Parallelization of an Algorithm for computing the Eigenelements of an Astrophysics Model

Maria Antónia Forjaz

Departamento de Matemática, Universidade do Minho, 4710 Braga, Portugal

Abstract.
The integral formulation of the transfer problem that represents the restriction of a strongly coupled system of nonlinear equations dealing with radiative transfer in stellar atmospheres is our test problem. Computational approaches in open source packages that consider parallel implementations, to solve the spectral problem is considered.

Keywords: Iterative refinement, spectral approximation, numerical methods, high performance computing
PACS: 02.60.-x, 02.60.Cb, 02.60.Dc

INTRODUCTION. THE INTEGRAL PROBLEM

The emission of photons in stellar atmospheres can be modeled by a strongly coupled system of nonlinear equations. Considering that temperature and pressure are given, we have a restriction of the system that represents a radiative transfer problem

$$T\varphi = z\varphi + f, \tag{1}$$

defined on a Banach space $X = L^1([0,\tau^*])$, expressed by the following integral equation

$$(Tx)(\tau) = \int_0^{\tau^*} g(|\tau-\tau'|)x(\tau')d\tau'.\ \ 0 \le \tau \le \tau^*. \tag{2}$$

For the astrophysics problem, the variable τ and τ^* represents, respectively, the optical depth and the optical thickness of a stellar atmosphere, z is on the resolvent set of the integral operator $T : X \to X$, and $w \in]0,1[$ is the albedo, that will is assumed to be constant. The free term f, is taken to be $f(\tau) = -1$ if $0 \le \tau \le \tau^*/2$, and $f(\tau) = 0$ if $\tau^*/2 \le \tau \le \tau^*$. The kernel g is defined as $g(\tau) = \frac{w}{2}E_1(\tau)$. where $E_1 = \int_1^{+\infty} \frac{exp(-\tau\mu)}{\mu} d\mu,\ \ \tau > 0$, is the first exponential-integral function of the sequence defined by

$$E_\nu(\tau) = \int_1^{+\infty} \frac{exp(-\tau\mu^\nu)}{\mu} d\mu,\ \ \tau \ge 0, \nu > 1, \tag{3}$$

which has a logarithmic behavior in the neighborhood of $\tau = 0$.

Suppose we are interested in the computation of a cluster of eigenvalues and the associated invariant subspace of dimension μ for the operator T.

PROJECTION PHASE. ITERATIVE REFINEMENT

The numerical solution used to solve this problem is usually based on the projection of the integral operador into a finite dimensional subspace. So, the operator T is thus approximated by T_n, its projection onto the finite dimensional subspace X_n, spanned by n linearly independent function in X (and we have $(T_n - zI)\varphi_n = f$). In the subspace X_n will be consider 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 $0 = \tau_{n,0} < \tau_{n,1} < \dots < \tau_{n,1} = \tau^*$.

Let's denote by $\underline{T}$ and $\underline{T_n}$, respectively, the operators that apply the operators T and T_n to each element of an ordered family of μ elements of X. We must note that $T_n x = \sum_{j=1}^{n} < x, l_{n,j} > e_{n,j} = e_n < x, l_n >$ with $l_{n,j} = T^* e^*_{n,j}$. We want to solve:

$$\underline{T}\Phi = \Phi\Theta \tag{4}$$

where X^μ is the product space having μ factors equal to X, $\Phi = [\Phi_1, \dots, \Phi_\mu] \in X^\mu$ is normalized by $< \Phi, \Psi_n >= I_\mu$, with I_μ the identity matrix of order μ,$\Psi_n \in (X^*)^\mu$ is given (were X^* is the Hilbert-adjoint space for X), and

$$\Phi\Theta = \left[\sum_{j=1}^{\mu} \Theta(j,1)\Phi(j), \dots, \sum_{j=1}^{\mu} \Theta(j,\mu)\Phi(j) \right] \tag{5}$$

for $\Phi = \left[\Phi(1), \dots, \Phi(\mu)\right]$ and Θ a complex matrix of order μ.

