and Astronomy, Louisiana State University, Baton Rouge, LA 70803, USA

Received 7 September / accepted 30 September 1993

Abstract. We perform 3D radiative transfer calculations in NLTE in the simple two-level atom approximation on the MasPar MP-1, which contains 8192 processors and is a single instruction multiple data (SIMD) machine, an example of the new generation of massively parallel computers. On such a machine, all processors execute the same command at a given time, but on different data. To make radiative transfer calculations efficient, we must re-consider the numerical methods and storage of data. To solve the transfer equation, we adopt the short characteristic method and examine different acceleration methods to obtain the source function. We use the ALI method and test local and non-local operators. Furthermore, we compare the Ng and the orthomin methods of acceleration. We also investigate the use of multi-grid methods to get fast solutions for the NLTE case. In order to test these numerical methods, we apply them to two problems with and without periodic boundary conditions.

Key words: radiative transfer – methods: numerical – stars: atmospheres

1. Introduction

The classical radiative transfer problem in astronomy has been concerned with calculating spectra of stellar atmospheres at rest (Mihalas 1978). Because of the spherical symmetry, this is essentially a one-dimensional problem with the radius as the only spatial coordinate. Several powerful methods for solving the radiative transfer equation for this case have been known for many years (Feautrier 1964; Auer & Mihalas 1969; Rybicki 1971), and it has become possible to include in such calculations many line transitions. In addition, the statistical and radiative equilibrium equations and the hydrostatic equations can be solved self-consistently (e.g. Werner 1989; Dreizler 1991). In 3D, however, it has not been possible to do this on present computers so far, both because the size of the problem exhausts the memory space, and because such computations would require far too much computing time. Three-dimensional calculations become necessary when astrophysical objects are not spherically symmetrical. Accretion disks are such objects (Adam 1990). Other examples are certain types of planetary nebulae (Icke et al. 1992) and gas clouds generated by collisions between stars (Ruffert 1992). To calculate spectra of such objects, one therefore needs to solve the radiative transfer equation in three dimensions. The increasing power of computers has now made it possible to tackle such problems. Several approaches to solving the transfer equations in more than one dimension have been made in recent years. Stenholm et al. (1991) computed the specific intensity at every point of a 3D cartesian grid by approximating the transfer equation as a difference equation. Another way of solving the transfer equation is by using the so-called short characteristics (Kunasz & Auer 1988). Recently Klein et al. (1989) applied the finite element method to this problem. Except for the finite difference method, all these methods have only been applied to 2D cases so far.

In the case of non-local thermal equilibrium (NLTE), the source function, on which the radiation field depends, is itself a function of the radiation field. The transfer equation then becomes an integral-differential equation. To solve this problem, one generally resorts to iterative methods, since the size of the problem can easily prohibit the storage of all the relevant quantities. The classical method is the Λ-iteration (Mihalas 1978). However, the convergence rate can be slow in interesting problems. For that reason, different ways of accelerating the convergence have been developed. Operator splitting was first used by Cannon (1973a, b) for radiative transfer problems. It was then further developed by, among others, Scharmer (1981) and Werner & Husfeld (1985), the latter introducing the term accelerated Λ-iteration (ALI). Klein et al. (1989) proposed a new double-splitting method to get a faster convergence. Recently, Turek (1993) proposed a conjugate gradient-like method to obtain the source function, and Turek & Wehrse (1993) applied it to astrophysical problems.

Traditionally, all radiative transfer calculations have been performed on scalar and vector computers. However, in recent years massively parallel computers have become available in a variety of different architectures distinguished in two major classes of parallel machines (Hockney & Jesshope 1988). On the one hand, there are the multiple instruction multiple data (MIMD) machines, on which each processor can make computations completely independently of other processors. But

the individual processors of such computers have to be fairly complex, are therefore relatively expensive, and the number of processors is usually small. The advantage of such computers is that a program can be run on different processors with different parameters without making adjustments within the program. Therefore, programming such machines can be relatively straightforward. In contrast to these MIMD computers are the single instruction multiple data (SIMD) machines, on which all processors perform exactly the same computations. The processors therefore can be relatively simple and are inexpensive, and such computers can contain a large number of processors. There the code mainly consists of array manipulations, and a language in which parallel array operations are easily defined is required to program such computers efficiently. Therefore, SIMD machines represent the more radical approach to parallel programming. Overall, it cannot be decided yet which approach in parallel computers will carry through, and currently applications are designed for both types of computers, which is reflected in conference proceedings on parallel computing (e.g. Meuer 1991).

In this paper we discuss NLTE radiative transfer calculations in three dimensions that have been performed on the SIMD machine MasPar MP-1. To our knowledge, no radiative transfer calculations have been performed so far on SIMD machines. First we describe the MasPar MP-1 in Sect. 2. Then we give a short description of the theory of radiative transfer and acceleration methods as far as it is necessary here. In the fourth section we describe the way we solve the transfer equation on this computer, and the way the computing time scales with the size of the problem. After that we describe the convergence properties of these calculations for two problems. In one we have Dirichlet boundary conditions, and in the other problem periodic boundary conditions. Final conclusions are presented in Sect. 6.

2. The MasPar MP-1

When programming on conventional computers, one needs to know very little to nothing at all about the architecture of the computer. This is not the case for massively parallel computers. There the architecture directly influences the style of computer codes, as one has to think in terms of how data are flowing macroscopically between different microprocessors when the program is executed. Even further, the existence of a certain architecture might make the solution of one problem very easy and fast, while essentially prohibiting the treatment of another problem on this particular computer. Because of these reasons, we have to give a fairly detailed description of the architecture of the MP-1. However, we mostly restrict ourselves to those features of the MP-1 that are important to us here. (The following description is based on the manuals available for the MasPar MP-1.)

The MP-1 is a SIMD machine, which means that all processors execute the same commands. It contains 8192 processing elements (PEs), although this number can be extended to 16384. The PEs are arranged in a two-dimensional grid with 128 PEs in the x and 64 PEs in the y direction. The data memory of the PEs then forms a natural z direction. Here each processor contains 64 kByte of memory space. As all processors perform the same operations, they need to be controlled by some central unit, which is the array control unit (ACU). While all parallel operations are performed on the PE grid, scalar calculations are done either on the ACU or the front end (FE), which is here a DEC 5000 workstation. Communication between the FE and the PE grid is possible through the ACU, which can broadcast scalar data simultaneously onto all PEs.

An ideal problem for such a SIMD machine would be one, in which all the processors could operate independently from one another, i.e. no data would need to be transfered between processors. However, in most realistic applications, this is not the case. We therefore have to describe the way processors can communicate with each other. On the MasPar there are essentially two ways to do that. Each processor on the PE grid is directly connected with the surrounding eight processors. This so-called X-NET enables the transfer of entire floating point numbers, and communication is therefore very fast. But this method of data transfer is not useful, if two processors which have to communicate with each other are very far apart on the PE grid. For this case exists the global router, which can connect any two PEs concurrently. Up to 512 simultaneous communications are possible on the MP-1. Because of the smaller number of possible connections between PEs, this type of communication is relatively slow.

To program a SIMD machine, one needs a computer language that allows parallel manipulation of data in a simple way. On the MP-1, two languages are currently available: MPL and MPFORTRAN. MPL is an enhancement of C and is the fundamental programming language on the MP-1. The code described here, however, is written in MPFORTRAN, which is close to the new FORTRAN 90 standard and is very suitable for parallel machines because of its array syntax. It also has the advantage that the programmer does not need to explicitly consider the number of available processors (although this number strongly influences the execution time of the program). If one has for example a three-dimensional array of some arbitrary size, then the first two dimensions are automatically allocated along the x and y axis of the PE grid, while the third dimension is stored in the memory of the PEs. If the x and y dimensions of the array are too large to fit onto the PE grid, the array is *cut* and *stacked* into the memory of the PEs and forms virtual layers there. (It is possible, but normally not necessary, to override this default.) Furthermore, the programmer does not need to decide explicitly how communications have to be performed, as is necessary in MPL, which makes the programming less tedious. It has the additional advantage that the code is less specialized for the architecture of the MP-1 and therefore can be adapted to different SIMD machines or conventional scalar machines more easily once the FORTRAN 90 standard is in general use.

