

**A combination of cyclic reduction and  
Kronecker product methods for the  
direct solution of Poisson's equation**

*Jef Hendrickx  
Marc Van Barel*

*Report TW 313, September 2000*



**Katholieke Universiteit Leuven**  
Department of Computer Science  
Celestijnenlaan 200A – B-3001 Heverlee (Belgium)

# A combination of cyclic reduction and Kronecker product methods for the direct solution of Poisson's equation

*Jef Hendrickx*  
*Marc Van Barel*

*Report TW 313, September 2000*

Department of Computer Science, K.U.Leuven

## Abstract

We present a fast direct method for the solution of a linear system  $M\mathbf{x} = \mathbf{y}$ , where  $M$  is a block tridiagonal Toeplitzmatrix with  $A$  on the diagonal and  $T$  on the two subdiagonals ( $A$  and  $T$  commute). Such matrices are obtained from a finite difference approximation to Poisson's equation (among others).

The new method is called KPCR( $l$ )-method and begins with  $l$  steps of cyclic reduction after which the remaining system is solved by a Kronecker product method. For an appropriate choice of  $l$  the asymptotic operation count for an  $n \times n$  grid is  $O(n^2 \log_2 \log_2 n)$ , which is faster than either cyclic reduction or the Kronecker product method itself. The algorithm is similar to and has the same complexity as the FACR( $l$ )-algorithm, which is a combination of cyclic reduction and Fourier analysis (or matrix decomposition). However, the FACR( $l$ )-algorithm only reaches this complexity if  $A$  (and  $T$ ) can be diagonalized by a fast transformation, where the new method is fast for every banded  $A$  and  $T$ . Moreover, the KPCR( $l$ )-method can be easily generalized to the case where  $A$  and  $T$  do not commute.

**Keywords :** Poisson equation, direct methods, cyclic reduction, Kronecker product, Fourier analysis, FACR, KPCR

**AMS(MOS) Classification :** Primary : 65F05, Secondary : 65F50, 65N06.

# A combination of cyclic reduction and Kronecker product methods for the direct solution of Poisson's equation<sup>3,4</sup>

Jef Hendrickx<sup>1</sup>, Marc Van Barel<sup>2</sup>

*Department of Computer Science, Katholieke Universiteit Leuven, Celestijnenlaan  
200 A, B-3001 Leuven, Belgium*

---

## Abstract

We present a fast direct method for the solution of a linear system  $M\mathbf{x} = \mathbf{y}$ , where  $M$  is a block tridiagonal Toeplitzmatrix with  $A$  on the diagonal and  $T$  on the two subdiagonals ( $A$  and  $T$  commute). Such matrices are obtained from a finite difference approximation to Poisson's equation (among others).

The new method is called KPCR( $l$ )-method and begins with  $l$  steps of cyclic reduction after which the remaining system is solved by a Kronecker product method. For an appropriate choice of  $l$  the asymptotic operation count for an  $n \times n$  grid is  $O(n^2 \log_2 \log_2 n)$ , which is faster than either cyclic reduction or the Kronecker product method itself. The algorithm is similar to and has the same complexity as the FACR( $l$ )-algorithm, which is a combination of cyclic reduction and Fourier analysis (or matrix decomposition). However, the FACR( $l$ )-algorithm only reaches this complexity if  $A$  (and  $T$ ) can be diagonalized by a fast transformation, where the new method is fast for every banded  $A$  and  $T$ . Moreover, the KPCR( $l$ )-method can be easily generalized to the case where  $A$  and  $T$  do not commute.

*Key words:* Poisson equation, direct methods, cyclic reduction, Kronecker product, Fourier analysis, FACR, KPCR

---

<sup>1</sup> E-mail: jef.hendrickx@cs.kuleuven.ac.be

<sup>2</sup> E-mail: marc.vanbarel@cs.kuleuven.ac.be

<sup>3</sup> The work of the authors is supported by the Belgian Programme on Interuniversity Poles of Attraction, initiated by the Belgian State, Prime Minister's Office for Science, Technology and Culture. The scientific responsibility rests with the authors.

<sup>4</sup> This research was partially supported by the K.U.Leuven (Bijzonder Onderzoeksfonds), project "SLAP: Structured Linear Algebra Package," grant #OT/00/16.

