Krylov Method Algorithms Adapted to the Grid Computing

Damien Tromeur-Dervout[*]

MCS and Center for the Development of Scientific Parallel Computing
CDCSP/ISTIL
University Lyon 1, 69622 Villeurbanne cedex, France

In this paper we define modified algorithms for general Krylov methods, that will be adapted to perform for the computing on the grid.

Motivations

Grid computer architecture are characterized by slow communication networks and high latencies [1]. Improvment of the communication network technology, may expect that standard numerical methodologies will be performant. Nevertheless, if the available networks are non dedicated, same problems of communication network with fluctuate bandwidths and high latencies will certainly appear. Indeed, as shown by the behavior of internet users, more great is the network capability, more great is the number of users and their needs.

From the numerical point of view, the Krylov methods became one of the key component of numerical solvers for large sparse linear systems coming from the discretisation of linear and non linear partial differential equations (see for example the Newton-Krylov-Schwarz method [3]). The two main components of a Krylov methods arc **Matrix-vector product** and **dot products**.

Renumbering techniques usualy lead to algorithms of matrix-vector product of PDEs problems with local communication pattern. Some, data distribution tools such as [2] allow to distribute the task computation with local communication pattern in order to minimize the communication between processes.

For the dot products, the situation is totaly different. They involve global communications between processors (at least *d-1* messages for 2^d processors with an hypercube topology). Thus, the dot product is the bottleneck for the performance of the Krylov methods implementation on the grid architecture.

The main idea of this paper is to modified the Krylov method algorithms, in order to delay as far as possible the dot products computation without destroying the Krylov method properties and convergence. For this purpose, with symbolic calculus software, we define induction formula for a Krylov method in order to go directly from the k^{th} iterate to the $(k+p)^{th}$ iterate.

Conjugate Gradient Algorithm

For example, we consider the Conjugate Gradient (CG) Krylov method. The CG method is based on the minimisation of functional J on the Krylov space $x^1 + K_r = x^1 + \{d^1,..., d_r\}$ with a scalar product $\phi(.,.)$ to define the relation of orthogonality between the Krylov vectors. The problem to solve $Ax = b$ with A symmetric is equivalent to the problem:

Find $x \in K_r$ that minimize $J(x) = \frac{1}{2}(x, Ax) - (b, x),$ (1)

$$\phi(d_i, d_j) = (d_i, Ad_j) = \begin{cases} = 0 & if\ i \neq j \\ \neq 0 & if\ i = j \end{cases}, \tag{2}$$

[*]E-mail: dtromeur@cdcsp.univ-lyon1.fr

where (.,.) represent the usual scalar product.

In the *GC* algorithm the new direction of descent at iterate *p* is computed in order to belong to the Krylov space K_p and to be orthogonal to the previous directions of descent with the scalar product ϕ. At each time step, GC performs 3 scalar products (with the third is dependant of the result of the two firsts). This is totally inefficient in the grid computing context with high latencies.

Conjugate Gradient adapted to the grid

We develop, a Maple code to program the fortran induction formula. The bilinear property of the scalar product ϕ is helpfully in this computation. The process to go directly from x_1 to x_{p+1}, is decomposed in three steps as follows:

1. Construct a Krylov space starting from x_1, compute $d_1 = r_1 = b - Ax_1$ and generate the space $\{d_1, Ad_1, A^2 d_1, ..., A^{p-1} d_1\} = K_p = \{\omega_1, \omega_2, ..., \omega_p\}$.

2. From $\omega_i = A^i \omega_1$, and $\omega_1 = d_1$, build orthogonal direction of descents searching d_i, $i > 1$ such that:

$$d_i = \omega_i + \sum_{j=1}^{j=i-1} \beta_{i,j} d_j, \; with \; \phi(d_i, d_j) = 0, \; j < i. \qquad (3)$$

3. compute the minimum of *J(x)* on $x_1 + K_p$

$$x_p = x_1 + \sum_{j=1}^{j=p} \alpha_j d_j. \qquad (4)$$

The problem is to generate recursively the $\beta_{i,j}$ and α_j with only the scalar product $H_{i,j} = (\omega_i, \omega_j)$. One can then computed once and for all at the beginning this scalar product with only one global communication of size 2**p-1* (see remark 3) or *(p* — l)*p*/2.