The approximate problem is

$$\underline{T_n}\Phi_n = \Phi_n\Theta_n. \tag{6}$$

where Φ_n contains μ elements of X which form a basis of the invariant subspace of T_n, associated to the spectrum of Θ_n, which are considered as approximation of those of Θ, and Ψ_n is chosen as a basis of the invariant subspace of T_n^* corresponding to the spectrum of Θ_n^* (see [3] and [4] for details). We must note that (6) can be solved by means of the n-dimensional matrix problem:

$$A_n u_n = u_n \Theta_n. \tag{7}$$

For n large enough, there is a set of n spectral values of T_n (with global algebraic multiplicity μ), approximating the set of eigenvalues of Θ. So, considering n fixed, to approximate the cluster of eigenvalues of Θ that we want, we can apply for a refinement method to the invariant subspace basis of operator T_n corresponding to the eigenvalues of Θ_n. The initial approximation $\varphi^{(0)}$ will be refined by Φ_n and $\Theta^{(k)}$ will denote the refined $k-$iterative matrix, whose eigenvalues approximate those of Θ.

A finer grid of $m+1$ points, $m \gg n$, $0 = \tau_{m,0} < \tau_{m,1} < \cdots < \tau_{m,m} = \tau^*$ is set to obtain a projection operator T_m which is only used to evaluate the operator T in the refinement scheme (and not to solve the corresponding spectral problem of dimension m).

The iterative refinement formulae is, obtained considering $F : X^\mu \to X^\mu$ and solving $F(x) = \underline{T}x - x < \underline{T}x, \Psi_n >$ with an inexact Newton method [3]. The following algorithm, allows the computation of the new refined iterative $\varphi^{(k+1)}$ from $\varphi_n^{(k)}$:

```
$\varphi^{(0)} = \Phi_n$
for $k = 0, 1, 2, \dots$
    $\Theta^{(k)} =< \underline{T}\varphi^{(k)}, \Psi_n >$
    $\xi^{(k)} = (\underline{T}\varphi^{(k)})(\Theta^{(k)^{-1}})$
    $\varphi^{(k+1)} = \xi^{(k)} - \Sigma_n(F(\xi^{(k)}))$
```

MATRIX COMPUTATION

Considering T_n, the operator approximation of T, corresponding to the its projection onto X_n, T_m, the operator approximation of T, corresponding to the its projection onto X_m, $m \gg n$ (with the basis $e_m = [e_{m,1} \dots e_{m,m}]$ in X_m), A_n the matrix ($n \times n$) that represents the operator T_n restricted to X_m, we denote by A_m the matrix ($m \times m$) that represents the operator T_n restricted to X_m. We will also denote by C the matrix ($n \times m$) representing the restriction of T_n to X_m and by D the matrix ($m \times n$) representing the restriction of T_n to X_n.

Under the assumption that the coarse grid is a subset of the fine grid, we can calculate A_m, a band and sparse matrix, using the exponential-integrals defined by (3), and the coeficientes matrices C and D satisfying $C = RA_m$ and $D = A_m E$, with $RE = I_n$, and:

$$R(j,k) = \begin{cases} h_{m,k}/h_{n,j} & \text{if } q \times (j-1) + 1 \leq q \times j \\ 0 & \text{onterwise} \end{cases}$$

and

$$E(k,j) = \begin{cases} 1 & \text{if } q \times (j-1) + 1 \leq q \times j \\ 0 & \text{onterwise} \end{cases}$$

for $q = m/n$ and $h_{n,j} = \tau_{n,j} - \tau_{n,j-1}$.

For the n-dimensional matrix problem (7), we can choose $\Phi_n = e_n u_n$ from the ordered set Φ_n, that forms a basis for the invariant subspace T_n corresponding to the spectrum of Θ_n. For the m-dimensional matrix problem, we have $< \Phi_n, e_m^* >=< e_n, e_m^* > u_n$ and so $u_m = E u_n$. For the approximation problem $\underline{T}_n^* \Psi_n = \Psi_n \Theta_n^*$, that can be solved by means of the matrix eigenvalue problem $A^* v_n = v_n \Theta_n^*$, with $v_n(i,j) = \Psi_{n,j}(e_{n,i}) \in \mathbb{C}^{n \times \mu}$, we have $< x, \Psi_n >= \Theta_n^{-1} < e_n, \Psi_n >< x, l_n >$ and so $v_m^* = \Theta_n^{-1} v_n^* C$. If we write $x = e_n x_n$ we will have $< x, \Psi_n >= v_m^* x_m$ (see [3] and [4] for details).