## 1 Introduction

The purpose of this paper is to find a fast direct method to solve the linear system of equations

$$M\mathbf{x} = \mathbf{y} \tag{1}$$

where  $M$  is a symmetric block tridiagonal Toeplitz matrix,

$$\begin{bmatrix} A & T & & & \\ T & A & \cdots & & \\ & \cdots & \cdots & T & \\ & & & T & A \end{bmatrix}. \tag{2}$$

Let us assume that  $M$  is of block order  $n - 1$  and that  $A$  and  $T$  are of order  $m - 1$ . Such a matrix arises for instance from a finite difference approximation to an elliptic partial differential equation on a rectangle with Dirichlet boundary conditions. A typical example is the Poisson equation, although the method will be applicable to a much larger class of differential equations. The most common direct methods for solving (1) are cyclic reduction and matrix decomposition or Fourier analysis. Both methods assume that  $A$  and  $T$  commute. Cyclic reduction was devised by Hockney [5] and is based on a reduction process that halves in every step the number of equations. In its original form, the method was not stable, but Buneman [3] developed a stable version. Fourier analysis or matrix decomposition was also developed by Hockney [5] and uses the simultaneous eigenvalue decomposition of  $A$  and  $T$  to transform the unknowns, such that the system (1) decouples into  $m - 1$  tridiagonal systems of order  $n - 1$ . For a detailed description of the two methods and some generalizations we refer to Buzbee, Golub and Nielson [2]. Hockney [6] combined the two methods into the FACR-algorithm: first  $l$  steps of cyclic reduction are performed after which the remaining system is solved by Fourier analysis. The combination is faster than the two methods used independently and Swarztrauber [10] found an optimal value for  $l$ . We remark however that the Fourier analysis method, and consequently the FACR-method as well, is only effective when some kind of fast Fourier transform can be used. Kronecker product methods do not have this drawback. As opposed to matrix decomposition, the methods do not use an eigenvalue decomposition of  $A$  and  $T$ , but they try to block diagonalize the matrix  $M$  in formula (2), by making use of the Kronecker product. Because of the specific form of  $M$ , this can be done by a sine transform matrix. Kronecker or tensor product methods were first described by Lynch, Rice and Thomas [7,8], but later several authors

used this technique. The methods hold for arbitrary  $A$  and  $T$ , even if they do not commute. In Hendrickx [4] one can find a broad class of elliptic partial differential equations with several types of boundary conditions and by using several types of grid, that can be solved fast by Kronecker product methods.

In this paper we will combine cyclic reduction and Kronecker product methods to the KPCR-method, similar to the FACR-method. The combined method will be faster than the individual methods. For the Poisson equation, the KPCR-method will have a speed comparable to the FACR-method, but the KPCR-method is more general, since it will hold for arbitrary  $A$  and  $T$  and we do not need the eigenvectors of  $A$  and  $T$ .

The outline of the paper is as follows. In section 2 and 3 we will review cyclic reduction and Kronecker product methods respectively. The KPCR-method, the new combined method, will be described in section 4. In section 5 we will see that the method can be generalized to the case where  $A$  and  $T$  do not commute. Finally, we will end with some numerical examples in section 6.

## 2 Cyclic reduction

In this section we will review the cyclic reduction method. Suppose we want to solve the linear system (1) where  $A$  and  $T$  commute. Originally the method was restricted to the case that  $n$  is a power of 2, but Sweet [11] generalized the method for arbitrary  $n$ . For simplicity however we will suppose in this section that  $n = 2^{k+1}$ . To conform with the matrix  $M$ , we write the vectors  $\mathbf{x}$  and  $\mathbf{y}$  in partitioned form,

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_{n-1} \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \\ \vdots \\ \mathbf{y}_{n-1} \end{bmatrix}.$$

We write down three succeeding equations of (1), for  $j$  even:

$$\begin{aligned} T\mathbf{x}_{j-2} + A\mathbf{x}_{j-1} + T\mathbf{x}_j &= \mathbf{y}_{j-1}, \\ T\mathbf{x}_{j-1} + A\mathbf{x}_j + T\mathbf{x}_{j+1} &= \mathbf{y}_j, \\ T\mathbf{x}_j + A\mathbf{x}_{j+1} + T\mathbf{x}_{j+2} &= \mathbf{y}_{j+1}. \end{aligned}$$

Multiplying the first and third equation by  $T$ , the second by  $-A$ , and adding, we have

$$T^2 \mathbf{x}_{j-2} + (2T^2 - A^2) \mathbf{x}_j + T^2 \mathbf{x}_{j+2} = T \mathbf{y}_{j-1} - A \mathbf{y}_j + T \mathbf{y}_{j+1}.$$

