Cyclic Reduction – History and Applications

Walter Gander and Gene H. Golub

ETH Zurich and Stanford University

Abstract. We discuss the method of Cyclic Reduction for solving special systems of linear equations that arise when discretizing partial differential equations. In connection with parallel computations the method has become very important.

This paper has been published in the Proceedings of the Workshop on Scientific Computing : 10-12 March, 1997, Hong Kong, editor-in-chief, Gene Howard Golub ; managing editor, Shiu Hong Liu ; editors, Franklin T. Luk, Robert James Plemmons. Springer Verlag, New York, 1997.

1 Introduction

Cyclic Reduction has proved to be an algorithm which is very powerful for solving structured matrix problems. In particular for matrices which are (block) Toeplitz and (block) tri-diagonal, the method is especially useful. The basic idea is to eliminate half the unknowns, regroup the equations and again eliminate half the unknowns. The process is continued *ad nauseum.* This simple idea is useful in solving the finite difference approximation to Poisson's equation in a rectangle and for solving certain recurrences. The algorithm easily parallelizes and can be used on a large variety of architectures [7]. New uses of Cyclic Reduction continue to be developed – see, for instance, the recent publication by Amodio and Paprzycki [1].

In this paper, we give a historical treatment of the algorithm and show some of the important applications. Our aim has not been to be exhaustive, rather we wish to give a broad view of the method.

In Section 2, we describe the algorithm for a general tri-diagonal matrix and a block tri-diagonal matrix. We show how the process need not be completed. Section 3 describes the method for solving Poisson's equation in two dimensions. We give details of the stabilized procedure developed by O. Buneman. We also describe a technique developed by Sweet [11] for making the method more useful in a parallel environment. The final Section describes the influence of the method on developing more general procedures.

2 Classical Algorithm

2.1 Scalar Cyclic Reduction

Consider the tridiagonal linear system $Ax = v$ where

$$A = \begin{pmatrix} d_1 & f_1 & & & \\ e_2 & d_2 & f_2 & & \\ & \ddots & \ddots & \ddots & \\ & & e_{n-1} & d_{n-1} & f_{n-1} \\ & & & e_n & d_n \end{pmatrix}. \tag{1}$$

The fundamental operation in Cyclic Reduction is the *simultaneous elimination of odd-indexed unknowns*. This operation may be described as a block LU-factorization of a permutation of the rows and columns of the matrix A. Let S be the permutation matrix such that

$$S(1,2,\ldots,n)^T = (1,3,\ldots,|2,4,\ldots)^T$$

The permuted matrix SAS^T becomes:

$$SAS^T = \left[\begin{array}{ccc|ccc} d_1 & & & f_1 & & \\ & d_3 & & e_3 & f_3 & \\ & & \ddots & & \ddots & \ddots \\ \hline e_2 & f_2 & & d_2 & & \\ & e_4 & \ddots & & d_4 & \\ & & \ddots & & & \ddots \end{array}\right].$$

Eliminating the odd-indexed unknowns is equivalent to computing a *partial LU-factorization* [2]; viz

$$SAS^T = \left[\begin{array}{ccc|ccc} 1 & & & 0 & & \\ & \ddots & & & \ddots & \\ & & 1 & & & 0 \\ \hline l_1 & m_1 & & 1 & & \\ & l_2 & \ddots & & \ddots & \\ & & \ddots & & & 1 \end{array}\right] \left[\begin{array}{ccc|ccc} d_1 & & & f_1 & & \\ & d_3 & & e_3 & f_3 & \\ & & \ddots & & \ddots & \ddots \\ \hline 0 & & & d_1^{(1)} & f_1^{(1)} & \\ & \ddots & & e_2^{(1)} & d_2^{(1)} & \ddots \\ & & 0 & & \ddots & \ddots \end{array}\right]. \tag{2}$$

For n even, this decomposition is computed for $i = 1, \ldots, \frac{n}{2} - 1$ as

$$m_i = f_{2i}/d_{2i+1}, \quad l_i = e_{2i}/d_{2i-1}, \quad l_{n/2} = e_n/d_{n-1}$$

$$f_i^{(1)} = -m_i f_{2i+1}, \quad e_{i+1}^{(1)} = -l_{i+1}e_{2i+1}$$

$$d_i^{(1)} = d_{2i} - l_i f_{2i-1} - m_i e_{2i+i}, \quad d_{n/2}^{(1)} = d_n - l_{n/2} f_{n-1}$$

