Appl. Phys. B 50, 303–306 (1990)

Applied Physics B Photophysics and Laser Chemistry

The Application of Conjugate Gradient Methods to NLTE Rate Matrix Equations

Sumanth Kaushik and Peter L. Hagelstein

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

Received 3 October 1989/Accepted 3 November 1989

Abstract. This paper brings to attention recent developments in algorithms to efficiently solve linear systems resulting from the discretization of systems of population rate equations which arise in X-ray laser kinetics modeling. Specifically, we report the success of two variants of the preconditioned gradient method, namely the preconditioned conjugate gradient square (PCGS) algorithm and the method of generalized conjugate residuals (GCR) in solving non-LTE rate equations.

PACS: 02.70.+d, 52.25 Dg.

The purpose of this paper is to bring to attention some recent work by the authors [1, 2] concerning the fast and efficient solution of non-local thermal equilibria (NLTE) rate equations. In non-equilibrium plasma kinetics calculations, a solution to NLTE rate equations often is required to determine the ionization balance and the fractional occupation of the atomic levels. These rate equations can be cast in the form

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

where $\mathbf{u}$ is a column vector of length N and $\mathbf{R}$ is a rate matrix. 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; the elements of the vector $\mathbf{u}, u_i$, represent the population density of state i. The rate matrix $\mathbf{R}$ is a function of both the population density and time.

A numerical solution to this rate equation typically requires a linearization and discretization of the equation (1). For example, if a modified backward Euler differencing were used, the resulting linear system to be solved would be given by

$$(\mathbf{I} - \Delta t \mathbf{R}) \cdot \mathbf{u}_{n+1} = \mathbf{u}_n. \tag{2}$$

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 and $\mathbf{R}$ is an approximation to $\mathbf{R}(\mathbf{u}, t)$ at t_{n+1}. Equation (2) can be cast into the matrix inversion problem of the form

$$\mathbf{A} \cdot \mathbf{x} = \mathbf{b},$$

where $\mathbf{A} \equiv (\mathbf{I} - \Delta t \mathbf{R})$, $\mathbf{x} = \mathbf{u}_{n+1}$, and $\mathbf{b} = \mathbf{u}_n$.

The above linear system can be solved by direct methods such as Gaussian elimination. Recent work by the authors has shown that iterative methods can provide a faster and more efficient alternative to standard Gaussian elimination for inverting NLTE rate matrices. This paper describes the use of two different iterative algorithms based on the preconditioned conjugate gradient method, namely the preconditioned conjugate gradient squared (PCGS) algorithm and the method of generalized conjugate residuals (GCR). The purpose of this work is to review the methods briefly in order to bring them to the attention of X-ray researchers.

The paper will be organized as follows. First, in Sect. 1, we shall present the algorithm. In presenting the algorithm, we will first show how the matrix equation (2) has to be modified in order for the iterative methods to converge rapidly. The modification will result in the solution of a new matrix equation, different from (2). We shall then present the PCGS and GCR algorithm to solve the new linear system. In Sect. 2, we will discuss the results of our numerical experiments and compare them with standard direct methods such as Gaussian elimination.

1. Algorithm

1.1. Reduction of Dimension and Preconditioning

As it stands, Eq. (2) is not in a form amenable for the either of the two conjugate gradient based methods because **A** is ill-conditioned. This ill-conditioning arises due to the conservative nature of the matrix **R** $\left(\text{i.e. } \sum_{i=1}^{N} R_{ij}=0\right)$ which makes **R** singular. Therefore, the first step in solving (2) is to remove this singularity.

The singularity can be removed in **R** (thereby improving the conditioning of **A**) by expressing the population of the last level N in terms of the levels $N-1$ levels $\left(\text{i.e. } u_N=N_0-\sum_{i=1}^{N-1} u_i\right)$. This leads to a new form of (1).

$$\frac{\partial \mathbf{u}}{\partial t}=\mathbf{R}'\cdot\mathbf{u}+\mathbf{f}, \tag{3}$$

where $\mathbf{R}'$ is a *non-singular* $(N-1)\times(N-1)$ 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 new matrix $\mathbf{R}'$ is nonsingular, the steady state population now can be found by solving $\mathbf{R}'\cdot\mathbf{u}=-\mathbf{f}$.

Equation (3) can be finite differenced to yield

$$(\mathbf{I}-\Delta t\mathbf{R}'_n)\cdot\mathbf{u}_{n+1}=\mathbf{u}_n+\Delta t\mathbf{f}_n. \tag{4}$$

This equation resembles (2). The problem again reduces to the form $\mathbf{A}\cdot\mathbf{x}=\mathbf{b}$ but now, $\mathbf{A}=\mathbf{I}-\Delta t\mathbf{R}'$ and $\mathbf{b}=\mathbf{u}_n+\Delta t\mathbf{f}_n$.

