The 8th European Congress on Computational Methods in Applied Sciences and Engineering
ECCOMAS Congress 2022
5–-9 June 2022, Oslo, Norway

A PARALLEL SOLVER FOR CFD BASED ON THE ALTERNATING ANDERSON-RICHARDSON METHOD

L. J. Chan[1], S. Marques[2] AND N. J. Hills[3]

[1,2,3] University Technology Centre, Department of Mechanical Engineering Sciences, University of Surrey, Stag Hill, Guildford GU2 7XH, UK

[1]l.chan@surrey.ac.uk; [2]s.marques@surrey.ac.uk; [3]n.hills@surrey.ac.uk

Key words: Alternating Anderson-Richardson, GMRES, linear solver, CFD

Abstract. Recent advancement in the usage and deployment of large supercomputing resources require the need for algorithmic improvements to make use of the increased parallelism architecture. The Alternating Anderson-Richardson (AAR) method has been recently shown to exhibit good performance when solving problems in distributed parallel computers. This research will extend and investigate the performance of the AAR method to solve CFD problems using a modern compressible flow solver. This work will compare its performance and scalability against commonly used linear solvers, such as the Richardson method and the Generalised Minimal RESidual (GMRES), for solving large, sparse linear systems of equations arising from CFD applications. Results using a range of turbomachinery test cases demonstrate that the current AAR implementation offers significant performance improvement over the Richardson method. The speedup of AAR with respect to GMRES is less significant due to the load imbalance across partitions.

1 INTRODUCTION

Linear systems of equations of the type $\mathbf{Ax} = \mathbf{b}$, are encountered in many fields of science and engineering as a result of discretising the partial differential equations describing physical processes such as those found in CFD applications. In order to achieve a steady state solution, a linear system of equations needs to be solved at each time step. It is well known that most of the time is spent in solving the linear system of equations and thus it is crucial to improve the current state-of-the-art linear solver to achieve better performance and scalability to fully exploit the increasing parallel hardware.

Krylov subspace method such as Biconjugate Gradient Stabilised (Bi-CGStab) [16] and GMRES [10] are the most commonly used iterative linear solvers for solving general, large linear system of equations. However, the efficiency of these methods tends to drop when running in parallel due to the high frequency of global reduction operations [1, 12, 18, 19]. In contrast, the Richardson [7] and Jacobi [9] methods do not perform any global reduction, which make them ideal for parallel computation. Nevertheless, they suffer from low convergence rate compared to Krylov subspace methods.

Recently, Alternating Anderson-Jacobi (AAJ) [6] and Alternating Anderson-Richardson (AAR) [13] are proposed as a way to improve the convergence rate of the Jacobi and Richardson iterations by performing Anderson extrapolations at periodic intervals. Both AAJ and AAR are shown to exhibit speedups, and better strong and weak scaling with respect to GMRES when solving Helmholtz and Poisson equations [13].

This research focuses on investigating the performance of AAR when solving the linear system of equations arising from CFD applications. The remainder of this paper is organised as follows: section 2 describes the theory and implementation of AAR; section 3 introduces the Hydra CFD solver and the test cases; section 4 includes the results and analysis of the performance of AAR; section 5 concludes the findings.

2 Alternating Anderson-Richardson Method

The AAR method consists of two stages of operations, which are the Richardson iterations [7] and the Anderson extrapolation [17]. The Anderson extrapolation is performed periodically to speedup the convergence of the Richardson iteration. The formulation is summarised in equations 1, 2 and 3.

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{x}_k + \mathbf{G}_k \mathbf{r}_k; & (1) \\ \mathbf{r}_k &= \mathbf{b} - \mathbf{A}\mathbf{x}_k; & (2) \end{aligned}$$

where the matrix $\mathbf{G}_k$ is given by:

$$\mathbf{G}_k = \begin{cases} \omega \mathbf{I} & \text{if } k/p \notin \mathbb{N} \\ \beta \mathbf{I} - (\mathbf{X}_k + \beta \mathbf{F}_k)\left(\mathbf{F}_k^T \mathbf{F}_k\right)^{-1} \mathbf{F}_k^T & \text{if } k/p \in \mathbb{N}. \end{cases} \quad (3)$$