Let us examplify with CG algorithm and *p*—3. The induction formula lead to:

$$\beta_{2,1} = -\frac{H_{2,2}}{H_{2,1}},$$

$$\beta_{3,2} = -\frac{H_{2,1}H_{3,3} - H_{3,1}H_{3,2}}{H_{2,1}H_{3,2} - H_{3,1}H_{2,2}},$$

$$\beta_{3,1} = \frac{H_{2,1}\beta_{2,1}H_{3,3} - \beta_{2,1}H_{3,1}H_{3,2} + H_{2,2}H_{3,3} - H_{3,2}^2}{H_{2,1}H_{3,2} - H_{3,1}H_{2,2}},$$

$$\alpha_1 = \frac{H_{1,1}}{H_{2,1}},$$

$$\alpha_2 = \frac{H_{1,1}\beta_{2,1} + H_{2,1}}{H_{2,1}\beta_{2,1}^2 + H_{3,1}\beta_{2,1} + H_{2,2}\beta_{2,1} + H_{3,2}},$$

$$\begin{aligned}\gamma_3 = {} & H_{2,1}\beta_{3,2}^2\beta_{2,1}^2 + 2H_{2,1}\beta_{3,2}\beta_{3,1}\beta_{2,1} \\ & + H_{3,1}\beta_{3,2}^2\beta_{2,1} + H_{2,2}\beta_{3,2}^2\beta_{2,1} + H_{2,1}\beta_{3,1}^2 \\ & + H_{3,1}\beta_{3,2}\beta_{3,1} + H_{4,1}\beta_{3,2}\beta_{2,1} + H_{2,2}\beta_{3,2}\beta_{3,1} \\ & + H_{3,2}\beta_{3,2}^2 + H_{3,2}\beta_{3,2}\beta_{2,1} + H_{4,1}\beta_{3,1} \\ & + H_{3,2}\beta_{3,1} + H_{4,2}\beta_{3,2} + H_{3,3}\beta_{3,2} + H_{4,3},\end{aligned}$$

$$\alpha_3 = \frac{H_{1,1}\beta_{3,2}\beta_{2,1} + H_{1,1}\beta_{3,1} + H_{2,1}\beta_{3,2} + H_{3,1}}{\gamma_3}.$$

Remark 1: The formula generated by the symbolic only involve operations with scalars. Only the $H_{i,j}$ terms and matrix vector products to compute W_i are operations with vectors.

Remark 2: These formula seem quite complex, nevertheless they are very cheap compared to operations with vectors for problems where the grid architecture is used.

Remark 3: The symmetry property of the matrix *A* allows to reduce the total number of scalar product to *2p*—1. We have $H_{i,j} = H_{1,i+j-1}$. Without taking in account this symmetry property, the number of scalar product should be *p*+*p(p*-l)/2. Nevertheless, the principle to compute once and for all the scalar product at the beginning still available.

Remark 4: The methodology developed here for the CG can be applied to any other krylov method. The knowledge of the functional J and the scalar product definition for the orthogonality constraint on the Krylov vectors are the only needs.

Preliminary Conclusions and Future work

We will present at the conference some tests of parallel efficiency of this method on a grid computing architecture. The meta-computer target consists in several beowulf systems, linked by the standard Ethernet network delivering between 3 and 5 Mbit/s. The numerical stability of theses computation will be investigated.

References

1. Nicolas Barberou, Marc Garbey, Mathias Hess, Mickael Resell, Tuomo Rossi, Jari Toivanen, and Damien Tromeur-Dervout. Efficient rneta-computing of elliptic linear and non linear problems, soumis a J. of Parallel and Distributed Computing.
2. Y.P. Chien, J.D. Chen, A. Ecer, and H.U. Akay. Dynamic Load Balancing for Parallel CFD on NT Networks. In D. Keyes al editors, editor, *Proc Int. Parallel CFD '99,* pages 165-171. Elsevier Science B. V., 2000.
3. D. K. Kaushik, D. E. Keyes, and B. F. Smith. Newton-krylov-schwarz methods for aero-dynamics problems: Compressible and incompressible flows on unstructured grids. In eds.C.-H. Lai, et al, editor, *Proceedings of the llth Intl. Conf. on Domain Decomposition Methods,* pages 513 520. DDM.org, 1998.