The computation of $x = \sum_n y$, for $x, y \in X^{\mu}$, where $\sum_n$ is the block resolvent of $\underline{T}_n$ corresponding to the spectrum of Θ_n, follow from the Sylvester equation:

$$A_n x_n - x_n \Theta_n = y_n - u_n v_n^* y_n$$

where $v_n^* x_n = 0$, and considering $x = (-y + \Phi_n < y, \Psi_n > + e_n A_n x_n) \Theta_n^{-1}$.

In the subspace X_m^{μ}, applying $< ., e_m^* >$, we have: $x_m = (-y_m + u_m v^* y_n + E A_n x_n) \Theta_n^{-1}$.

PARALLEL EXPERIMENTS

Due to the large dimensional cases of interest to the astrophysicists and due to the memory limitation of computers, we need to use high performance computers as well as a scalable software. We are interested in investigating the trade offs and capabilities implemented in some of the packages available in the DOE Advanced CompuTational Software (ACTS) Collection [5], such as PETSc and SLEPc. SLEPc is based on the PETSc data structures and it employs the MPI [9] standard for message-passing communication.

The Libraries PETSc and SLEPc

The Portable, Extensible Toolkit for Scientific Computation, PETSc [7], has successfully demonstrated that the use of modern programming paradigms can ease the development of large-scale scientific application codes in Fortran, C, and C++. One of the most importance feature of PETSc is it's object-oriented design, witch provides important advantages over traditional design, for instance, hiding details of parallel execution, and allows the user to work at a higher level of abstraction. PETSc is built around a variety of data structures and algorithmic objects and the application programmer works directly with these objects rather than concentrating on the underlying data structures.

The Scalable Library for Eigenvalue Problem Computations, SLEPc [8], is a software library for computing a few eigenvalues and associated eigenvectors of a large sparse matrix or matrix pencil. It has been developed on top of PETSc and enforces the same programming paradigm. SLEPc is not only an extension of PETSc but it is also portable, scalable, efficient, and flexible, and very important are the features related with is extensibility and interoperability. The new functionality provided by the SLEPc library is organized around two objects, EPS (Eigenvalue Problem Solver), is the main object provided by SLEPc, and ST (Spectral Transformation), which encapsulates the functionality required for acceleration techniques based on the transformation of the spectrum such as shift to origin, shift and invert, and the Cayley transformation.

Results and discussion

As it was explained the ideia of the method is to solve the eigenvalue problem in a low dimensional discretization space and then, make use of an iteratively refinement of the corresponding spectral elements to yield an approximation to the spectral elements of T. In order to go throw higher dimensions, the development and implementation of parallel codes to run on high performance computers must be consider; the aim is to reduce the computing times, providing that, to be efficient these codes need to minimize as much as possible the communication overheads, and as it is always the case with parallel algorithms, a major challenge here is the so-called *scalability*, that is, we would like to go *massively parallel* and still attain good efficiency.

The eigenvalue problem was implemented in a sequencial code, for dense matrix [3], [4]. A fixed value for albedo w was considered ($w = 0.75$), as as well as possible range for τ^* (from 1000 to 4000).

Matrizes A_n and A_m (replace n by m) were obtained using the exponential-integrals (3):

$$A_n(i,j) = \begin{cases} \dfrac{w}{2h_{n,i}}\Big[E_3(|\tau_{n,i-1}-\tau_{n,j}|) - E_3(|\tau_{n,i}-\tau_{n,j}|) + E_3(|\tau_{n,i}-\tau_{n,j-1}|) - E_3(|\tau_{n,i-1}-\tau_{n,j-1}|)\Big] & ,\text{if} \quad i \neq j \\ w\Big[1 + \dfrac{1}{2h_{n,i}}(E_3(h_{n,i}-0.5))\Big] & ,\text{ if} \quad i = j \end{cases}$$

and, this is the most time consuming phase, due to the large number of exponential calculations. Nevertheless, we claim that the parallelization of this phase will reduce very significantly the time to solution. As we are interested in very large dimensions and because of that, matrix A_m does not fit in only one processor, it has to be computed in a distributed way. One possible distribution for the elements of matrix A_m is to consider a cyclic distribution by columns. As matrix A_m is a band matrix, scalability will be achieved. and the coeficientes matrices C, satisfying $C = RA_m$, can *easly* be obtain since we consider C distributed by block columns.