Since the Schur complement is tridiagonal we may iterate and again eliminate the odd-indexed unknowns of the reduced system. G. Golub recognized (cf. [10]) that Cyclic Reduction is therefore equivalent to Gaussian elimination without pivoting on a permuted system $(PAP^T)(Px) = Pv$. The permutation matrix P reorders the vector $(1, 2, \dots, n)$ such that the odd multiples of 2^0 come first, followed by the odd multiples of 2^1, the odd multiples of 2^2, etc.; e.g. for $n = 7$ we get the permuted system

$$\begin{pmatrix} d_1 & & & & f_1 & & \\ & d_3 & & & e_3 & & f_3 \\ & & d_5 & & & f_5 & e_5 \\ & & & d_7 & & e_7 & \\ e_2 & f_2 & & & d_2 & & \\ & & e_6 & f_6 & & d_6 & \\ & e_4 & f_4 & & & & d_4 \end{pmatrix} \begin{pmatrix} x_1 \\ x_3 \\ x_5 \\ x_7 \\ x_2 \\ x_6 \\ x_4 \end{pmatrix} = \begin{pmatrix} v_1 \\ v_3 \\ v_5 \\ v_7 \\ v_2 \\ v_6 \\ v_4 \end{pmatrix}. \tag{3}$$

The connection to Gaussian elimination allows us to conclude that if elimination with diagonal pivots is stable then Cyclic Reduction will also be stable. If A is strictly diagonal dominant or symmetric positive definite then this holds for PAP^T. No pivoting is necessary in these cases and therefore Cyclic Reduction is well defined and stable. However, fill-in is generated in the process and this increases the operation count. Cyclic Reduction requires about 2.7 times more operations than Gaussian elimination and thus has a *redundancy* of 2.7 [2].

2.2 Block Cyclic Reduction

We use the notation of Heller [8] and consider a block tridiagonal linear system $Ax = v$, i.e.

$$E_j x_{j-1} + D_j x_j + F_j x_{j+1} = v_j, \quad j = 1, \dots, n, \quad E_1 = F_n = 0. \tag{4}$$

where x_j and $v_j \in \mathbf{R}^m$ and the blocks are $m \times m$ matrices E_j, D_j and F_j so that the dimension of the matrix A is nm.

Consider three consecutive equations which we arrange in a tableau:

$Eq\#$	x_{2j-2}	x_{2j-1}	x_{2j}	x_{2j+1}	x_{2j+2}	rhs
$2j-1$	E_{2j-1}	D_{2j-1}	F_{2j-1}			v_{2j-1}
$2j$		E_{2j}	D_{2j}	F_{2j}		v_{2j}
$2j+1$			E_{2j+1}	D_{2j+1}	F_{2j+1}	v_{2j+1}

In order to eliminate x_{2j-1} and x_{2j+1} we multiply equation $\#(2j-1)$ with $-E_{2j}(D_{2j-1})^{-1}$, equation $\#(2j+1)$ with $-F_{2j}(D_{2j+1})^{-1}$ and add these to equation $\#(2j)$. The result is a new equation

$Eq\#$	x_{2j-2}	x_{2j-1}	x_{2j}	x_{2j+1}	x_{2j+2}	rhs
j	E'_j	0	D'_j	0	F'_j	v'_j

where

$$\begin{aligned}
E'_j &= -E_{2j}(D_{2j-1})^{-1}E_{2j-1} && (5)\\
D'_j &= D_{2j} - E_{2j}(D_{2j-1})^{-1}F_{2j-1} - F_{2j}(D_{2j+1})^{-1}E_{2j+1} && (6)\\
F'_j &= -F_{2j}(D_{2j+1})^{-1}F_{2j+1} && (7)\\
v'_j &= v_{2j} - E_{2j}(D_{2j-1})^{-1}v_{2j-1} - F_{2j}(D_{2j+1})^{-1}v_{2j+1}. && (8)
\end{aligned}$$

Eliminating in this way the odd indexed unknowns, we obtain a block tridiagonal reduced system. The elimination can of course only be performed if the diagonal block matrices are non-singular. Instead of inverting these diagonal blocks (or better solving the corresponding linear systems) it is sometimes possible to eliminate the odd unknowns by matrix multiplications: Multiply equation #$(2j-1)$ with $E_{2j}D_{2j+1}$, equation #$(2j)$ with $D_{2j-1}D_{2j+1}$, equation #$(2j+1)$ with $F_{2j}D_{2j-1}$ and add. If

$$\begin{aligned}
D_{2j-1}D_{2j+1}E_{2j} &= E_{2j}D_{2j+1}D_{2j-1} \quad \text{and}\\
D_{2j-1}D_{2j+1}F_{2j} &= F_{2j}D_{2j-1}D_{2j+1},
\end{aligned}$$

then the matrices of the unknowns with odd indices vanish. However, if the matrices do not commute then we cannot eliminate by matrix multiplications. In case of the discretized Poisson equation (20) the matrices do commute.