If we do this for every even  $j$ , the system decouples in 2 new systems:

$$\begin{bmatrix} A^{(1)} & T^{(1)} & & & \\ T^{(1)} & A^{(1)} & \ddots & & \\ & \ddots & \ddots & T^{(1)} & \\ & & & T^{(1)} & A^{(1)} \end{bmatrix} \begin{bmatrix} \mathbf{x}_2 \\ \mathbf{x}_4 \\ \vdots \\ \mathbf{x}_{n-2} \end{bmatrix} = \begin{bmatrix} \mathbf{y}_2^{(1)} \\ \mathbf{y}_4^{(1)} \\ \vdots \\ \mathbf{y}_{n-2}^{(1)} \end{bmatrix},$$

and

$$\begin{bmatrix} A & & & & \\ & A & & & \\ & & \ddots & & \\ & & & & A \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_3 \\ \vdots \\ \mathbf{x}_{n-1} \end{bmatrix} = \begin{bmatrix} \mathbf{y}_1 - T \mathbf{x}_2 \\ \mathbf{y}_3 - T \mathbf{x}_2 - T \mathbf{x}_4 \\ \vdots \\ \mathbf{y}_{n-1} - T \mathbf{x}_{n-2} \end{bmatrix},$$

with

$$A^{(1)} = 2T^2 - A^2, \quad T^{(1)} = T^2, \\ \mathbf{y}_j^{(1)} = T \mathbf{y}_{j-1} - A \mathbf{y}_j + T \mathbf{y}_{j+1}, \quad j = 2, 4, \dots, n-2.$$

The matrix of the first system has the same form as (2), so we can apply the reduction repeatedly, until only 1 equation remains. The eliminated unknowns can be found by the second equation. This process is called cyclic reduction. The method was developed by Hockney [5] but we refer to [2] for a detailed description.

### 3 Kronecker product method

Let  $S_n$  (or shortly  $S$ ) denote the sine transform matrix of order  $n$ :

$$S_n = \left[ \sqrt{\frac{2}{n+1}} \sin \frac{ij\pi}{n+1} \right]_{i,j=1}^n.$$

The matrix is symmetric and orthogonal, and  $S$  can be applied to a vector in  $O(n \log n)$  operations as long as  $n+1 = 2^k$  for some integer  $k$  (or has at least small prime factors). We consider the class of matrices that can be diagonalized by  $S$

$$\tau_n = \{C = S_n \Lambda S_n | \Lambda \text{ any diagonal matrix of order } n\}$$

This class is sometimes called the  $\tau$ -class [1] or the class of  $S$ -matrices [9,4] and the matrices have a very typical structure. A characterization for the matrices in this class is the cross-sum property, this is for any matrix  $C$  in  $\tau_n$  holds

$$c_{i,j-1} + c_{i,j+1} = c_{i-1,j} + c_{i+1,j},$$

where we have assumed the “boundary conditions”  $c_{0,j} = c_{n+1,j} = c_{i,0} = c_{i,n+1} = 0$ . Reversely, every matrix that satisfies the cross-sum property (inclusive the “boundary conditions”) lies in the  $\tau$ -class. As a consequence, such a matrix can be completely build from its first row. By calculating the first row in the relation  $S\Lambda = CS$ , we see that the matrix of eigenvalues  $\Lambda$  can be computed from the first row as well. As an example, a symmetric tridiagonal Toeplitz matrix (i.e. a matrix of the form (2) with  $A$  and  $T$  scalars) satisfies the cross-sum property and therefore can be diagonalized by  $S_n$ .

By use of the Kronecker product we can extend this class to block matrices. For instance, consider a matrix  $C$  of the form

$$C = (S_n \otimes S_m)\Lambda(S_n \otimes S_m)$$

with  $\Lambda$  an arbitrary diagonal matrix of order  $mn$ , and where  $\otimes$  denotes the Kronecker product between matrices. The matrix  $C$  is considered as a block matrix of block order  $n$  and we denote the  $(i, j)$ -block in  $C$  as  $C_{i,j}$ . The cross-sum condition holds both between and inside the blocks:

