A Direct Elliptic Solver Based on Hierarchically Low-Rank Schur Complements

Gustavo Chávez, George Turkiyyah, and David E. Keyes

1 Introduction

Cyclic reduction was conceived in 1965 for the solution of tridiagonal linear systems, such as the one-dimensional Poisson equation (Hockney 1965). Generalized to higher dimensions by recursive blocking, it is known as block cyclic reduction (BCR) (Buzbee et al. 1970). It can be used for general (block) Toeplitz and (block) tridiagonal linear systems; however, it is not competitive for large problems, because its arithmetic complexity grows superlinearly. Cyclic reduction can be thought of as a direct Gaussian elimination that recursively computes the Schur complement of half of the system. The complexity of Schur complement computations is dominated by the inverse. By considering a tridiagonal system and an even/odd ordering, cyclic reduction decouples the system such that the inverse of a large block is the block-wise inverse of a collection of independent smaller blocks. This addresses the most expensive step of the Schur complement computation in terms of operation complexity and does so in a way that launches concurrent subproblems. Its concurrency feature, in the form of recursive bisection, makes it interesting for parallel environments, provided that its arithmetic complexity can be improved.

We address the time and memory complexity growth of the traditional cyclic reduction algorithm by approximating dense blocks as they arise with hierarchical matrices ($\mathcal{H}$-Matrices). The effectiveness of the block approximation relies on the rank structure of the original matrix. Many relevant operators are known to have blocks of low rank off the diagonal. This philosophy follows recent work

G. Chávez • G. Turkiyyah • D.E. Keyes (✉)
Extreme Computing Research Center, King Abdullah University of Science and Technology, 23955 Thuwal, Saudi Arabia
e-mail: gustavo.chavezchavez@kaust.edu.sa; george.turkiyyah@kaust.edu.sa; david.keyes@kaust.edu.sa

C.-O. Lee et al. (eds.), *Domain Decomposition Methods in Science and Engineering XXIII*, Lecture Notes in Computational Science and Engineering 116, DOI 10.1007/978-3-319-52389-7_12

discussed below, but to our knowledge this is the first demonstration of the utility of complexity-reducing hierarchical substitution in the context of cyclic reduction.

The synergy of cyclic reduction and hierarchical matrices leads to a parallel fast direct solver of log-linear arithmetic complexity, $O(N \log^2 N)$, with controllable accuracy. The algorithm is purely algebraic, depending only on a block tridiagonal structure. We call it Accelerated Cyclic Reduction (ACR). Using a well-known implementation of $\mathcal{H}$-LU (Grasedyck et al. 2009), we demonstrate the range of applicability of ACR over a set of model problems including the convection-diffusion equation with recirculating flow and the wave Helmholtz equation, problems that cannot be tackled with the traditional FFT enabled version of cyclic reduction, FACR (Swarztrauber 1977). We show that ACR is competitive in time to solution as compared with a global $\mathcal{H}$-LU factorization that does not exploit the cyclic reduction structure. The fact that ACR is completely algebraic expands its range of applicability to problems with arbitrary coefficient structure within the block tridiagonal sparsity structure, subject to their amenability to rank compression. This gives the method robustness in some applications that are difficult for multigrid. The concurrency and flexibility to tune the accuracy of individual matrix block approximations makes it interesting for emerging many-core architectures. Finally, as with other direct solvers, there are complexity-accuracy tradeoffs that would naturally lead to the development of a new scalable preconditioner based on ACR.

2 Related Work

Exploiting underlying low-rank structure is a trending strategy for improving the performance of sparse direct solvers.

