JOURNAL OF COMPUTATIONAL PHYSICS 101, 360–367 (1992)

The Application of the Preconditioned Biconjugate Gradient Algorithm to NLTE Rate Matrix Equations

SUMANTH KAUSHIK AND PETER L. HAGELSTEIN

Research Laboratory of Electronics, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139

Received December 21, 1988; revised February 16, 1990

This paper reports the success of the preconditioned biconjugate gradient (PBCG) and the conjugate gradient square (CGS) algorithms in solving the matrix equations resulting from the discretization of systems of population rate equations which arise in nonequilibrium kinetics modeling. The success of the PBCG and CGS can be attributed to two main ideas: First, the singularity of the rate matrix resulting from population conservation requirement was removed through a reduction of matrix order so as to improve the condition number of the matrix. Second, an efficient preconditioner was found to reduce the reduce the eigenvalue spread of the rate matrix. The preconditioning matrix was selected on the basis of retaining the largest few rates in each column of the well conditioned rate matrix. This preconditioner, along with the reduced rate matrix, enabled the algorithms to converge very rapidly so as to make them an attractive alternative to standard direct methods.

1. INTRODUCTION

The subject of interest here is the solution of the ODE system

$$\frac{\partial \mathbf{u}}{\partial t} = \mathbf{R} \cdot \mathbf{u}, \tag{1}$$

where $\mathbf{u}$ is a column vector of length N and $\mathbf{R}$ is a rate matrix. A solution to this rate equation is required in many nonlocal thermal equilibria (NLTE) problems. In a typical NLTE application, the matrix $\mathbf{R}$ is composed of the detailed rates at which particles enter and leave a particular quantum state and the elements of the vector $\mathbf{u}$, u_i, represent the population density of state i.

Although $\mathbf{R}$ can be quite dense and randomly structured, it has an important property that holds true in many physical systems; namely, it conserves particle number. Conservation of particles means that $\sum_{i=1}^{N} u_i = N_0$, where N_0 is the total number of particles at $t = 0$. It is simple to show that this condition implies the following important relation:

$$\sum_{i=1}^{N} R_{ij} = 0. \tag{2}$$

An analytical solution of (1) is often intractable, particularly if $\mathbf{R}$ is nonlinear. In order to solve (1) one can either use an ODE package (for example, the GEAR package), or else for certain applications (such as reactive flow or transient line transfer problems) one might linearize $\mathbf{R}$ about $\mathbf{u}$ and solve the resulting linear problem with a simple one-step difference equation. For example, if a modified backward Euler differencing were used, the resulting linear system to be solved would be given by

$$(\mathbf{I} - \Delta t\, \mathbf{R}_n) \cdot \mathbf{u}_{n+1} = \mathbf{u}_n. \tag{3}$$

The subscript n denotes the timestep ($\mathbf{u}_n = \mathbf{u}(t_n)$). The quantity Δt is $t_{n+1} - t_n$ and the matrix $\mathbf{I}$ is the identity matrix. If the matrix $\mathbf{R}$ is nonlinear, then the use of $\mathbf{R}_n$ in (3) is perhaps the most convenient one, although clearly not the most accurate one.

Since $\mathbf{R}$ is in general time-dependent, one is faced with the task of solving the matrix equation (3) at every time step. This is a very significant computational task, particularly since the dimension of the matrix $\mathbf{R}$ can be very large ($N \sim 1000$). As an example, if one were to solve (3) using Gaussian elimination, roughly 3×10^8 multiplication and addition operations would be needed if $\mathbf{R}$ were full (an operation count of roughly $fN^3/3$ nonzero additions and multiplications occurs for these matrices with fill factor f). In addition to the large computational expense, Gaussian elimination also presents serious storage problems. In typical NLTE applications, $\mathbf{R}$ is somewhat sparse ($f = 1$–20%). However, the LU decomposition of $\mathbf{I} - \Delta t\, \mathbf{A}$ results in $\mathbf{L}$ and $\mathbf{U}$ matrices which are not sparse and require the storage of a full matrix. In many large vector processors, storage is an important issue, thereby making Gaussian elimination an unattractive prospect.

An alternative to solution by Gaussian elimination is to adopt some type of iterative method. Iterative methods are typically memory-efficient and may require in addition fewer cycles. Conjugate gradient (CG) methods have become important in many applications recently, although

0021-9991/92 $5.00

to our knowledge such methods have not previously been successfully applied to the linear system of interest here. If a class of efficient iterative methods could be developed for NLTE problems, then they would present a very attractive alternative to Gaussian elimination. In this paper we present one such iterative method that "beats" Gaussian elimination. In particular, the preconditioned biconjugate gradient (PBCG) algorithm is shown to be a much faster and more efficient alternative to Gaussian elimination in inverting the large matrices that arise in NLTE modelling.

