Evaluation of Linear Solvers for Astrophysics Transfer Problems*

Osni Marques[1] and Paulo B. Vasconcelos[2]

[1] Lawrence Berkeley National Laboratory,
1 Cyclotron Road, MS 50F-1650, Berkeley, CA 94720-8139, USA
oamarques@lbl.gov
[2] Faculdade de Economia da Universidade do Porto,
Rua Dr. Roberto Frias s/n, 4200-464 Porto, Portugal
pjv@fep.up.pt

Abstract. In this work we consider the numerical solution of a radiative transfer equation for modeling the emission of photons in stellar atmospheres. Mathematically, the problem is formulated in terms of a weakly singular Fredholm integral equation defined on a Banach space. Computational approaches to solve the problem are discussed, using direct and iterative strategies that are implemented in open source packages.

Keywords: High performance computing, Fredholm integral equation, weakly singular kernel, projection approximation, numerical methods.
AMS subject classification: 32A55, 45B05, 65D20, 65R20, 68W10.

1 Introduction and Problem Overview

The emission of photons in stellar atmospheres can be modeled by a strongly coupled system of nonlinear equations. In this work we consider a restriction of the system by taking into account the temperature and pressure. We refer the reader to [2] and [13] for details on the model. The resulting integral equation, a radiative transfer problem, is expressed by

$$T\varphi - z\varphi = f, \quad \varphi \in L^1(I), \quad I = [0, \tau^\star], \tag{1}$$

defined on a Banach space $L^1(I)$, where the integral operator T is defined as

$$(T\varphi)(\tau) = \frac{\varpi}{2} \int_0^{\tau^\star} E_1\left(|\tau - \tau'|\right) \varphi\left(\tau'\right) d\tau'. \tag{2}$$

The variable τ represents the optical depth, $\tau^\star$ is the optical thickness of a stellar atmosphere, z is in the resolvent set of T and $\varpi \in]0, 1[$ is the albedo (assumed

* This work was partly supported by the Centro de Matemática da Universidade do Porto, through the programs POCTI and POSI, and partly by the Director, Office of Science, Division of Mathematical, Information, and Computational Sciences of the U.S. Department of Energy under contract No. DE-AC02-05CH11231.

to be constant in the present work). The free term f is taken to be $f(\tau) = -1$ if $0 \leq \tau \leq \tau^*/2$, and $f(\tau) = 0$ if $\tau^*/2 < \tau \leq \tau^*$. The first exponential-integral function E_1, defined by

$$E_1(\tau) = \int_1^\infty \frac{\exp(-\tau\mu)}{\mu} d\mu, \quad \tau > 0, \tag{3}$$

has a logarithmic behavior in the neighborhood of 0.

The numerical approach used to solve this problem is based on the projection of the integral operator into a finite dimensional subspace. By evaluating the projected problem on a specific basis function we obtain a linear system of equations whose coefficient matrix is banded, sparse and nonsymmetric. In order to obtain a good accuracy for the solution it is necessary to use a large dimension for the space where the problem is projected into. One possible approach is to compute an approximate initial solution in a subspace of moderate (small) size and then iteratively refine it by a Newton-type method. This approach was adopted with success in [17]. Alternatively, one can discretize the problem on a finer grid and then solve a large banded sparse algebraic linear system. In this case, depending on the dimension of the problem, we can employ either direct or iterative methods.

This work aims to explore the second approach mentioned above. Due to the large dimensional cases of interest to the astrophysicists and due to the memory limitation of computers, one needs to have access to scalable parallel versions of the code. Scalability is crucial either for the generation phase of the matrix coefficients as well as for the solution phase. In [17], scalability of the solution phase was not achieved because the (moderate size) systems were not solved in a distributed way. The parallelization of the solution phase was limited to the iterative refinement process. MPI was used for this purpose.

A large number of computational models and simulations that are analyzed and solved on nowadays high end computers benefit from the use of advanced and promptly available software tools and libraries to achieve performance, scalability and portability. In these lines, we are interested in investigating the trade offs and capabilities implemented in several packages, in particular the ones that are available in the DOE Advanced CompuTational Software (ACTS) Collection [9]. In the following sections we outline the projection and matrix formulation that we use to tackle the integral operator. Next, we give a brief description of the ACTS Collection, and the tools that are pertinent to our application. Finally, we present some numerical results and drawn up some conclusions.