Eliminating the unknowns with odd indices thus results in a new tridiagonal system for the even unknowns. The new system is half the size of the original system. We can iterate the procedure and eliminate again the odd unknowns in the new system.

2.3 Divide and Conquer Algorithm

If we reorder Equation (4) for the first step of Cyclic Reduction by separating the unknowns with odd and even indices we obtain:

$$\begin{pmatrix} D_1 & F \\ G & D_2 \end{pmatrix}\begin{pmatrix} x_o \\ x_e \end{pmatrix} = \begin{pmatrix} v_o \\ v_e \end{pmatrix} \tag{9}$$

where D_1 and D_2 are diagonal block-matrices and F and G are block-bidiagonal. Eliminating the unknowns with odd indices $x_o = (x_1, x_3, \ldots)^T$ means computing the reduced system of equations for the unknowns with even indices $x_e = (x_2, x_4, \ldots)^T$ by forming the Schur complement:

$$(D_2 - GD_1^{-1}F)x_e = v_e - GD_1^{-1}v_o. \tag{10}$$

It is also possible to eliminate the unknowns with even indices, thus forming the equivalent system:

$$\begin{pmatrix} D_1 - FD_2^{-1}G & 0 \\ 0 & D_2 - GD_1^{-1}F \end{pmatrix}\begin{pmatrix} x_o \\ x_e \end{pmatrix} = \begin{pmatrix} v_o - FD_2^{-1}v_e \\ v_e - GD_1^{-1}v_o \end{pmatrix}. \tag{11}$$

Notice that by this operation the system (11) splits in two independent block-tridiagonal systems for the two sets of unknowns x_o and x_e. They can be solved independently and this process can be seen as the first step of a divide and conquer algorithm. The two smaller systems can be split again in the same way in two subsystems which can be solved independently on different processors.

2.4 Incomplete Cyclic Reduction

Cyclic Reduction reduces $Ax = v$ in k steps to $A^{(k)}x^{(k)} = v^{(k)}$. At any level k it is possible to solve the reduced system and compute by back-substitution the solution $x = x^{(0)}$.

Hockney [9] already noticed that often the off-diagonal block-elements of the reduced system decrease in size relatively to the diagonal elements at a quadratic rate. Therefore it is possible to terminate the reduction phase earlier, neglect the off diagonal elements and solve approximately the reduced system: $y^{(k)} \approx x^{(k)}$. Back-substitution with $y^{(k)}$ yields $y \approx x$. This procedure is called *incomplete Cyclic Reduction.*

Considering the block-tridiagonal matrices

$$B^{(k)} = \left(-(D_j^{(k)})^{-1}E_j^{(k)}, 0, -(D_j^{(k)})^{-1}F_j^{(k)}\right),$$

Heller [8] proves that for diagonally dominant systems i.e. if $\|B^{(0)}\|_\infty < 1$, then

1. Cyclic Reduction is well defined and $\|x - y\|_\infty = \|x^{(k)} - y^{(k)}\|_\infty$
2. $\|B^{(i+1)}\|_\infty \leq \|B^{(i)}\|_\infty^2$
3. $\|x^{(k)} - y^{(k)}\|_\infty / \|x^{(k)}\|_\infty \leq \|B^{(k)}\|_\infty$.