3. Radiative transfer

To calculate the spectrum of an object in three dimensions, one must solve the radiative transfer equation (Mihalas 1978)

$$\begin{aligned} \boldsymbol{n}\nabla I_\nu(\boldsymbol{r},\boldsymbol{n}) &= (d/ds)I_\nu(\boldsymbol{r},\boldsymbol{n}) \\ &= \chi_\nu(\boldsymbol{r},\boldsymbol{n})(S_\nu(\boldsymbol{r},\boldsymbol{n}) - I_\nu(\boldsymbol{r},\boldsymbol{n})) \end{aligned} \tag{1}$$

where χ is the opacity, S is the source function, I is the specific intensity, and no time dependence is considered. The vector $\boldsymbol{n}$ gives the direction of the ray, ν is the frequency, and $\boldsymbol{r}$ gives the position. The parameter s is the length along the ray propagating in the direction $\boldsymbol{n}$. The transfer equation can easily be integrated along the line of sight, and thus one obtains the formal solution

$$\begin{aligned} I_\nu(s) =& I_\nu(0)\exp(-\tau_\nu(s)) + \\ & \int_0^{\tau_\nu(s)} d\tau'_\nu S_\nu(\tau'_\nu)\exp(\tau'_\nu - \tau_\nu(s)) \end{aligned} \tag{2}$$

$$\tau_\nu(s) = \int_0^s \chi_\nu(s')ds' \tag{3}$$

where τ is the optical depth along the line of sight. In stellar atmospheres the radius is normally the only spatial variable, and one needs to consider only one angle for the vector $\boldsymbol{n}$. However, if the object has no spherical symmetry, all quantities depend in general on three spatial variables, and $\boldsymbol{n}$ is a function of two angles. As the opacity is normally also frequency dependent, the specific intensity becomes a function of six variables.

For a two-level atom in statistical equilibrium, the source function can be written as

$$S = (1-\epsilon)J + \epsilon B \tag{4}$$

where B is the Planck function, J is the mean intensity, and ϵ is the thermalization parameter. The latter represents the probability that a photon is absorbed and therefore converted into thermal energy. The mean intensity for the two-level atom becomes

$$J = (1/4\pi)\int d\Omega \int d\nu \Phi_\nu(\boldsymbol{n}) I_\nu(\boldsymbol{n})\,. \tag{5}$$

The function Φ is the absorption profile. Clearly the source function is not only dependent on local quantities when $\epsilon < 1$, but also depends on the radiation field, which the transfer equation makes a strongly non-local quantity. The specific intensity and therefore J depend on the source function, and one can write $J = \Lambda[S]$ where the Λ-operator (in the case of the two-level atom a linear operator) is acting on S. After discretization of the problem onto a spatial grid, one obtains a system of linear equations $J = \Lambda S$ where Λ is now a matrix. In principle, one can directly solve for the source function by matrix inversion

$$S = [\mathbf{1} - (1-\epsilon)\Lambda]^{-1}\epsilon B \tag{6}$$

where $\mathbf{1}$ is the unity matrix. However, if one has a three-dimensional grid with 64 points in each dimension, the Λ-matrix has about $6.9\ 10^{10}$ elements. Such a matrix cannot be stored on present machines, and an inversion becomes impossible. The problem has to be solved iteratively. The source function of the $(i+1)$th iteration derives from the ith source function by

$$S^{i+1} = (1-\epsilon)\Lambda S^i + \epsilon B\,. \tag{7}$$

This is the classical Λ-iteration.

As we already pointed out above, the convergence can be accelerated significantly with the ALI method. It is now in common use and has been reviewed by several authors in recent years (Kalkofen 1987; Rybicki 1991; Hubeny 1992). One writes $\Lambda = \Lambda^* + (\Lambda - \Lambda^*)$, where Λ^* is an approximation to the true Λ-operator. The iteration scheme can then be written as

$$S^{i+1} - S^i = [\mathbf{1} - (1-\epsilon)\Lambda^*]^{-1}(S^{\mathrm{FS}} - S^i) \tag{8}$$

where S^{FS} is the source function obtained through a classical Λ-iteration from S^i. This is essentially a Jacobi or block Jacobi iteration depending on the form of the approximate operator (Stoer & Bulirsch 1990). When one uses the diagonal of the Λ-operator, the above matrix inversion becomes a simple division. At large optical depths $\Lambda \to \mathbf{1}$, and when $\epsilon \ll 1$ the approximate operator acts as a large amplification factor. Hamann (1985) and Werner & Husfeld (1985) were the first to interprete the operator splitting that way.

Now arises the question of how to construct the approximate operator. There have been a number of proposals, and we refer to the review articles for references. However, it was Olson et al. (1986) who showed mathematically that a nearly optimal operator is simply the diagonal of the Λ-matrix. One can find the diagonal by the definition of the Λ-matrix that

$$\begin{pmatrix} J_1 \\ \vdots \\ J_n \end{pmatrix} = \begin{pmatrix} \Lambda_{11} & \dots & \Lambda_{1n} \\ \vdots & \ddots & \vdots \\ \Lambda_{n1} & \dots & \Lambda_{nn} \end{pmatrix} \begin{pmatrix} S_1 \\ \vdots \\ S_n \end{pmatrix}\,. \tag{9}$$

The ith column of the Λ-matrix is therefore the mean intensity, if the source function is zero at all gridpoints except at the ith position, where it is one. One can therefore use any method that gives the mean intensity to calculate the Λ-matrix as well. The mean intensity at the grid point i is the ith diagonal element of Λ.

As Olson & Kunasz (1987) already pointed out, a diagonal Λ^* is a local operator as it does not couple different spatial points. Using this local operator has many advantages computationally, as it requires few calculations (no matrix inversion has to be performed) and little memory space, and as it nevertheless performs well in many cases (MacFarlane 1992). However, in the optically thin areas the radiation field is highly non-local, and therefore the off-diagonal elements of the Λ-matrix are no longer negligible in comparison to the diagonal elements. Therefore, one can expect to improve the convergence by including off-diagonal elements in Λ^*, which then becomes a non-local operator including spatial coupling. However, what was a simple division in Eq. (8) in the case of a diagonal Λ^* has now become a matrix inversion. In the one-dimensional case, including the nearest neighbors results in a tridiagonal matrix (Olson & Kunasz 1987; Werner 1989). In the two-dimensional case, one obtains a matrix with nine non-zero elements per row, which was

used by Steiner (1990). But in the 3D case, a matrix inversion becomes quite difficult as we get twenty-seven non-zero elements per row with many zero elements between non-zero ones in each row. When we therefore include off-diagonal elements, we will solve Eq. (8) not by directly inverting the matrix, but by solving the corresponding linear system of equations iteratively using the Jacobi method. When we have some approximation S' for the source function, then the jth step of this iteration can be written as

$$\boldsymbol{x}^{j+1} = \boldsymbol{x}^j + [\mathbf{1} - (1-\epsilon)\Lambda^{\mathrm{D}}]^{-1}\{(1-\epsilon)\Lambda^{\star}\boldsymbol{x}^j + \boldsymbol{b} - \boldsymbol{x}^j\} \quad (10)$$

where $\boldsymbol{x}^j = S^j - S'$, $\boldsymbol{b} = S^{\mathrm{FS}} - S'$, and S^{FS} is the source function we obtain from S' using a classical Λ-iteration. The matrix $\Lambda^{\star}$ is an approximation to the Λ-matrix that includes off-diagonal elements, and Λ^{D} is the diagonal of the Λ-matrix. While we avoid the complicated matrix inversion, we need to make several iteration steps before we get the vector $\boldsymbol{x}$ with a sufficient accuracy. But as the elements of the diagonal of the Λ-matrix are always the largest elements per row and column, we can expect to get a sufficiently accurate solution quite rapidly, especially when we use Λ^{D} to get a good initial estimate for the vector $\boldsymbol{x}$.

There are additional ways of accelerating the convergence. We test the Ng method (Ng 1974) that was first introduced by Buchler & Auer (1985) for radiative transfer problems. Furthermore, we use the orthomin method (Vinsome 1976), which was first applied by Klein et al. (1989) to radiative transfer problems. Both can be combined with the ALI method. In addition to the descriptions of these methods in the above-mentioned papers, they have been discussed and compared by Auer (1991), and we therefore will not give a detailed description here. The essential idea behind both methods is to use several previously calculated source functions in order to estimate the correct source function. The Ng method gets the new estimate by minimizing the residual directly, while the orthomin method minimizes the residual with respect to a set of conjugate vectors. In comparison, the ALI method uses the previously calculated source function and calculates the next source function by a different matrix multiplication. The ALI method therefore represents a different concept from the Ng or orthomin acceleration. The free parameter in these acceleration methods is the number of previously calculated source functions. Here we use three residuals for the Ng method and two conjugate vectors for the orthomin method.