$\mathbf{X}_k$ is an array that stores the difference between two successive approximate solutions from the Richardson iterations as shown in equation 4, whereas $\mathbf{F}_k$ stores the difference between two successive residuals as shown in equation 5.

$$\mathbf{X}_k = \left[(\mathbf{x}_{k-m+1} - \mathbf{x}_{k-m}) \ (\mathbf{x}_{k-m+2} - \mathbf{x}_{k-m+1}) \ \dots \ (\mathbf{x}_k - \mathbf{x}_{k-1}) \right], \quad (4)$$

$$\mathbf{F}_k = \left[(\mathbf{r}_{k-m+1} - \mathbf{r}_{k-m}) \ (\mathbf{r}_{k-m+2} - \mathbf{r}_{k-m+1}) \ \dots \ (\mathbf{r}_k - \mathbf{r}_{k-1}) \right]; \quad (5)$$

ω and β are relaxation parameters used in the Richardson and Anderson updates respectively. They can be viewed as a tool to scale the residual vectors. However, these parameters do not affect the end solution in exact arithmetic. Thus, by convention, they are set to one. The parameter k is the iteration counter and p represents the frequency of performing the Anderson extrapolation. The parameter m is the maximum number of basis vectors.

AAR is a class of Krylov subspace method, where the differences between solutions of two successive Richardson iterations form the basis vectors that belong to subspace $\mathcal{K}_m(\mathbf{A}, \mathbf{r}_0)$, where

$$\mathcal{K}_m(\mathbf{A}, \mathbf{r}_0) = \text{span}\{\mathbf{r}_0, \mathbf{A}\mathbf{r}_0, \dots, \mathbf{A}^{m-1}\mathbf{r}_0\}. \quad (6)$$

The Anderson extrapolation on the other hand solves the least square problem by finding the solution, $\bar{\mathbf{x}}$. In the original AAR algorithm proposed in [13], the Anderson extrapolation performs one extra Richardson iteration after solving the least square problem and thus the final solution lies within subspace $\mathcal{K}_{m+1}(\mathbf{A}, \mathbf{r}_0)$ as shown in equation 7.

$$\mathbf{x}_{m+1} \in \mathcal{K}_{m+1}(\mathbf{A}, \mathbf{r}_0) = \operatorname{span}\{\mathbf{r}_0, \mathbf{A}\mathbf{r}_0, \ldots, \mathbf{A}^{m-1}\mathbf{r}_0, \mathbf{A}^m\mathbf{r}_0\}. \tag{7}$$

Although the final solution, $\mathbf{x_{m+1}}$ spans one more dimension than $\bar{\mathbf{x}}$, experiments show that $\bar{\mathbf{x}}$ gives better convergence rate than $\mathbf{x_{m+1}}$ for our applications. This is because $\mathbf{x_{m+1}}$ is not the solution that minimises the residual in $\mathcal{K}_{m+1}(\mathbf{A}, \mathbf{r}_0)$. Thus, the algorithm is slightly modified so that the Anderson extrapolation stops after solving the least square problem. The modified preconditioned AAR is summarised in algorithm 1.

Algorithm 1 Modified Preconditioned Alternating Anderson Richardson

```
1:  Initialise $\mathbf{x}_0, \omega, \beta, p, m, tol, maxits$
2:  Compute $\mathbf{r}_0 = \mathbf{M}^{-1}(\mathbf{b} - \mathbf{A}\mathbf{x}_0)$
3:  Compute $\mathbf{x}_1 = \mathbf{x}_0 + \omega\mathbf{r}_0$
4:  for $k = 1$ to $maxits$ do
5:      Compute $\mathbf{r}_k = \mathbf{M}^{-1}(\mathbf{b} - \mathbf{A}\mathbf{x}_k)$
6:      Compute $l = min\{k, m\}$
7:      Compute $i = mod\left(\frac{k-1}{p}\right) + 1$
8:      $\mathbf{X}(:, i) = \mathbf{x}_k - \mathbf{x}_{k-1} \in \mathbb{R}^{n \times l}$
9:      $\mathbf{F}(:, i) = \mathbf{r}_k - \mathbf{r}_{k-1} \in \mathbb{R}^{n \times l}$
10:     if $mod\left(\frac{k}{p}\right) \neq 0$ then
11:         Compute $\mathbf{x}_{k+1} = \mathbf{x}_k + \omega\mathbf{r}_k$
12:     else
13:         Compute $\mathbf{y} = -\left(\mathbf{F}^T\mathbf{F}\right)^{-1}\mathbf{F}^T\mathbf{r}_k \in \mathbb{R}^l$
14:         Compute $\bar{\mathbf{x}}_k = \mathbf{x}_k + \mathbf{X}\mathbf{y}$
15:         Update $\mathbf{r}_k = \mathbf{r}_k + \mathbf{F}\mathbf{y}$
16:         if $\frac{\|\mathbf{r}_k\|_2}{\|\mathbf{r}_0\|_2} < tol$ or $k == maxits$ then
17:             Exit Loop
18:         else
19:             Update $\mathbf{x}_k = \bar{\mathbf{x}}$
20:             Compute $\mathbf{x}_{k+1} = \mathbf{x}_k + \beta\mathbf{r}_k$
21:         end if
22:     end if
23: end for
```