2 Projection Phase and Matrix Formulation

Integral equations as the one described in the previous section are usually solved by discretization mechanisms, for instance by projection into a finite dimensional subspace. The operator T is thus approximated by T_n, with its projection into the finite dimensional subspace given by $X_n = \text{span}\{e_{n,j}, j = 1, \ldots, n\}$ (spanned by n linearly independent functions in X). In this case, we will take for X_n the

basis $e_n = [e_{n,1} \dots \ e_{n,n}]$ of piecewise constant functions on each subinterval of $[0, \tau^*]$ determined by a grid of $n+1$ points $0 = \tau_{n,0} < \tau_{n,1} < \dots < \tau_{n,n} = \tau^*$.

For $x \in X$ let

$$\langle x, e^*_{n,j} \rangle = e^*_{n,j}(x) = \frac{1}{\tau_{n,j} - \tau_{n,j-1}} \int\limits_{\tau_{n,j-1}}^{\tau_{n,j}} x(\tau) d\tau,$$

and define

$$T_n x = \pi_n T x = \sum_{j=1}^{n} \langle x, T^* e^*_{n,j} \rangle e_{n,j}, \tag{4}$$

where $\pi_n x = \sum\limits_{j=1}^{n} \langle x, e^*_{n,j} \rangle e_{n,j}$ and $T^* e^*_n \in X^*$(the adjoint space of X). The approximate problem

$$(T_n - zI)\varphi_n = f \tag{5}$$

is then solved by means of an algebraic linear system of equations

$$(A - zI)x = b, \tag{6}$$

where A is a non singular matrix of order n, and $A(i,j) = \langle e_{n,j}, T^* e^*_{n,i} \rangle$, $b(i) = \langle f, T^* e^*_{n,i} \rangle$, $x(j) = \langle \varphi_n, T^* e^*_{n,j} \rangle$ (see [2]). The relation between x and φ_n is given by

$$\varphi_n = \frac{1}{z} \left(\sum_{j=1}^{n} x(j) e_{n,j} - f \right).$$

In order to achieve an approximate solution φ_n with good accuracy by this method it may be necessary to use a very large dimensional linear system.

To obtain the elements of A we need to compute

$$A(i,j) = \frac{\varpi}{2\left(\tau_{n,i} - \tau_{n,i-1}\right)} \int_{\tau_{n,i-1}}^{\tau_{n,i}} \int_0^{\tau^*} E_1\left(|\tau - \tau'|\right) e_{n,j}\left(\tau'\right) d\tau' d\tau$$

for $i, j = 1, ..., n$. Using the fact that $E_3(0) = 1/2$, we obtain

$$A(i,j) = \begin{cases} \frac{\varpi}{2(\tau_{n,i} - \tau_{n,i-1})} \left[-E_3\left(\tau_{n,i} - \tau_{n,j}\right) + E_3\left(\tau_{n,i-1} - \tau_{n,j}\right) + \right. & \text{if } i \neq j \\ \left. + E_3\left(\tau_{n,i} - \tau_{n,j-1}\right) + E_3\left(\tau_{n,i-1} - \tau_{n,j-1}\right) \right] & \\ \varpi \left[1 + \frac{1}{\tau_{n,i} - \tau_{n,i-1}} \left(-E_3\left(\tau_{n,i} - \tau_{n,i-1}\right) - 1 \right) \right] & \text{if } i = j \end{cases}, \tag{8}$$

where

$$E_3(\tau) = \int_1^{\infty} \frac{\exp(-\tau\mu)}{\mu^3} d\mu. \tag{9}$$

For computational purposes, this function is evaluated according to [1].

3 ACTS: Tools of the Trade

The ACTS Collection consists of a set of computational tools for the solution of common and important computational problems. The tools were developed in various laboratories and universities and have allowed a wide spectrum of important computational problems to be solved to content [10]. We refer the reader to [9] for an overview of the project and available numerical tools, and also to the ACTS Information Center [12] for details about all tools available in the Collection.