S. Bondeli and W. Gander [3] considered scalar incomplete Cyclic Reduction for the the special $n \times n$ tridiagonal system $Ax = v$

$$\begin{pmatrix} a & 1 & & & \\ 1 & a & 1 & & \\ & 1 & \ddots & \ddots & \\ & & \ddots & a & 1 \\ & & & 1 & a \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{pmatrix} = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_{n-1} \\ v_n \end{pmatrix}. \tag{12}$$

Such systems have to be solved in the Fourier algorithm discussed in Section 3.4. Cyclic Reduction reduces this system in step k to $A^{(k)}x^{(k)} = v^{(k)}$ where

$$A^{(k)} = \begin{pmatrix} a_k & b_k & & & \\ b_k & a_k & b_k & & \\ & b_k & \ddots & \ddots & \\ & & \ddots & a_k & b_k \\ & & & b_k & a_k \end{pmatrix} . \tag{13}$$

As shown in [3], it is also possible to obtain an explicit expression for a_k:

1. If $|a| = 2$ then the diagonal and off-diagonal elements are

$$b_k = -\operatorname{sign}(a)/2^k \tag{14}$$
$$a_k = \quad \operatorname{sign}(a)/2^{k-1} . \tag{15}$$

2. If $|a|>2$ the diagonal elements produced by the reduction phase are given by

$$a_k = \operatorname{sign}(a)\sqrt{a^2-4}\ \coth\left(|y_0|2^k\right) , \tag{16}$$

where

$$y_0 = \begin{cases} \ln\left(\sqrt{a^2-4}+a\right) - \ln 2 & \text{if } a > 2 \\ -\ln\left(\sqrt{a^2-4}-a\right) + \ln 2 & \text{if } a < -2 . \end{cases}$$

While $b_k \to 0$ the sequence a_k converges quadratically to

$$\operatorname{sign}(a)\sqrt{a^2-4}, \tag{17}$$

and it is possible to take advantage of that fact. One can predict, depending on a and the machine precision, the number of Cyclic Reduction-steps until a_k may be assumed to be constant. E.g. for $a = 2.5$ the 14 leading decimal digits of a_k are constant for for $k \geq 5$ [3]. Thus we can apply incomplete Cyclic Reduction to solve the special system. Furthermore by making use of the explicit expressions for a_k the computation can be vectorized.

3 Poisson's Equation

We will now consider a special block tridiagonal system of equations that arises when discretizing Poisson's equation

$$\Delta u(x, y) = f(x, y) \tag{18}$$

on some rectangular domain defined by $a < x < b$, $c < y < d$. Let

$$\Delta x = \frac{b-a}{m+1}, \quad \Delta y = \frac{d-c}{n+1} \tag{19}$$

be the step sizes of the grid and denote by u_{ij} the approximation for the function value $u(x_i, y_j)$ at $x_i = a + i\Delta x$ and $y_j = c + j\Delta y$. For simplicity, we shall assume hereafter that $\Delta x = \Delta y$. If we use central difference approximations for the Laplace operator Δ and if we assume Dirichlet zero boundary conditions then we obtain the linear system

$$Tu = g \tag{20}$$

with a block tridiagonal matrix

$$T = \begin{pmatrix} A & I & & \\ I & A & \ddots & \\ & \ddots & \ddots & I \\ & & I & A \end{pmatrix} \in \mathbf{R}^{nm \times nm}, \tag{21}$$

$I \in \mathbf{R}^{m \times m}$ is the identity matrix and

$$A = \begin{pmatrix} -4 & 1 & & \\ 1 & -4 & \ddots & \\ & \ddots & \ddots & 1 \\ & & 1 & -4 \end{pmatrix} \in \mathbf{R}^{m \times m}. \tag{22}$$

The unknown vector u contains the rows of the matrix u_{ij} i.e.

$$u = \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{pmatrix} \in \mathbf{R}^{nm} \quad \text{where} \quad u_j = \begin{pmatrix} u_{1j} \\ u_{2j} \\ \vdots \\ u_{mj} \end{pmatrix} \in \mathbf{R}^{m}. \tag{23}$$

The right hand side is partitioned similarly:

$$g_{ij} = (\Delta y)^2 f(x_i, y_j), \quad g_j = \begin{pmatrix} g_{1j} \\ g_{2j} \\ \vdots \\ g_{mj} \end{pmatrix} \in \mathbf{R}^{m} \quad \text{and} \quad g = \begin{pmatrix} g_1 \\ g_2 \\ \vdots \\ g_m \end{pmatrix} \in \mathbf{R}^{nm}.$$

3.1 Cyclic Reduction with Matrix Multiplications

Elimination of the unknowns with odd indices can be done by matrix multiplications since for the system (20) the matrices $D_j = A$ and $F_j = E_j = I$ do commute. As pointed out in [5], the block matrices of the system of equations one obtains when discretizing Poisson's equation in a rectangle using the nine-point formula do also commute. Thus in this case elimination can also be done with matrix multiplication.