In exact arithmetic, the modified AAR is equivalent to GMRES, see for example [3, 17] for proof. In line 13 of Algorithm 1, both the matrix-matrix multiplication, $\mathbf{F}^T\mathbf{F}$ and matrix-vector multiplication, $\mathbf{F}^T\mathbf{r}_k$, each require global reductions. However, these two operations are independent and their results are not dependent on each other. Therefore, the global reduction can be reduced to one operation by storing both results into a single array before communicating. The inversion of the matrix $\mathbf{F}^T\mathbf{F}$ is achieved implicitly by solving a small dense linear system of equations by performing QR factorisation with a Householder Transformation. These two operations can be easily optimised using level 3 and level 2 BLAS functions respectively.

The frequency of global reductions is proportional to the frequency of performing the Anderson extrapolation step. Thus, for efficient use of AAR, the Anderson extrapolation should be performed as infrequently as possible. Ideally, it should only be performed once at the last iteration unless a restarting strategy is adopted. Thus, the AAR method is less attractive for applications requiring the solution to meet a fixed relative tolerance. Unlike GMRES where the residual norm can be obtained cheaply at every iteration, AAR needs to solve the least square problem by the Anderson extrapolation before the residual norm can be calculated. This property has a limited impact on iterative methods, such as those found in most CFD solvers, as the linear system of equations is solved approximately. For AAR, it is thus advantageous to fix the number of linear solver iterations only.

3 Experiment Setup

The linear solvers are implemented in Hydra CFD solver, which solves the compressible Reynolds-averaged Navier-Stokes (RANS) flow equations using the finite volume method, [2, 5]. Several turbulence models are available, including Spalart-Allmaras turbulence model [11] and $k-\omega$ SST turbulence model [4]. Hydra employs node-centred spatial discretisation, where each node forms a dual volume around itself and multiple faces at its boundary. The convective flux is evaluated at each boundary face with Roe scheme [8].

For temporal discretisation, Hydra employs an implicit 3-step Runge-Kutta (RK) scheme, which was introduced in [15] and further optimised in [14].

Although the residual is evaluated with second order accurate scheme, the implemented Jacobian matrix is only first order accurate. This strategy has the benefit of reducing the computation time of linear solvers but at the cost of the convergence rate. The linear solvers tested include the Richardson method, GMRES and AAR. The linear solvers are preconditioned by point-block ILU0 [9]. Note that for this study, we are only interested in the optimum performance of these solvers. Therefore, the parameters such as the CFL value and the number of linear solver iterations are tuned for optimum performance. The convergence tolerance of the Newton iteration is set to 10^{-16} for all test cases.

Two test cases representing different typical gas turbine engine components or flows, are used to assess the linear solvers implemented into Hydra by this work. These include a rotor stator cavity and a low-pressure turbine (LPT) blade.

The first model tested is a cavity located in between a stator and a rotor rotating at a fixed angular velocity as shown in figure 1. The two rectangular sides of the sector are set to periodic boundary condition, whereas the other boundary walls are set to no-slip viscous wall. It consists of approximately 3.96M number of nodes and adopts the Spalart-Allmaras turbulence model using a wall function formulation.