1.2. Preconditioning

In order to further improve the condition number of the matrix **A**, the matrix is preconditioned [3, 4, 7]. Preconditioning involves the selection of a matrix **Q** such that

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

This **Q** is then used to solve the preconditioned system

$$\mathbf{Q}^{-1}\cdot\mathbf{A}\cdot\mathbf{x}=\mathbf{Q}^{-1}\cdot\mathbf{b}. \tag{5}$$

The matrix $\mathbf{Q}^{-1}$ is selected such the computation of $\mathbf{Q}^{-1}$ is a computationally cheap task. One simple way to generate a $\mathbf{Q}^{-1}$ is to have $\mathbf{Q}^{-1}=\mathbf{U}_{\text{inc}}^{-1}\cdot\mathbf{L}_{\text{inc}}^{-1}$ where $\mathbf{L}_{\text{inc}}$ and $\mathbf{U}_{\text{inc}}$ are strictly lower and upper diagonal matrices respectively. One way to generate the matrices $\mathbf{L}_{\text{inc}}$ and $\mathbf{U}_{\text{inc}}$ is through the incomplete Crout decomposition [8].

Typically, in an incomplete decomposition, $(L_{\text{inc}})_{ij}=(U_{\text{inc}})_{ij}=0$ if $(i,j)\notin\mathscr{P}$ where $\mathscr{P}$ is the sparsity pattern. One simple way to assign a sparsity pattern is to have the same as that of **A**. However, we have found that this is not necessary. In fact, only a few elements of **A** need to be retained in the sparsity pattern. These few elements that are retained are the m largest (absolute value) off-diagonal terms in each column of **R**. Here, m is an adjustable parameter. We have found that for a 258×258 matrix which may have a fill factor of 20%, a good value for m is $m\sim10$ which implies that $\mathbf{L}_{\text{inc}}$ and $\mathbf{U}_{\text{inc}}$ have a fill factor of only 4%. Physically, the above prescription 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.

1.3. Preconditioned CGS Algorithm

With the singularity removed and the matrix preconditioned, it is now possible to solve the linear system using conjugate gradient methods. There are numbers of good references to the standard conjugate gradient algorithm (e.g. see [5, 6]). The conjugate gradient algorithm improves the current estimate of $\mathbf{x}_k$ by adding a correction vector $\alpha_k\mathbf{p}_k$. The algorithm generates optimal direction vectors $\mathbf{p}_{k+1}$ and the coefficient α by minimizing the error from iteration to iteration.

However, the standard conjugate gradient method is not appropriate for this problem because it is only suited for solving symmetric matrices, and **R** is not symmetric. This is the motivation for considering the PCGS algorithm because the CGS algorithm is known be a robust method for solving asymmetric matrix equations [11]. The preconditioned algorithm is given as

$$\begin{aligned}
&\mathbf{h}_{k+1}=\mathbf{e}_k-\alpha_k\mathbf{y}_k,\\
&\mathbf{x}_{k+1}=\mathbf{x}_k+\alpha_k(\mathbf{e}_k+\mathbf{h}_k),\\
&\mathbf{r}_{k+1}=\mathbf{r}_k-\alpha_k\mathbf{A}\cdot(\mathbf{e}_k+\mathbf{h}_k),\\
&\beta_k=\frac{\mathbf{r}_0^T\cdot\mathbf{r}_{k+1}}{\mathbf{r}_0^T\cdot\mathbf{r}_k},\\
&\mathbf{e}_{k+1}=\mathbf{U}^{-1}\cdot\mathbf{L}^{-1}\cdot\mathbf{r}_{k+1}+\beta_k\mathbf{h}_{k+1},\\
&\mathbf{p}_{k+1}=\mathbf{e}_{k+1}+\beta_k(\mathbf{h}_{k+1}+\beta_k\mathbf{p}_k),\\
&\mathbf{y}_{k+1}=\mathbf{A}\cdot\mathbf{p}_{k+1},\\
&\mathbf{w}_{k+1}=\mathbf{U}^{-1}\cdot\mathbf{L}^{-1}\cdot\mathbf{y}_{k+1},\\
&\alpha_{k+1}=\frac{\mathbf{r}_0^T\cdot\mathbf{r}_{k+1}}{\mathbf{r}_0^T\cdot\mathbf{y}_{k+1}},
\end{aligned} \tag{6}$$

where $\mathbf{r}_0\equiv\mathbf{b}-\mathbf{A}\cdot\mathbf{x}_0$, $\mathbf{e}_0=0$, and $\mathbf{y}_0=0$.

