Available online at www.sciencedirect.com

SCIENCE DIRECT®

Journal of Computational Physics 207 (2005) 354–374

JOURNAL OF COMPUTATIONAL PHYSICS

www.elsevier.com/locate/jcp

A multigrid preconditioner and automatic differentiation for non-equilibrium radiation diffusion problems

Roland Glowinski [a], Jari Toivanen [b,*]

[a] Department of Mathematics, PGH 651, University of Houston, Houston, TX 77204-3008, United States
[b] Center for Research in Scientific Computation, Box 8205, North Carolina State University, Raleigh, NC 27695-8205, United States

Received 2 August 2004; received in revised form 20 December 2004; accepted 21 January 2005
Available online 14 April 2005

Abstract

We study the efficient solution of non-equilibrium radiation diffusion problems. An implicit time discretization leads to the solution of systems of non-linear equations which couple radiation energy and material temperature. We consider the implicit Euler method, the mid-point scheme, the two-step backward differentiation formula, and a two-stage implicit Runge–Kutta method for time discretization.

We employ a Newton–Krylov method in the solution of arising non-linear problems. We describe the computation of the Jacobian matrix for Newton's method using automatic differentiation based on the operator overloading in Fortran 90. For GMRES iterations, we propose a simple multigrid preconditioner applied directly to the coupled linearized problems.

We demonstrate the efficiency and scalability of the proposed solution procedure by solving one-dimensional and two-dimensional model problems.

Keywords: Non-equilibrium radiation diffusion; Multigrid method; Preconditioning; Newton–Krylov method; Automatic differentiation

1. Introduction

The solution of non-equilibrium radiation diffusion problems is considered. This problem is described by a highly non-linear system of parabolic partial differential equations for radiation energy and material temperature [1]. Traditionally the time integration of this system has been performed using explicit, semi-implicit or operator splitting schemes. During the recent years fully implicit time discretizations have

* Corresponding author. Tel.: +1 919 515 6545; fax: +1 919 515 1636.
E-mail addresses: roland@math.uh.edu (R. Glowinski), jatoivan@ncsu.edu (J. Toivanen).

0021-9991/$ - see front matter © 2005 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2005.01.032

become more common [2–14]. They offer much higher accuracy and they do not pose a time step limitation due to a stability condition. On the other hand, they require the solution of a system of non-linear equations at each time step. The purpose of this paper is to study the efficient solution of these systems of non-linear equations for simplified model problems [5].

The most common way to perform the spatial discretization has been to use finite volumes with the quantities located at cell centers being the unknowns. Also, low order finite difference and finite element discretizations have been employed. A more special non-conforming discretization was used in [9]. Typically the used grids and meshes have been uniform. In the following, we will use finite differences for nodal values on uniform grids. In [10,12] adaptive mesh refinement strategies have been considered. Since typical solutions have sharp thermal fronts, this gives an apparent way to improve the accuracy of the approximation without introducing excessive amount of new unknowns.

Among implicit temporal discretizations the implicit Euler, the Crank–Nicolson method, the mid-point scheme and the backward differentiation formulas (BDFs) have been used for radiation diffusion problems. In [3] a problem specific time step selection was introduced which is based on estimating the wave speed and then using it to maintain a desired CFL number. The study [7] showed that the Crank–Nicolson method yields accurate results even with a CFL number close to one while semi-implicit and implicit Euler methods require much smaller CFL number to achieve the same accuracy. In [6] a generic ODE solver based on variable order BDFs and variable time step sizes was employed. We will compare the implicit Euler, the mid-point scheme, the two-step backward differentiation formula (BDF2) and a two-stage implicit Runge–Kutta method. We employ the problem specific time step control from [3].

Newton's method, inexact Newton's methods, Picard iterations, and non-linear multigrid methods have been employed to solve the system of non-linear equations arising at each implicit time step. The convergence of Picard iterations were found out to be much slower than the convergence of Newton's method in [2,4]. A full approximation storage (FAS) multigrid was compared to Newton's method in [8,10]. Both papers concluded that the convergence of both methods are similar but Newton's method is computationally much faster due to the repeated expensive evaluations of non-linear terms in the FAS multigrid. We will use Newton's method which is the most commonly used method for radiation diffusion problems and which also seems to the most efficient method for these problems.

Newton's method can be implemented in many different ways. The Jacobian matrix has to be formed or approximated somehow. If the systems of linear equations with Jacobian matrices are solved with a Krylov subspace method then it is only necessary to be able to multiply a vector by the Jacobian matrix. This can be done using a finite difference approximation without forming the matrix; see [15] and references therein. This approach has been used in [4,5], for example. Another possibility is to hand code the computation of the Jacobian matrix like was done in [8], for example. This is often laborious and error prone to implement. An alternative way to form the exact Jacobian is to compute it using automatic differentiation. We will employ this approach implemented using the operator overloading in Fortran 90.

Each iteration in Newton's method, an inexact Newton's method, or a Picard iteration requires the solution of a system of linear equations. Usually with radiation diffusion equations these problems are solved using an iterative Krylov subspace method and, particularly, using the GMRES method [16]. Such a method is called a Newton–Krylov method [17] when it is combined with Newton's method. The arising systems of linear equations are not well conditioned mainly due to large variations in the diffusion coefficients and small grid step sizes. It is advantageous to introduce a preconditioner which improves the conditioning and, hence, the convergence of the iterative method. The study of preconditioners is the most active field of research related to the solution of radiation diffusion problems. Most of them employ a multigrid method [18–20] as a part of the preconditioner. For example, in [5] a preconditioner based on an operator splitting was described and then a modification of it was considered in [11]. An effective Schur complement preconditioner was introduced for non-equilibrium radiation diffusion problems without diffusion for the material temperature in [6]. A multigrid method suitable for systems like the one in [21] can be used directly as a preconditioner. We will pursue this approach by describing a simple but still effective multigrid preconditioner.

The outline of the paper is the following. We begin by considering Newton's method and the automatic differentiation for computation of the Jacobian matrices. Then, we describe a model non-equilibrium radiation diffusion problem. The temporal and spatial discretizations are introduced in the next section. Then the multigrid preconditioner is introduced for solving the arising systems of linear equations using a preconditioned GMRES method. The next section gives numerical results for one-dimensional and two-dimensional model problems. After some conclusions a short example of implementing automatic differentiation using the operator overloading in Fortran 90 is shown in the appendix.

2. Newton's method and automatic differentiation

We consider the solution of the system

$$\mathscr{B}(\mathscr{E}) = 0 \tag{1}$$

of non-linear equations, where $\mathscr{E}$ is a vector and $\mathscr{B}$ is a non-linear vector-valued function. The implicit time discretizations of non-linear partial differential equations lead to the solution of this kind of problems. For our radiation diffusion model problems these non-linear equations are described in Section 4.1.

We use Newton's method to solve Eq. (1). In the following, the superscript (j) in $\mathscr{E}$ denotes the iteration number and $\mathscr{E}^{(0)}$ is the initial guess for the solution. In time-dependent problems the solution at the previous time step gives a natural initial guess. The (j + 1)th iterant in Newton's method is given by

$$\mathscr{E}^{(j+1)} = \mathscr{E}^{(j)} + \mathscr{F}^{(j)}, \tag{2}$$

where the vector $\mathscr{F}^{(j)}$ is the solution of the system of linear equations

$$\mathscr{J}(\mathscr{E}^{(j)})\mathscr{F}^{(j)} = -\mathscr{B}(\mathscr{E}^{(j)}). \tag{3}$$

Here, $\mathscr{J}(\mathscr{E}^{(j)})$ is the Jacobian matrix of $\mathscr{B}(\mathscr{E}^{(j)})$. In Section 5, we consider the efficient solution of (3) for the radiation diffusion problems.

In principle, the computation of the Jacobian matrix is a mechanical and straightforward task, but in practice it is laborious and error prone to code it. Thus, a typical approach is to approximate the Jacobian matrix. In [2,5], the matrix multiplications by the Jacobian are approximated using finite differences leading to efficient solution procedure for radiation diffusion problems.