The second test model is an LPT blade as shown in figure 1. The model consists of an LPT blade with subsonic inflow and outflow at both ends. The subsonic inflow profile for total pressure, total temperature, turbulent kinematic viscosity, whirl angle and pitch angle are assigned to the inflow surface of the model, whereas the subsonic outflow profile for static pressure is assigned to the outflow surface of the model. No-slip viscous wall boundary conditions are enforced at the wall of the LPT blade, and the upper and lower sides are set as periodic boundaries. This model consists of approximately 11.7M number of nodes and is modelled with the Spalart-Allmaras turbulence model using a wall function formulation.

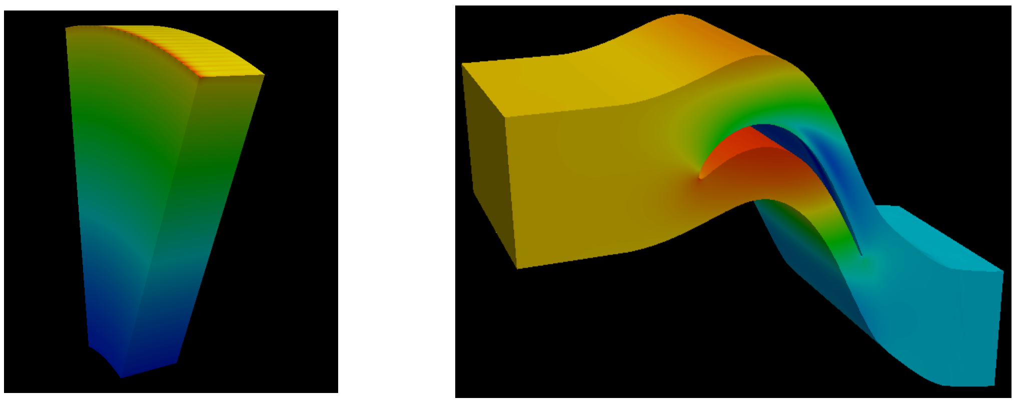
Figure 1: Rotor Stator (left) and LPT Blade (right)

4 Results and Discussion

4.1 Performance Comparison of Linear Solvers

The speedup of GMRES and AAR with respect to the Richardson method is shown in figure 2. The figure shows that both GMRES and AAR achieve significant speedup with respect to the Richardson method. However, the speedup of AAR with respect to GMRES is marginal. Additionally, AAR also exhibits similar strong scaling when compared to GMRES as shown in figure 3. The outcome is in contrast with the results obtained in the literature [13], in which AAR achieved better scalability and performance than GMRES. The literature reports the speedup of 1.38 with respect to GMRES with ILU(0) preconditioner when solving a Helmholtz equation [13].

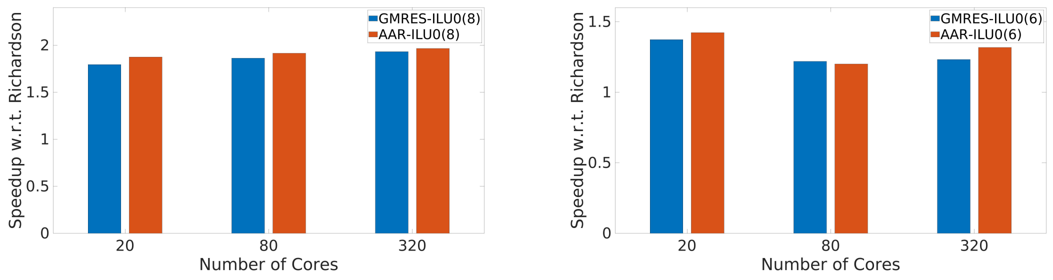

Figure 2: Speedup with respect to Richardson for Rotor Stator (left) and LPT Blade (right) Test Case