Nested dissection based clustering of an $\mathcal{H}$-Matrix is known as $\mathcal{H}$-Cholesky by Ibragimov et al. (2007) and $\mathcal{H}$-LU by Grasedyck et al. (2009), the main idea being to introduce $\mathcal{H}$-Matrix approximation on Schur complements based on domain decomposition. This is accomplished by a nested dissection ordering of the unknowns, and the advantage is that large blocks of zeros are preserved after factorization. The non-zero blocks are replaced with low-rank approximations, and an LU factorization is performed, using hierarchical matrix arithmetics. Recently, Kriemann (2013) demonstrated that $\mathcal{H}$-LU implemented with a task-based scheduling based on a directed acyclic graph is well suited for modern many-core systems when compared with the conventional recursive algorithm. A similar line of work by Xia and Gu (2010) also proposes the construction of a rank-structured Cholesky factorization via the HSS hierarchical format (Chandrasekaran et al. 2006). Figure 1 illustrates the differences between nested dissection ordering and the even/odd (or red/black) ordering of cyclic reduction.

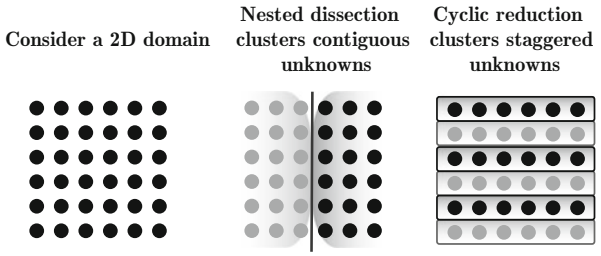

Fig. 1 The nested dissection ordering recursively clusters contiguous unknowns by bisection, whereas the red/black ordering recursively clusters staggered unknowns, allowing isolation of a new readily manipulated diagonal block

Multifrontal factorization, with low-rank approximations of frontal matrices, as in the work of Xia et al. (2010) also relies on nested dissection as the permutation strategy, but it uses the multifrontal method as a solver. Frontal matrices are approximated with the HSS format, while the solver relies on the corresponding HSS algorithms for elimination (Xia et al. 2010). A similar line of work is the generalization of this method to 3D problems and general meshes by Schmitz and Ying (2012, 2014). More recently, Ghysels et al. (2015) introduced a method based on a fast ULV decomposition and randomized sampling of HSS matrices in a many-core environment, where HSS approximations are used to approximate fronts of large enough size, as the complexity constant in building an HSS approximation is only convenient for large matrices.

This strategy is not limited to any specific hierarchical format. Aminfar et al. (2016) proposed the use of the HODLR matrix format Ambikasaran and Darve (2013), also in the context of the multifrontal method. The well known solver MUMPS now also exploits the low-rank property of frontal matrices to accelerate its multifrontal implementation, as described in Amestoy et al. (2014).

3 Accelerated Cyclic Reduction

Consider the two-dimensional linear variable-coefficient Poisson equation (1) and its corresponding block tridiagonal matrix structure resulting from a second order finite difference discretization, as shown in (2):

$$-\nabla \cdot \kappa(\mathbf{x}) \nabla u = f(\mathbf{x}), \tag{1}$$

$$A = \text{tridiag}(E_i, D_i, F_i) = \begin{bmatrix} D_1 & F_1 & & & \\ E_2 & D_2 & F_2 & & \\ & \ddots & \ddots & \ddots & \\ & & E_{n-1} & D_{n-1} & F_{n-1} \\ & & & E_n & D_n \end{bmatrix}. \tag{2}$$

We leverage the fact that for arbitrary $\kappa(\mathbf{x})$, the tridiagonal blocks D_i are *exactly* representable by rank 1 $\mathcal{H}$-Matrix since the off-diagonal blocks have only one entry regardless of their coefficient, and the blocks E_i and F_i are diagonal. As cyclic reduction progresses, the resulting blocks will have a bounded increase in the numerical ranks of their off-diagonal blocks. This numerical off-diagonal rank may be tuned to accommodate for a specified accuracy. We choose the $\mathcal{H}$-Matrix format proposed in Hackbusch (1999) by Hackbusch, although ACR is not limited to a specific hierarchical format. In terms of admissibility condition, we choose weak admissibility, as the sparsity structure is known beforehand and it proved effective in our numerical experiments.