In this paper we are interested in solving equation (1) on a fine mesh. ACTS incorporates the packages ScaLAPACK [6], SuperLU [7], PETSc [5] and Trilinos [11]. ScaLAPACK provides routines for distributed-memory message-passing MIMD architectures, in particular routines for solving systems of linear equations, least squares, eigenvalue problems and singular value problems. SuperLU is a library for the direct solution of large, sparse, nonsymmetric systems of linear equations, but that can also be applied efficiently to many symmetric systems. Working precision iterative refinement subroutines are provided for improved backward stability. PETSc provides a number of functionalities for the numerical solution of PDEs that require solving large-scale, sparse linear and nonlinear systems of equations. It includes nonlinear and linear equation solvers that employ a variety of Newton techniques and Krylov subspace methods. Trilinos is one the the last additions to ACTS. It targets the development of parallel solver algorithms and libraries within an object-oriented software framework. It provides self-contained packages, each one with its own set of requirements. One of this packages is AztecOO, which superseded the widely used package Aztec.

In order to solve the problem for larger values of $\tau^\star$ we need to use high performance computers as well a scalable software. Taking into account the characteristics of the coefficient matrix here we will focus on SuperLU and PETSc, for the direct and iterative solution, respectively, of a large, sparse, nonsymmetric system of linear equations.

4 Numerical results

The numerical results showed in this section were obtained on an SGI Altix 350, an AMD Opteron cluster, and an IBM SP. The Altix is configured with 32 64-bit 1.4 GHz Intel Itanium-2 processors, with 192 GBytes of shared memory. The cluster is configured with 356 dual-processor nodes, each processor running at a clock speed of 2.2 GHz, with 6 GB of memory per node, interconnected with a high-speed InfiniBand network. The IBM SP is configured with 380 compute nodes with 16 Power 3+ processors per node. Most nodes have 16 GB of memory. These three systems are located at the National Energy Research Scientific Computing Center (NERSC), Lawrence Berkeley National Laboratory, of the US Department of Energy. To validade our implementation, our experiments used only a small fraction of the computer power provided by those systems. For the physical problem we considered $\varpi = 0.75$ and $\varpi = 0.90$, and explored the band

	generation	solution	
m		factor	solve
1000	3.26E+03	6.95E+01	1.00E+00
2000	2.12E+04	1.65E+02	3.00E+00
4000	9.71E+04	3.59E+01	6.00E+00
8000	4.26E+05	7.51E+02	1.80E+01
16000	1.80E+06	1.54E+03	3.00E+01
32000	7.36E+06	3.12E+03	5.35E+01

Table 1. Normalized times for the generation of the matrix and solution of the system of equations with SuperLU, for various matrix sizes (m) and $\varpi = 0.75$, on the SGI Altix.

		generation	SuperLU	GMRES
m	nnz		(factor+solve)	(22 iterations)
1000	104918	5.79E+01	1.14E+00	1.00E+00
2000	215718	2.27E+02	2.75E+00	1.90E+00
4000	445264	8.83E+02	6.11E+00	3.36E+00
8000	880518	3.46E+03	1.26E+01	6.96E+00

Table 2. Normalized times for the generation of matrices of various sizes (m), with the corresponding number of nonzeros in the matrix (nnz) for $\varpi = 0.75$, and solution with two distinct solvers, on one processor of the IBM SP.

and sparse characteristics of the coefficient matrix obtained for this particular kernel as mentioned earlier.

In Table 1 we show normalized times required for the generation of the matrix (and right-hand side) and for the solution of problem with SuperLU on one processor of the SGI Altix, for $\varpi = 0.75$. As can be seen in the table, the most time consuming part of the simulation is the generation of the matrix, due to the large number of exponential evaluations. This phase is orders of magnitude more expensive that the other calculations and grows exponentially. The factor phase of the solution is then the second most time consuming part. One of the main advantages of the LU factorization is the potential gain that we can achieve if there is a need to solve several linear systems with the same coefficient matrix. However, this is not the case here. In addition, for higher dimensional problems direct methods usually becomes less competitive.