Our approach is to compute the Jacobian matrix using automatic differentiation. We will use an approach based on an operator overloading technique available in modern programming languages like Fortran 90 and C++ [22–24]. An alternative way would be to use a preprocessor, for example, ADIFOR [25], which generates the necessary code for computing the desired derivatives. The basic idea is to form an new data type for floating point numbers which also includes derivative information and, then, to redefine the elementary operations and functions like addition, multiplication, exponential function, etc., for this new data type so that they also compute desired derivatives using the chain rule of the differential calculus.

With operator overloading based automatic differentiation the code typically requires only a minor modification in order to compute the desired derivatives. In our case, we only modify the code computing the residual vector $\mathscr{B}(\mathscr{E}^{(j)})$ in (1). In the following, we call independent variables the variables with respect to which we want to compute the derivatives. Then the three required modifications for the code are: the type of variables which depend directly or indirectly on independent variables are defined to be the new data type, the independent variables are initialized and in the end, the derivative information is copied to its data structures like into a matrix in our case. In Appendix A, we give a short example how the Jacobian can be computed using Fortran 90 and automatic differentiation for one-dimensional equilibrium radiation diffusion problem. In the same manner, we have computed the Jacobians for all problems considered in Section 6.

The cost of forming the Jacobian using the automatic differentiation depends how the residual vector is computed. We analyze this cost under the assumption that the residual is evaluated component wise like in Appendix A. For a more detailed analysis of computational cost and memory usage of the automatic differentiation we refer to [22,23]. When computing the residual at a given grid point the unknowns appearing in the finite difference stencil are declared as independent variables which means that the derivatives are computed with respect to all of them. These derivatives of the residual then define the corresponding row of the Jacobian. In Section 4.2, we describe nine-point finite difference stencils for both radiation energy and material temperature for two-dimensional non-equilibrium radiation diffusion problems. Furthermore, the equations for energy and temperature are coupled. Thus, we have 18 independent variables.

Whenever we perform an arithmetical operation with variables which depend directly or indirectly on the independent variables the overloaded operation compute also the derivatives of the result with respect to all independent variables using the chain rule. Each arithmetical operation requires from one to five additional operations to compute the derivative with respect to one independent variable. On average one operation requires about two operations for each independent variables to compute the derivatives. For example, with 18 independent variables one arithmetical operation can generate from 18 to 90 additional operations to compute the derivatives.

The actual CPU time required by the automatic differentiation depends on a great extent how well the Fortran 90 compiler can optimize the operator overloading code. Many contemporary compilers are not good in this optimization and, thus, the operator overloading caries a heavy overhead, but still the increase in the number of arithmetical operations can give a pessimistic estimate for the CPU time usage. A rough estimate of CPU time usage is obtained by multiplying the running time by the number of independent variables.

One alternative way to form the Jacobian is to compute it row by row using finite differences. With this approach for each row it is necessary to compute as many finite differences as there are points in the finite difference stencil which is the same as the number of independent variables with the automatic differentiation. Thus, the computational cost of this approach is comparable to the automatic differentiation.

By hand coding the analytical expressions for the Jacobian it is possible to reduce the cost compared to the automatic differentiation. A large part of computations with the automatic differentiation are performed with zeros, since many of the intermediate variables do not depend on all independent variables. These arithmetical operations can be omitted when using the analytical Jacobian. Also the overhead associated to the operator overloading can be avoided. Thus, an efficiently coded analytical Jacobian can be a few times faster than the automatic differentiation.

With a Newton–Krylov method it is sufficient to be able to multiply vectors by the Jacobian. This product can be approximated by using a finite difference approximation of the Jacobian; see [15] and references therein. A forward finite difference gives an approximation

$$\mathscr{J}(\mathscr{E})\mathscr{F} \approx \frac{1}{\epsilon}(\mathscr{B}(\mathscr{E}+\epsilon\mathscr{F}) - \mathscr{B}(\mathscr{E})), \tag{4}$$

where ϵ is a small positive number. Thus, the cost of a multiplication is about the same as the evaluation of the residual $\mathscr{B}(\mathscr{E}+\epsilon\mathscr{F})$ when $\mathscr{B}(\mathscr{E})$ has been computed beforehand. One approximate multiplication (4) is several times faster than forming the Jacobian. Furthermore, a considerable amount of memory is saved by not storing the Jacobian. On the other hand, the multiplication of a vector by the stored Jacobian is several times faster than the approximation (4). Therefore, we can expect that the use of (4) is faster when the number of Krylov iterations is small and the use of the stored Jacobian is faster when the number of Krylov iterations is large. Some preconditioners like the multigrid preconditioner based on the Galerkin coarsening in Section 5 require the Jacobian matrix.

An advantage of the Jacobian computed using the automatic differentiation and the analytic Jacobian is that their accuracy is usually comparable to the machine precision. Generally finite difference approximations are much less accurate and they require to choose a finite difference step size which has a strong impact on accuracy. The typical accuracy of a first-order difference approximation is the square root of the machine precision, that is, a half of the decimals are correct. With higher order finite differences accuracy can be improved, but they are computationally much more expensive. The accuracy of the first-order difference approximation is sufficiently high for many problems. For example, for radiation diffusion problems they have been used with good success in [4,5].

3. Model problem

We consider the solution of the gray approximation for an idealized thermal non-equilibrium radiation diffusion problem [5]. This is a coupled system for the radiation energy E and the material temperature T. For our model problem, let Ω be one-dimensional or two-dimensional rectangular domain whose sides have unit length, that is, $\Omega = (0,1)^d$, where d is the dimension of the problem.

The gray approximation for the radiation diffusion equation reads

$$\frac{\partial E}{\partial t} - \nabla \cdot (D_{\mathrm{r}}(T,E)\nabla E) = \sigma(T)(T^4 - E) \quad \text{in } \Omega \tag{5}$$

and the material energy balance equation reads

$$\frac{\partial T}{\partial t} - \nabla \cdot (D_{\mathrm{t}}(T)\nabla T) = -\sigma(T)(T^4 - E) \quad \text{in } \Omega. \tag{6}$$

In a thermal equilibrium the radiation energy and the material temperature are in the balance $E = T^4$. In this special case, the radiation energy can be obtained by solving Eq. (5) in which the right-hand side is zero. We will not consider this special case here.

In (5) and (6), $\sigma(T)$ denotes the photon absorption cross-section given by

$$\sigma(T) = \frac{z^3}{T^3}, \tag{7}$$

where z is the atomic mass number of the medium which may vary from one point to another. The diffusion coefficient for the radiation energy in (5) is

$$D_{\mathrm{r}}(T,E) = \frac{1}{3\sigma(T) + |\nabla E|/E}, \tag{8}$$

where the second term in the denominator is a flux limiter preventing the flux of energy to move faster than the speed of light. Alternative flux limiters have been considered in [26] and references therein. The diffusion coefficient for the material energy balance equation (6) is

$$D_{\mathrm{t}}(T) = kT^{5/2}, \tag{9}$$

where k is a constant.

In the model problems considered in this article, there is a unit radiation flux on the left boundary of the domain [2,5]. This corresponds to the Robin boundary condition

$$\frac{1}{4}E - \frac{1}{6\sigma}\frac{\partial E}{\partial x_1} = 1 \quad \text{on } x_1 = 0. \tag{10}$$

On the right boundary, we pose the homogeneous Robin boundary condition

$$\frac{1}{4}E - \frac{1}{6\sigma}\frac{\partial E}{\partial x_1} = 0 \quad \text{on } x_1 = 1. \tag{11}$$

In the two-dimensional case, the solution is chosen to be symmetric (or insulated) over the lower and upper boundaries, that is,

$$\frac{\partial E}{\partial x_2} = 0 \quad \text{on } x_2 = 0 \quad \text{and} \quad x_2 = 1. \tag{12}$$

For the material temperature, we pose homogeneous Neumann boundary conditions on all boundaries, that is,

$$\frac{\partial T}{\partial x_i} = 0 \quad \text{on } x_i = 0 \quad \text{and} \quad x_i = 1 \tag{13}$$

for $i = 1$ for one-dimensional problems and $i = 1, 2$ for two-dimensional problems.