Approximating each block as an $\mathcal{H}$-Matrix, we use the corresponding hierarchical arithmetic operations as cyclic reduction progresses, instead of the conventional linear algebra arithmetic operations. The following table summarizes the complexity estimates in terms of time and memory while dealing with a $n \times n$ block in a typical dense format and as a block-wise approximation with a rank-r $\mathcal{H}$-Matrix.

	Inverse	Storage
Dense Block	$\mathcal{O}(n^3)$	$\mathcal{O}(n^2)$
$\mathcal{H}$ Block	$\mathcal{O}(r^2 n \log^2 n)$	$\mathcal{O}(rn \log n)$

The following table summarizes the complexity estimates of the methods discussed so far in a two-dimensional square mesh where N is the total number of unknowns, neglecting the dependence upon rank. The derivation of the complexity estimates for $\mathcal{H}$-LU can be found in Bebendorf (2008).

	Operations	Memory
BCR	$\mathcal{O}(N^2)$	$\mathcal{O}(N^{1.5})$
$\mathcal{H}$-LU	$\mathcal{O}(N \log^2 N)$	$\mathcal{O}(N \log N)$
ACR	$\mathcal{O}(N \log^2 N)$	$\mathcal{O}(N \log N)$

With block-wise approximations in place, block cyclic reduction becomes ACR. BCR consists of two phases: reduction and back-substitution. The reduction phase is equivalent to block Gaussian elimination without pivoting on a permuted system $(PAP^T)(Pu) = Pf$. Permutation decouples the system, and the computation of the Schur complement reduces the problem size by half. This process is recursive and finishes when a single block is reached, although the recursion can be stopped when the system is small enough to be solved directly.

As an illustration, consider a system of $n = 8$ points per dimension, which translates into a $N \times N$ sparse matrix, with $N = n^2$. The first step is to permute the system, which with an even/odd ordering becomes:

$$\left[\begin{array}{cccc|cccc} D_0 & & & & F_0 & & & \\ & D_2 & & & E_2 & F_2 & & \\ & & D_4 & & & E_4 & F_4 & \\ & & & D_6 & & & E_6 & F_6 \\ \hline E_1 & F_1 & & & D_1 & & & \\ & E_3 & F_3 & & & D_3 & & \\ & & E_5 & F_5 & & & D_5 & \\ & & & E_7 & & & & D_7 \end{array}\right] \left[\begin{array}{c} u_0 \\ u_2 \\ u_4 \\ u_6 \\ \hline u_1 \\ u_3 \\ u_5 \\ u_7 \end{array}\right] = \left[\begin{array}{c} f_0 \\ f_2 \\ f_4 \\ f_6 \\ \hline f_1 \\ f_3 \\ f_5 \\ f_7 \end{array}\right]. \tag{3}$$

Consider the above 2×2 partitioned system (3) as H. The upper-left block is block-diagonal, which means that its inverse can be computed as the inverse of each individual block (D_0, D_2, D_4, and D_6), in parallel and with hierarchical matrix arithmetics. The Schur complement of the upper-left partition may then be computed as follows:

$$\left[\begin{array}{c|c} H_{11} & H_{12} \\ \hline H_{21} & H_{22} \end{array}\right] \left[\begin{array}{c} u_{even} \\ \hline u_{odd} \end{array}\right] = \left[\begin{array}{c} f_{even} \\ \hline f_{odd} \end{array}\right]. \tag{4}$$

$$(H_{22} - H_{21}H_{11}^{-1}H_{12})u_{odd} = f^{(1)}, \qquad f^{(1)} = f_{odd} - H_{21}H_{11}^{-1}f_{even}. \tag{5}$$