This paper is organized in the following manner. First, the issues relevant to the CG method is briefly reviewed in the context of the present application. This leads to a discussion of the origin and consequences of numerical ill-conditioning as it pertains to the solution of rate equations; following which, a simple method is presented to avoid ill-conditioned systems. Once these important issues relating to the CG method have been resolved, the results of the present effort, in particular the PBCG and PCGS algorithms, are discussed.

2. CONJUGATE GRADIENT METHODS

During the last decade, the preconditioned conjugate gradient algorithm or variants thereof have become favored iterative methods to solve large sparse linear systems of the form

$$\mathbf{A} \cdot \mathbf{x} = \mathbf{b}. \tag{4}$$

In the present application, $\mathbf{A}$ is defined to be

$$\mathbf{A} \equiv \mathbf{I} - \Delta t\, \mathbf{R}. \tag{5}$$

Although in the strict sense the CG method is a direct method, it has become referred to as an iterative method [1, 3].

CG-type methods hitherto have not been used for the present application. There are difficulties associated with the application of conjugate gradient methods to systems which are both nonsymmetric and which have a large condition number. For example, the CG method and the biconjugate gradient method can be viewed as a algorithms which construct a polynomial fit to the inverse of the eigenvalue spectrum [4, 5]:

$$\frac{1}{\lambda_i} = \sum_{k=1}^{N} f_k (\lambda_i)^k. \tag{6}$$

A large spread in the eigenvalue spectrum correspondingly requires many terms in the polynomial expansion; hence, slower convergence rates. The spectral condition number measures this spread and is defined to be

$$\mathscr{C}(\mathbf{A}) \equiv \|\mathbf{A}\|_2 \, \|\mathbf{A}^{-1}\|_2 = \frac{\sigma_{\max}}{\sigma_{\min}},$$

where $\sigma_{\max}$ and $\sigma_{\min}$ are the largest and smallest singular values of the matrix. The largest and smallest singular values correspond to the largest and smallest nonzero eigenvalues of the matrix. Since microscopic processes occur at widely ranging time scales, rate matrices with large condition numbers are very common. In typical NLTE problems, $\mathscr{C}(\mathbf{A})$ can be as high as 10^{10}. For matrices with such a broad eigenvalue spectrum, the standard CG-type algorithms fail.

However, the above problems are not insurmountable. For example, condition number can be reduced by preconditioning $\mathbf{A}$ [6]. This involves the selection of a matrix $\mathbf{Q}$ such that

$$\mathbf{Q}^{-1} \cdot \mathbf{A} \approx \mathbf{I},$$

where $\mathbf{Q}$ is chosen so that its inverse is easily computable. This $\mathbf{Q}$ is then used to solve the preconditioned system

$$\mathbf{Q}^{-1} \cdot \mathbf{A} \cdot \mathbf{x} = \mathbf{Q}^{-1} \cdot \mathbf{b}. \tag{7}$$

This method is essentially what we have used for the NLTE problem.

An alternate route to developing an iterative method is to symmetrize the initial linear system (4), and then to precondition (or precondition and then symmetrize) as

$$\mathbf{A}^{\mathrm{T}} \cdot \mathbf{A} \cdot \mathbf{x} = \mathbf{A}^{\mathrm{T}} \cdot \mathbf{b}, \tag{8}$$

where $\mathbf{A}^{\mathrm{T}} \cdot \mathbf{A}$ is now a symmetric matrix. This approach involves squaring the condition number of the matrix; as a result, it is more difficult to precondition.

3. FINITE DIFFERENCING ILL-CONDITIONED SYSTEMS

3.1. *Condition Numbers*

In many NLTE applications (e.g. stellar plasmas), the time evolution of the population vector is not of primary interest; instead, the desired information in many of these applications is the steady state distribution. For these applications, it would be very useful to set $\Delta t = \infty$ in (3) and solve directly for the steady state.

Unfortunately, this is very difficult if the system is differenced as (3). The reason is rooted in the singularity of the rate matrix $\mathbf{A}$. The rate matrix is always singular if it

conserves population. This can easily be seen in (2), which makes one of the N rows dependent on the remaining $N-1$ rows. As a result of this singularity, CG methods cannot converge effectively. But it is important to realize that this singularity has a physical interpretation. The eigenvector corresponding to the zero eigenvalue corresponds to the steady state solution when $\partial \mathbf{u}/\partial t = 0$, $\mathbf{u} \neq 0$.

It may by argued, however, that even though **R** is singular, **A**, as a result of the identity matrix, is not singular. This is indeed true, but it is shown that the loss of singularity is due to the numerics. As a consequence, it is shown that the condition number is highly dependent on the parameters of the numerical scheme. In particular, it is seen that the matrix system (3) is singular if $\Delta t = \infty$. This behavior is clearly undesirable, since it implies that steady state solutions are not easily computable with the same differencing scheme. In the subsequent discussion we use the term singular value and eigenvalue interchangably since the matrix corresponding to the finite differenced system is no longer singular; hence, the singular value is simply the square root of the magnitude of the eigenvalue.