$$C_{i-1,j} + C_{i+1,j} = C_{i,j-1} + C_{i,j+1} \quad (3)$$

and

$$c_{i,j-1;k,l} + c_{i,j+1;k,l} = c_{i-1,j;k,l} + c_{i+1,j;k,l}$$

where  $c_{i,j;k,l}$  denotes element  $(i, j)$  in the block  $C_{k,l}$ , and again with the “boundaries” set to zero. Equivalently, for a matrix  $C$  defined as

$$C = (S_n \otimes I_m)\Lambda(S_n \otimes I_m), \quad (4)$$

where  $I_m$  denotes the identity matrix of order  $m$  and where  $\Lambda$  is an arbitrary block diagonal matrix of block order  $n$ , only the relation (3) between the blocks holds. Reversely, every block matrix that fulfils the block relation (3), can be written in the form (4). As a consequence, the matrix  $M$  of (2) can be block diagonalized as

$$M = (S_{n-1} \otimes I_{m-1})B(S_{n-1} \otimes I_{m-1}). \quad (5)$$

The matrix  $B$  is a block diagonal matrix and its diagonal blocks can be found from the first row of  $M$ :

$$B = \text{diag}\{B_1, \dots, B_{n-1}\},$$

$$B_j = A + 2T \cos \frac{j\pi}{n}, \quad j = 1, \dots, n-1.$$

Notice that another way to get this result is to write  $M$  as  $M = I_{n-1} \otimes A + P \otimes T$ , with  $P$  the matrix

$$P = \begin{bmatrix} 0 & 1 & & \\ 1 & 0 & \cdots & \\ & \cdots & \cdots & 1 \\ & & & 1 & 0 \end{bmatrix}.$$

The matrix  $P$  is a  $\tau$ -matrix and can therefore be written as  $S_{n-1}\Omega S_{n-1}$ , with  $\Omega = \text{diag}\{2\pi/n, \dots, 2(n-1)\pi/n\}$ , such that

$$M = (S_{n-1} \otimes I_{m-1})(I \otimes A + \Omega \otimes T)(S_{n-1} \otimes I_{m-1}).$$

The matrix in the middle of the right hand side is a block diagonal matrix and is precisely the latter matrix  $B$ .

We can use the factorization (5) to solve the linear system  $M\mathbf{x} = \mathbf{y}$  as  $\mathbf{x} = (S \otimes I)B^{-1}(S \otimes I)\mathbf{y}$ . This leads to the following algorithm.

**Algorithm 1 (Kronecker product method)** Solve the linear system  $M\mathbf{x} = \mathbf{y}$ , where  $M$  is a block symmetric tridiagonal Toeplitz matrix of block order  $n-1$  with  $A$  on the diagonal and  $T$  on the subdiagonal.

- (1)  $\hat{\mathbf{y}} = (S \otimes I)\mathbf{y}$
- (2) Solve the systems  $(A + 2T \cos \frac{j\pi}{n}) \hat{\mathbf{x}}_j = \hat{\mathbf{y}}_j$ , for  $j = 1, \dots, n-1$
- (3)  $\mathbf{y} = (S \otimes I)\hat{\mathbf{x}}$

The first and last step imply  $m-1$  sine transforms of order  $n-1$ . The second step consists of solving  $n-1$  linear systems with matrices  $B_1, \dots, B_{n-1}$  of order  $m-1$ . Often, the matrices  $A$  and  $T$  are tridiagonal or diagonal matrices, so this can be computed in  $O(nm)$ . Therefore, the total operation count is  $O(nm \log n)$  if  $A$  and  $T$  are tridiagonal and if  $n$  has small prime factors.

We notice that this method is similar to the matrix decomposition method or Fourier analysis [5,2]. If  $A$  and  $T$  commute and are symmetric, then they are simultaneously diagonalizable by some orthogonal matrix  $Q$ . We can use this to decouple the system in  $m-1$  tridiagonal systems of order  $n-1$ . The method consists then of three steps similar to the Kronecker product method: first perform an orthogonal transformation on the unknowns with transformation matrix  $Q$ , next solve  $m-1$  tridiagonal systems, and finally perform again some orthogonal transforms with the matrix  $Q$ . In case of the Poisson equation, the matrices  $A$  and  $T$  have a very specific form and the matrix  $Q$  is equal to the sine transform matrix  $S_{m-1}$ . In this case, the operation count is  $O(nm \log m)$

flops (if  $m$  has small prime factors). The Kronecker product method however, has operation count  $O(nm \log n)$  for every banded  $A$  and  $T$ . Moreover,  $A$  and  $T$  do not have to commute or have to be symmetric and we do not need to compute the eigenvalues or eigenvectors. Finally, we remark that in Hendrickx [4] the Kronecker product method is used to solve a broad class of elliptic partial differential equations with several types of boundary conditions and on different types of grid. The linear systems that arise have a different form compared to (1), but one can use other trigonometric transforms to block diagonalize the matrices.