Superscripts indicates algorithmic steps. A key property of the Schur complement of a block tridiagonal matrix is that it yields another block tridiagonal matrix, as can been seen in the resulting permuted matrix system (5):

$$\left[\begin{array}{ccc|ccc} D_0^{(1)} & & & F_0^{(1)} & & \\ & D_2^{(1)} & & E_2^{(1)} & F_2^{(1)} & \\ \hline E_1^{(1)} & F_1^{(1)} & & D_1^{(1)} & & \\ & E_3^{(1)} & & & D_3^{(1)} & \end{array}\right] \left[\begin{array}{c} u_0^{(1)} \\ u_2^{(1)} \\ \hline u_1^{(1)} \\ u_3^{(1)} \end{array}\right] = \left[\begin{array}{c} f_0^{(1)} \\ f_2^{(1)} \\ \hline f_1^{(1)} \\ f_3^{(1)} \end{array}\right]. \tag{6}$$

One step further, the computation of the Schur complement of the permuted system (6), results in:

$$\left[\begin{array}{cc} D_0^{(2)} & F_0^{(2)} \\ E_1^{(2)} & D_1^{(2)} \end{array}\right] \left[\begin{array}{c} u_0^{(2)} \\ u_1^{(2)} \end{array}\right] = \left[\begin{array}{c} f_0^{(2)} \\ f_1^{(2)} \end{array}\right]. \tag{7}$$

A last round of permutation and Schur complement computation leads to the $D_0^{(3)}$ block, which is the last step of the reduction phase of Cyclic Reduction. A back-substitution phase to recover the solution also consists of $\log n$ steps. Each

step involves matrix-vector products involving the off-diagonal blocks $E^{(i)}$ and $F^{(i)}$ and the inverses of the diagonal $D^{(i)}$ blocks computed during the elimination phase. These matrix-vector operations are done efficiently with hierarchical matrix arithmetics.

4 Numerical Results in 2D

We select two test cases to provide a baseline of performance and robustness as compared with the $\mathcal{H}$-LU implementation in HLIBpro Hackbusch et al. (xxxx), and with the AMG implementation in Hypre Lawrence Livermore National Laboratory (2017). Tests are performed in the shared memory environment of a 36-core Intel Haswell processor.

The first test is the wave Helmholtz equation.

$$\begin{aligned} \nabla^2 u + k^2 u = f(\mathbf{x}), \qquad & \mathbf{x} \in \Omega = [0,1]^2 \quad u(\mathbf{x}) = 0,\ x \in \Gamma \\ & f(\mathbf{x}) = 100 e^{-100((x-0.5)^2+(y-0.5)^2)}. \end{aligned} \tag{8}$$

For large values of kh, where h is the mesh spacing, discretization leads to an indefinite matrix. Performance over a range of k is shown in Fig. 2, for $h = 2^{-10}$. We compare ACR and $\mathcal{H}$-LU with AMG as a direct solver and as a preconditioner in combination with GMRES. For small α AMG outperforms the direct methods, but AMG loses robustness with rising indefiniteness.

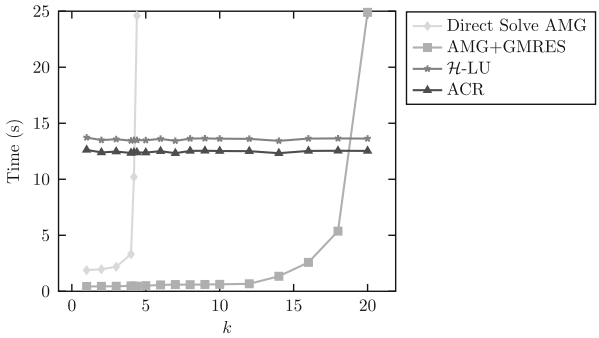

Fig. 2 Runtime versus wavenumber for fixed mesh size in the Wave Helmhotz equation. AMG is the method of choice for small k, but loses robustness with indefiniteness