Assume $n = 2^{k+1} - 1$ and consider again three consecutive equations

$Eq\#$	u_{2j-2}	u_{2j-1}	u_{2j}	u_{2j+1}	u_{2j+2}	rhs
$2j-1$	I	A	I			g_{2j-1}
$2j$		I	A	I		g_{2j}
$2j+1$			I	A	I	$g_{2j+1}.$

In order to eliminate u_{2j-1} and u_{2j+1} we multiply equation $\#(2j)$ with $-A$ and add all three equations. The result is the new equation

$Eq\#$	u_{2j-2}	u_{2j-1}	u_{2j}	u_{2j+1}	u_{2j+2}	rhs
j	I	0	$2I - A^2$	0	I	$g_{2j-1} - Ag_{2j} + g_{2j+1}.$

The recurrence relations corresponding to (5) are simpler. With $A^{(0)} = A$ and $g_j^{(0)} = g_j$ the r-th reduction step is described by

$$A^{(r+1)} = 2I - (A^{(r)})^2 \tag{24}$$

$$g_j^{(r+1)} = g_{2j-1}^{(r)} - A^{(r)} g_{2j}^{(r)} + g_{2j+1}^{(r)}, \quad j = 1, \dots, 2^{k+1-r} - 1. \tag{25}$$

Thus after r reduction steps the remaining system of equation has the size $(2^{k+1-r} - 1) \times (2^{k+1-r-1})$:

$$u_{(j-1)2^r} + A^{(r)} u_{j2^r} + u_{(j+1)j2^r} = g_j^{(r)}, \quad j = 1, \dots, 2^{k+1-r} - 1. \tag{26}$$

(Notice that in (26) we have assumed that $u_0 = u_{n+1} = 0$).

After k reduction steps we are left with one block $m \times m$ system

$$A^{(k)} u_{2^k} = g_1^{(k)}. \tag{27}$$

After determining u_{2^k} back-substitution is performed in which Equation (26) is solved for u_{j2^k} while $u_{(j-1)2^r}$ and $u_{(j+1)j2^r}$ are known from the previous level.

For Equation (27) and in the back-substitution phase, linear equations with the matrices $A^{(r)}$ must be solved. Furthermore transforming the right hand side (25) needs matrix-vector multiplications with $A^{(r)}$. We show in the next section how these operations can be executed efficiently.

As shown in [5] block-Cyclic Reduction can also for be applied for matrices generated by periodic boundary conditions, i.e. when the $(1, n)$ and the $(n, 1)$ block of T (20) is non-zero.

Unfortunately as mentioned in [5] the process described above is numerically unstable.

3.2 Computational Simplifications

From (24) we notice that $A^{(r)} = P_{2^r}(A)$ is a polynomial in A of degreee 2^r which as we will see can be factorized explicitly. Thus $A^{(r)}$ does not need to be computed numerically – the operator $A^{(r)}$ will be stored in factorized form. From (24) it follows that the polynomials satisfy the recurrence relation:

$$P_1(x) = x, \quad P_{2^{r+1}}(x) = 2 - P_{2^r}^2(x).$$

It is interesting that the polynomials P_{2^r} have a strong connection to the Chebychev polynomials $T_n(t)$ which are defined by expanding $\cos n\varphi$ in powers of $\cos\varphi$:

$$T_n(\cos\varphi) = \cos n\varphi.$$

Using the well known trigonometric identity

$$\cos(k+l)\varphi + \cos(k-l)\varphi = 2\cos l\varphi \cos k\varphi \tag{28}$$

we obtain for $l = 1$ the famous three term recurrence relation ($t = \cos\varphi$)

$$T_{k+1}(t) = 2tT_k(t) - T_{k-1}(t). \tag{29}$$

More generally, relation (28) translates to the "short cut" recurrence

$$T_{k+l}(t) = 2T_l(t)T_k(t) - T_{k-l}(t).$$

Specializing by choosing $k = l$ we obtain

$$T_{2k}(t) = 2T_k^2(t) - T_0(t) = 2T_k^2(t) - 1 \tag{30}$$

almost the same recurrence as for P_{2^r}. Multiplying (30) by -2 and substituting $k = 2^r$, we obtain