In our model problems, the material is initially cold and the radiation energy is in the equilibrium with the material temperature. At the initial time $t = 0$, the radiation energy is chosen to be $E = 10^{-5}$ and, thus, the material temperature is $T = E^{1/4} = 10^{-5/4}$.

4. Discretization

4.1. Temporal discretizations

We consider the temporal discretization of the system

$$\frac{\partial \mathscr{E}}{\partial t} = \mathscr{C}(\mathscr{E}) \tag{14}$$

of differential equations, where $\mathscr{E}$ is a vector and $\mathscr{C}$ is a non-linear vector-valued function. In Section 4.2, the spatial discretization of the radiation diffusion problem leads to this kind of a system.

We sought $\mathscr{E}$ at the times t_k, $k = 0, 1, \ldots, M$, where t_M is some final time. For t_ks it holds that $t_0 = 0$ and $t_k < t_{k+1}$. In the following formulas, we denote the length of the time step from t_k to t_{k+1} by Δt_{k+1}. Furthermore, $\mathscr{E}^k$ denotes the vector $\mathscr{E}$ at the time t_k. We will consider several fully implicit time discretizations which have better stability properties than semi-implicit or explicit discretizations. Typically, for implicit methods the choice of the length of time steps can be based on accuracy considerations without limits due to stability.

For (14), the simplest implicit time discretization is the implicit (backward) Euler method defined by

$$\mathscr{E}^{k+1} = \mathscr{E}^k + \Delta t_{k+1} \mathscr{C}(\mathscr{E}^{k+1}). \tag{15}$$

Let Δt denote the length of largest time step, that is, $\Delta t = \max_{1 \leqslant k \leqslant M} \Delta t_k$. Then the accuracy of the final solution E^M is $\mathscr{O}(\Delta t)$ with this scheme when $\mathscr{C}$ is a linear operator [27].

A more accurate time discretization is obtained by using the mid-point quadrature rule. This leads to the mid-point scheme

$$\mathscr{E}^{k+1} = \mathscr{E}^k + \Delta t_{k+1} \mathscr{C}((\mathscr{E}^{k+1} + \mathscr{E}^k)/2). \tag{16}$$

For a linear $\mathscr{C}$ the formula (16) gives a second-order accurate $\mathscr{O}((\Delta t)^2)$ final solution $\mathscr{E}^M$. Furthermore, for a linear $\mathscr{C}$ this discretization coincides with the Crank–Nicolson scheme which is obtained as the average of the implicit and explicit Euler methods.

The BDF2 is a popular second-order accurate implicit multistep method. For arbitrary time steps Δt_k and Δt_{k+1}, it is given by

$$\mathscr{E}^{k+1} = (1 + r_k)\mathscr{E}^k - r_k \mathscr{E}^{k-1} + (\Delta t_{k+1} - r_k \Delta t_k)\mathscr{C}(\mathscr{E}^{k+1}), \tag{17}$$

where the scalar r_k is defined as

$$r_k = \frac{(\Delta t_{k+1})^2}{(\Delta t_k)^2 + 2\Delta t_k \Delta t_{k+1}}. \tag{18}$$

In the case that Δt_k and Δt_{k+1} are equal, r_k has the value 1/3 and the backward differentiation formula (17) simplifies to a more familiar form

$$\mathscr{E}^{k+1} = \frac{4}{3}\mathscr{E}^k - \frac{1}{3}\mathscr{E}^{k-1} + \frac{2}{3}\Delta t_{k+1}\mathscr{C}(\mathscr{E}^{k+1}). \tag{19}$$

Since the BDF2 step requires two previous solutions, the time integration has to be started with some other method. One possibility is to use the implicit Euler method for the first time step. This is likely to reduce the accuracy of the solution $\mathscr{E}^M$, but the asymptotic second-order accuracy is still maintained.

Another second-order accurate scheme is the two-stage implicit Runge–Kutta method [28] defined by

$$\tilde{\mathscr{E}}^{k+1} = \mathscr{E}^k + \Delta t_{k+1}\Big((1-\theta)\mathscr{C}(\mathscr{E}^k) + \theta\mathscr{C}(\tilde{\mathscr{E}}^{k+1})\Big), \tag{20}$$

$$\mathscr{E}^{k+1} = \mathscr{E}^k + \Delta t_{k+1}\left(\frac{1}{2}\mathscr{C}(\mathscr{E}^k) + \left(\frac{1}{2} - \theta\right)\mathscr{C}(\tilde{\mathscr{E}}^{k+1}) + \theta\mathscr{C}(\mathscr{E}^{k+1})\right),$$

where $\tilde{\mathscr{E}}^{k+1}$ is an intermediate stage value and θ is a constant having the value $1 - 1/\sqrt{2}$.

The implicit Euler method and the Runge–Kutta method are L-stable [27–29] which is a desirable property for methods used to integrate parabolic partial differential equations. The mid-point and Crank–Nicolson schemes are not L-stable and the solutions computed using these methods can have unphysical oscillations unless very small time steps are used.

For the time step size selection we use the approach proposed in [3], further analyzed in [7], and used in [11,14]. The basic idea is to estimate the propagation speed of the dominant wave. An approximation of this speed is obtained by computing

$$s_k = \left\|\frac{\partial E^k}{\partial t}\right\|_1 \Big/ \|\nabla E^k\|_1, \tag{21}$$

where $\|\cdot\|_1$ denotes the L_1-norm and E^k is the radiation energy at the time t_k. The gradient of E is approximated using the central differences in the interior of Ω and one sided differences on the boundaries $\partial\Omega$. The time derivative of E is approximated using the backward difference in time. The time step is controlled by defining a desired CFL number. We estimate the current CFL number by

$$\mathrm{CFL}_k = \frac{\Delta t_k}{\Delta x}s_k. \tag{22}$$

Then the new time step is given by

$$\Delta t_{k+1} = \min\left(\frac{\mathrm{CFL}}{\mathrm{CFL}_k}, 1.25\right)\Delta t_k, \tag{23}$$

where the factor 1.25 limits the growth rate of the time step. The size of the first time step is chosen to be

$$\Delta t_1 = \mathrm{CFL}\Delta x/10.$$

4.2. Spatial discretization

We define a grid with a uniform grid step size $\Delta x = 1/(I-1)$ to all d directions, where I is the number of grid points in each direction. In the following, we consider a two-dimensional model problem. One-dimensional and three-dimensional problems can be discretized in the same manner. The grid point (i,j), $1 \leqslant i \leqslant I$,

$1 \leqslant j \leqslant I$, is located at $((i-1)\Delta x, (j-1)\Delta x)$. Our unknowns are the grid point values and not the cell center values like in [5]. We denote the value of the radiation energy E and the material temperature T at the grid point (i,j) by $E_{i,j}$ and $T_{i,j}$, respectively.