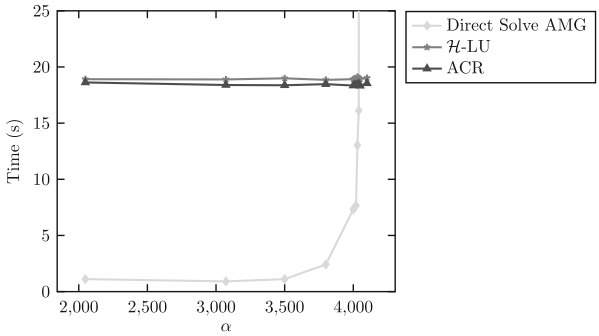

Fig. 3 Runtime versus velocity magnitude in convection-diffusion. AMG is the method of choice in the diffusion dominated limit, but loses robustness with skew-symmetry

The second test is convection-diffusion equation with recirculating flow.

$$\begin{aligned} -\nabla^2 u + \alpha b(\mathbf{x}) \cdot \nabla u &= f(\mathbf{x}), \qquad \mathbf{x} \in \Omega = [0,1]^2 \;\; u(\mathbf{x}) = 0, \; x \in \Gamma \\ b(\mathbf{x}) &= \begin{pmatrix} \sin(4\pi x)\sin(4\pi y) \\ \cos(4\pi x)\cos(4\pi y) \end{pmatrix} \qquad f(\mathbf{x}) = 100 e^{-100((x-0.5)^2+(y-0.5)^2)}. \end{aligned} \tag{9}$$

Discretization of this equation, again with $h = 2^{-10}$, leads to a nonsymmetric matrix, whose eigenvalues go complex (with central differencing) when the cell Peclet number exceeds 2. Direct algebraic methods are unaffected.

We progressively increase the convection dominance with α. For small α AMG outperforms the direct methods, but AMG is not robust with respect to the rising skew-symmetry. ACR maintains its performance for any α, as shown in Fig. 3.

5 Extensions

The discretization of 3D elliptic operators also leads to a block tridiagonal structure, with the difference that each block is of size $n^2 \times n^2$, instead of $n \times n$, as in the 2D discretization. A similar reduction strategy in the outermost dimension is possible, and leads to a solver with log-linear complexity in N and similar parallel structure, except that ranks grow.

The controllable accuracy feature of hierarchical matrices suggests the possibility of using ACR as a preconditioner, with rank becoming a tuning parameter balancing the cost per and the number of iterations, while preserving the rich concurrency features of the method.

6 Concluding Remarks

We present a fast direct solver, ACR, for structured sparse linear systems that arise from the discretization of 2D elliptic operators. The solver approximates every block using an $\mathcal{H}$-Matrix, resulting in a log-linear arithmetic complexity of $\mathcal{O}(N \log^2 N)$ with memory requirements of $\mathcal{O}(N \log N)$.

Robustness and applicability are demonstrated on model scalar problems and contrasted with established solvers based on the $\mathcal{H}$-LU factorization and algebraic multigrid. Multigrid maintains superiority in scalar problems with sufficient definiteness and symmetry, whereas hierarchical matrix-based replacements of direct methods tackle some problems where these properties are lacking. Although being of the same asymptotic complexity as $\mathcal{H}$-LU, ACR has fundamentally different algorithmic roots which produce a novel alternative for a relevant class of problems with competitive performance, and concurrency that grows with the problem size.

In Chávez et al. (2016) we expand on the consideration of cyclic reduction as a fast direct solver for 3D elliptic operators.

References

S. Ambikasaran, E. Darve, An $\mathcal{O}(N \log N)$ fast direct solver for partial hierarchically semiseparable matrices. J. Sci. Comput. **57**(3), 477–501 (2013)

P. Amestoy, A. Buttari, G. Joslin, J.-Y. L'Excellent, M. Sid-Lakhdar, C. Weisbecker, M. Forzan, C. Pozza, R. Perrin, V. Pellissier, Shared-memory parallelism and low-rank approximation techniques applied to direct solvers in FEM simulation. IEEE Trans. Mag. **50**(2), 517–520 (2014)