It should be noted that the above problems are not unique to the differencing scheme in (3). A standard practice by which the singularity is removed is to replace the last row by a conservation vector. Pictorially, this matrix looks like

$$\mathbf{B} = \begin{pmatrix} 1-\Delta t\, R_{1,1} & -\Delta t\, R_{1,2} & \cdots & \cdots & -\Delta t\, R_{1,n} \\ -\Delta t\, R_{2,1} & 1-\Delta t\, R_{2,2} & \cdots & \cdots & -\Delta t\, R_{2,n} \\ \vdots & \vdots & & \ddots & \vdots \\ \vdots & \vdots & \vdots & & -\Delta t\, R_{n-1,n} \\ 1 & 1 & 1 & \cdots & 1 \end{pmatrix}. \tag{9}$$

Although **A** and **B** appear to be very different from each another, they nevertheless have very similar eigenvalue spectrums. The eigenvalue spectrum of **A** is related to the eigenvalue spectrum of **R** by the simple relation $\lambda_i = 1 - \mu_i \Delta t$, where μ_i's are the eigenvalues of **R**. Since $\mu_{\min} = 0$ for **R**, the condition number for **A** is simply

$$\mathscr{C}(\mathbf{A}) = \frac{1-\mu_{\max}\Delta t}{1-\mu_{\min}\Delta t} = 1 - \mu_{\max}\Delta t, \tag{10}$$

where all eigenvalues μ_i have a negative real parts. This condition number has a disagreeable behaviour in the limit of large time steps; as $\Delta t \to \infty$, $\mathscr{C}(A) \to \infty$. In this sense, **A** is ill-conditioned.

The exact eigenvalue spectrum of **B** is quite difficult to calculate. Intuitively, it would seem that there should be some relationship between the eigenvalue spectrum of **B** and **A**. To understand this, several matrices of the form of **A** and **B** were generated and their respective condition numbers were examined. It was found that the condition numbers were virtually identical in the limit of large time steps. One possible reason for this behaviour is the following. As $\Delta t \to \infty$, the coupling between the top rows and the bottom row becomes very small. In the limit of total decoupling, only the diagonal element is important; hence, the lowest eigenvalue is $\lambda_{\min}(\mathbf{B}) = 1$. Similarly, the largest eigenvalues do not change appreciably from **A**. The physical process responsible for the large eigenvalues (for typical non-LTE plasmas) are collisions. Since, these processes are relatively insensitive to the changes to any single quantum state, the perturbation of the one the states (e.g. in (9), the lowest row) should not really affect these large eigenvalues. So based on these arguments, in the limit $\Delta t \to \infty$, it can be seen that $\mathscr{C}(\mathbf{B}) \approx \mathscr{C}(\mathbf{A})$.

In the limit that Δt becomes very large, the condition number of **A** and **B** approaches $-\mu_{\max}\Delta t$, which is linearly divergent in Δt. If the lowest eigenvalue $\lambda_{\min}$ (which corresponds to the steady state solution) could be removed in some way, then the resulting system would have a condition number

$$\mathscr{C}(\mathbf{A}') \to \frac{\mu_{\max}}{\mu'_{\min}} \tag{11}$$

in the limit of large Δt, where $\mathbf{A}'$ is the matrix corresponding to the modified system where $\lambda_{\min}$ has been removed. The eigenvalue $\mu'_{\min}$ is the eigenvalue with the smallest magnitude of **R** other than the zero eigenvalue corresponding to the steady state solution.

This condition number ($\mathscr{C}(\mathbf{A}')$) is due to the detailed physics and atomic rates which occur in the problem, and is not related to either the time step or differencing scheme. It is, in a sense, a "physical" condition number for the system. Although condition numbers smaller than this may occur, it can be hoped that a numerical scheme can be devised that would do no worse than this under any circumstances (the schemes described in this section can have condition numbers which are arbitrarily large). In the next section we discuss a scheme involving a minor modification to (3) which has this desirable property.

We may summarize the discussion of the section as follows: The matrix **R** is singular and the zero eigenvalue corresponds to the steady state solutions of the system. Although the numerical scheme (3) removes this singularity, the condition number of the discretized system is still much larger than the physical condition number. Although the ill-conditioning is not a serious issue for direct inversion, it becomes a serious issue for iterative methods since the convergence properties are directly related to the condition number.

3.2. *Reduction of Dimension*

From the previous section, it is clear that the goal is to somehow eliminate $\lambda_{\min} = 1$ eigenvalue. This can be accomplished by taking advantage of conservation. Since the system is conservative, only $N-1$ equations need to be solved explicitly at the $n+1$ th timestep. The population of the Nth level can be determined through

$$u_N = N_0 - \sum_{j=1}^{N-1} u_j. \tag{12}$$

The use of backward Euler differencing on the rate equations for the other $N-1$ levels leads to a set of linear equations with an associated matrix that is well conditioned for large time steps.