Our discretization is based on central finite differences and, thus, the accuracy of the spatial discretization is formally second-order accurate with respect to the grid step size Δx. For two-dimensional problems we obtain a nine point difference scheme for E and a five point scheme for T. Furthermore, these schemes are coupled due to the equilibration coupling in (5) and (6). A similar discretization was used in [5]. For an interior grid point (i,j) the difference formula for the diffusion for the radiation energy in (5) reads

$$\begin{aligned}\nabla\cdot(D_{\mathrm{r}}(T,E)\nabla E) \approx \big(&D_{\mathrm{r},i,j+1/2}(E_{i,j+1}-E_{i,j}) + D_{\mathrm{r},i-1/2,j}(E_{i-1,j}-E_{i,j}) + D_{\mathrm{r},i+1/2,j}(E_{i+1,j}-E_{i,j}) \\ &+ D_{\mathrm{r},i,j-1/2}(E_{i,j-1}-E_{i,j})\big)/(\Delta x)^2,\end{aligned} \tag{24}$$

where diffusion coefficients are evaluated at the mid points between the grid points. They are given by

$$D_{\mathrm{r},i+a/2,j+b/2} = \frac{1}{3\sigma_{i+a/2,j+b/2} + 2G_{i+a/2,j+b/2}/(E_{i+a,j+b}+E_{i,j})}, \tag{25}$$

where the values of a and b can be -1 or 1 so that the subscript matches one of the subscripts appearing in (24). Here, the photon absorption cross-section is given by

$$\sigma_{i+a/2,j+b/2} = \left(\frac{z((i+a/2-1)\Delta x, (j+b/2-1)\Delta x)}{\frac{1}{2}(T_{i+a,j+b}+T_{i,j})}\right)^3, \tag{26}$$

where $z(x_1,x_2)$ is the function giving the atomic mass number of the medium at the point (x_1,x_2). Our approximation for the norm of the gradient reads

$$G_{i+a/2,j+b/2} = \frac{1}{\Delta x}\left[\left(E_{i+a,j+b}-E_{i,j}\right)^2 + \frac{1}{16}\left(E_{i+b,j+a+b}+E_{i+a+b,j+a}-E_{i-b,j-a+b}-E_{i+a-b,j-a}\right)^2\right]^{1/2}.$$

The diffusion $\nabla\cdot(D_{\mathrm{t}}(T)\nabla T)$ in the material balance equation (6) is discretized using the scheme (24) with the coefficients $D_{\mathrm{r},i+a/2,j+b/2}$ replaced by

$$D_{\mathrm{t},i+a/2,j+b/2} = k\left(\frac{1}{2}(T_{i+a,j+b}+T_{i,j})\right)^{5/2}. \tag{27}$$

The discrete form of the equilibration coupling term $\sigma(T)(T^4-E)$ in (5) and (6) at the grid point (i,j) is

$$\sigma_{i,j}\left(T_{i,j}^4 - E_{i,j}\right), \tag{28}$$

where $\sigma_{i,j}$ is given by (26).

Let us consider the discretization of the diffusion for the radiation energy at the grid points $(1,j)$ on the boundary $x_1 = 0$. The discretization is performed in the same way for the diffusion for the material temperature and on the other boundaries. We introduce fictitious grid points $(0,j)$ and the associated fictitious values $E_{0,j}$. By using the central finite difference for the partial derivative in (10), we obtain the approximation

$$\frac{1}{4}E_{1,j} - \frac{1}{12\sigma_{1,j}\Delta x}(E_{2,j}-E_{0,j}) = 1. \tag{29}$$

Solving $E_{0,j}$ from (29) gives us the expression

$$E_{0,j} = E_{2,j} - 3\sigma_{1,j}\Delta x(E_{1,j}-4) \tag{30}$$

for the fictitious values. Now, we obtain the discretization on the boundary by using the same difference scheme (24) for the grid points $(1,j)$ and by replacing the fictitious values in the scheme by the expression in (30).

The use of the described spatial discretization leads to the semidiscrete problem (14), where

$$\mathscr{E} = \begin{pmatrix} \mathbf{E} \\ \mathbf{T} \end{pmatrix}. \tag{31}$$

Here, the vectors $\mathbf{E}$ and $\mathbf{T}$ contain the values of E and T, respectively, at the grid points.

5. GMRES and multigrid preconditioning

In this section for simplicity we denote the Jacobian matrix $\mathscr{J}(\mathscr{E}^{(j)})$ by $\mathscr{A}$. We use the GMRES method [16] with a multigrid preconditioner to solve a system of linear equations

$$\mathscr{A}\mathscr{F} = \mathscr{G}$$

for a given vector $\mathscr{G}$. Let the matrix $\mathscr{B}$ correspond the action of our multigrid preconditioner. Then, we solve a left preconditioned system of linear equations

$$\mathscr{B}\mathscr{A}\mathscr{F} = \mathscr{B}\mathscr{G}, \tag{32}$$

with the iterative GMRES method.

The Jacobian matrix has the block form

$$\mathscr{A} = \begin{pmatrix} A_{EE} & A_{ET} \\ A_{TE} & A_{TT} \end{pmatrix},$$

where the first and second block row corresponds to the unknown values of the energy E and the temperature T, respectively, at the grid points.

In the following, we describe the multigrid method corresponding to the preconditioner $\mathscr{B}$. Coarse grids are obtained by doubling the grid step sizes from one level to the next. We denote the matrix associated to an interpolation (prolongation) operation for a scalar grid function by P. For one-dimensional, two-dimensional, and three-dimensional problems we use linear, bilinear, and trilinear interpolation operation, respectively. Another possibility would be to use operator dependent interpolation operations like in [21], but our simpler interpolations already yielded good results in our numerical experiments. The interpolation for a function pair (E, T) corresponds to the matrix

$$\mathscr{P} = \begin{pmatrix} P & 0 \\ 0 & P \end{pmatrix}.$$

We employ the full weighting restriction operator [18]. Thus, the restriction of a vector $\mathscr{R}$ is given by

$$\mathscr{R}^{\mathrm{c}} = \frac{1}{2d}\mathscr{P}^{\mathrm{T}}\mathscr{R},$$

where d is the dimension of the problem. The operators for the coarser grids are obtained using the Galerkin coarsening [18]. Thus, the matrix corresponding to the next coarser grid is given by

$$\mathscr{A}^{\mathrm{c}} = \mathscr{P}^{\mathrm{T}}\mathscr{A}\mathscr{P}.$$

In our multigrid we smooth grid functions by performing a few Gauss–Seidel iterations. For this the unknowns are ordered in such a way that the grid point value of T follows the value of E at the same grid point. Thus, this is not the block ordering used for other operations. Now we have considered all details of our multigrid method and we can introduce Algorithm 1.

Algorithm 1. (A recursive multigrid method)

```
**Multigrid** $(\mathscr{A}, \mathscr{F}, \mathscr{G})$
If the size of $\mathscr{A}$ is small then solve $\mathscr{A}\mathscr{F} = \mathscr{G}$ and return
Smooth $\mathscr{F}$ with a few Gauss–Seidel sweeps
Compute residual $\mathscr{R} = \mathscr{A}\mathscr{F} - \mathscr{G}$
Compute restriction $\mathscr{R}^{\mathrm{c}} = \frac{1}{2d}\mathscr{P}^{\mathrm{T}}\mathscr{R}$
Compute Galerkin coarsening $\mathscr{A}^{\mathrm{c}} = \mathscr{P}^{\mathrm{T}}\mathscr{A}\mathscr{P}$
Set $\mathscr{F}^{\mathrm{c}} = 0$
Call **Multigrid**$(\mathscr{A}^{\mathrm{c}}, \mathscr{F}^{\mathrm{c}}, \mathscr{G}^{\mathrm{c}})$
Compute interpolation $\mathscr{R} = \mathscr{P}\mathscr{F}^{\mathrm{c}}$
Set $\mathscr{F} = \mathscr{F} - \mathscr{R}$
Smooth $\mathscr{F}$ with a few Gauss–Seidel sweeps
```

An alternative way to construct coarse grid operators is to restrict $\mathscr{E}^{(j)}$ to coarsers grids and then form the Jacobian matrices for the coarsers grids. We have also made numerical experiments with this approach. In these experiments, which are not reported in this article, the convergence of the GMRES methods was almost as fast as with the Galerkin coarsening.

6. Numerical results

In all numerical experiments we have performed time integration for three time units. We have used two presmooth Gauss–Seidel iterations and two postsmooth Gauss–Seidel iterations in Algorithm 1. Unless otherwise stated the stopping criterion in Newton's method has been the l_2-norm of the residual of (1) is less than 10^{-4} and in the GMRES method it has been the l_2-norm of the residual of (32) is reduced by the factor of 10^{-4}. According to several test runs these conditions seem to be sufficient to reduce the error due to the termination of iterations well below the discretization error for all discretizations used in the experiments. For coarser discretizations less strict criteria would have been sufficient to accomplish the same, but for uniformity and simplicity we have used the same criteria for all discretizations.