Finally, we test the multi-grid method as first introduced to radiative transfer problems by Steiner (1991). Again we will not go into the details here, as they can all be found in depth in his paper, and as there exists a vast literature on this subject (e.g. Hackbusch 1985), but we will describe the general idea of this method. When one has some approximation to a linear equation on a grid of a certain coarseness, then this approximation will have deviations from the exact solution on this grid. There will be variations over the scale of the separation of the grid points (short-period errors) and variations over the entire mesh (long-period errors). Acceleration methods that try to minimize the residual of a linear equation can only reduce the short-period errors efficiently, not the long-period errors. The idea of multi-grid methods is to introduce grids of varying coarseness. On the coarse grids, the large scale variations of the error can be minimized; and on the fine grids, the small-scale variations can be reduced. To get grid quantities from fine grids to coarser grids, one applies a *restriction*, which we do here by averaging the quantities. The reverse process from the coarse grid to the fine grid is called the *prolongation*, for which we use linear interpolation. On the coarsest grid one solves the so-called *coarse grid equation*, which has the same structure as Eq. (7) for the linear Λ-operator. To do this we make four steps of the ALI method with a local operator. There exists a vast number of different algorithms on how best to combine the use of grids of varying coarseness. We adopt here the algorithm as described by Steiner (1991).

4. Solution of the NLTE radiative transfer problem on the MP-1

4.1. Short characteristics

When the thermalization parameter is much smaller than one, the source function depends strongly on the radiation field, and we have to use Eqs. (7) or (8) to get the source function by iteration. After each step we need to calculate the specific intensities for all direction vectors $\boldsymbol{n}$ and frequencies ν on all the grid points. As was pointed out by Castor et al. (1991), we potentially need a very large number $\mathrm{N}_{\boldsymbol{n}}$ of vectors $\boldsymbol{n}$ for 3D cases to adequately resolve the integrand of Eq. (5). Additionally, a large number N_{ν} of frequency points can be required when velocities become important due to the Doppler shift (Adam 1990). Therefore, the calculation of the mean intensity will be the most time-consuming part of 3D radiative transfer calculations, and we need a fast numerical method to get the specific intensities on the grid points. If one works on a SIMD machine, one must have a numerical method that uses the PE grid optimally, i.e., most or all processors are active at any given time. This implies that all the information needed by any processor is located on or near to that processor. Otherwise, communication may take up an unreasonable amount of time in comparison to actual numerical calculations. But the radiation field is intrinsically a non-local quantity through the radiative transfer equation, and therefore the problem of solving this equation is not ideally suited for parallel machines. In the following, we describe a method with which the transfer equation can nevertheless be solved efficiently on a SIMD machine.

A fast method to obtain the specific intensities on a cartesian grid are the short characteristics as described by Kunasz & Auer (1988) for the two-dimensional case, but the concept easily generalizes to three dimensions. In Fig. 1 we show one x-y layer of a 3D cartesian grid and the short characteristic for point p. This is simply the characteristic of the transfer equation in this cell of the grid. If τ is the optical depth along this characteristic, then the specific intensity I_{p} at the point p is

$$I_{\mathrm{p}} = I_{\mathrm{p}'} \exp(-\tau) + \int_0^{\tau} d\tau' S(\tau') \exp(\tau' - \tau) \quad (11)$$

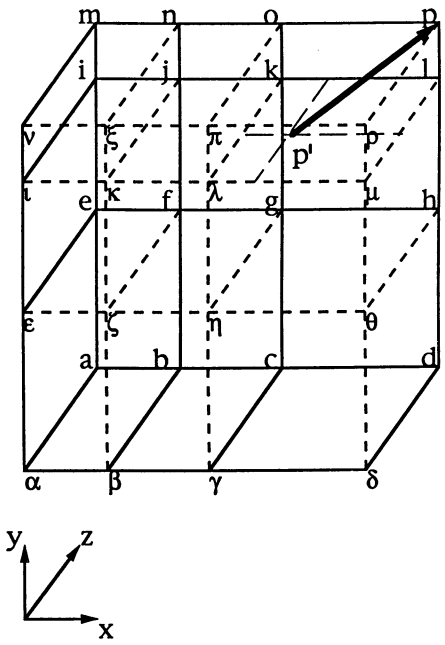

Fig. 1. x-y layer of the 3D cartesian grid showing a short characteristic for point p

where $I_{p'}$ is the specific intensity at the point p′, from which the short characteristic starts. In order to calculate the integral, one has to decide on the way to interpolate the source function along the short characteristic. We choose linear interpolation here. Furthermore, we have to get $I_{p'}$ by interpolating the specific intensities from neighboring points. In the simplest case of linear interpolation, these are in Fig. 1 the points k, l, λ and μ. Because these intensities have to be known first before $I_{p'}$ can be calculated, we have to go through the grid cell by cell in order to get the intensities at the grid points. It is hereby assumed, that the radiation coming into the grid is known, i.e., that we have Dirichlet boundary conditions.

When one works on the MP-1 with its two-dimensional PE grid, a useful way of storing the data of the cartesian grid with its $N_x \times N_y \times N_z$ points is to map the x-y axis of the input grid onto the PE grid and to identify the z axis with the memory of the PEs. Then, ideally, all the specific intensities of one x-y layer should be calculated in one step, and in the following we will investigate whether this is possible. First we have to calculate the integral of Eq. (11) at every point of a x-y layer. As all the data needed to calculate the integral at some point are stored on or nearby the processor that corresponds to that point, we can indeed calculate this integral simultaneously at every point of the x-y layer. Necessary data transfer can be done using the X-Net and is therefore fast. Now we have to get the specific intensity at the start of every short characteristic. But as we pointed out above, this leads to a stepping through the input grid, and essentially only one processor is active at a time. To get a more optimal use of the PEs, we require the points of the cartesian input grid to be equidistant in the x and y dimension (the total dimensions of the x and y axis still do not need to be identical). Then all short characteristics in one x-y layer start from the same side. Therefore, the geometrical data needed to characterize them scale as $N_n \times N_z$ instead of as $N_n \times N_x \times N_y \times N_z$. Furthermore, in the case that the short characteristics in one x-y layer start from the x-y layer below, where the specific intensities have already been calculated, we can get the specific intensities in the current x-y layer in one step. If the short characteristics start from the x-z side or the y-z side, we need to sweep through the x-y layer in $N_y - 1$ or $N_x - 1$ steps, and N_x or N_y PEs respectively are active at a time.

As we saw in the previous paragraph, we do not have an optimal use of the PE grid when we have to sweep through the x-y layer to get the specific intensities at the grid points. We can get around this problem by viewing the intensities in one x-y layer as the solution to a system of linear equations. The specific intensity at the point p can be written as a linear combination of the weighted specific intensities at the points k, l, λ and μ plus a constant which corresponds to the integral of Eq. (11). As the intensities at the points λ and μ are already known, we can simply write I_p as a linear combination of I_k and I_l and a constant, which contains now the integral and the weighted intensities from the neighboring x-y layer. This rewriting can be done for every point of the current x-y layer. If we use a higher order to interpolate intensities, we simply get more complicated linear equations for the grid points. Overall we can write these linear relations as

$$AI = I_0 \tag{12}$$

where I_0 is a constant vector, and I contains the specific intensities. This vector is unknown prior to the solution of this linear equation except for the components that correspond to points on the boundary in the case of Dirichlet boundary conditions. For the above matrix we have $A = 1 - W$, where W is the matrix containing as its elements the weights, which also include the factor $\exp(-\tau)$, and these elements are therefore much smaller than one at large optical depths between grid points. It is obvious that the diagonal of W is zero and that all elements of W are at most one and not negative. Furthermore, there always exists a way to number the grid points such that W becomes a lower triangular matrix, and the solution to the above algebraic system can be obtained by forward substitution. But we can also solve this equation iteratively by writing

$$I^{i+1} = WI^i + I_0 \tag{13}$$

where i counts the iterations. Using the Jacobi iteration instead of this simple iteration scheme does not make a difference, as the diagonal of A is the unity matrix. The advantage on a SIMD machine is that we can have $N_x \times N_y$ active processors instead of only N_x or N_y when sweeping through the grid. Furthermore, because of the special form of W as a lower triangular matrix, we know that we get the exact solution after $N_x - 1$ or $N_y - 1$ iterations. Even further, we can drop the requirement of having equidistant spacing in the x and y direction and still have the exact solution in $\max(N_x, N_y) - 1$ iterations, as it takes only that many steps to transfer the information from any point on the boundary to any grid point that can be influenced by the conditions at the boundary point. However, more memory space and