#### 4 KPCR-method

In this section we shall describe the KPCR-method which combines the methods of section 2 and 3. The method is similar to the FACR-method [6], which is a combination of cyclic reduction and Fourier analysis. The KPCR-method however will be more general than the FACR-method.

We will assume again that  $AT = TA$  and  $n = 2^{k+1}$ . Instead of performing cyclic reduction until only one equation remains, the KPCR-method begins with only  $l$  steps of cyclic reduction ( $l < k$ ), exactly as the FACR-method does. In the latter method, the remaining systems will be solved by Fourier analysis, in the KPCR-method we will use the Kronecker product method. If we want to state explicitly the number of cyclic reduction steps  $l$ , we will denote the methods as the FACR( $l$ )- and the KPCR( $l$ )-method. After  $l$  steps of cyclic reduction we have a system

$$M^{(l)} \mathbf{x}^{(l)} = \mathbf{y}^{(l)},$$

with  $M^{(l)}$  the matrix

$$M^{(l)} = \begin{bmatrix} A^{(l)} & T^{(l)} & & & \\ T^{(l)} & A^{(l)} & \cdots & & \\ & \cdots & \cdots & T^{(l)} & \\ & & & T^{(l)} & A^{(l)} \end{bmatrix},$$

and where  $A^{(l)}$  and  $T^{(l)}$  are defined by the recursion

$$A^{(r)} = 2 \left( T^{(r-1)} \right)^2 - \left( A^{(r-1)} \right)^2, \quad A^{(0)} = A,$$

$$T^{(r)} = \left( T^{(r-1)} \right)^2, \quad T^{(0)} = T,$$

for  $r = 1, \dots, l$ . Furthermore

$$\mathbf{x}^{(l)} = \begin{bmatrix} \mathbf{x}_{2^l} \\ \mathbf{x}_{2 \cdot 2^l} \\ \vdots \\ \mathbf{x}_{n-2^l} \end{bmatrix}, \quad \mathbf{y}^{(l)} = \begin{bmatrix} \mathbf{y}_{2^l}^{(l)} \\ \mathbf{y}_{2 \cdot 2^l}^{(l)} \\ \vdots \\ \mathbf{y}_{n-2^l}^{(l)} \end{bmatrix},$$

where  $\mathbf{y}_j^{(l)}$  is defined by the recursion

$$\mathbf{y}_j^{(r)} = T^{(r-1)} \left( \mathbf{y}_{j-2^{r-1}}^{(r-1)} + \mathbf{y}_{j+2^{r-1}}^{(r-1)} \right) - A^{(r-1)} \mathbf{y}_j^{(r-1)}.$$

Since the matrix  $M^{(l)}$  has the same form as  $M$  in (2), we can use the Kronecker product method from section 3 to solve the systems. By other words,  $M^{(l)}$  can be block diagonalized by the sine transform

$$(S_{n/2^l-1} \otimes I) B^{(l)} (S_{n/2^l-1} \otimes I),$$

where

$$B^{(l)} = \text{diag}\{B_1^{(l)}, B_2^{(l)}, \dots, B_{n/2^l-1}^{(l)}\},$$

and

$$B_j^{(l)} = A^{(l)} + 2T^{(l)} \cos \frac{j\pi}{n/2^l}. \quad (6)$$

If we use algorithm 1 to solve the system, we have to solve in a second step the systems  $B_j^{(l)} \hat{\mathbf{x}}_{j2^l} = \hat{\mathbf{y}}_{j2^l}^{(l)}$ . Even if  $A$  and  $T$  have small bandwidth, the bandwidth of  $B_j^{(l)}$  can become very large, for instance if  $A$  and  $T$  are tridiagonal, the bandwidth of  $B_j^{(l)}$  is  $2^{l+1} + 1$ , just like  $A^{(l)}$ . However, in [2]  $A^{(l)}$  is written as a product of matrices built up with  $A$  and  $T$ :

$$A^{(l)} = - \prod_{j=1}^{2^l} \left( A + 2T \cos \frac{(2j-1)\pi}{2^{l+1}} \right). \quad (7)$$