The differential equation describing the level population of state i can be written

$$\frac{\partial u_i}{\partial t} = \sum_{j=1}^{N} R_{ij} u_j = \sum_{j=1}^{N-1} R_{ij} u_j + R_{iN} u_N. \tag{13}$$

This population of level N can be eliminated by using the conservation condition to yield

$$\frac{\partial u_i}{\partial t} = \sum_{j=1}^{N-1} (R_{ij} - R_{iN}) u_j + R_{iN} N_0. \tag{14}$$

This implies the system of rate equations

$$\frac{\partial \mathbf{u}}{\partial \mathbf{t}} = \mathbf{R}' \cdot \mathbf{u} + \mathbf{f}, \tag{15}$$

where $\mathbf{R}'$ is a *nonsingular* $(N-1)\times(N-1)$ matrix. The elements of $\mathbf{R}'$ and $\mathbf{f}$ are $R'_{ij} = R_{ij} - R_{iN}$ and $f_i = R_{in} N_0$, respectively. Since the matrix $\mathbf{R}'$ is nonsingular, the steady state population can be found by solving $\mathbf{R}' \cdot \mathbf{u} = -\mathbf{f}$.

Finite differencing (15) yields the linear system

$$(\mathbf{I} - \Delta t\, \mathbf{R}'_n) \cdot \mathbf{u}_{n+1} = \mathbf{u}_n + \Delta t\, \mathbf{f}_n. \tag{16}$$

This equation resembles (3); however, the matrix

$$\mathbf{A}' = \mathbf{I} - \Delta t\, \mathbf{R}' \tag{17}$$

has a reduced condition number

$$\mathscr{C}(\mathbf{A}') = \frac{1 - \Delta t\, \mu_{\max}}{1 - \Delta t\, \mu'_{\min}} \tag{18}$$

following the discussion of the last section. In the best case ($\Delta t = 0$), $\mathscr{C}(\mathbf{A}') = 1$ and in the worst case ($\Delta t = \infty$), $\mathscr{C}(\mathbf{A}') = \mu_{\max}/\mu_{\min}$. One cannot do better than this because the condition number cannot be made smaller (without preconditiong) than the spread in eigenvalues due to the physical processes.

In this sense, the task has been accomplished; by reducing the dimension of $\mathbf{A}$ and solving the $\mathbf{A}'$ system, the condition number remains finite in the large Δt limit. For situations where $|\Delta t\, \mu_{\min}|$ is less than unity little improvement is expected. For $|\Delta t\, \mu_{\min}|$ larger than unity, the reduced system can have a substantially smaller condition number.

The method described in this section has the additional feature that it is algebraically conservative. Gaussian elimination of the system (3) is known to lead to moderate to large errors in total population, and the usual practice in NLTE calculations is to replace one of the rate equations by the conservation condition as described in the previous section. This modified system gives dramatic improvement in the conservation of population, as is also well known. The reduced system discussed in this section is operationally equivalent in that conservation is guaranteed algebraically on each timestep.

4. PRECONDITIONING

Although $\mathbf{A}'$ is not singular for large Δt, the condition number can be nevertheless be quite large. Under these conditions, it is necessary to precondition $\mathbf{A}'$ in order for a conjugate gradient method to be effective. To simplify notation, the prime on $\mathbf{A}'$ is dropped for the remainder of the paper.

As mentioned before, preconditioning involves the determination of an easily invertible matrix $\mathbf{Q}$, where $\mathscr{C}(\mathbf{Q}^{-1}\mathbf{A}) \ll \mathscr{C}(\mathbf{A})$. One class of $\mathbf{Q}$'s that are easily invertible are those that are of the form $\mathbf{Q} = \mathbf{LU}$, where $\mathbf{L}$ and $\mathbf{U}$ are the lower diagonal and upper diagonal matrices, respectively. A simple way to generate $\mathbf{L}$ and $\mathbf{U}$ is to use the incomplete Crout decomposition (ICD) [8]. ICD defines these matrices to be

$$\begin{aligned} L_{mm} &= 1 \qquad m = (1, \ldots, n) \\ U_{mj}|_{(m,j)\in\mathscr{P}} &= A_{mj} - \sum_{\substack{k<j \\ (m,k)\in\mathscr{P} \\ (k,j)\in\mathscr{P}}} L_{mk} U_{kj} \qquad j \geqslant m \\ L_{im}|_{(i,m)\in\mathscr{P}} &= \frac{1}{U_{mm}} \left[A_{im} - \sum_{\substack{k<i \\ (i,k)\in\mathscr{P} \\ (k,m)\in\mathscr{P}}} L_{mk} U_{km} \right] \qquad i \geqslant m. \end{aligned} \tag{19}$$

Here the summation is done over the sparsity pattern $\mathscr{P}$.