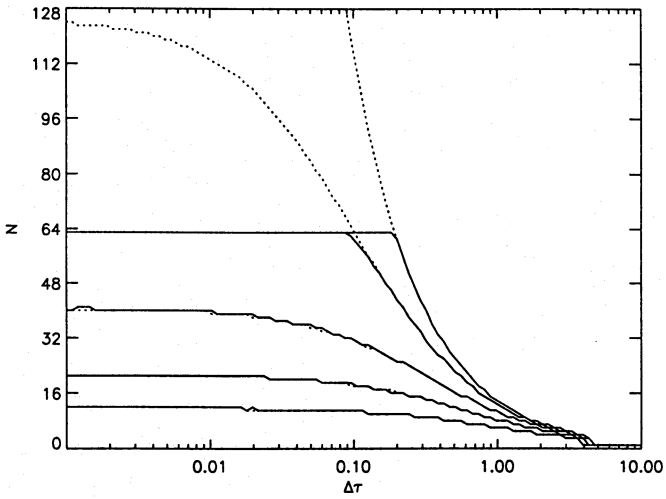

Fig. 2. Number of steps needed for convergence to get specific intensities in one x-y layer for Dirichlet (solid) and periodic boundary conditions (dotted) as a function of the optical depth between grid points for different angles θ. From top to bottom $\theta = 0°, 5°, 15°, 25°, 35°$

more arithmetic calculations are then needed, and therefore we keep the grid points equidistant in x and y. But the main advantage of using this iteration is the possibility that much fewer steps are required to obtain sufficient accuracy when the elements of W are small. This is the case when the optical depth between grid points is large because of the factor $\exp(-\tau)$. The vector $\boldsymbol{n}$ also influences the elements of W through the weights of the interpolation. But most important is the dependence on the angle θ between $\boldsymbol{n}$ and the x-y layer. By comparing the intensities obtained by sweeping and by iteration, we find that the iteration has converged when $(\boldsymbol{I}^{i+1} - \boldsymbol{I}^{i})/\boldsymbol{I}^{i+1} < 10^{-6}$. Figure 2 shows the required steps until convergence as a function of optical depth $\Delta\tau$ between neighboring grid points for different angles θ. The x-component of $\boldsymbol{n}$ is zero, and an isotropic grid with $\mathrm{N}_x = \mathrm{N}_y = 64$ points is used. The source function is set to one everywhere. As expected, the convergence is fastest for large θ. For angles larger than $\approx 10°$ the number of iterations is less than $\max(\mathrm{N}_x, \mathrm{N}_y) - 1$. Furthermore, the number of iterations decreases strongly when the optical depth between neighboring points approaches one, and for depths larger than four we can calculate the specific intensities in a x-y layer in one step instead of in N_x or N_y steps.

This iterative solution turns out to be even more useful when we have periodic boundary conditions in x and y. In this case, the specific intensities at the side $x = x_{\min}$ are equal to the specific intensities at the side $x = x_{\max}$. The same is the case for the y-axis, and there are Dirichlet boundary conditions only for rays coming from the top or the bottom of the grid. Such boundary conditions can be important when investigating radiative transfer in stellar atmospheres with inhomogeneities through convection (Nordlund 1991; Steffen 1991) or magnetic fields (Kalkofen et al. 1989). In 2D calculations, in which there are periodic boundary conditions only along one axis, it is possible to calculate the specific intensity at the boundary analytically when using the short characteristic method (Steiner 1990). However, in the 3D case we have for every x-y layer $\mathrm{N}_x + \mathrm{N}_y - 1$ unknown specific intensities at the boundary. We can express these intensities by an algebraic equation similar to Eq. (12), but with a lower dimension as we have $\mathrm{N}_x + \mathrm{N}_y - 1$ fewer independent intensities. It follows that the matrix A is no longer triangular and the equation cannot be solved by forward substitution. Therefore, when we solve it iteratively, we cannot expect to find the correct solution after at most $\max(\mathrm{N}_x, \mathrm{N}_y) - 1$ steps, as in the case with Dirichlet boundary conditions. On a SIMD machine one can solve this problem iteratively using again Eq. (13) and resetting the periodic boundary conditions after each iteration. As in the case with Dirichlet boundary conditions, we show in Fig. 2 the number of steps required to achieve convergence for different angles θ as a function of the optical depth between neighboring grid points. The number of required iterations diverges for $\theta \to 0$ and $\Delta\tau \to 0$. This is not surprising, as there is no matter and therefore no radiation along the line of sight in this case, but the source function is set to one nevertheless. In order to achieve convergence in less than 128 steps on this grid, the angle θ must be greater than 5°. In a realistic application one can generally fulfill such a restriction.

We now have to discuss how we construct the approximation to the Λ-matrix for the ALI method. This is done equivalently to Kunasz & Olson (1988). When we want to construct the diagonal element of Λ at point p for example (see Fig. 1), we set the source functions to zero at all gridpoints except at the point p, where it is set to one. If we use only linear interpolation for the source function, the intensity at point p′ is zero, and we get the exact element Λ_{pp} by only calculating the intergral of Eq. (11). Therefore, we can calculate the diagonal of Λ for all points in one x-y layer simultaneously. However, when we want to construct for example Λ_{pk}, we have to set the source function to one at point k and zero at all others. Then the intensity $I_{\mathrm{p}'}$ is not zero. In order to use the processors optimally, we will set $I_{\mathrm{p}'}$ to zero. Therefore, we underestimate the values of the off-diagonal elements. But we can expect this error to be small, especially in the optically thick areas, where the ALI method is most useful.

Finally we have to address the problem of when the iterative scheme has converged to a solution and with what approximation to best start the iteration. As was shown by Olson et al. (1986) and pointed out by Auer (1991) as well, an upper bound of the convergence rate of the classical Λ-iteration is given by $1 - \epsilon$. Therefore, the solution has converged when $\Delta S/S \ll \epsilon$ where ΔS is the difference between two source functions obtained by a Λ-iteration.

To get a reasonably good guess for the initial source function, we have to estimate the mean intensity. We can say that $J \approx (1 - p_e)S$ where p_e is the escape probability of photons. This probability is given by (Mihalas, 1978)

$$p_e = (1/4\pi) \int d\nu d\Omega \exp(\tau_\nu(\boldsymbol{n})) \qquad (14)$$

when the effects of scattering are ignored. The optical depth $\tau_\nu(\boldsymbol{n})$ can be calculated using short characteristics.

4.2. Timing

Because the calculations are done simultaneously on many processors, one can expect a significantly different scaling for this code than when the same calculations are done on a conventional scalar or vector machine. This is indeed the case. While the timing on a conventional scalar machine scales as the number of grid points $N_x \times N_y \times N_z$, the calculation on a SIMD machine scales as N_z when the processors are used optimally (at least when numerical methods comparable to ours are used). From this we can see the potential power of parallel machines, as we can increase the number of grid points in x and y arbitrarily without increasing the computing time, as long as there are sufficient processors and as long as the processors are used optimally. We therefore have to discuss the timing of the numerical methods that we use here in some depth, in order to compare the efficiency of performing the calculation on this SIMD machine with the efficiency when done on a conventional computer. In the following, we will assume that the number of points on the x and y axis of the input grid is at most the number of PEs of the x and y axis of the PE grid. If this is not the case, then the calculations have to be repeated for all the virtual layers, which we described in Sect. 2.