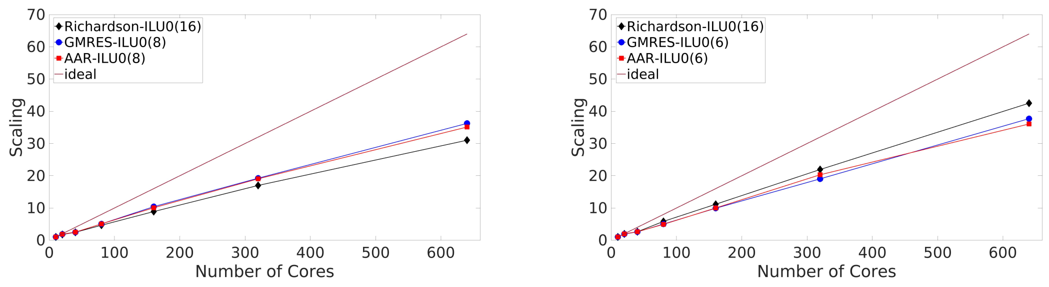

Figure 3: Scalability for Rotor Stator (left) and LPT Blade (right) Test Case

4.2 Performance Analysis of AAR

A test model described in Algorithm 2, is setup to investigate how the $\mathbf{M}^{-1}\mathbf{Ax}$ operation impacts the global synchronisation times across processes and determine how this type of operation contributes to the low speedup achieved by the AAR method. In GMRES, a global reduction is executed after performing an $\mathbf{M}^{-1}\mathbf{Ax}$ operation, whereas in AAR, a global reduction is executed only after performing n number of $\mathbf{M}^{-1}\mathbf{Ax}$ operations. Therefore, this test is a proxy for the cost of the global reductions performed by GMRES and AAR. Based on figure 4, the global synchronisation time increases as the number of $\mathbf{M}^{-1}\mathbf{Ax}$ operations increases. This implies that the global reduction performed by AAR is significantly more expensive than that of GMRES.

Algorithm 2 Global Synchronisation Test Model

```
 1: Set a barrier for global synchronisation
 2: for $i = 1$ to $n$ do
 3:     Perform $\mathbf{y} = \mathbf{A}\mathbf{x}$
 4:     Perform $\mathbf{x} = \mathbf{M}^{-1}\mathbf{y}$
 5: end for
 6: $t1$ = wall clock time
 7: Set a barrier for global synchronisation
 8: $t2$ = wall clock time
 9: $dt = t2 - t1$
10: Perform global reduction on $dt$
11: Divide $dt$ by the total number of processor to get the average communication time
```

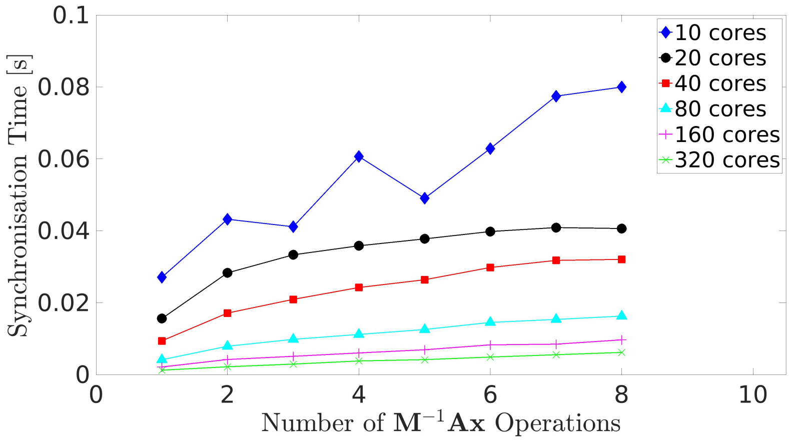

Figure 4: Global Synchronisation Time Vs Number of $\mathbf{M}^{-1}\mathbf{A}\mathbf{x}$ Operations

The increase in the global synchronisation time is likely to be caused by the load imbalance across processes, in which the distribution of the load is shown in table 1. Load imbalancing can be interpreted as the difference in the number of operations between the fastest and the slowest processes. As more $\mathbf{M}^{-1}\mathbf{A}\mathbf{x}$ operations are performed, the difference in the number of operations accumulates and thus increases the waiting time of the processes in the event of global communication.

Number of Processes	Minimum Number of Nodes	Maximum Number of Nodes	Average Number of Nodes	Difference between Maximum and Minimum
10	389180	403590	396000	14410
20	191510	202950	198000	11440
40	93830	104170	99000	10340
80	47080	51920	49500	4840
160	23034	26070	24750	3036
320	11440	13337	12375	1897

Table 1: Partitioning Data

5 Conclusion

In conclusion, both GMRES and AAR show significant speedup with respect to the Richardson method in solving compressible flow problems for turbomachinery applications. However, the speedup of AAR with respect to GMRES is less than expected. AAR and GMRES are also shown to exhibit similar strong scaling.