There are several possible patterns $\mathscr{P}$ that one could choose. A natural choice for $\mathscr{P}$ is to select the coordinates of the largest m off-diagonal terms in each column of $\mathbf{R}$. Physically, it means that for each level u_i, the m largest states to which it couples is retained in the sparsity pattern; the coupling strength is measured (crudely) by the off-diagonal

matrix element. The following is an illustration of the sparsity selection with $m = 2$:

$$\mathbf{A} = \begin{pmatrix} -2 & 3 & 4 & 2 & 1 \\ 1 & -2 & 3 & 4 & 2 \\ 2 & 4 & -2 & 1 & 3 \\ 3 & 2 & 1 & -2 & 4 \\ 4 & 1 & 2 & 3 & -2 \end{pmatrix} \qquad (20)$$

$$\mathscr{P} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 \end{pmatrix}.$$

Note that the regardless of the magnitude of A_{ii}, the diagonal is always included in the sparsity pattern.

The selection of the fastest rates *out* of a level in the sparsity pattern rather than the fastest rates *into* a level was purely an arbitrary decision. It may appear at first glance that the preconditioner could be improved if both the rates were included in the sparsity pattern. This, however, did not proved to be the case. A preconditioner matrix was generated by selecting *all* the nonzero elements of **A** to be in the sparsity pattern. It was found that the convergence rates did not change as a result of the better preconditioner. As a result, it was concluded that inclusion of both the rates into the sparsity pattern would probably not improve the conditioning. Furthermore, due to the pecularity of the data structure used for efficient storage and manipulation of sparse matrices, it is very cumbersome to implement a scheme in which both the fastest rates into and out of a level are included for preconditioning. For these reasons, only one set of rates were included in the sparsity pattern.

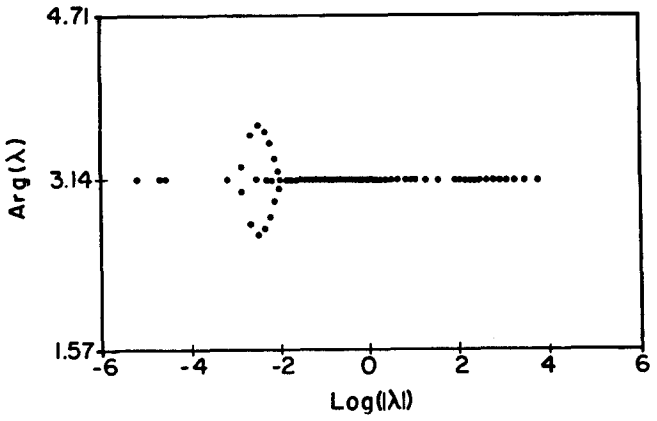

FIG. 1. Location of eigenvalues for unpreconditioned matrix **R** in the complex plane.

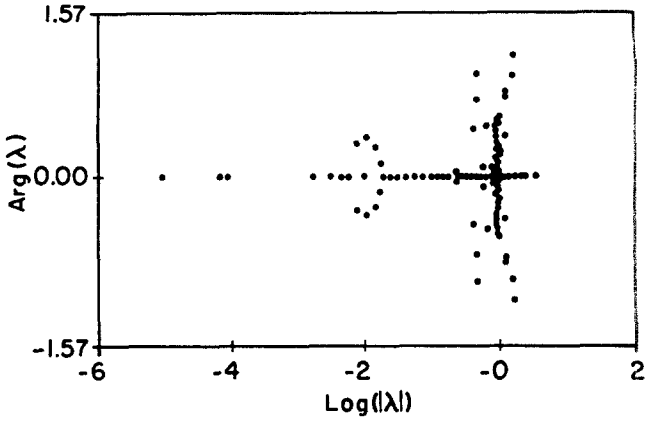

FIG. 2. Location of eigenvalues for preconditioned matrix $\mathbf{Q}^{-1}\mathbf{R}$ in the complex plane.

In order to test the efficacy of this preconditioning method, a typical matrix encountered in NLTE problems was generated (for a nickel-like Mo model, similar to that used in Ref. [9]). The generated test **R** was a 258×258 matrix with a fill factor of 19%. It had an unpreconditioned condition number $\mathscr{C}(\mathbf{R}') \approx 10^8$, where $\mathbf{R}'$ is the rate matrix with the singularity removed according to the above prescription. By generating the **Q** using the ICD described above with $m = 7$, the condition number of the preconditioned matrix dropped to $\mathscr{C}(\mathbf{Q}^{-1}\mathbf{R}') = 10^5$. The eigenvalue plot is given in Figs. 1 and 2. Although the condition number does not fall dramatically, it can be seen from Fig. 2 that the eigenvalues are clustered strongly about $1 + 0i$ [10].

5. SCG AND PBCG METHODS

As discussed previously, the rate matrices are inherently asymmetric due to the details of the atomic processes. Symmetrization can be easily accomplished by solving the system in (8). However, it should be noted that $\mathscr{C}(\mathbf{A}^T\mathbf{A}) = |\mathscr{C}(\mathbf{A})|^2$. This squaring of the condition number often makes SCG (symmetrized conjugate gradient method) unattractive. In the present application, the SCG has no hope of converging if **A** is unpreconditioned.