The most time-consuming part of getting the source function in the case of NLTE is the computation of the mean intensity, because it consists of solving transfer equation for many angles and frequencies, and it has to be repeated for every iteration. We therefore discuss this first. The total timing for it using short characteristics on the MP-1 scales linearly with N_n and N_ν. We can obtain the timing for the calculation of the specific intensities for a given $\boldsymbol{n}$ and ν in the following way. We first have to calculate the integral of Eq. (11), for which we need $A(N_z\text{-}1)$ seconds, where the constant is $A \approx 1.0$ ms. To calculate the final specific intensities by sweeping one either needs $B(N_z\text{-}1)$ seconds, if all short characteristics start at a x-y layer, or $B'(N_y\text{-}1)(N_z\text{-}1)$, if they start from a x-z layer, or $B'(N_x\text{-}1)(N_z\text{-}1)$ seconds, if they start from a y-z layer. For the constant B we obtain $B \approx B' \approx 0.25$ ms (for periodic boundary conditions $B' \approx 0.45$ ms). Of course it is also possible that the short characteristics start from different sides in different x-y layers, if the grid is not equidistant along the z-axis. But we are interested here in how long a calculation takes on average, and we therefore assume that we have many different viewing directions and a grid with equidistant spacing in all dimensions. Then one obtains for the time needed to calculate the mean intensity

$$T = N_\nu N_n \left\{A + \frac{1}{3}\left[B + B(N_x - 1) + B(N_y - 1)\right]\right\}(N_z - 1)\,. \quad (15)$$

To simplify the discussion we will assume from now on that we have a grid of N points in each direction with $N \gg 1$. Table 1 lists how the computing time scales for different tasks. For large N the time scales as N^2 for the sweeps. When calculating the specific intensities iteratively, the timing depends on the average iteration steps $\bar{N}$ necessary to get convergence. As discussed before, this is at most N in the case of Dirichlet boundary conditions, while it can be larger for periodic boundary conditions.

Table 1. Scaling of the computing time for the different calculations on an isotropic cartesian grid of N^3 grid points. The constants are $A \approx 1.0$ ms and $B \approx 0.25$ ms. For details see text

Calculation	Computing time
Mean intensity by sweeping	$= N_\nu N_n \left\{AN + \frac{2}{3}BN^2\right\}$
	$\approx N_\nu N_n \frac{2}{3}BN^2$
Mean intensity by iteration	$= N_\nu N_n \{A + \frac{2}{3}B\bar{N}\}N$
ALI, Ng, orthomin	$\propto N$
Multi-grid method: $T_0 \propto N^3$	$= 2.5\,T_0$ for $n = 2$
	$= 2.5\,T_0$ for $n = 3$
	$= 2.5\,T_0$ for $n = 4$
$T_0 \propto N^2$	$= 3.0\,T_0$ for $n = 2$
	$= 3.25\,T_0$ for $n = 3$
	$= 3.375\,T_0$ for $n = 4$
$T_0 \propto N$	$= 4.0\,T_0$ for $n = 2$
	$= 5.5\,T_0$ for $n = 3$
	$= 7.0\,T_0$ for $n = 4$

However, in the most optimal case $\bar{N} = 1$, and the computing time scales as N.

After we have calculated the mean intensity, we have to get the source function iteratively. To accelerate the convergence, we use ALI and/or Ng or the orthomin acceleration. The computing times for all these methods do not depend on N_n and N_ν and scale simply as N. Only in the case of ALI with a non-local operator can the acceleration methods become important for the overall execution time. For one iteration step of Eq. (10) we need for a 64^3 grid 0.15 s with Dirichlet boundary conditions, while it takes 0.45 s in the case of periodic boundary conditions. However, when we use multi-grid methods, where one uses different numbers n of grids, the calculation of the computing time is more difficult, as one step of the multi-grid method essentially requires the repeated computation of the mean intensity on different grids. The time for one iteration step can therefore be written as aT_0, where T_0 is the time needed to calculate the mean intensity on a cartesian grid with N^3 grid points. As can be seen from Table 1, the factor a depends on how T_0 scales with N. As a is always larger than one, the convergence rate for a problem using multi-grid methods must be larger by a factor a compared to that of some other acceleration method, or this other acceleration method is faster in total computing time. Clearly, multi-grid methods are of greater advantage on a scalar computer than on an optimally used SIMD machine. To decide whether the multi-grid methods have any advantages over the Ng and orthomin acceleration, we ultimately have to compare their actual performance when applied to radiative transfer problems.

5. Discussion

5.1. Dirichlet boundary conditions

In the previous section, we have described how to calculate the mean intensity, and how to find the source function by iterative methods. In order to investigate the convergence properties, we

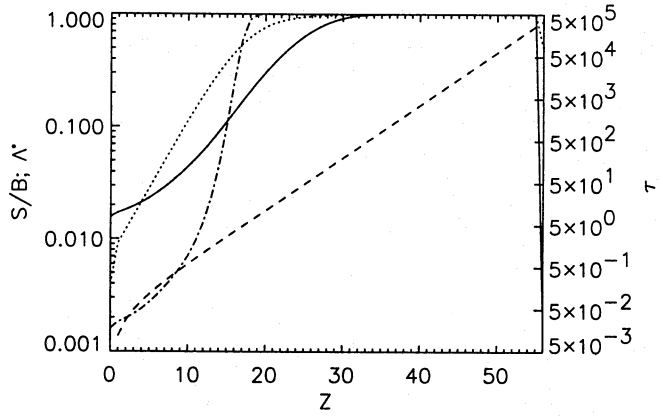

Fig. 3. Solution (solid) and initial guess (dash dotted) for the source function, diagonal of the Λ-matrix (dotted) and optical depth τ (dashed)

assume the 'grey' line here, i.e., we only consider one frequency. The 3D cartesian grid we choose has 57 points along each axis and unity spacing between grid points along mesh lines. We use 26 viewing directions. The Planck function is set to one everywhere, and the thermalization parameter is set to $\epsilon = 10^{-3}$, which represents an intermediate value. There is no incoming radiation. The opacity we use is $\chi(x, y, z) = 10^{z-2}$, and we interpolate it linearly within the grid cells. This choice of model has the advantage that it contains both optically thin and thick regions. The optical depth along a ray parallel to the z axis is shown in Fig. 3 as a function of the z coordinate. As one can see, the optical depth increases from zero to about $5\ 10^5$. Finally we want to point out that we can compare our results with the plane parallel calculations of Auer (1991) because χ is constant in each x-y layer. The two source functions must be identical in the optically thick regions. Only in the optically thin regions do 3D effects become important, and the source function is smaller in the 3D case because of the increased escape probability.

As we described above, one can get an estimate of the source function by calculating escape probabilities using Eq. (14). This estimated source function is shown in Fig. 3. This figure (and unless stated otherwise all following figures) shows the source function along the middle axis of the grid parallel to the z axis. It also shows the 'correct' source function S_c, which was obtained by making many iterations. One can see that the escape probability formulation gives a good approximation to the source function in the effectively optically thick regions where $\epsilon\tau > 1$, which causes the photons to be thermalized and the source function to be close to the Planck function. On the other hand, it differs very much from the correct source function in the other regions and only reflects the qualitative behavior of it, as this approximation does not take into account the effects of scattering. But in the following we will always use ϵB as an initial guess for the source function, as we want to demonstrate the convergence behavior of different models, and this value is a lower limit for the source function.

The basic properties of the classical Λ-iteration are shown by Fig. 4a. The source function is still constant at large optical depths, while at small optical depths the convergence has obviously slowed down very much, even though it is still far away from the solution. This can be understood in the following way (Mihalas 1978). The mean free path of photons is about one optical depth. As ϵ is the probability of thermalization of photons, they travel by random walk a distance of $\epsilon^{-1/2}$ before they are thermalized (also called the thermalization length). Therefore, the source function will only become the Planck function at a depth of $\epsilon^{-1/2}$. As it takes that many steps before information about the boundary propagates into the interior, the Λ-iteration requires $\epsilon^{-1/2}$ steps before it approaches the solution. If there are effectively optically thick regions, one therefore needs to find a different way to obtain the solution.