6.1. One-dimensional test problem

Our first numerical experiments are performed with one-dimensional problems having the atomic mass number $z = 1$ in the whole interval $(0, 1)$. We start by studying the convergence of different time discretizations when the coefficient k in the material energy balance equation (6) is zero. For this purpose, we define the L_2 error of the material temperature T to be

$$\left[\frac{1}{2}\left(T_1 - T_1^{(\mathrm{r})}\right)^2 \Delta x + \sum_{i=2}^{I-1}\left(T_i - T_i^{(\mathrm{r})}\right)^2 \Delta x + \frac{1}{2}\left(T_I - T_I^{(\mathrm{r})}\right)^2 \Delta x\right]^{1/2}, \tag{33}$$

where $T_1^{(\mathrm{r})}$ is the reference solution. Fig. 1 shows the time convergence of the L_2 error at the final time $t = 3$ for the discretizations when the grid step is $\Delta x = 1/128$. The reference solution has been computed with CFL = 0.001, the same grid step size, and using 10^{-7} instead of 10^{-4} in the stopping criteria for both iterations. In Fig. 1, IE denotes the implicit Euler method, MP denotes the mid-point scheme, IRK denotes the two-stage implicit Runge–Kutta method, and BDF denotes the two-step backward differentiation formula.

The implicit Euler method approaches first-order convergence with respect to the time step size when the CFL number is decreased. The convergence rate of the Runge–Kutta method varies slightly with the

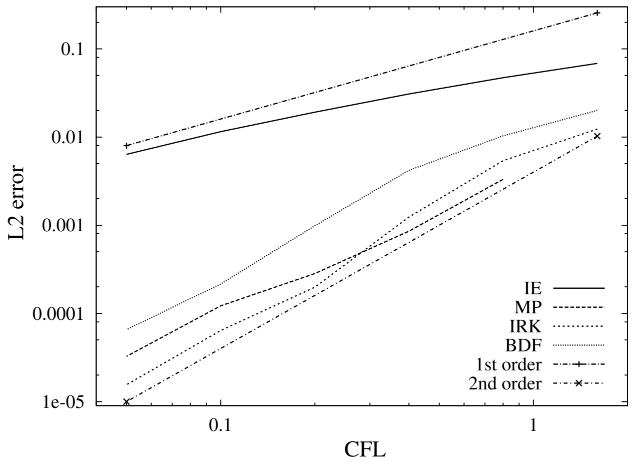

Fig. 1. The time convergence of different time discretizations for the one-dimensional problem with $\Delta x = 1/128$.

average rate being second-order over the considered CFL range. The BDF is second-order accurate once CFL is less than 0.4 and the mid-point scheme seems to be second-order when CFL is less than 0.1. When the CFL number is 0.1 the L_2 error of the mid-point scheme coincides quite well with the L_2 error reported for the Crank–Nicolson method in [14]. When the CFL number was one or larger the mid-point scheme led to a negative value of energy at some point of space and time and the simulation failed due to this. This and the need to have CFL number less than 0.1 to obtain second-order convergence rate are probably due to the lack of L-stability of the mid-point scheme.

Next, we study the space–time convergence of the discretizations. For this comparison the reference solution is computed using $\Delta x = 1/2048$ and the mid-point scheme with CFL = 0.05. We have used the straight injection of the reference solution to the coarser grid when computing the L_2 error defined by (33). Fig. 2 shows the L_2 error for the different time discretizations and for three grid step sizes: $\Delta x = 1/32$, $\Delta x = 1/128$, and $\Delta x = 1/512$. With the implicit Euler method it is necessary to decrease the CFL

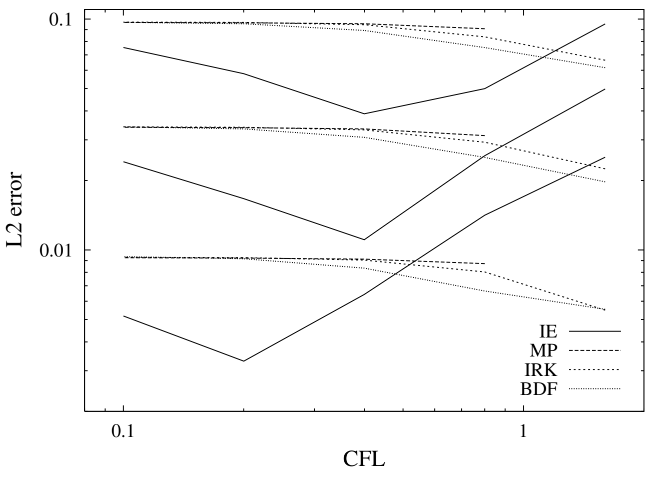

Fig. 2. The convergence of different time discretizations for one-dimensional problems with $\Delta x = 1/32$ (top), $\Delta x = 1/128$, and $\Delta x = 1/512$ (bottom).

number when the grid is refined in order to obtain smallest possible error. This is a consequence of the first-order convergence of the implicit Euler method. For other methods a constant CFL number is sufficient to obtain smallest possible error. Interestingly the largest used CFL number leads to the smallest error for other methods. With the mid-point scheme and CFL = 0.1 the errors for $\Delta x = 1/128$ and $\Delta x = 1/512$ are about 3.40×10^{-2} and 9.26×10^{-3}, respectively. The ratio of these is about 3.67 which suggest that the convergence of the space discretization could be first-order. This is a typical convergence rate for grid point value based discretizations when the solution is non-smooth at the scale defined by the grid. Similar convergence rates were observed also in [13,14].

Fig. 3 shows the material temperature T at the final time $t = 3$ computed using three different grids and the mid-point scheme with CFL = 0.8. The temperature distributions are in good agreement with the plots of T for the one-dimensional problem shown in [5].

Table 1 reports the convergence of the Newton iterations and the preconditioned inner GMRES iterations. In Table 1, we report results for the values 0 and 0.1 for the coefficient k in (6). A smaller CFL number, that is, a smaller time step makes both iterations converge faster. Also, the iterations converge faster with a larger coefficient k. The number of preconditioned GMRES iterations is rather insensitive to the grid step size which makes the preconditioner scalable.

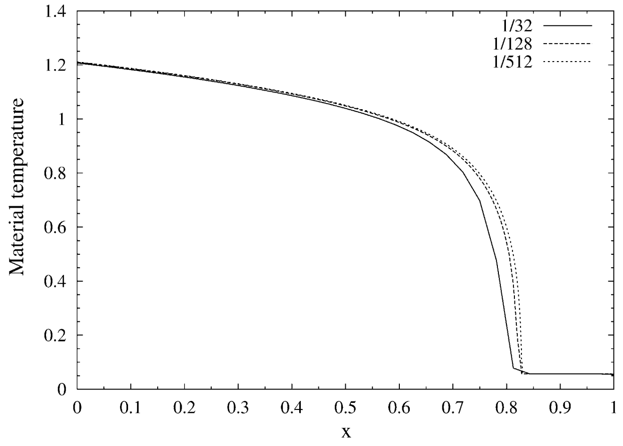

Fig. 3. The material temperature T after three time units computed using three different grid step sizes Δx, the mid-point scheme, and CFL = 0.8.