A version of the preconditioned SCG algorithm was implemented and results were disappointing. The algorithm did not converge on the test problem described in Section 4. This is not entirely surprising considering that the condition number of the symmetrized preconditioned system is in the vicinity of 10^{10}.

As a result of the failure with SCG-type algorithms, attention had to turn elsewhere. Two algorithms that have shown much success in solving asymmetric systems are the biconjugate gradient algorithm (BCG) and the conjugate gradient squared algorithms (CGS). The results of these algorithms are now discussed.

5.1. *Preconditioned BCG Algorithm*

Unlike SCG, BCG, does not square the condition number as SCG; as a result, faster convergence can be expected. The BCG algorithm requires basis vectors in both the **A**-space and its adjoint space $\mathbf{A}^{\mathrm{T}}$. Details of this algorithm can be found in the literature [10, 11].

The preconditioned BCG algorithm is given by

$$\begin{aligned}
\alpha'_k &= \frac{[(\mathbf{U}^{\mathrm{T}})^{-1}\cdot\mathbf{s}_k]^{\mathrm{T}}\cdot[\mathbf{L}^{-1}\cdot\mathbf{r}_k]}{\mathbf{q}_k^{\mathrm{T}}\mathbf{A}\cdot\mathbf{p}_k}\\
\mathbf{x}_{k+1} &= \mathbf{x}_k + \alpha'_k\mathbf{p}_k\\
\mathbf{r}_{k+1} &= \mathbf{r}_k - \alpha'_k\mathbf{A}\cdot\mathbf{p}_k\\
\mathbf{s}_{k+1} &= \mathbf{s}_k - \alpha'_k\mathbf{A}^{\mathrm{T}}\cdot\mathbf{q}_k \qquad (21)\\
\beta'_k &= -\frac{[(\mathbf{U}^{\mathrm{T}})^{-1}\cdot\mathbf{s}_{k+1}]^{\mathrm{T}}\cdot[\mathbf{L}^{-1}\cdot\mathbf{r}_{k+1}]}{[(\mathbf{U}^{\mathrm{T}})^{-1}\cdot\mathbf{s}_k]^{\mathrm{T}}\cdot[\mathbf{L}^{-1}\cdot\mathbf{r}_k]}\\
\mathbf{p}_{k+1} &= \mathbf{U}^{-1}\cdot\mathbf{L}^{-1}\cdot\mathbf{r}_{k+1} - \beta'_k\mathbf{p}_k\\
\mathbf{q}_{k+1} &= (\mathbf{L}^{\mathrm{T}})^{-1}\cdot(\mathbf{U}^{\mathrm{T}})^{-1}\cdot\mathbf{s}_{k+1} - \beta'_k\mathbf{q}_k
\end{aligned}$$

with the initial conditions $\mathbf{r}_0 = \mathbf{s}_0 = \mathbf{p}_0 = \mathbf{q}_0 = \mathbf{b} - \mathbf{A}\cdot\mathbf{x}_0$, and where $\mathbf{x}_0$ is the initial guess vector. This algorithm implements the biconjugate gradient method on the system

$$\mathbf{L}^{-1}\cdot\mathbf{A}\cdot\mathbf{U}^{-1}\cdot(\mathbf{U}\cdot\mathbf{x}) = \mathbf{L}^{-1}\cdot\mathbf{b}, \qquad (22)$$

which is ultimately equivalent to (7).

This algorithm was tried on our test matrix. As it can be seen in Fig. 3, the results were quite successful. There are two curves shown in Fig. 3. The top curve shows the iteration plot for an initial state that is away from equilibrium: ($\|\mathbf{x}^{t=0} - \mathbf{x}^*\|_2 = 7.0$); the bottom curve shows for an initial state close to equilibrium $\|\mathbf{x}^{t=0} - \mathbf{x}^*\|_2 = 9.6\times10^{-1}$. The algorithm converged in about 30 iterations for the former and in about 8 in the latter. In transient NLTE calculations,

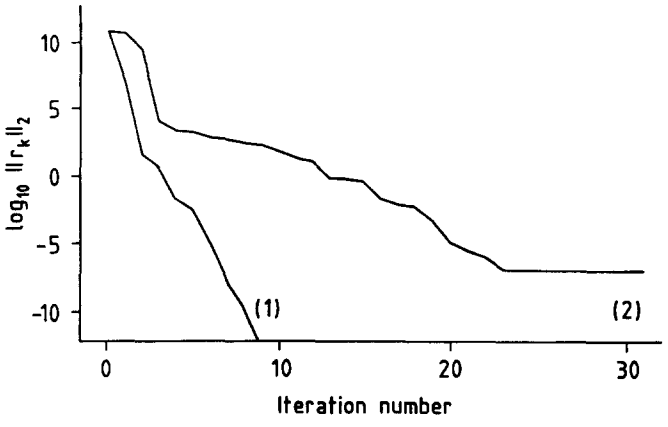

FIG. 3. $\|r_k\|_2$ vs iteration number for PBCG algorithm: (1) PBCG near equilibrium; (2) PBCG away from equilibrium.

the initial guess at timestep $n+1$ is the solution at timestep n, and this guess tends to be closer to the final solution at timestep $n+1$ on average than our "good initial guess" example here.

