The 18th International Conference on Finite Elements in Flow Problems (FEF2015)
16-18 March 2015, Taipei, Taiwan

ABCD-SOLVER : A HYBRID METHOD FOR SOLVING LARGE SPARSE SYSTEMS

Iain Duff[1], Ronan Guivarch[2], Daniel Ruiz[2], and Mohamed Zenadi[2]

[1] STFC-RAL, UK and CERFACS, France,
iain.duff@stfc.ac.uk
[2] CNRS-IRIT, Université de Toulouse, France,
{guivarch,ruiz,zenadi}@enseeiht.fr

Key words: *Iterative methods, hybrid solver, CG acceleration, multiprocessing environment.*

Although current work on direct methods has enabled the solution of very large systems efficiently, there are still applications that can cause problems particularly because of memory requirements. On the other hand, iterative methods can often be very slow to converge even with a good preconditioner. We are thus drawn to examine techniques that combine the best features of both approaches, and we term these hybrid methods.

We look in particular at the Block Cimmino method which is a rather elementary iterative technique but can be accelerated in a number of ways. We examine a derivative of this method which is effectively a direct method and compare the performance of this against the MUMPS multifrontal solver and Block Cimmino accelerated by block conjugate gradients. A major strength of the block Cimmino method is that it is easily parallelizable. We also comment on this aspect of the approach.

The derivation of the Block-Cimmino based dierect solver, wich we denote as ABCD Solver, the acronym standing for "*Augmented Block Cimmino Distributed Solver*", can be summarized in the following way. Consider the solution of the system

$$Ax = b \tag{1}$$

where A is an $n \times n$ sparse matrix, x is an n-vector and b is an n-vector. We use the block Cimmino algorithm which involves subdividing the system into strips of rows, viz.

$$\begin{bmatrix} A_1 \\ \vdots \\ A_p \end{bmatrix} x = \begin{bmatrix} b_1 \\ \vdots \\ b_p \end{bmatrix}$$

and solving iteratively using

$$\begin{aligned} u_i &= A_i^+ b_i - P_{\mathcal{R}(A_i^T)} x^{(k)} \quad i = 1, ...p \\ x^{(k+1)} &= x^{(k)} + \omega \sum_{i=1}^{p} u_i, \end{aligned}$$

where, A_i^+ is the Moore-Penrose pseudo-inverse and $P_{\mathcal{R}(A_i^T)} = A_i^+ A_i$ is the projector onto the range of A_i^T. The convergence is dependent on the products $A_i A_j^T$ and we develop an augmented apporach, in which system (1) is embedded in a super space with additional variables, as well as with extra constraint equations to recover the same solution, and where these products become zero. The resulting system is given by

$$\begin{bmatrix} A & C \\ B & S \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} b \\ f \end{bmatrix},$$

in which we can denote by $\bar{A}$ the submatrix $\begin{bmatrix} A & C \end{bmatrix}$ where C as been chosen to enforce the p subspaces to be orthogonal, so that we have:

$$\begin{array}{rcl} & \bar{A}^+ b &= \sum_{i=1}^{p} \bar{A}_i^+ b_i \\ P = & P_{\mathcal{R}(\bar{A}^T)} &= \sum_{i=1}^{p} P_{\mathcal{R}(\bar{A}_i^T)} \, . \end{array}$$

We construct f and S as :

$$f = -Y \bar{A}^+ b \quad \text{and} \quad S = Y (I - P) Y^T,$$

where $Y = \begin{bmatrix} 0 & I \end{bmatrix}$, and the solution is given by :

$$\begin{bmatrix} x \\ y \end{bmatrix} = \bar{A}^+ b + (I - P) Y^T S^{-1} f.$$

To obtain the solution, we currently build S and factorize it using a direct solver. The added value of this approach is the fact that the columns of S can be built in an embarrassingly parallel fashion. The memory cost can be prohibitive in the case where S is not small or sparse enough, but we observe in many cases that S remains reasonnable enough to make this approach computationnally effective. The fact that S is symmetric positive definite also offers the possibility of computing $S^{-1} f$ iteratively using conjugate gradients, without building S explicitly.

We illustrate and compare the computing and memory costs of both direct and iterative approaches on a set of large systems arising either from structural mechanics or computational fluid dynamics problems. The ABCD Solver package (release 1.0) is freely accessible at *http://abcd.enseeiht.fr*, including installation instructions and documentation.