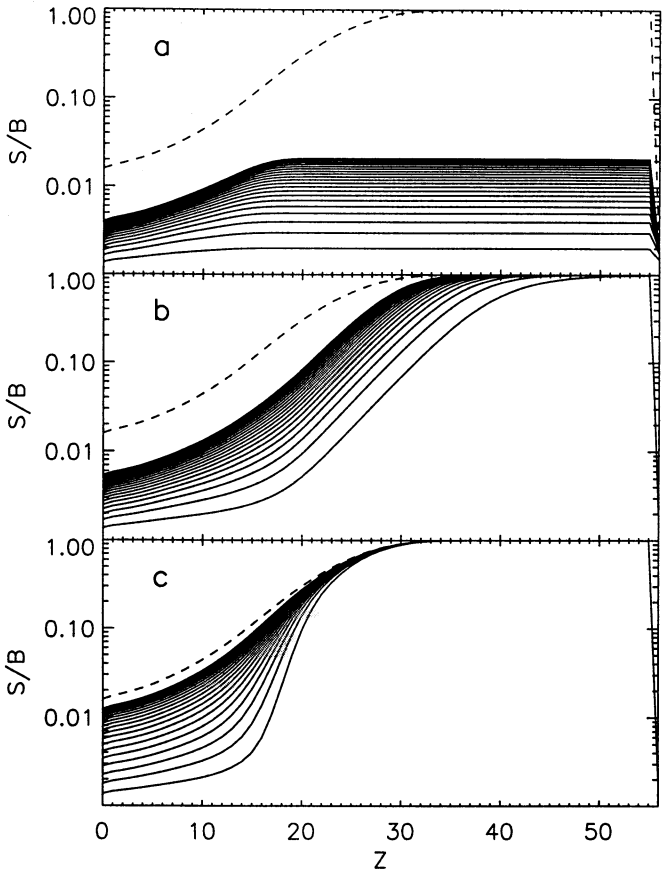

Fig. 4a–c. Convergence of **a** the classical Λ-iteration, **b** ALI with local Λ^*, and **c** ALI with non-local Λ^*. The solution is shown by dashed line

The solution in the regions where the photons are thermalized can be obtained quickly using the ALI method, which is demonstrated by Fig. 4b when using the diagonal of the Λ-matrix, i.e. a local operator, as an approximation. The diagonal of the Λ-matrix for points along the z axis is shown in Fig. 3. At large depths the Λ-matrix is almost a diagonal matrix. Therefore, we get the final solution there after only a few steps of ALI. However, the ALI method using a diagonal matrix does not give a sufficiently good convergence in the effectively optically thin regions, where the photons are not thermalized. As astrophysically interesting cases will normally contain optically thin regions, we will not achieve sufficiently fast overall convergence. This is illustrated in Fig. 5, which shows the maximum relative

error $|S^i - S_c|/S_c$ of the source function after each iteration step (the maximum relative error for the classical Λ iteration is close to one). A way to improve the convergence is by using a better approximation to the Λ-matrix. In Fig. 4c we show the convergence when we include the 26 off-diagonal elements in Λ^* which correspond to nearest neighbors, and use Eq. (10) as the iterative scheme to perform one step of ALI. As we can see, the convergence is significantly faster with this non-local operator than when we only use a local operator. In the areas where photons are thermalized, we get the solution after only one step. In the effectively optically thin regions the convergence is also accelerated in comparison to the local operator. But to decide on the effectiveness of this iteration scheme, we have to look at how many iterations of Eq. (10) are needed for one ALI step. Here we limit the maximal number of iterations to at most 128. Then for the first step of ALI, we need this maximal number (though after 20 iterations we already have convergence in the effectively optically thick regions). But for the next step it takes 90 steps to get a relative change smaller than 10^{-4} between successive source functions as calculated by Eq. (10). In the following, the number of required steps decreases, and after 10 ALI steps we need 43 iterations, which is better than performing a matrix inversion for every ALI step. However, the major disadvantage of this method is the relatively large size of the array required to store the 26 off-diagonal elements. In many cases (e.g. multiple lines) this will exceed the memory space.

The convergence can be significantly accelerated by using the Ng or the orthomin methods of acceleration. Figure 6a shows the convergence of the Ng acceleration, while Fig. 6b shows the convergence of the orthomin acceleration. In both figures we have combined the acceleration methods with ALI using a local operator. Obviously, we get a significantly faster convergence than when only using ALI with a local operator. When comparing these figures one has to keep in mind that one step of the orthomin method essentially requires the calculation of two formal solutions. Then it becomes clear that these two methods are about equally fast in this example. But as is clearly seen in Fig. 5 (and it was noted by Auer 1991 as well), the convergence is much smoother using the orthomin acceleration, and it is therefore preferable. The orthomin method is also tested in combination with ALI using a non-local operator, and Fig. 6c shows the result. This clearly gives the best convergence, but the major drawback is again the large memory space required to store the non-local operator. The number of iterations for one ALI step is as follows. For the first step we again need the maximal number of allowed steps. After that the number of steps rapidly decreases, with 14 iterations for the 10th orthomin step and one iteration for the 16th orthomin step.

Finally, we show the convergence behavior when using multi-grid methods. Figures 7a, b and c show the convergence behavior when n = 2, 3, and 4, respectively. It can be seen directly that using only two grids does not give a satisfactory convergence. There is little difference between the n = 3 and n = 4 case. But as one step using four grids requires more time than one step using three grids (see Table 1), the three grid method is preferable. Overall, these two multi-grid methods are faster

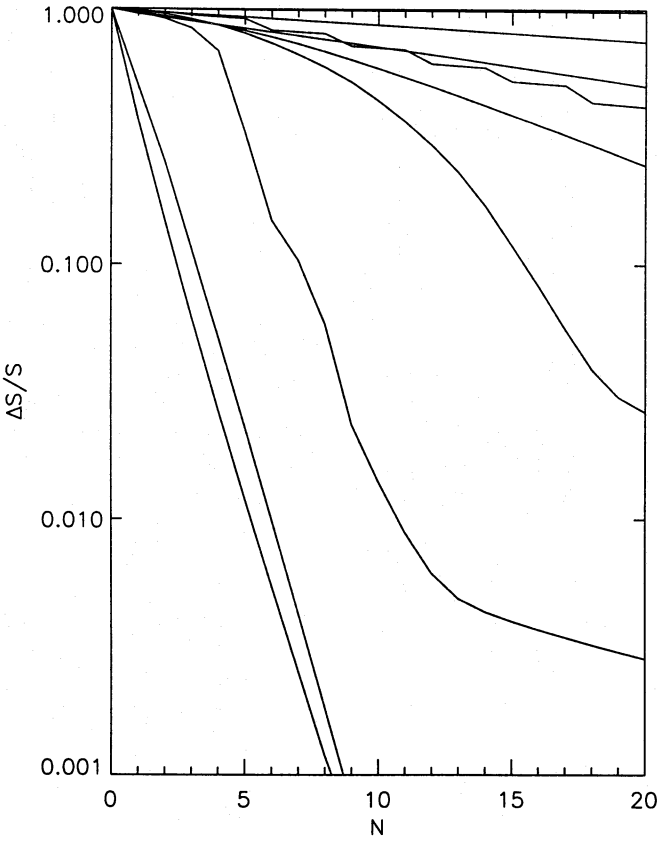

Fig. 5. Maximum relative error of the source function after each iteration step. From top to bottom (at N = 20): Classical Λ-iteration, ALI with local Λ^*, multi-grid method with n=2, Ng acceleration, ALI with non-local Λ^*, orthomin with local Λ^*, orthomin with non-local Λ^*, multi-grid method with n=3, multi-grid method with n=4. The convergence using orthomin with a non-local Λ^* slows down for N > 12 because the correction to the source function is smaller than its machine accuracy

than any other method. But because the time needed for one step of the multi-grid method is longer than the time required for one step of orthomin, we have to compare the number of steps needed to achieve a certain convergence. Looking at Fig. 5 and using the numbers from Table 1, we see that the multi-grid method is fastest when the timing scales as N^2 (when we have to sweep through the x-y layers to get the specific intensities), but not if it scales as N (when we only need one iteration to get the specific intensities in one x-y layer). Then the orthogonal minimization using only a local operator still converges faster. Even though there are viewing directions for which the iteration to get the specific intensities requires the maximal number of steps, on average we only need $\bar{N}$ = 6 iterations to get the specific intensities for some viewing direction. Therefore, the timing scales closely to N, and the multi-grid methods are of no advantage in our example.