The success of the PBCG over the Gaussian elimination must be compared in terms of both memory used and operation count. In order to efficiently use memory, the nonzero matrix elements of the matrices **A**, **L**, and **U** were all stored as lists and the coordinates corresponding the nonzero matrix elements were stored in a integer array. The memory required by the PBCG algorithm as implemented requires

$$M_{\mathrm{PBCG}} = fN^2 + 2mN + \frac{fN^2}{4} + \frac{mN}{2} + 11N,$$

where fN^2 is for **A** stored as a list; where $2mN$ is for the **L** and the **U** stored as a list; $11N$ is for the vectors and the remaining for the integer index vectors need for the list. As before, f denotes the fill factor. For the test matrix above ($f = 19\%$, $m = 7$, $N = 258$), the savings in memory is 70%.

To compare speed, the number of floating point multiplications and additions required for Gaussian elimination is roughly

$$N_{\mathrm{Gauss}} \approx \frac{fN^3}{3} + \frac{N^2}{2},$$

under the assumption that the coding is such that all "zero" multiplies and additions are avoided. This optimistic limit is not usually obtained in hand-coded Gaussian elimination routines, however, it could in principle be obtained on some future computer whose architecture permitted fast concurrent zero checks. The PBCG iterations requires roughly

$$N_{\mathrm{PBCG}} \approx k(2fN^2 + 2mN + 8N)$$

additions and multiplications under similar assumptions. There is some time required for setup, but we have found this to be rather small in comparison with the time spent on the iterations. The ratio of operations for the two methods is approximately

$$\frac{N_{\mathrm{PBCG}}}{N_{\mathrm{Gauss}}} \approx \frac{6k}{N}$$

for a matrix with a moderately large fill factor. In this case the full matrix multiples of the PBCG method dominate the computation time. For the examples considered above, the ratios are 0.7 and 0.2 for the poor and good initial guesses, respectively. This estimate is a rather ideal operation count

estimate, and in our experience so far, the observed ratio of run times tends to favor the PBCG by larger factors.

5.2. *Preconditioned CGS Algorithm*

The CGS algorithm is very similar to the BCG algorithm; however, its convergence rate can be expected to be nearly twice as that of the BCG algorithm [12]. The residual error at each iteration k for the BCG can be written in the form

$$\mathbf{r}_k = \phi_k(\mathbf{A})\,\mathbf{r}_0,$$

where $\phi_k(\mathbf{A})$ is a polynomial of order k. This polynomial is generated by the BCG algorithm. The CGS algorithm generates, not $\phi_k(\mathbf{A})$, but rather, its *square* $\phi_k^2(\mathbf{A})$; hence, the convergence rate is twice as fast. The preconditioned version of the algorithm is given as

$$\begin{aligned}
\mathbf{h}_{k+1} &= \mathbf{e}_k - \alpha_k \mathbf{y}_k \\
\mathbf{x}_{k+1} &= \mathbf{x}_k + \alpha_k(\mathbf{e}_k + \mathbf{h}_k) \\
\mathbf{r}_{k+1} &= \mathbf{r}_k - \alpha_k \mathbf{A}\cdot(\mathbf{e}_k + \mathbf{h}_k) \\
\beta_k &= \frac{\mathbf{r}_0^{\mathrm{T}}\cdot\mathbf{r}_{k+1}}{\mathbf{r}_0^{\mathrm{T}}\cdot\mathbf{r}_k} \\
\mathbf{e}_{k+1} &= \mathbf{U}^{-1}\cdot\mathbf{L}^{-1}\cdot\mathbf{r}_{k+1} + \beta_k\mathbf{h}_{k+1} \\
\mathbf{p}_{k+1} &= \mathbf{e}_{k+1} + \beta_k(\mathbf{h}_{k+1} + \beta_k\mathbf{p}_k) \\
\mathbf{y}_{k+1} &= \mathbf{A}\cdot\mathbf{p}_{k+1} \\
\mathbf{w}_{k+1} &= \mathbf{U}^{-1}\cdot\mathbf{L}^{-1}\cdot\mathbf{y}_{k+1} \\
\alpha_{k+1} &= \frac{\mathbf{r}_0^{\mathrm{T}}\cdot\mathbf{r}_{k+1}}{\mathbf{r}_0^{\mathrm{T}}\cdot\mathbf{y}_{k+1}},
\end{aligned} \tag{23}$$

where $\mathbf{r}_0 \equiv \mathbf{b} - \mathbf{A}\cdot\mathbf{x}_0$. Sonneveld found CGS to be as much as 40% faster than BCG for inverting matrices arising from discretization of diffusion type equations. In many of the test cases, Sonneveld found CGS to not only be faster, but more accurate as well.