Table 1
The convergence of iterative methods for one-dimensional problems

k	CFL	$\Delta x = 1/64$		$\Delta x = 1/128$		$\Delta x = 1/256$	
		Newton it./time step	GMRES it./ Newton it.	Newton it./time step	GMRES it./ Newton it.	Newton it./time step	GMRES it./ Newton it.
0.0	0.8	4.32	2.66	4.40	2.88	4.35	2.93
0.0	0.4	3.23	1.97	3.23	1.98	2.98	1.98
0.0	0.2	2.52	1.96	2.40	1.98	2.15	1.98
0.0	0.1	1.95	1.92	1.98	1.92	1.99	1.86
0.1	0.8	3.63	2.16	3.60	2.21	3.19	2.27
0.1	0.4	2.82	1.94	2.85	1.96	2.34	1.98
0.1	0.2	2.07	1.91	1.99	1.93	1.98	1.97
0.1	0.1	1.95	1.70	1.98	1.79	1.32	1.90

A similar convergence study has been performed in [5]. It is difficult to compare their results to ours, since their discretizations and stopping criteria for iterations are different. The Newton iteration counts reported in [5] are slightly larger, but their stopping criterion for Newton's method is also slightly more strict. Thus, the convergence of Newton's method seems to be comparable in here and [5]. The stopping criteria for the GMRES method are quite different and due to this we do not compare the convergence of the preconditioner GMRES methods.

6.2. Two-dimensional test problem with $z = 1$

The first two-dimensional test problem is a trivial extension of the previous one-dimensional problem. The boundary conditions were described in Section 3. The exact solution is obtained from the solution of the one-dimensional problem by setting it to be constant in the x_2-direction.

We have only used the mid-point scheme for the time discretization for the two-dimensional problems. The L_2 error of the solution has been defined analogously to the L_2 error in (33) for one-dimensional problems. According to our experiments at the final time $t = 3$ the L_2 errors of the solutions of the two-dimensional problems coincides with three or four decimals to the L_2 errors of the solutions of the corresponding one-dimensional problems.

The convergence of Newton's method and the preconditioned GMRES method are reported in Table 2. These experiments are the same ones as with the one-dimensional problem. The behavior is very similar with the only difference being a slight growth in the number of iterations. Thus, the same conclusions as in the case of the one-dimensional problems are also valid for this problem.

For a comparison we also reported in Table 3 the average number of iterations when we do not have a preconditioner. Furthermore, in Table 4 we have given the iteration counts when the finite difference approximation (4) of the Jacobian is used. According to [5], we have chosen the finite difference length to be

Table 2
The convergence of iterative methods for two-dimensional problems with $z = 1$

k	CFL	$\Delta x = 1/64$		$\Delta x = 1/128$		$\Delta x = 1/256$	
		Newton it./time step	GMRES it./ Newton it.	Newton it./time step	GMRES it./ Newton it.	Newton it./time step	GMRES it./ Newton it.
0.0	0.8	4.62	3.08	4.87	3.17	4.71	3.30
0.0	0.4	3.71	2.51	3.69	2.36	3.53	2.24
0.0	0.2	2.91	1.95	2.94	1.97	2.93	1.98
0.0	0.1	2.15	1.95	2.00	1.97	2.00	1.98
0.1	0.8	3.86	2.81	3.86	2.80	3.80	2.65
0.1	0.4	3.08	2.08	2.97	1.97	2.97	1.98
0.1	0.2	2.86	1.94	2.40	1.94	2.10	1.96
0.1	0.1	1.98	1.80	1.99	1.86	2.00	1.94

Table 3
The convergence of iterative methods without a preconditioner for two-dimensional problems with $z = 1$

k	CFL	$\Delta x = 1/64$		$\Delta x = 1/128$		$\Delta x = 1/256$	
		Newton it./time step	GMRES it./ Newton it.	Newton it./time step	GMRES it./ Newton it.	Newton it./time step	GMRES it./ Newton it.
0.0	0.8	4.62	43.24	4.88	65.12	4.71	96.14
0.0	0.4	3.71	29.25	3.69	43.16	3.53	63.06
0.0	0.2	2.91	19.13	2.94	27.64	2.93	40.58
0.0	0.1	2.14	12.75	2.00	17.80	2.00	25.84

Table 4
The convergence of Jacobian-free iterative methods without a preconditioner for two-dimensional problems with $z = 1$

k	CFL	$\Delta x = 1/64$		$\Delta x = 1/128$		$\Delta x = 1/256$	
		Newton it./time step	GMRES it./ Newton it.	Newton it./time step	GMRES it./ Newton it.	Newton it./time step	GMRES it./ Newton it.
0.0	0.8	4.62	26.62	4.88	36.20	4.71	53.96
0.0	0.4	3.71	19.69	3.69	25.16	3.53	36.62
0.0	0.2	2.91	13.96	2.94	16.96	2.93	24.42
0.0	0.1	2.14	9.74	2.00	11.34	2.00	15.99

$$\epsilon = \frac{5 \times 10^{-8} \|\mathscr{E}\|_1}{N\|\mathscr{F}\|_2},$$

where $\|\cdot\|_1$ and $\|\cdot\|_2$ denote the l_1 and l_2 vector norms, respectively, and $N = 2I^2$ is the length of the vectors. With the finite difference approximation of the Jacobian we cannot use the proposed multigrid preconditioner, because it requires the Jacobian matrix.

The use of the multigrid preconditioner or the finite difference approximation of the Jacobian does not have essential effect to the number of Newton iterations. The number of GMRES iterations is from 6 to 30 times larger without a preconditioner. When the mesh step size is halved the number of GMRES iterations grows roughly by the factor of 1.5 when no preconditioner is used while with the preconditioner there is no growth or it is very mild. For this test problem, the finite difference approximation of the Jacobian leads to better conditioned systems of linear equations.

6.3. Two-dimensional test problem with jump in z

In the second two-dimensional problem the atomic mass number is given by

$$z = \begin{cases} 10, & x \in [1/3, 2/3] \times [1/3, 2/3], \\ 1, & \text{otherwise} \end{cases}$$

and $k = 0.01$ is the diffusion coefficient in the material energy balance equation. In order to improve the accuracy of the discretization we make grid lines to coincide with the lines $x_1 = 1/3$, $x_1 = 2/3$, $x_2 = 1/3$, and $x_2 = 2/3$. This is accomplished by choosing the number of grid points in both directions to be multiple of three plus one.

Table 5 reports the convergence of Newton's method and the preconditioned GMRES method. Newton's method takes slightly less iterations when there is a jump in z. This is probably due to the stopping criterion which requires the norm of the residual to be less than 10^{-4}. The larger value of z in part of the domain makes the diffusion of the radiation energy smaller and also the residual. The preconditioned GMRES method requires more iterations in this problem, but the number of iterations is still small.

Table 5
The convergence of iterative methods for two-dimensional problems with a jump in z

CFL	$\Delta x = 1/96$		$\Delta x = 1/192$		$\Delta x = 1/384$	
	Newton it./time step	GMRES it./ Newton it.	Newton it./time step	GMRES it./ Newton it.	Newton it./time step	GMRES it./ Newton it.
0.4	4.01	6.79	3.74	5.21	3.58	4.06
0.2	3.24	5.85	2.98	4.28	2.94	3.43
0.1	2.54	4.93	2.39	3.79	2.04	2.89

The reason for the larger number of iterations is that the standard multigrid is not so effective when there is a jump in coefficients. An interesting property of this problem is that the number of both iterations decrease as the grid is made finer. This same behavior was also observed in [5]. The convergence of the preconditioned GMRES is rather good even though the coefficients have very large jumps in this problem.

Tables 6 and 7 give the average iteration counts for the two coarser discretizations without a preconditioner and with the finite difference approximation (4) of the Jacobian, respectively. The finite difference length is the same as in Section 6.2. Again the number of Newton iterations is essentially the same for three different versions. Unlike for $z = 1$ now the number of GMRES iterations are about same when the Jacobian matrix is computed or its finite difference approximation is used. The number of GMRES iterations is from 15 to 34 times larger without a preconditioner. Thus, the impact of the multigrid preconditioner to the efficiency much greater when there is a jump in the atomic mass number z.

Table 8 studies the accuracy of the discretizations. We use the solution computed using the finest grid and smallest time steps as a reference solution, since we cannot compute more accurate reference solution with a reasonable amount of time. Table 8 suggests that the convergence of the space discretization could be first-order, since the difference is approximately halved when we move to the right in Table 8. Based on the last column the convergence of the mid-point time discretization seems to be second-order. Fig. 4 shows contour lines of the material temperature at the final time $t = 3$ computed using the finest grid.