$$-2T_{2^{r+1}}(t) = 2 - (2T_{2^r}(t))^2 = 2 - (-2T_{2^r}(t))^2.$$

Thus the polynomials $-2T_{2^r}(t)$ obey the same recurrence relation as P_{2^r}. However, since $P_1(x) = x$ and $-2T_1(t) = -2t$ we conclude that $x = -2t$ and therefore

$$P_{2^r}(x) = -2T_{2^r}\left(-\frac{x}{2}\right), \quad r \geq 0. \tag{31}$$

The properties of the Chebychev polynomials are well known and by (31), we can translate them to $P_{2^r}(x)$. The zeros of P_{2^r} are

$$\lambda_i^{(k)} = -2\cos\left(\frac{2i-1}{2^{r+1}}\pi\right), \quad i = 1, 2, \ldots, 2^r$$

and since for $r \geq 1$ the leading coefficient of $P_{2^r}(x)$ is -1, the factorization is

$$P_{2^r}(x) = -\prod_{i=1}^{2^r}(x - \lambda_i^{(k)}), \quad \text{thus} \quad A^{(r)} = -\prod_{i=1}^{2^r}(A - \lambda_i^{(k)} I). \tag{32}$$

Multiplication. To compute $u = A^{(r)}v = P_{2^r}(A)v$ we have now several possibilities:

1. Using (24) we could generate $A^{(r)}$ explicitly. Notice that A is tridiagonal but $A^{(r)}$ will soon be a full matrix with very large elements.
2. Using the recurrence relation for Chebychev polynomials (29) we obtain for $t = -x/2$ using (31) a three term recurrence relation for $P_n(x)$
$$P_{n+1}(x) = -xP_n(x) - P_{n-1}(x).$$
Now we can generate a sequence of vectors $z_i = P_i(A)v$ starting with $z_0 = -2v$ and $z_1 = Av$ by
$$z_i = -Az_{i-1} - z_{i-2}, \quad i = 2, 3, \ldots, 2^r.$$
Then $u = z_{2^r} = P_{2^r}(A)v = A^{(r)}v$. In [5] it is proved that the resulting algorithm *cyclic odd-even reduction and factorization* (CORF) based on this procedure is numerically unstable.
3. Using the product of linear factors (32) we compute $z_0 = -v$ and
$$z_i = (A - \lambda_i^{(k)}I)z_{i-1}, \quad i = 1, 2, \ldots, 2^r$$
and obtain $u = z_{2^r}$. This seems to be the best way to go.

Solving Block Equations. To solve $A^{(r)}v = u$ for v we have also several possibilities:

1. Generate $A^{(r)}$ explicitly and use Gaussian elimination. This simple approach is not a good idea since $A^{(r)}$ will be an ill-conditioned full matrix with large elements [5].
2. Using again the product of linear factors (32) we initialize $z_0 = -u$, solve *sequentially* the tridiagonal systems
$$(A - \lambda_i^{(k)}I)z_i = z_{i-1}, \quad i = 1, 2, \ldots, 2^r \tag{33}$$
and obtain $v = z_{2^r}$.
3. A *parallel* algorithm was proposed using the following idea of R. Sweet [11] using a partial fraction expansion. Since the zeros of $P_{2^r}(x)$ are all simple the partial fraction expansion of the reciprocal value is
$$\frac{1}{P_{2^r}(x)} = -\frac{1}{\prod_{i=1}^{2^r}(x - \lambda_i^{(k)})} = \sum_{i=1}^{2^r} \frac{c_i^{(r)}}{x - \lambda_i^{(k)}},$$
where
$$c_i^{(r)} = \frac{1}{P'_{2^r}(\lambda_i^{(k)})} = -\prod_{\substack{j=1 \\ j\neq i}}^{2^r} \left(\lambda_j^{(k)} - \lambda_i^{(k)}\right)^{-1}.$$

Therefore we can express

$$v = \left(A^{(r)}\right)^{-1} u = \left(P_{2^r}(A)\right)^{-1} u = \sum_{i=1}^{2^r} c_i^{(r)} (A - \lambda_i^{(k)} I)^{-1} u$$

where now the 2^r linear systems $(A - \lambda_i^{(k)} I)^{-1} u$ in the sum may be computed independently and in parallel. As discussed by Calvetti *et al.* in [6], partial fraction expansion may be more sensitive to roundoff errors in the presence of close poles.