The reason for the low speedup of AAR with respect to GMRES is due to the increased cost of global synchronisation of AAR by having low frequency of global reduction. The increased cost of global synchronisation is likely caused by the load imbalance across the processes. As more iterations are conducted without global synchronisation, the load difference between the fastest and the slowest processes builds up and causes longer waiting time in the event of global communication.

Acknowledgements

The authors thank Dr. Dario Amirante for providing the test cases used in this paper. The first author gratefully thanks Rolls Royce and University of Surrey for the PhD studentship.

REFERENCES

[1] Ashby, T. J., Ghysels, P., Heirman, W., and Vanroose, W. The impact of global communication latency at extreme scales on krylov methods. In *International Conference on Algorithms and Architectures for Parallel Processing* (2012), Springer, pp. 428–442.

[2] Lapworth, L. Hydra-cfd: a framework for collaborative cfd development. In *International conference on scientific and engineering computation (IC-SEC)* (2004), vol. 30.

[3] Lupo Pasini, M. Convergence analysis of anderson-type acceleration of richardson's iteration. *Numerical Linear Algebra with Applications 26*, 4 (2019), e2241.

[4] Menter, F. R. Improved two-equation k-omega turbulence models for aerodynamic flows. *Nasa Sti/recon Technical Report N 93* (1992), 22809.

[5] Moinier, P. *Algorithm developments for an unstructured viscous flow solver.* PhD thesis, Oxford University Oxford, UK, 1999.

[6] Pratapa, P. P., Suryanarayana, P., and Pask, J. E. Anderson acceleration of the jacobi iterative method: An efficient alternative to krylov methods for large, sparse linear systems. *Journal of Computational Physics 306* (2016), 43–54.

[7] Richardson, L. F. Ix. the approximate arithmetical solution by finite differences of physical problems involving differential equations, with an application to the stresses in a masonry dam. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character 210*, 459-470 (1911), 307–357.

[8] Roe, P. L. Approximate riemann solvers, parameter vectors, and difference schemes. *Journal of computational physics 43*, 2 (1981), 357–372.

[9] Saad, Y. *Iterative methods for sparse linear systems.* SIAM, 2003.

[10] Saad, Y., and Schultz, M. H. Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on scientific and statistical computing 7*, 3 (1986), 856–869.

[11] Spalart, P., and Allmaras, S. A one-equation turbulence model for aerodynamic flows. In *30th aerospace sciences meeting and exhibit* (1992), p. 439.

[12] Sturler, E. d., and van der Vorst, H. A. Communication cost reduction for krylov methods on parallel computers. In *International Conference on High-Performance Computing and Networking* (1994), Springer, pp. 190–195.

[13] Suryanarayana, P., Pratapa, P. P., and Pask, J. E. Alternating anderson–richardson method: An efficient alternative to preconditioned krylov methods for large, sparse linear systems. *Computer Physics Communications 234* (2019), 278–285.

[14] Swanson, R., Turkel, E., and Yaniv, S. Analysis of a rk/implicit smoother for multigrid. In *Computational Fluid Dynamics 2010.* Springer, 2011, pp. 409–417.

[15] Swanson, R. C., Turkel, E., and Rossow, C.-C. Convergence acceleration of runge–kutta schemes for solving the navier–stokes equations. *Journal of Computational Physics 224*, 1 (2007), 365–388.

[16] Van d VHA, B. Cgstab: a fast and smoothly converging variant of bi-cg for the solution of nonsymmetric linear systems. *Siam Journal on Scientific & Statistical Computing 13*, 2 (1992), 631–644.

[17] Walker, H. F., and Ni, P. Anderson acceleration for fixed-point iterations. *SIAM Journal on Numerical Analysis 49*, 4 (2011), 1715–1735.

[18] Yang, L. T., and Brent, R. P. The improved krylov subspace methods for large and sparse linear systems on bulk synchronous parallel architectures. In *Proceedings International Parallel and Distributed Processing Symposium* (2003), IEEE, pp. 11–pp.

[19] Zhu, S.-X., Gu, T.-X., and Liu, X.-P. Minimizing synchronizations in sparse iterative solvers for distributed supercomputers. *Computers & Mathematics with Applications 67*, 1 (2014), 199–209.