We must note that, considering the already done implementations of the sequencial version of the method [3], [4], we can say that it performs very well, if the desired eigenpairs are few of the largest ones. We also have that, it is needed an higher number of iterations to compute the largest eigenvalues than to compute the one's with smaller absolute value (for $m = 4000$ and $n = 800$, the largest one eigenpair only required 125.8 seconds, and 49 iterations, and to the [7] fourth eigenpair are required 194.0 seconds, and 106 iterations). This may be occur more drastically, when we have very high matrix dimensions and we want to obtain eigenvalues with very good precision. Taking in account the dimension of the initial matrix A_n, and the dimension of A_m, considering the *ratio* $q = m/n$ constant, as the dimension m grows, the cpu times required to obtain the largest eigenpairs tends to decrease. It is also true that, for larger fixed m, as the *ratio* $q = m/n$ increases, the number of iterations tend to be increase, although the method converge (for $m = 4000$ with $q = m/n = 10$, are required 143 iterations, and with $q = m/n = 40$ are required 839 iterations). This can be very important when we want to solve very high dimensional problems, although it usually requires greater number of iterations, but not cpu time. However, if the eigenpairs to refine are good approximations, (large n), the cost to get so good inicial approximations could have been so higher that it overcomes the smaller number of iterations. As expected, we also have that, as more precision is required, the number of iterations grows.

We also must refer that, for those eigenvalues that are not very close, the iterative refinement may be done in parallel, without any communication need among processors, which will certain give significant gains

Due to parallel proprieties of the algorithm, our parallel implementation will scales well but this needs to be supported by results obtained on parallel platforms with a larger number of processors (preliminary results are being obtained on a parallel computer with 30 processors available).

REFERENCES

1. O. Marques and P. B. Vasconcelos, *Evaluation of Linear Solvers for Astrophysics Transfer Problems*, Lectures Notes in Computer Science, Springer Verlag, **4395**, 466–474, (2007).
2. P. B. Vasconcelos, and F.D. d'Almeida, *Performance evalution of a parallel algorithm for a radiative tranfer problem*, Lectures Notes in Computer Science, Springer Verlag, 864–871 (2006).
3. M, Ahues, F.D. d'Almeida, A. Largillier and P. B. Vasconcelos, *Defect Correction for Spectral Computations for Integral Operator*, Communications on Pure and Apllied Analysis, **Volume 5, Number 2**, 241–250 (2006).
4. M, Ahues, F.D. d'Almeida, A. Largillier and P. B. Vasconcelos, *Comparison between two matrix implementation of an oblique spectral projection approximation of for integral operator* (to be published).
5. L.A., Drummond, O. Marques, *An Overview of the Advanced CompuTational Software (ACTS) Collection*, ACM TOMS, **31** 282–301 (2005).
6. L.A., Drummond, V. Hernandes, O. Marques, J. E. Roman, and V. Vidal, *A Survey of High-Quality Computational Libraries and their Impact in Sciences and Engineering Applications Collection*, Lectures Notes in Computer Science, Springer Verlag, **3402** 37–50 (2005).
7. S. Balay, K. Buschelman, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, B. F. Smith, H. Zhang, *PETSc Users Manual*, ANL-95/11 - Revision 2.1.5, Argonne National Laboratory (2004).
8. V. Hernandez, J. E. Roman and V. Vidal, *SLEPc: A Scalable and Flexible Toolkit for the Solution of Eigenvalue Problems*, ACM Transactions on Mathematical Software, **Volume 31, Number 3**, 351–362,(2005).
9. M. Snir, S. Otto, S.Huss-Lederman, D. Walker and J. Dongarra, *MPI: The Complete Reference*, The MIT Press, (1996).