Table 2 shows normalized times required for the generation of the matrix (and right-hand side), and for the solution of problem with SuperLU and GMRES with Jacobi preconditioner on one processor of the IBM SP. The tolerance for the iterative method was set to 10^{-10}. We notice that for the parameters we have used in defining the problem an iterative method is very adequate in the sequential case.

m	p	generation	GMRES (22 iterations)	BiCGstab (14 iterations)
10000	2	7.70E+03	8.77E+00	9.16E+00
	4	3.87E+03	6.32E+00	6.11E+00
	8	1.93E+03	3.06E+00	3.24E+00
	16	9.72E+02	1.90E+00	2.43E+00
	32	4.87E+02	1.33E+00	1.00E+00
25000	16	6.13E+03	4.95E+00	3.62E+00
	32	3.00E+03	2.47E+00	2.04E+00
	64	1.49E+03	1.76E+00	1.36E+00

Table 3. Normalized times and number of iterations for various matrix sizes (m) and for $\varpi = 0.75$ on up to 64 processors (p) of the IBM SP.

The numbers in Tables 1 and 2 stress the need for parallelization in order to solve the problem for higher dimensions. It turns out that the terms in (8) can be analytically developed such that their generation becomes embarrassing parallel, that is, without any communication needed among the processors [17]. As a result, the data distribution can also be done accordingly to the solver used.

In Table 3 we list (normalized) times for the generation phase and for two preconditioned iterative methods implemented in PETSc on the IBM SP, for up to 64 processors, for $\varpi = 0.75$. We observe that there are significant gains by using the parallel version of the code. The generation phase is still the most time consuming part of the algorithm. The two sparse iterative solvers show similar times, although for the parameters chosen BICGstab requires fewer iterations.

In Table 4 we list (normalized) times for the generation phase and for two preconditioned iterative methods implemented in PETSc on the Opteron cluster, for $\varpi = 0.75$. Once more, we observe that there are significant gains by using the parallel version of the code. The generation phase dominates the computational costs. The two sparse iterative solvers show similar times, although, as before, for the parameters chosen BICGstab requires fewer iterations. The achieved speedup is not ideal but together with the generation phase the performance of the code is almost linear, see Figure 1. The stagnation of the speedup curve for the linear solver in Figure 1 only indicates that for $m = 10000$ it is not worthy to use more than 16 processors. In fact, the speedup gets better for larger values of m. The time required to generate the matrix for $m = 2.5 \times 10^4$ on 8 processors was similar to the time to generate the matrix for $m = 5 \times 10^4$ on 32 processors. For the solver, the time to solve the linear system for $m = 2.5 \times 10^4$ on 8 processors was similar to the time required for the solution of a linear system for $m = 5 \times 10^4$ on 16 processors, showing a good relative speedup.

In Table 5 we list, as in Table 4, the normalized times on the Operon cluster but now for $\varpi = 0.90$. The time for the generation phase was the same but, as expected, for higher values of the albedo the linear system becomes more difficult to solve. In fact, for $\varpi = 0.90$ GMRES performed better than BiCGstab,

m	p	generation	GMRES (22 iterations)	BiCGstab (14 iterations)
10000	1	5.40E+03	7.54E+00	7.95E+00
	2	2.67E+03	4.02E+00	4.58E+00
	4	1.39E+03	2.32E+00	2.56E+00
	8	6.90E+02	1.80E+00	1.97E+00
	16	3.51E+02	1.15E+00	1.25E+00
	32	1.79E+02	1.15E+00	1.36E+00
25000	4	8.41E+03	5.42E+00	5.61E+00
	8	4.28E+03	3.02E+00	3.15E+00
	16	2.16E+03	2.05E+00	1.83E+00
	32	1.07E+03	1.00E+00	1.15E+00
50000	16	8.57E+03	3.14E+00	3.20E+00
	32	4.24E+03	1.53E+00	1.86E+00

Table 4. Normalized times and number of iterations for various matrix sizes (m) and for $\varpi = 0.75$ on up to 32 processors (p) on the Opteron cluster.