However, the one drawback with this algorithm is that it requires two full matrix-vector multiplications. Matrix-vector multiplications are the largest fraction of the computational task; hence, it would make sense that an algorithm that requires only one matrix

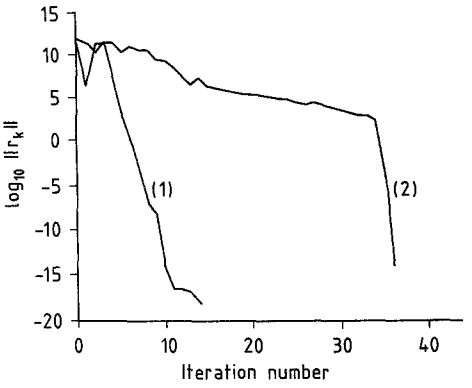

Fig. 1. $\|\mathbf{r}_k\|_2$ vs. iteration no. for PCGS algorithm. (1) PCGS near equilibrium (2) PCGS away from equilibrium

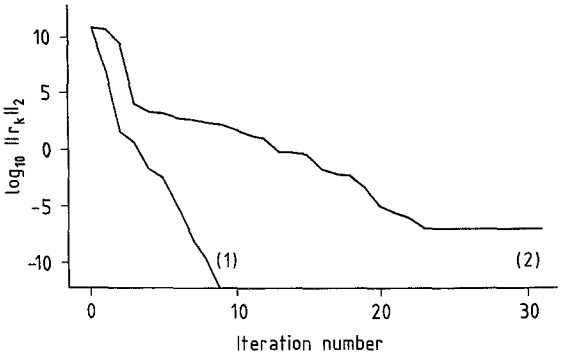

Fig. 2. $\|\mathbf{r}_k\|_2$ vs. iteration no. for GCR (1) GCR near equilibrium (2) GCR away from equilibrium

multiplication and can converge with similar rates is far preferable. This is the motivation for examining the second algorithm, the method of generalized conjugate residuals.

1.4. Generalized Conjugate Residuals

The method of generalized conjugate residuals (GCR) is another popular method to solve asymmetric matrix equations [12]. The GCR algorithm, like the CG method, chooses the optimal α to minimize the error; however, GCR does not choose the optimal direction. By relaxing the optimality constraint on the directions, the GCR method has to do less work per iteration. As mentioned earlier, the PCGS algorithm requires two full matrix vector multiplications, whereas GCR requires only one full matrix vector multiply. Thus, it can be nearly twice as fast if the convergence rate is the same. Reviews of GCR can be found in the literature [12].

The preconditioned version of the algorithm which we have proposed is:

$$
\begin{aligned}
\alpha_0 &= \frac{\mathbf{r}_0\ \mathbf{U}^{-1}\cdot\mathbf{L}^{-1}\cdot\mathbf{r}_0}{\mathbf{p}_0\cdot\mathbf{A}\cdot\mathbf{p}_0} \qquad k=0,\\
\alpha_k &= \frac{\mathbf{p}_k^T\cdot[\mathbf{A}(\mathbf{LU})^{-1}]^T\cdot\mathbf{r}_k}{\mathbf{y}_k^T\cdot\mathbf{y}_k} \quad \text{for} \quad k=(1,2,\ldots,k),\\
\mathbf{x}_{k+1} &= \mathbf{x}_k+\alpha_k\mathbf{p}_k,\\
\mathbf{r}_{k+1} &= \mathbf{r}_k-\alpha_k\mathbf{y}_k,\\
\mathbf{s}_{k+1} &= \mathbf{A}\cdot\mathbf{r}_{k+1},\\
\beta_{k,l} &= \frac{(\mathbf{A}\cdot\mathbf{p}_l)^T\cdot\mathbf{s}_{k+1}}{\mathbf{y}_l^T\cdot\mathbf{y}_l} \quad \text{for} \quad l=(0,1,\ldots,k),\\
\mathbf{p}_{k+1} &= \mathbf{r}_{k+1}-\sum_{l=0}^{l=k}\beta_{k,l}\mathbf{p}_l,\\
\mathbf{y}_{k+1} &= \mathbf{A}\cdot\mathbf{p}_{k+1}=\mathbf{s}_{k+1}-\sum_{l=0}^{l=k}\beta_{k,l}\mathbf{y}_l.
\end{aligned}
\tag{7}
$$

The algorithm above is a modification of the algorithm in the literature [12]. The definition of α has been modified for the very first iteration. It was found that the algorithm works much better with this modification. The motivation for this has been discussed in length in [2] and is beyond the scope of this paper.

2. Results

To test the algorithm, a rate matrix was constructed based on a model for a nickel-like Mo plasma (similar to the one used in [9]). The rate matrix was a 258×258 matrix having a fill factor of 19%.