Table 6
The convergence of iterative methods without a preconditioner for two-dimensional problems with a jump in z

CFL	$\Delta x = 1/96$		$\Delta x = 1/192$	
	Newton it./time step	GMRES it./Newton it.	Newton it./time step	GMRES it./Newton it.
0.4	4.00	112.00	3.74	176.11
0.2	3.25	89.55	2.98	129.94
0.1	2.54	78.50	2.39	116.14

Table 7
The convergence of Jacobian-free iterative methods without a preconditioner for two-dimensional problems with a jump in z

CFL	$\Delta x = 1/96$		$\Delta x = 1/192$	
	Newton it./time step	GMRES it./Newton it.	Newton it./time step	GMRES it./Newton it.
0.4	4.01	113.17	3.74	177.25
0.2	3.25	90.50	2.98	131.02
0.1	2.54	79.24	2.39	116.84

Table 8
The L_2 differences between a given material temperature and the material temperature computed with $\Delta x = 1/384$ and CFL = 0.1

CFL	$\Delta x = 1/48$	$\Delta x = 1/96$	$\Delta x = 1/192$	$\Delta x = 1/384$
0.4	6.44×10^{-2}	3.17×10^{-2}	1.19×10^{-2}	2.63×10^{-4}
0.2	6.46×10^{-2}	3.20×10^{-2}	1.21×10^{-2}	6.56×10^{-5}
0.1	6.49×10^{-2}	3.21×10^{-2}	1.22×10^{-2}	0

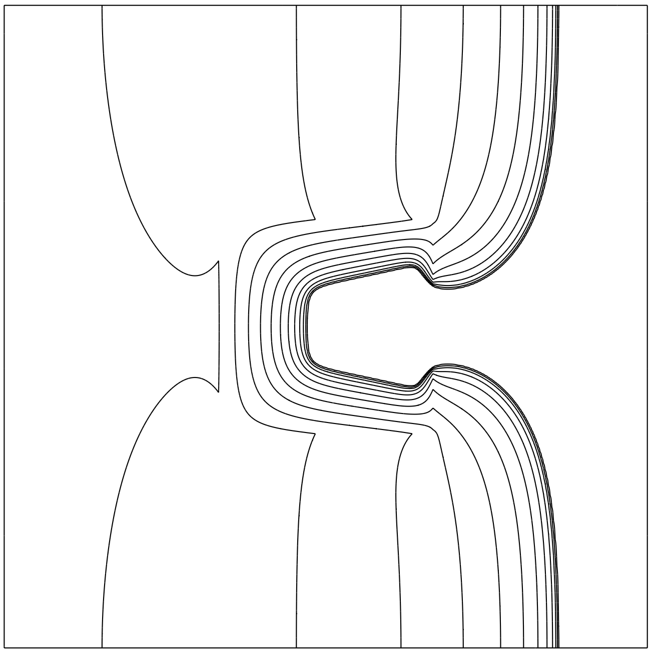
Fig. 4. Contour lines corresponding the material temperatures $T = 0.1, 0.2, \ldots, 1.2$ computed using the 385×385 grid.

7. Conclusions

For Newton's method we computed the Jacobian matrix using automatic differentiation which was implemented using the operator overloading in Fortran 90. This required only a slight modification to the computation of the non-linear residual while hand coding this same computation would have been quite laborious. For the iterative solution of systems of linear equations with the Jacobian matrix we introduced a simple multigrid preconditioner. Many of the earlier preconditioners for non-equilibrium radiation diffusion problems have used some physics based knowledge while the multigrid preconditioner does not use such information and, thus, it is more general. Based on numerical experiments the convergence of GMRES iterations with the multigrid preconditioner seems to be comparable to the convergence with the operator splitting preconditioner introduced in [5].

The implicit mid-point scheme leads often to negative radiation energy or material temperature with large time steps corresponding to a CFL number close to one. With the implicit Euler method, the BDF2 and a two-stage implicit Runge–Kutta method it is possible to use larger time steps without encountering negative values. With the second-order methods the accuracy of the solution does not improve when the CFL number is reduced from the value around one. This suggests that the spatial discretization error dominates the temporal discretization error when the CFL number is about one or less.

A more systematic comparison of the accuracy of different discretizations and the efficiency of different solvers would be useful future research topic. A more advanced multigrid preconditioner using possibly special smoothers and matrix-dependent prolongations/restrictions might increase efficiency. This could be especially true for problems with larger jumps in coefficients and when larger time steps are used. One obvious way to improve the efficiency would be to consider the local refinements of grids/meshes like it has been done for equilibrium radiation diffusion problems in [10,12].

Acknowledgments

We thank Dr. Michael Pernice for proposing to study the solution of non-equilibrium radiation diffusion problems and references to articles on this topic. We are grateful for Dr. Dana Knoll for providing information on the spatial discretization and Dr. Raino A.E. Mäkinen for giving us an example Fortran 90 code for automatic differentiation. We also thank Dr. Janne Martikainen for fruitful discussions on multigrid methods. We are grateful for the anonymous reviewers for suggestions which improved the paper considerably. The research was supported by the Los Alamos Computer Science Institute (LACSI).

Appendix A. Example of automatic differentiation

The following Fortran 90 subroutine Code 1 computes the residual for the one-dimensional equilibrium radiation diffusion problem. The code has been augmented with the computation of the Jacobian matrix using automatic differentiation based on the operator overloading. User supplies to the routine the number of unknowns (grid points) `n`, the grid step size `dx` and the vector `e` containing the radiation energy at grid points. The routine returns the residual vector `r` and the tridiagonal Jacobian matrix `j`.

Code 1 computes the residual by going through the grid points in a loop. The vector `ed` contains the radiation energy for the grid point under consideration and the grid points left and right from it. For the Jacobian computation the vector `ed` contains the independent variables. Here, the independent variables mean the variables with respect to which the derivatives are computed.

The new data type containing also the derivative information is `dreal` and it is shown in Code 2. The following modifications have been made to the code in order to compute the Jacobian: (1) the types of the variables `ed`, `re` and the functions `sigma`, `diff` have been defined to be the new data type, (2) the independent variable is initialized using the function `adindep`, (3) one row of the Jacobian is extracted from `re` using the function `adder`.

Code 1. An automatic differentiation example

```
!Subroutine for computing the residual vector and
!the Jacobian matrix. Parameters:
!
!  n   the number of grid points (input)
!  dx  the grid step size (input)
!  e   the grid point values of radiation energy (input)
!  r   the residual vector (output)
!  j   the tridiagonal Jacobian matrix (output)
!
subroutine resjac(n,dx,e,r,j)
  use ad
  integer, intent(in)::n
  real(wp), intent(in)::dx, e(n)
  real(wp), intent(out)::r(n), j(-1:1,n)
  !
  real(wp)::el(3) !three local values
  type(dreal)::ed(3) !three local values and their derivatives
  type(dreal)::re !one component of residual and its derivatives
```

```
    type(dreal), external::sigma, diff
    integer::i
    !
    !Go through the grid points from left to right
    do i=1,n
      !Copy the values of energy at the grid points
      !i-1, i, and i+1 to el
      el=0.d0
      if (i > 1) el(1)=e(i-1)
      el(2)=e(i)
      if (i < n) el(3)=e(i+1)
      !
      !Copy the local values from el to ed and define that we
      !want the derivatives with respect to the components of el
      ed=adindep(el)
      !
      !On the left boundary compute the value of the fictitious
      !grid point outside the interval using the Robin b.c.
      if (i==1) ed(1)=ed(3) - 3.d0*sigma(ed(2))*dx*(ed(2) - 4.d0)
      !
      !On the right boundary compute the value of the fictitious
      !grid point outside the interval using the Robin b.c.
      if (i==n) ed(3)=ed(1) - 3.d0*sigma(ed(2))*dx*ed(2)
      !
      !Compute the residual at the ith grid point
      re=diff(ed(2),ed(3),dx)*(ed(3) - ed(2))/(dx**2) &
        - diff(ed(1),ed(2),dx)*(ed(2) - ed(1))/(dx**2)
      !
      !Copy the residual at the ith grid point to re
      r(i)=re
      !Copy the derivatives to the ith row of the Jacobian matrix
      j(-1:1,i)=adder(re)
    end do
  end subroutine resjac
```

Code 2 gives a part of module containing the definitions needed for the automatic differentiation. The new data type `dreal` contains a vector whose first component contains the value of the variable and others contain the derivatives with respect to the independent variables. We have shown how the multiplication operation is performed with the new data type. In the same manner all other elementary operations and functions can be defined for the data type `dreal`. Also, the functions for initializing the independent variables (`adindep`) and for returning the derivative information (`adders`) are given.