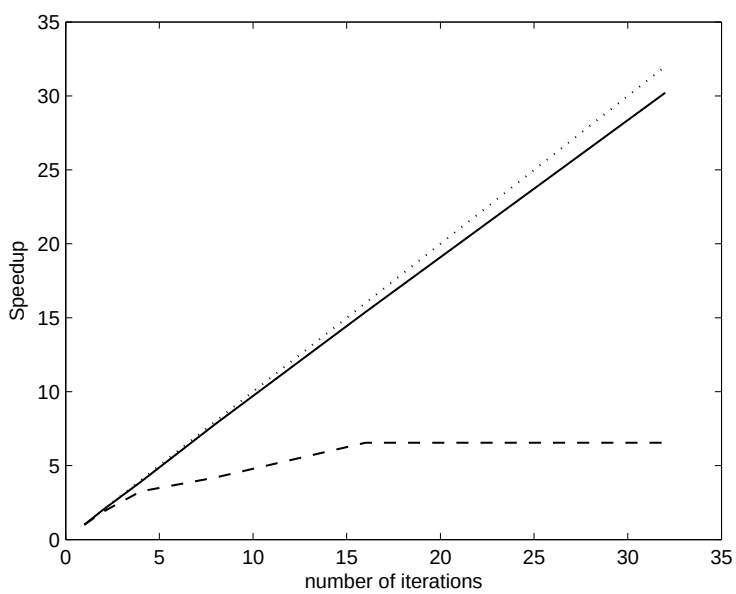

Fig. 1. Speedup for $m = 10^4$, solid line: ideal, dashed line: preconditioned GMRES.

m	p	generation	GMRES (24 iterations)	BiCGstab (37 iterations)
10000	1	2.83E+03	6.14E+00	6.99E+00
	2	1.36E+03	3.62E+00	4.12E+00
	4	7.20E+03	2.20E+00	2.31E+00
	8	3.59E+02	1.67E+00	1.76E+00
	16	1.80E+02	1.11E+00	1.30E+00
	32	9.19E+02	1.02E+00	1.00E+00
25000	4	2.83E+03	4.96E+00	5.37E+00
	8	2.22E+03	2.96E+00	3.39E+00
	16	1.11E+03	1.78E+00	2.11E+00
	32	5.55E+03	1.33E+00	1.46E+00
50000	16	4.36E+03	2.82E+00	3.14E+00
	32	2.21E+03	2.15E+00	2.05E+00

Table 5. Normalized times and number of iterations for various matrix sizes (m) and for $\varpi = 0.90$ on up to 32 processors (p) on the Opteron cluster.

requiring only a few more iterations than for $\varpi = 0.75$. That was not the case of BiCGstab, which required almost three times as many iterations and therefore a degradation in performance for this system.

The entries of the coefficient matrices show a high decay in magnitude from the diagonal. This property allowed us to successfully employ the highly parallel Jacobi preconditioner for the iterative methods. As we can see in Fig. 2, both iterative solvers performed better with the simple Jacobi preconditioner than with the block-Jacobi.

5 Conclusions and Future Work

In this contribution we discussed the numerical solution of a radiative transfer equation for modeling the emission of photons in stellar atmospheres, in particular mechanisms that we have implemented to enable the solution of large systems. This is necessary because the generation of the matrix associated to the model requires a significant amount of time. The parallelization of generation phase, as dicussed in the previous section, dramatically reduces the time to solution. At the same time, it is also important to select an appropriate solver for the resulting system of linear equations. Here we focused on tools available in the DOE ACTS Collection, and in particular (the sequential version of) SuperLU and iterative methods implemented in PETSc. These tools have delivered capability and portability and thus have been very useful in the development of a number of applications, including the one discussed here.