We will deduce a similar factorization for  $B_j^{(l)}$ . It can be easily seen that  $A^{(l)}$  and  $B_j^{(l)}$  are homogeneous polynomials of degree  $2^l$  in  $A$  and  $T$ , so that

$$A^{(l)} = \sum_{j=0}^{2^l} c_j A^j T^{2^l-j} \equiv P_{2^l}(A, T), \quad B_j^{(l)} = \sum_{j=0}^{2^l} d_j A^j T^{2^l-j} \equiv Q_{2^l}(A, T).$$

Let  $p_{2^l}(a, t)$  and  $q_{2^l}(a, t)$  be the scalar analogues of the matrix polynomials  $P_{2^l}$

and  $Q_{2^l}$ :

$$p_{2^l}(a, t) = \sum_{j=0}^{2^l} c_j a^j t^{2^l-j}, \quad q_{2^l}(a, t) = \sum_{j=0}^{2^l} d_j a^j t^{2^l-j},$$

then it follows from (6) that

$$q_{2^l}(a, t) = p_{2^l}(a, t) + 2t^{2^l} \cos \frac{j\pi}{n/2^l}.$$

If we make the substitution  $a/t = -2 \cos \theta$ , then Buzbee, Golub and Nielson [2] derived that

$$p_{2^l}(a, t) = -2t^{2^l} T_{2^l}(-a/2t),$$

where  $T_n(-a/2t) = \cos(n\theta)$  denotes the Chebyshev polynomial of the first kind of degree  $n$ . It then follows that

$$q_{2^l}(a, t) = -2t^{2^l} \left( T_{2^l}(-a/2t) - \cos \frac{j\pi}{n/2^l} \right).$$

The second factor vanishes for  $2^l\theta = \pm j\pi 2^l/n + 2r\pi$  with  $r \in \mathbb{Z}$ , or in terms of  $-a/t$  for

$$\frac{-a}{2t} = \cos \left( \frac{j\pi}{n} + r \frac{2\pi}{2^l} \right), \quad r = 1, \dots, 2^l.$$

Finally, since the leading coefficient of  $T_n(x)$  is equal to  $2^{n-1}$ , we can write  $q_{2^l}$  as

$$q_{2^l}(a, t) = - \prod_{r=1}^{2^l} \left( a + 2t \cos \left( \frac{j\pi}{n} + r \frac{2\pi}{2^l} \right) \right).$$

Consequently

$$B_j^{(l)} = - \prod_{r=1}^{2^l} \left( A + 2T \cos \left( \frac{j\pi}{n} + r \frac{2\pi}{2^l} \right) \right). \quad (8)$$

If  $A$  and  $T$  are tridiagonal matrices, a linear system with  $B_j^{(l)}$  as matrix can be solved as  $2^l$  repeated tridiagonal systems. Now the KPCR( $l$ )-method is complete, and we can formulate the algorithm. However, for the cyclic reduction method to be stable, matrix-vector products of the form  $A^{(r)}\mathbf{y}$  have to be avoided. Buneman [3] developed a stable version. The basic idea is to introduce vectors  $\mathbf{p}_j^{(r)}$  and  $\mathbf{q}_j^{(r)}$  such that  $\mathbf{y}_j^{(r)} = A^{(r)}\mathbf{p}_j^{(r)} + \mathbf{q}_j^{(r)}$ , and to rewrite the algorithm in function of  $\mathbf{p}_j^{(r)}$  and  $\mathbf{q}_j^{(r)}$  and without matrix-vector products with  $A^{(r)}$  as matrix. If we apply the same adjustments to the KPCR( $l$ )-method, then after  $l$  steps of cyclic reduction, we have to solve the system

$$T^{(l)}\mathbf{x}_{j-2^l} + A^{(l)}\mathbf{x}_j + T^{(l)}\mathbf{x}_{j+2^l} = A^{(l)}\mathbf{p}_j^{(l)} + \mathbf{q}_j^{(l)},$$

for  $j = 2^l, 2 \cdot 2^l, \dots, n - 2^l$  (with  $\mathbf{x}_0 = \mathbf{x}_n = \mathbf{0}$ ), so again matrix-vector products with  $A^{(l)}$  appear in the right-hand side. However, we can rewrite the system