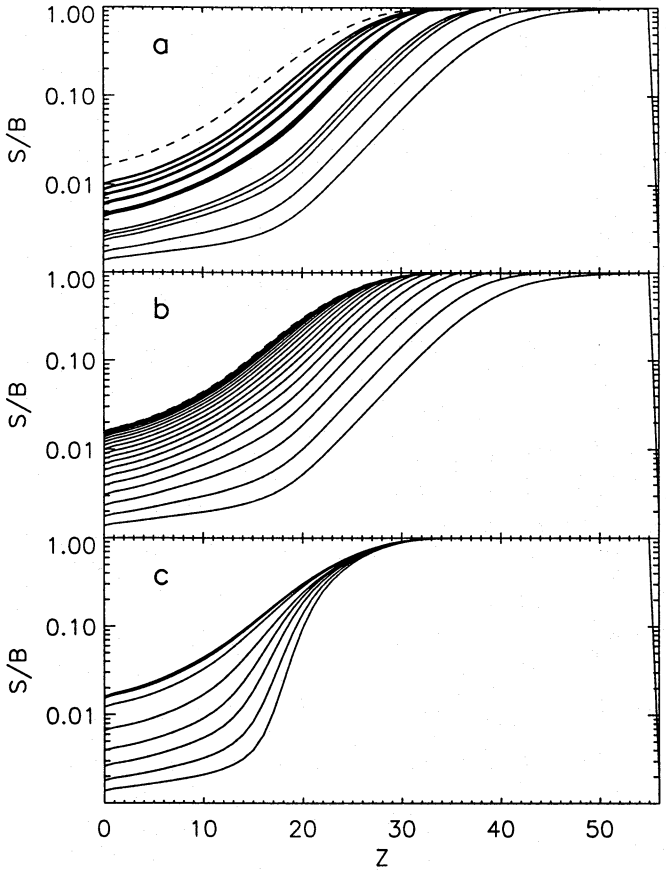

Fig. 6a–c. Convergence of **a** Ng acceleration, **b** orthomin with local Λ^*, and **c** orthomin with non-local Λ^*. The solution is shown by dashed line

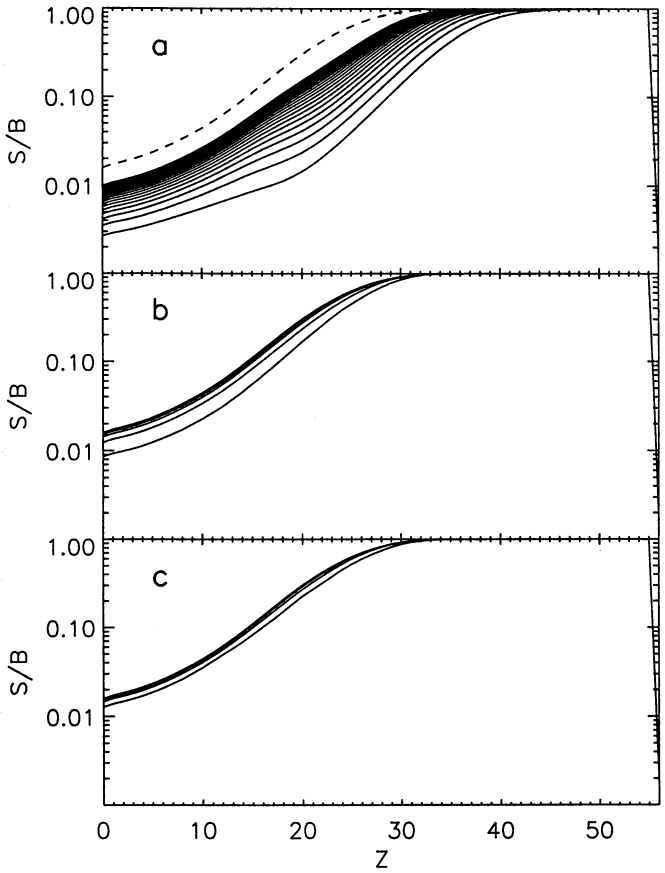

Fig. 7a–c. Convergence of multi-grid methods with **a** n=2, **b** n=3, and **c** n=4. The solution is shown by dashed line

5.2. *Periodic boundary conditions*

To illustrate the ability of our program to handle radiative transfer calculations with periodic boundary conditions, we adapt the two-dimensional model of a stellar atmosphere by Steiner (1991) to three dimensions. In Fig. 8 we show the model. The mesh has $49 \times 49 \times 25$ points and the length of the mesh in x and y is two, while it is one in z (in our calculations we do not take advantage of any symmetries). We have periodic boundary conditions for the radiation in x and y. There is no radiation entering into the computational domain from the top, but we have isotropic radiation entering from the bottom. The opacity outside the dashed cube is denoted with χ_e. Inside the cube the opacity is reduced by a factor 5, and it is denoted with χ_i. Here we use $\chi_e = 24$. Physically, this is a simplified model of a magnetic flux tube in an otherwise normal stellar atmosphere. In this tube the density and therefore the opacity are reduced.

For the radiative transfer calculations we assume radiative equilibrium and again the 'grey' case. From this it follows that the source function is equal to the mean intensity, i.e.

$$S = J \,. \tag{16}$$

The mean intensity J can be written as

$$J = \Lambda S + G \tag{17}$$

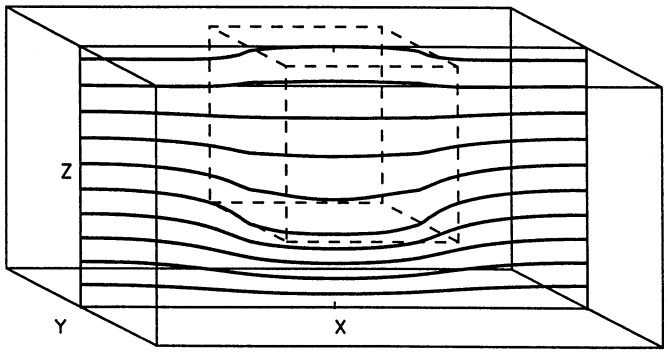

Fig. 8. Part of grey 3D atmosphere with periodic boundary conditions along x and y. Isotropic radiation is entering from the bottom. The length of the cube along x and y is 2, while it is unity along z. The opacity within the dashed cube is $\chi_i = 4.8$, while it is $\chi_e = 24$ outside. A computational mesh of $49 \times 49 \times 25$ mesh points is used. The contour lines of the source function in the x-z plane are shown

where Λ is the usual Λ-operator, and G is the mean intensity due to the incident radiation. By combining the above equations, we get for the source function

$$S = \Lambda S + G \,. \tag{18}$$

We show lines of constant S in the x-z plane in Fig. 8. Under the additional assumption of LTE, we have for the source function $S = B = \sigma T^4/\pi$, where T is the temperature. Therefore, the contour lines in Fig. 8 are also isotherms. As can be seen in the figure, the reduced opacity in the flux tube causes a cooling of the matter at the bottom and a heating at the top of the tube relative to the matter outside the flux tube (Kalkofen et al. 1989). This is qualitatively the same result as that of the 2D calculation of Steiner (1991).

In order to get the specific intensities in the x-y layers within a reasonable number of steps, we rotate the set of vectors $\boldsymbol{n}$ so that no vector has a corresponding angle θ smaller than 8.5°. Then we need at most 25 steps to get the specific intensities of any x-y layer in our example, but on average we have $\bar{\mathrm{N}} = 12$. We summarize the convergence properties of different iteration schemes in Fig. 9 similarly to Fig. 5 for the Dirichlet boundary conditions. As an initial guess we have set the source function everywhere equal to the mean intensity at the bottom, where the radiation is approximately isotropic and the incoming radiation is given. As in the previous example, convergence is achieved in the fewest steps with the multi-grid method using three and four grids. But because of the relatively small number $\bar{\mathrm{N}}$ here, the multi-grid method is of no advantage. Overall it is the orthomin method with a non-local operator which converges fastest. Using Eq. (10), it only takes nine iterations for the first step the same accuracy as in the previous example. Again this number decreases rapidly. However, if memory space is insufficient, the orthomin method with a local operator works best.

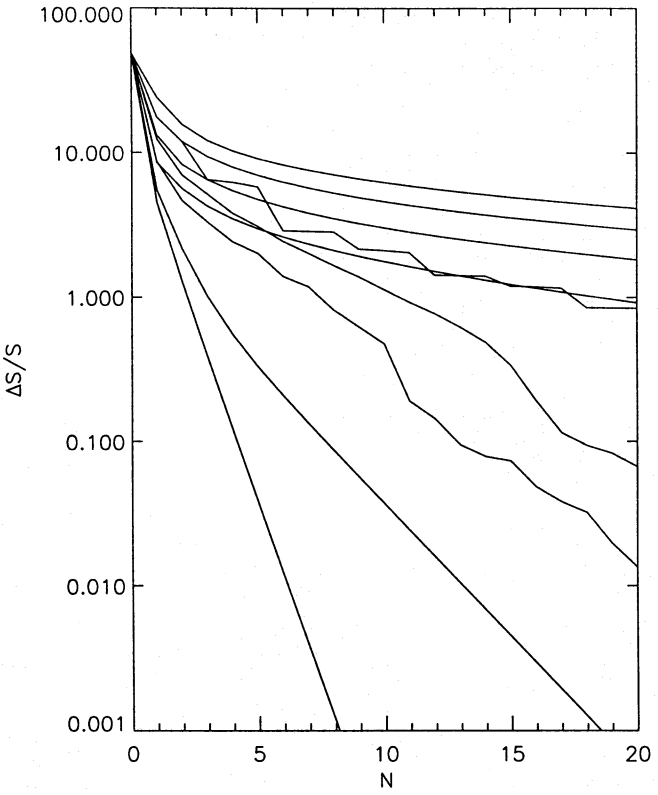

Fig. 9. Maximum relative error of the source function after each iteration step. From top to bottom (at N = 20): classical Λ-iteration, ALI with local $\Lambda^\star$, ALI with non-local $\Lambda^\star$, multi-grid method with n=2, Ng acceleration, orthomin with local $\Lambda^\star$, orthomin with non-local $\Lambda^\star$, multi-grid method with n=3, multi-grid method with n=4. The orthomin acceleration overshoots slightly during the convergence. Therefore, the maximum relative error occurs at different points after different iteration steps, and it is not as smooth as other methods