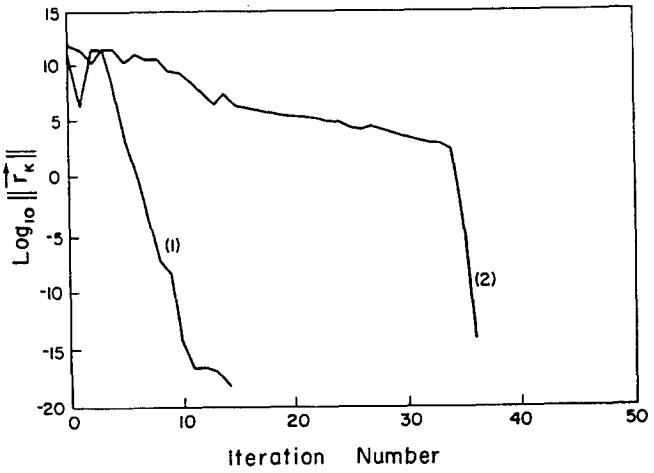

FIG. 4. $\|r_k\|_2$ vs iteration number for PCGS algorithm: (1) PCGS near equilibrium; (2) PCGS away from equilibrium.

Figure 4 shows the iteration plot for PCGS for the present application. Comparing this with Fig. 3, it is apparent that the PCGS algorithms converges 20% faster than the PBCG algorithm. Though the speedup is not as large as that achieved by Sonneveld [12], it is nevertheless a significant improvement. In terms of operation counts, the PCGS algorithms requires only a trivial number of extra operations (n more operations with n being the dimension of the matrix); therefore, the 20% speedup in convergence almost directly translates to a 20% computational speedup.

6. SUMMARY AND CONCLUSIONS

The preconditioned biconjugate gradient and its variant the preconditioned conjugate gradient squared algorithms have been applied to the linear system arising from a simple backward Euler differencing of the rate equations encountered in NLTE simulations, and the iterations have been found to be rapidly convergent.

The matrix arising from the discretization of the rate equations is singular, corresponding to the existence of a steady state solution. This singularity was removed through a reduction of the order of the system by one, and the determination of the population of the extra level through enforcement of conservation.

A preconditioning matrix was selected which is simply an incomplete LU decomposition of a sparse version of the reduced matrix. The time required to decompose the preconditioning matrix can be minimized by keeping only a small number of matrix elements. We have found that a suitable preconditioning matrix can be constructed by keeping only the largest matrix elements in each row.

A preconditioner based on this principle has been implemented and has been found to be successful in reducing the condition number of the original rate matrix. The resulting PBCG and PCGS algorithm has been found to be rapidly convergent and is relatively efficient, and provides a competitive alternative to Gaussian elimination in solving rate equations.

Based on our success with the PBCG and PCGS algorithms, we are presently investigating modifications and variations of the above algorithms to enhance the computational efficiency. The results of the research effort will be presented in forthcoming publications.

ACKNOWLEDGMENTS

The authors acknowledge support for this work from the Lawrence Livermore National Laboratory under Contract BO48704. The authors thank the reviewers for their perceptive and useful comments. In particular, we thank one of the reviewers for suggesting the use of the CGS algorithm.

REFERENCES

1. J. K. Reid, in *Proceedings of the Conference on Large Sparse Systems of Linear Equations*, edited by J. K. Reid (Academic Press, New York, 1971).
2. D. Kershaw, *J. Comput. Phys.* **26**, 43 (1978).
3. M. R. Hestenes and E. Stieffel, *J. Res. Nat. Bur. Stand. (US)* **49**, 409 (1952).
4. David G. Luenberger, *Linear and Nonlinear Programming* (Addison–Wesley, MA, 1984).
5. O. Axelsson, in *Sparse Matrix Techniques*, edited by V. A. Barker, Copenhagen 1976, Lecture Notes in Mathematics, Vol. 571 (Springer-Verlag, Berlin, 1977).
6. J. A. Meijerink and H. A. van der Vorst, *Math. Comput.* **31**, 148 (1977).
7. Ben Noble, *Applied Linear Algebra* (Prentice–Hall, New Jersey, 1969).
8. H. A. van der Vorst, *J. Comput. Phys.* **44**, 1 (1981).
9. P. Hagelstein, "Something New, Something Old," Proceedings of the OSA conference on Short Wavelength Coherent Radiation, Cape Cod, September 1988.
10. Z. Mikic and E. C. Morse, *J. Comput. Phys.* **61**, 154 (1985).
11. T. K. Sarkar, *J. Electromagn. Waves Appl.* **1**, 343 (1987).
12. Peter Sonneveld, *SIAM J. Sci. Stat. Comput.* **10**, 36 (1989).