A. Aminfar, S. Ambikasaran, E. Darve, A fast block low-rank dense solver with applications to finite-element matrices. J. Comput. Phys. **304**, 170–188 (2016)

M. Bebendorf, *Hierarchical Matrices: A Means to Efficiently Solve Elliptic Boundary Value Problems*. Lecture Notes in Computational Science and Engineering, vol. 63 (Springer, Berlin, 2008)

B.L. Buzbee, G.H. Golub, C.W. Nielson, On direct methods for solving Poisson equation. SIAM J. Numer. Anal. **7**(4), 627–656 (1970)

S. Chandrasekaran, M. Gu, T. Pals, A fast *ULV* decomposition solver for hierarchically semiseparable representations. SIAM J. Matrix Anal. Appl. **28**(3), 603–622 (2006)

G. Chavez, G. Turkiyyah, S. Zampini, H. Ltaief, D. Keyes, Accelerated cyclic reduction: a distributed-memory fast direct solver for structured linear systems (2016). Preprint, arXiv:1701.00182

P. Ghysels, X.S. Li, F.-H. Rouet, S. Williams, A. Napov, An efficient multi-core implementation of a novel HSS-structured multifrontal solver using randomized sampling. arXiv:1502.07405 [cs.MS], pp. 1–26, 2015

L. Grasedyck, R. Kriemann, S. Le Borne, Domain decomposition based $\mathcal{H}$-LU preconditioning. Numer. Math. **112**(4), 565–600 (2009)

W. Hackbusch, A sparse matrix arithmetic based on $\mathcal{H}$-Matrices. Part I: introduction to $\mathcal{H}$-Matrices. Computing **62**(2), 89–108 (1999)

W. Hackbusch, S. Börm, L. Grasedyck, HLib 1.4. http://hlib.org, 1999–2012. Max-Planck-Institut, Leipzig

R.W. Hockney, A fast direct solution of Poisson's equation using Fourier analysis. J. ACM **12**(1), 95–113 (1965)

I. Ibragimov, S. Rjasanow, K. Straube, Hierarchical Cholesky decomposition of sparse matrices arising from curl-curl-equation. J. Numer. Math. **15**(1), 31–57 (2007)

R. Kriemann, Parallel $\mathcal{H}$-Matrix arithmetics on shared memory systems. Computing **74**(3), 273–297 (2005). ISSN=0010-485X

R.D. Falgout, U.M. Yang, Hypre: a library of high performance preconditioners, in *Computational Science - ICCS 2002: International Conference Amsterdam*, ed. by P.M. A. Sloot, A.G. Hoekstra, C.J.K. Tan, J.J. Dongarra. 2002 Proceedings, Part III (Springer, Berlin/Heidelberg, 2002), pp. 632–641. ISBN:978-3-540-47789-1

P.G. Schmitz, L. Ying, A fast direct solver for elliptic problems on general meshes in 2D. J. Comput. Phys. **231**(4), 1314–1338 (2012)

P.G. Schmitz, L. Ying, A fast nested dissection solver for Cartesian 3D elliptic problems using hierarchical matrices. J. Comput. Phys. **258**, 227–245 (2014)

P. Swarztrauber, The methods of cyclic reduction, Fourier analysis and the FACR algorithm for the discrete solution of Poisson equation on a rectangle. SIAM Rev. **19**(3), 490–501 (1977)

J. Xia, S. Chandrasekaran, M. Gu, X. Li, Superfast multifrontal method for large structured linear systems of equations. SIAM J. Matrix Anal. Appl. **31**(3), 1382–1411 (2010)

J. Xia, S. Chandrasekaran, M. Gu, X.S. Li, Fast algorithms for hierarchically semiseparable matrices. Numer. Lin. Alg. Appl. **17**(6), 953–976 (2010)

J. Xia, M. Gu, Robust approximate Cholesky factorization of rank-structured symmetric positive definite matrices. SIAM J. Matrix Anal. Appl. **31**(5), 2899–2920 (2010)