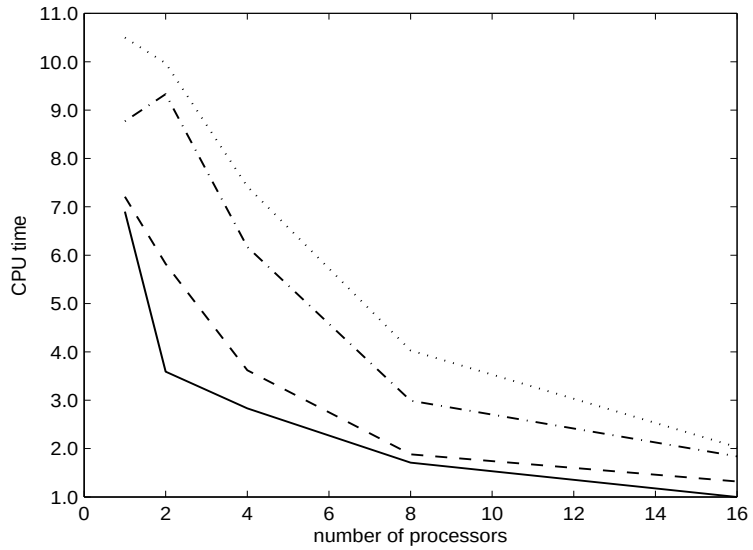

Fig. 2. CPU time in seconds for $m = 5 \times 10^4$, solid line: GMRES/Jacobi, dashed line: BiCGstab/Jacobi, dashdotted line: GMRES/blockJacobi, dotted line: BiCGstab/blockJacobi.

References

1. M. Abramowitz and I.A. Stegun. *Handbook of Mathematical Functions.* Dover, New York, 1960.
2. M. Ahues, F.D. d'Almeida, A. Largillier, O. Titaud and P. Vasconcelos. An L^1 Refined Projection Approximate Solution of the Radiation Transfer Equation in Stellar Atmospheres. *J. Comput. Appl. Math.*, 140:13–26, 2002.
3. F.D. d'Almeida and P.B. Vasconcelos, A Parallel Implementation of the Atkinson Algorithm for Solving a Fredholm Equation. *Lecture Notes in Computer Science*, 2565:368–376, 2003.
4. E. Anderson and Z. Bai and C. Bischof and S. Blackford and J. W. Demmel and J. J. Dongarra and J. Du Croz and A. Greenbaum and S. Hammarling and A. McKenney and D. C. Sorensen, *LAPACK User's Guide*, SIAM, Philadelphia, USA, 1999.
5. S. Balay and K. Buschelman and V. Eijkhout and W. D. Gropp and D. Kaushik and M. G. Knepley and L. Curfman McInnes and B. F. Smith and H. Zhang, PETSc Users Manual, Technical Report ANL-95/11, Revision 2.1.5, Argonne National Laboratory, 2004.
6. L.S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, and R.C. Whaley. *ScaLAPACK Users' Guide.* SIAM, Philadelphia, PA, 1997.
7. J. Demmel and J. Gilbert and X. Li. SuperLU Users' Guide, LBNL-44289, 1999.

8. J.J. Dongarra, I.S. Duff, D.C. Sorensen and H.A. van der Vorst. *Numerical Linear Algebra for High-Performance Computers*. Society for Industrial and Applied Mathematics, Philadelphia, 1998.
9. L.A. Drummond and O. Marques, An Overview of the Advanced CompuTational Software (ACTS) Collection. *ACM TOMS*, 31:282–301, 2005.
10. L.A. Drummond, V. Hernandez, O. Marques, J.E. Roman, and V. Vidal, A Survey of High-Quality Computational Libraries and their Impact in Science and Engineering Applications Collection. *Lecture Notes in Computer Science*, Springer Verlag, 3402:37–50, 2005.
11. M. Heroux and J. Willenbring, Trilinos Users Guide, Technical Report SAND2003-2952, Sandia national Laboratories, 2003.
12. O.A. Marques and L.A. Drummond, The DOE ACTS Information Center. http://acts.nersc.gov.
13. B. Rutily. Multiple Scattering Theoretical and Integral Equations. *Integral Methods in Science and Engineering: Analytic and Numerical Techniques*, Birkhauser, 211-231, 2004.
14. Y. Saad, SPARSKIT: A basic tool kit for sparse matrix computations, Technical Report RIACS-90-20, NASA Ames Research Center, Moffett Field, CA, 1990.
15. Y. Saad. *Iterative Methods for Sparse Linear Systems*. PWS Publishing Company, 1996.
16. M. Snir, S. Otto, S. Huss-Lederman, D. Walker, J.J. Dongarra. *MPI: The Complete Reference*. The MIT Press, 1996.
17. P.B. Vasconcelos and F.D. d'Almeida, Performance evaluation of a parallel algorithm for a radiative transfer problem. *Lecture Notes in Computer Science*, Springer Verlag, 3732: 864–871, 2006.