3.3 Buneman's Algorithm

It has been observed that block Cyclic Reduction using the recurrence relation (25)

$$g_j^{(r+1)} = g_{2j-1}^{(r)} - A^{(r)} g_{2j}^{(r)} + g_{2j+1}^{(r)}, \quad j = 1, \ldots, 2^{k+1-r} - 1$$

to update the right hand side is numerically unstable. O. Buneman [4,5] managed to stabilize the algorithm by rearranging the computation of the right hand side in a clever way.

In his approach the right hand side is represented as

$$g_j^{(r)} = A^{(r)} p_j^{(r)} + q_j^{(r)}$$

and the vectors $p_j^{(r)}$ and $q_j^{(r)}$ are computed recursively in the following way. Initialize $p_j^{(0)} = 0$ and $q_j^{(0)} = g_j$. Then for $j = 1, \ldots, 2^{k+1-r} - 1$

1. Solve $A^{(r-1)} v = p_{2j-1}^{(r-1)} + p_{2j+1}^{(r-1)} - q_j^{(r-1)}$ for v
2. $p_j^{(r)} = p_j^{(r-1)} - v$
3. $q_j^{(r)} = q_{2j-1}^{(r-1)} + q_{2j+1}^{(r-1)} - 2p_j^{(r)}$.

There is also a simplification in the back-substituting phase. In order to compute u_{j2^r} from

$$u_{(j-1)2^r} + A^{(r)} u_{j2^r} + u_{(j+1)j2^r} = A^{(r)} p_j^{(r)} + q_j^{(r)}$$

we rearrange the equation to

$$A^{(r)} \underbrace{(u_{j2^r} - p_j^{(r)})}_{v} = q_j^{(r)} - u_{(j-1)2^r} - u_{(j+1)j2^r}$$

solve for v and obtain $u_{j2^r} = p_j^{(r)} + v$.

3.4 Fourier Algorithm

Using the eigen-decomposition of the matrix A it is possible to transform the block tridiagonal system (20) to m independent tridiagonal linear $n \times n$ systems of the form (12). These systems can then be solved e.g. in parallel by incomplete Cyclic Reduction.

The key for this procedure is the fact that the eigenvalues and -vectors of the matrix A (22) can be computed explicitly. It is well known that

$$B = \begin{pmatrix} 2 & -1 & & \\ -1 & 2 & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{pmatrix} \in \mathbf{R}^{m\times m}$$

has the eigenvalues λ_i eigenvectors Q with

$$q_{ij} = \sqrt{\frac{2}{m+1}} \sin \frac{ij\pi}{m+1}, \quad \lambda_i = 2 - 2\cos\frac{i\pi}{m+1}. \tag{34}$$

This result may be generalized to matrices of the form

$$C = \begin{pmatrix} a & b & & \\ b & a & \ddots & \\ & \ddots & \ddots & b \\ & & b & a \end{pmatrix}.$$

Since $C = -bB + (a+2b)I$ we conclude that the eigenvectors are the same and that the eigenvalues of C are $\tilde{\lambda}_i = -b\lambda_i + a + 2b$. The eigenvalues of A are obtained for $b = 1$ and $a = -4$.

Thus we can decompose the matrix A (22) as $A = Q\Lambda Q^T$. Introducing this decomposition in the system (20) and multiplying each block equation from the left by Q^T we obtain

$$Q^T u_{j-1} + \Lambda Q^T u_j + Q^T u_{j+1} = Q^T g_j, \quad j = 1, \ldots, n.$$

If we introduce new variables $\hat{u}_j = Q^T u_j$ the system (20) becomes

$$\hat{u}_{j-1} + \Lambda \hat{u}_j + \hat{u}_{j+1} = \hat{g}_j, \quad j = 1, \ldots, n.$$

Finally by permuting the unknowns by grouping together the components with the same index of $\hat{u}_j$

$$\tilde{u}_j = (\hat{u}_{i,1}, \hat{u}_{i2}, \ldots, \hat{u}_{in})^T$$

and by permuting the equations in the same way, the system (20) is transformed into m decoupled tridiagonal systems of equations

$$\begin{pmatrix} T_1 & & & \\ & T_2 & & \\ & & \ddots & \\ & & & T_m \end{pmatrix} \begin{pmatrix} \tilde{u}_1 \\ \tilde{u}_2 \\ \vdots \\ \tilde{u}_m \end{pmatrix} = \begin{pmatrix} \tilde{g}_1 \\ \tilde{g}_2 \\ \vdots \\ \tilde{g}_m \end{pmatrix} \tag{35}$$

$$T_i = \begin{pmatrix} \lambda_i & 1 & & \\ 1 & \lambda_i & \ddots & \\ & \ddots & \ddots & 1 \\ & & 1 & \lambda_i \end{pmatrix} \in \mathbf{R}^n.$$