Code 2. A part of automatic differentiation module

```
  module ad
    integer, parameter::wp=selected_real_kind(12)
```

```
!
!nmx is the maximum number of independent variables
integer, parameter::nmx=16
!
type dreal
  !r is array containing the value of variable r(0) and the
  !derivatives r(1:n) with respect to the independent variables
  real(wp)::r(0:nmx)
  !n is the number of independent variables
  integer::n
end type dreal

!Defines the multiplication to use the function mulss
!for the product of two dreals
interface operator (*)
  module procedure mulss
⋮
!Function for performing the multiplication of two dreals
function mulss(a,b) result(c)
  type(dreal), intent(in)::a, b
  type(dreal)::c
  !
  !Compute the result c%r(0) of multiplication
  c%r(0)=a%r(0)*b%r(0)
  !Compute the derivatives c%r(1:c%n) using the product rule
  c%r(1:c%n)=a%r(1:a%n)*b%r(0)+a%r(0)*b%r(1:b%n)
end function mulss

!Function for defining the values and independent variables
!in dreal array
function adindep(a) result(c)
  real(wp), intent(in)::a(:)
  type(dreal)::c(size(a))
  !
  !Copy the values
  c%r(0)=a
  !Set the number of independent variables
  c%n=size(a)
  !Initialize the derivatives with respect to the independent
  !variables. The derivative of the ith variable is one
  !with respect to the ith independent variables and all
  !other derivatives are zero
  do i=1,c%n
    c(i)%r(1:c%n)=0.d0
    c(i)%r(i)=1.d0
  end do
end function adindep
```

```
    !Function for returning the derivatives from dreal
    function adder(a) result(c)
      type(dreal), intent(in)::a
      real(wp)::c(a%n)
      !
      !Copy the derivatives from r(1:a%n)
      c=a%r(1:a%n)
    end function adder
  end module ad
```

References

[1] R.L. Bowers, J.R. Wilson, Numerical Modeling in Applied Physics and Astrophysics, Jones and Bartlett, Boston, MA, 1991.
[2] D.A. Knoll, W.J. Rider, G.L. Olson, An efficient nonlinear solution method for non-equilibrium radiation diffusion, J. Quant. Spectros. Radiat. Transfer 63 (1999) 15–29.
[3] W.J. Rider, D.A. Knoll, Time step size selection for radiation diffusion calculations, J. Comput. Phys. 152 (1999) 790–795.
[4] W.J. Rider, D.A. Knoll, G.L. Olson, A multigrid Newton–Krylov method for multimaterial equilibrium radiation diffusion, J. Comput. Phys. 152 (1999) 164–191.
[5] V.A. Mousseau, D.A. Knoll, W.J. Rider, Physics-based preconditioning and the Newton–Krylov method for non-equilibrium radiation diffusion, J. Comput. Phys. 160 (2000) 743–765.
[6] P.N. Brown, C.S. Woodward, Preconditioning strategies for fully implicit radiation diffusion with material-energy transfer, SIAM J. Sci. Comput. 23 (2001) 499–516.
[7] D.A. Knoll, W.J. Rider, G.L. Olson, Nonlinear convergence, accuracy, and time step control in nonequilibrium radiation diffusion, J. Quant. Spectros. Radiat. Transfer 70 (2001) 25–36.
[8] D.J. Mavriplis, An assesment of linear versus nonlinear multigrid methods for unstructured mesh solvers, J. Comput. Phys. 175 (2002) 302–325.
[9] K.S. Kang, P_1 nonconforming finite element multigrid method for radiation transport, SIAM J. Sci. Comput. 25 (2003) 369–384.
[10] L. Stals, Comparison of non-linear solvers for the solution of radiation transport equations, Electron. Trans. Numer. Anal. 15 (2003) 78–93.
[11] V.A. Mousseau, D.A. Knoll, New physics-based preconditioning of implicit methods for non-equilibrium radiation diffusion, J. Comput. Phys. 190 (2003) 42–51.
[12] M. Pernice, Performance of a Newton–Krylov–FAC method for equilibrium radiation diffusion on locally refined grids, Talk in the Eighth Copper Mountain Conference on Iterative Methods, Copper Mountain, CO (2004).
[13] C.C. Ober, J.N. Shadid, Studies on the accuracy of time-integration methods for the radiation-diffusion equations, J. Comput. Phys. 195 (2004) 743–772.
[14] R.B. Lowrie, A comparison of implicit time integration methods for nonlinear relaxation and diffusion, J. Comput. Phys. 196 (2004) 566–590.
[15] D.A. Knoll, D.E. Keyes, Jacobian-free Newton–Krylov methods: a survey of approaches and applications, J. Comput. Phys. 193 (2004) 357–397.
[16] Y. Saad, M.H. Schultz, GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM J. Sci. Statist. Comput. 7 (1986) 856–869.
[17] C.T. Kelley, Solving Nonlinear Equations with Newton's Method, Fundamentals of Algorithms, vol. 1, SIAM, Philadelphia, PA, 2003.
[18] W.L. Briggs, V.E. Henson, S.F. McCormick, A Multigrid Tutorial, second ed., SIAM, Philadelphia, PA, 2000.
[19] W. Hackbusch, Multigrid Methods and Applications, Springer Series in Computational Mathematics, vol. 4, Springer-Verlag, Berlin, 1985.
[20] P. Wesseling, An Introduction to Multigrid Methods, John Wiley & Sons Ltd, Chichester, 1992.
[21] J.E. Dendy Jr., Black box multigrid for systems, Appl. Math. Comput. 19 (1986) 57–74.
[22] A. Griewank, Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation, Frontiers in Applied Mathematics, vol. 19, SIAM, Philadelphia, PA, 2000.
[23] J. Haslinger, R.A.E. Mäkinen, Introduction to Shape Optimization, Advances in Design and Control, vol. 7, SIAM, Philadelphia, PA, 2003.

[24] E. Tijskens, D. Roose, H. Ramon, J. De Baerdemaeker, Automatic differentiation for solving nonlinear partial differential equations: an efficient operator overloading approach, Numer. Algorithms 30 (2002) 259–301.
[25] C. Bischof, A. Carle, P. Khademi, A. Mauer, ADIFOR 2.0: Automatic differentation of Fortran 77 programs, IEEE Computat. Sci. Eng. 3 (1996) 18–32.
[26] G.L. Olson, L.H. Auer, M.L. Hall, Diffusion, P_1, and other approximate forms of radiation transport, J. Quant. Spectros. Radiat. Transfer 64 (2000) 619–634.
[27] E. Hairer, G. Wanner, Solving Ordinary Differential Equations. II: Stiff and differential-algebraic problems, second ed., Springer Series in Computational Mathematics, vol. 14, Springer-Verlag, Berlin, 1996.
[28] J.R. Cash, Two new finite difference schemes for parabolic equations, SIAM J. Numer. Anal. 21 (1984) 433–446.
[29] R. Glowinski, Finite Element Methods for Incompressible Viscous Flow, Handbook of Numerical Analysis, vol. IX, North-Holland, Amsterdam, 2003.