Our algorithm was tested on this matrix. As it can be seen in Fig. 1, the algorithm was quite successful. There are two curves shown in Fig. 1: the top curve shows the iteration plot for an initial state that is away from equilibrium: ($\|\mathbf{x}^{t=0}-\mathbf{x}^{t=\infty}\|_2=7.0$); the bottom curve shows for an initial state close to equilibrium $\|\mathbf{x}^{t=0}-\mathbf{x}^{t=\infty}\|_2=9.6\times10^{-1}$. The algorithm converged in about thirty iterations for the former and in about eight in the latter. The slower rate observed in the former can be readily explained; however, the details are somewhat technical and can be found in [2].

Figure 2 plots the result of the GCR. As it can be seen the algorithm is converged roughly 30% faster. Furthermore, since the algorithm multiplies only one full matrix, it is roughly a factor of two faster per iteration step.

However, the GCR algorithms is somewhat fragile in comparison with the PCGS algorithm which is robust for all initial conditions. Though GCR has faster convergence than PCGS for the present problem, the algorithm tends to break down when the system is very far from steady state. If the number of iterations required for convergence is moderately large (as is the case when the system is very far from equilibrium), GCR algorithm does not converge well. In Fig. 2 it can

be seen that the solutions computed by the GCR algorithm does not converge to the same level of accuracy as that by the PCGS algorithm (Fig. 1) for the second case.

The reason for this is as follows. At each iteration, the GCR algorithm involves a Grahm-Schmidt type orthogonalization with respect to the correction vectors selected in the previous steps (see the algorithm above). As the number of iterations increases, the number of required orthogonalizations also increases which in turn makes the algorithm very susceptible to numerical roundoff errors.

For applications where the system is very far away from equilibrium, the CGS method seems to be the algorithm of choice. It is robust for all initial conditions. As the system approaches steady state, however, GCR can be used with an expected 30% speedup.

3. Computational Costs

It would be useful at this point to compare the computational work (operation counts) of the above two algorithms with Gaussian elimination. The number of operations required for Gaussian elimination is roughly

$$N_{\text{Gauss}} \approx f\frac{N^2}{3} + \frac{N^2}{2}. \tag{8}$$

For simplicity, we have assumed that multiplications and additions take roughly the same time to execute.

The number of operations required for the PCGS algorithms are

$$N_{\text{PCGS}} \approx (9N + 4fN^2 + 4mN^2)k. \tag{9}$$

Here, m is the number of non-zero elements in each column of the matrix $\mathbf{L}+\mathbf{U}$ (see [1]) and k is the number of iterations required for convergence. The number of operations required for GCR is

$$N_{\text{GCR}} \approx (4N + 2fN^2 + 2mN^2)k + 2Nk(k+1). \tag{10}$$

For the test problem above ($f=19\%$, $m=7$, $N=258$, $k=9$), GCR is three times faster than PCGS and over five times faster than Gaussian elimination.

4. Conclusion

This paper has reviewed two CG based iterative methods, the preconditioned conjugate square and the method of generalized conjugate residuals, as an efficient alternative to standard Gaussian elimination for inverting rate matrices in NLTE problems. Of the two, the GCR algorithm is much faster; however, it is not as robust as the PCGS algorithm and must not be used when the system is far away from steady state.

Acknowledgement. The authors would like to acknowledge support for the work from the Lawrence Livermore National Laboratory under the contract number B048704.

References

1. S. Kaushik, P. Hagelstein: Accepted for publication, J. Comp. Phys.
2. S. Kaushik, P. Hagelstein: Submitted for publication, J. Comp. Phys.
3. J.K. Reid: *Proceedings of the Conference on Large Sparse Systems of Linear Equations*, ed. by J.K. Reid (Academic, New York 1971)
4. D. Kershaw: J. Comput. Phys. **26**, 43–65 (1978)
5. M.R. Hestens, E. Stieffel: J. Res. Natl. Bur. Stand. **49**, 409 (1952)
6. D.G. Luenberger: *Linear and Nonlinear Programming* (Addison-Wesley, MA 1984)
7. J.A. Meijerink, H.A. van der Vorst: Math. Comput. **31**, 148 (1977)
8. H.A. van der Vorst: J. Comp. Phys. **44**, 1–19 (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, E.C. Morse: J. Comp. Phys. **61**, 154–185 (1985)
11. P. Sonneveld: SIAM J. on Sci. Stat. Comput. **10**, 36
12. H. Elman: *Iterative Methods for Large, Sparse, Nonsymmetric Systems of Linear Equations*, Report 229, Yale University, Dept. of Comp. Sci., New Haven, April 1982