Summarizing we obtain the following algorithm to solve Equation (20):

1. Compute the eigen-decomposition $A = Q\Lambda Q^T$ using the explicit expression (34).
2. Transform the right hand side $\hat{g}_j = Q^T g_j$, $j = 1, \ldots, n$.
3. Permute equations and unknowns and solve the m decoupled equations (35) in parallel.
4. Compute the solution by the back-transformation $u_j = Q\hat{u}_j$.

Notice that for the transformations in the second and forth step the fast Fourier transform may be used [9].

For solving the special decoupled linear systems we have the choice of several methods. First they may be solved efficiently using Gaussian elimination in parallel on different processors. Because of the special structure $[1, \lambda_k, 1]$ of the tridiagonal matrix the elements of the LU decomposition converge and may be assigned their limit [3].

A second possibility is to compute the eigen-decomposition of $[1, \lambda_k, 1] = \tilde{Q}D_k\tilde{Q}^T$. The eigenvectors are the same for all the decoupled systems, so only one decomposition has to be computed. However, this method is more expensive than Gaussian elimination. To solve one system by applying $\tilde{Q}D_k^{-1}\tilde{Q}^T$ to the right hand side we need $O(n^2)$ operations while Gaussian elimination only needs $O(n)$.

Finally we may solve these special systems in parallel using the incomplete Cyclic Reduction described in Section 2.4. The operation count is higher than for Gaussian elimination but still of $O(n)$. The advantage of this method is that one can make use of a vector arithmetic unit if it is available on a single processor [3].

4 Conclusions

Cyclic Reduction was originally proposed by Gene Golub as a very simple recursive algorithm to solve tridiagonal linear systems. This algorithm turned out to be extremely useful and efficient for modern computer architectures (vector- and parallel processing).

Applied and specialized to the block tridiagonal linear system that is obtained by discretizing Poisson's equation, it became the key for the development of *Fast Poisson Solvers*. Fast Poisson solvers have, in addition, stimulated very much the ideas and development of *Domain Decomposition* and embedding techniques.

References

1. Amodio, P., Paprzycki, M.: A Cyclic Reduction Approach to the Numerical Solution of Boundary Value ODEs. SIAM J. Sci. Comput. Vol. **18**, No. 1, (1997) 56–68
2. Arbenz, P., Hegland M.: The Stable Parallel Solution of General Banded Linear Systems. Technical Report #252, Computer Science Dept. ETH Zürich, 1996.
3. Bondeli, S., Gander, W.: Cyclic Reduction for Special Tridiagonal Matrices. SIAM J. for Matrix Analysis Vol. **15** (1994)
4. Buneman, O.: A Compact Non-iterative Poisson Solver. Rep. 294, Stanford University, Institute for Plasma Research, Stanford, Calif. (1969)
5. Buzbee, B. L., Golub, G. H., Nielson, C. W.: On Direct Methods for Solving Poisson's Equations. SIAM J. Numerical Analysis, **7** (4) (1970) 627–656
6. Calvetti, D., Gallopoulos, E., Reichel, L.: Incomplete partial fractions for parallel evaluation of rational matrix functions. Journal of Computational and Applied Mathematics **59** (1995) 349–380
7. Golub, G. H., Ortega, J. M.: Scientific Computing. An Introduction with Parallel Computing. Academic Press, Inc., (1993)
8. Heller, D.: Some Aspects of the Cyclic Reduction Algorithm for Block Tridiagonal Linear Systems. SIAM J. Numer. Anal., **13** (4) (1976) 484–496
9. Hockney, R.W.: A Fast Direct Solution of Poisson's Equation Using Fourier Analysis. J. Assoc. Comput. Mach. **12** (1965) 95–113
10. Lambiotte, J., Voigt, R.: The Solution of Tridiagonal Linear Systems on the CDC Star100 Computer. ACM Trans. Math. Softw. **1** (1975) 308–329
11. Sweet, R. A.: , A Parallel and Vector Variant of the Cyclic Reduction Algorithm. SIAM J. Sci. and Statist. Computing **9** (4) (1988) 761–765