6. Conclusion

We have developed a computer code to solve radiative transfer problems in three dimensions on the MasPar MP-1, which is a SIMD machine with 8192 processors. We adapt the short characteristic method (Kunasz & Auer 1988) for a 3D cartesian grid in order to solve the radiative transfer equation. To parallelize the computation, we map the cartesian grid onto the two-dimensional grid of the processors. Therefore, we are successful in parallelizing the radiative transfer calculations if the scaling of the computing time is independent of the number of grid points in the x and y axis, and if it is only linearly dependent on the number of grid points in the z axis. This has to be seen in comparison to calculations on conventional scalar or vector machines, where the computing time scales as N^3 when there are N grid points in each dimension. In contrast, it only scales as N on this SIMD machine if the problem is parallelized completely. Because obtaining the solution of the radiative transfer equation is a highly non-local problem, it is not possible to achieve complete parallelization. However, in the case of Dirichlet boundary conditions, we can reduce the scaling of the computing time to N^2 in the worst case, and in a wide range of problems it reduces to $\bar{\mathrm{N}} \times \mathrm{N}$ where $\bar{\mathrm{N}}$ is much smaller than N and close to one. Even in the case of periodic boundary conditions, where potentially $\bar{\mathrm{N}} \gg \mathrm{N}$, one has in praxis $\bar{\mathrm{N}} \ll \mathrm{N}$. A SIMD machine is therefore very suitable for studying inhomogeneities in stellar atmospheres, where periodic boundary conditions become important.

In order to achieve the final source function in the NLTE case, we have to make iterations. We have tested various methods to accelerate the convergence. Most of these can be completely parallelized. Only the multi-grid method turns out not to be well suited for SIMD machines, because the timing depends only weakly on N in comparison to a conventional computer, and because many processors will be inactive during the computations. Overall, the orthomin acceleration in combination with the ALI method gives the smoothest convergence. As an approximate Λ-operator we test a purely diagonal matrix and a matrix constructed by including 26 nearest neighbors. As the inversion of this latter matrix is difficult, we solve the corresponding equation using the Jacobi iteration. While the diagonal matrix already gives a good solution in the effectively optically thick areas, it does not accelerate the convergence sufficiently fast in the optically thin areas, as expected. There the operator's including the 26 nearest neighbors gives a significant improvement. But because of the large memory space required to store this operator, it may not always be possible to use it.

Overall we find that it is possible to efficiently solve NLTE radiative transfer problems on a SIMD machine. We were careful to write the program without overly specializing for the particular SIMD machine that we have available. As the entire code is written in a computer language that is close to the new FORTRAN 90 standard, we will even be able to adapt the code easily to conventional machines once this new standard is generally available. Furthermore, it should not be difficult to convert this program onto a different kind of SIMD machine, as we do not explicitly use specific hardware tools of the MasPar MP-1. The only way we have specialized the code for this particular machine is by assuming a 2D grid of processors, but this assumption should not prove to be very restrictive when using other SIMD machines.

Acknowledgements. The author thanks Dr. D. Koester for many discussions, and K. Powell for proofreading. This work was supported in part by grants from the National Science Foundation (AST 90 8244) and NASA (NAGW 2447) and by the Louisiana State Board of Regents in providing funds to establish the Concurrent Computing Laboratory at the LSU Department of Physics and Astronomy.

References

Adam J., 1990, A&A 240, 541
Auer L.H., 1991, in: Kalkofen W. (ed.) Numerical Radiative Transfer. Cambridge Univ. Press, Cambridge, p. 101
Auer L.H., Mihalas D., 1969, ApJ 158, 641
Buchler J.R., Auer L.H., 1985, in: Davis J., Hooper C., Lee R., Merts A., Rozsnyai B. (eds.) Proc. 2nd Intern. Conf. and Workshop on Radiat. Properties of Hot Dense Matter. World Scientific, Singapore, p. 48
Cannon C.J., 1973a, JQSRT 13, 627
Cannon C.J., 1973b, ApJ 185, 621
Castor J.I., Dykema P.G., Klein R.I., 1991, in: Crivellari L., Hubeny I., Hummer D.G. (eds.) Stellar Atmospheres: Beyond Classical Models, NATO ASI Series C 341. Kluwer, Dordrecht, p. 49
Dreizler S., 1992, in: Heber U., Jeffery C.S. (eds.) The Atmospheres of Early-Type Stars. Springer-Verlag, Berlin, p. 270
Feautrier P., 1964, C.R.Acad.Sci.Paris 258, 3189
Hackbusch W., 1985, Multi-grid methods and applications. Springer-Verlag, Berlin
Hamann W.-R., 1985, A&A 148, 364
Hockney R.W., Jesshope C.R., 1988, Parallel Computers 2. Adam Hilger, Bristol
Hubeny, I., 1992, in: Heber U., Jeffery C.S. (eds.) The Atmospheres of Early-Type Stars. Springer-Verlag, Berlin, p. 377
Icke V., Balick B., Frank A., 1992, A&A 253, 224
Kalkofen W., 1987, in: Kalkofen W. (ed.) Numerical Radiative Transfer. Cambridge Univ. Press, Cambridge, p. 23
Kalkofen W., Bodo G., Massaglia S., Rossi P., 1989, in: Rutten R.J., Severino G. (eds.) Solar and Stellar Granulation, Third International Workshop of the OAC-NATO Advanced Research Workshop. p. 571
Klein R.I., Castor J.I., Greenbaum A., Taylor D., Dykema P.G., 1989, JQSRT 41, 199
Kunasz P., Auer L.H., 1988, JQSRT 39, 67
Kunasz P., Olson G.L., 1988, JQSRT 39, 1
MacFarlane J.J., 1992, A&A 264, 153
Meuer H.W., 1991, Parallelisierung komplexer Probleme. Springer, Berlin
Mihalas D., 1978, Stellar Atmospheres. Freeman, San Francisco
Ng K.C., 1974, J. Chem. Phys. 61, 2680
Nordlund Å., 1991, in: Crivellari L., Hubeny I., Hummer D.G. (eds.) Stellar Atmospheres: Beyond Classical Models, NATO ASI Series C 341. Kluwer, Dordrecht, p. 61
Olson G.L., Auer L.H., Buchler J.R., 1986, JQSRT 35, 431
Olson G.L., Kunasz P.B., 1987, JQSRT 38, 325
Ruffert M., 1992, A&A 265, 82
Rybicki G., 1971, JQSRT 11, 589
Rybicki G.B., 1991, in: Crivellari L., Hubeny I., Hummer D.G. (eds.) Stellar Atmospheres: Beyond Classical Models, NATO ASI Series C 341. Kluwer, Dordrecht, p. 1
Scharmer G.B., 1981, ApJ 249, 720
Steffen M., 1991, in: Crivellari L., Hubeny I., Hummer D.G. (eds.) Stellar Atmospheres: Beyond Classical Models, NATO ASI Series C 341. Kluwer, Dordrecht, p. 247
Steiner O., 1990, A&A, 231, 278
Steiner O., 1991, A&A 242, 290
Stenholm L.G., Störzer H., Wehrse R., 1991, JQSRT 45, 47
Stoer J., Bulirsch R., 1990, Numerische Mathematik 2. Springer, Berlin
Turek S., 1993, Impact of Computing in Science and Engineering 166 (in press)
Turek S., Wehrse R., 1993, A&A (submitted)
Vinsome P.K.W., 1976, in: Proc. of the 4th Symp. on Reservoir Simulation, Soc. of Petroleum Engineers. Boulder, p. 149
Werner K., Husfeld D., 1985, A&A 148, 417
Werner K., 1989, A&A 226, 265

This article was processed by the author using Springer-Verlag TEX A&A macro package 1992.