as

$$T^{(l)} \left( \mathbf{x}_{j-2^l} - \mathbf{p}_{j-2^l}^{(l)} \right) + A^{(l)} \left( \mathbf{x}_j - \mathbf{p}_j^{(l)} \right) + T^{(l)} \left( \mathbf{x}_{j+2^l} - \mathbf{p}_{j+2^l}^{(l)} \right) = \mathbf{q}_j^{(l)} - T^{(l)} \left( \mathbf{p}_{j-2^l}^{(l)} + \mathbf{p}_{j+2^l}^{(l)} \right),$$

and consider it as a system in the unknowns  $\mathbf{x}_j - \mathbf{p}_j^{(l)}$ . We now state the complete algorithm.

**Algorithm 2 (KPCR( $l$ ), Buneman-variant 1)** Solve the linear system  $M\mathbf{x} = \mathbf{y}$ , where  $M$  is a block symmetric tridiagonal Toeplitz matrix of block order  $n - 1$  with  $A$  on the diagonal and  $T$  on the subdiagonal. The matrices  $A$  and  $T$  commute and are of order  $m - 1$ . We assume that  $n = 2^{k+1}$ .

- (1) *Initialisation*
  - (a)  $\mathbf{p}_j^{(0)} = \mathbf{0}$ ,  $j = 0, 1, \dots, n$
  - (b)  $\mathbf{q}_j^{(0)} = \mathbf{y}_j$ ,  $j = 1, 2, \dots, n - 1$
- (2) *Reduction*

For  $r = 1, 2, \dots, l$  do

For  $j = 2^r, 2 \cdot 2^r, \dots, n - 2^r$  do

  - (a)  $\mathbf{p}_j^{(r)} = \mathbf{p}_j^{(r-1)} - \left( A^{(r-1)} \right)^{-1} \left( T^{2^{r-1}} \left( \mathbf{p}_{j-2^{r-1}}^{(r-1)} + \mathbf{p}_{j+2^{r-1}}^{(r-1)} \right) - \mathbf{q}_j^{(r-1)} \right)$ , with  $A^{(r-1)}$  as in (7).
  - (b)  $\mathbf{q}_j^{(r)} = T^{2^{r-1}} \left( \mathbf{q}_{j-2^{r-1}}^{(r-1)} + \mathbf{q}_{j+2^{r-1}}^{(r-1)} \right) - 2T^{2^r} \mathbf{p}_j^{(r)}$
- (3) *Solve remaining system*
  - (a)  $\tilde{\mathbf{q}}_j^{(l)} = \mathbf{q}_j^{(l)} - T^{2^l} \left( \mathbf{p}_{j-2^l}^{(l)} + \mathbf{p}_{j+2^l}^{(l)} \right)$ , voor  $j = 2^l, 2 \cdot 2^l, \dots, n - 2^l$
  - (b)  $\hat{\mathbf{q}}^{(l)} = \left( S_{n/2^l-1}^I \otimes I \right) \tilde{\mathbf{q}}^{(l)}$
  - (c) Solve  $B_j^{(l)} \hat{\mathbf{x}}_j = \hat{\mathbf{q}}_j^{(l)}$  for  $j = 2^l, 2 \cdot 2^l, \dots, n - 2^l$ , with  $B_j^{(l)}$  as in (8).
  - (d)  $\tilde{\mathbf{x}}^{(l)} = \left( S_{n/2^l-1}^I \otimes I \right) \hat{\mathbf{x}}^{(l)}$
  - (e)  $\mathbf{x}_j^{(l)} = \tilde{\mathbf{x}}_j^{(l)} + \mathbf{p}_j^{(l)}$ , voor  $j = 2^l, 2 \cdot 2^l, \dots, n - 2^l$
- (4) *Backsubstitution* (we assume  $\mathbf{x}_0 = \mathbf{x}_n = \mathbf{0}$ )

For  $r = l - 1, l - 2, \dots, 0$  do

For  $j = 2^r, 3 \cdot 2^r, \dots, n - 2^r$  do

$$\mathbf{x}_j = \mathbf{p}_j^{(r)} + \left( A^{(r)} \right)^{-1} \left( \mathbf{q}_j^{(r)} - T^{2^r} \left( \mathbf{x}_{j-2^r} + \mathbf{x}_{j+2^r} \right) \right), \text{ with } A^{(r)} \text{ as in (7).}$$

We still have a freedom in choosing the value of  $l$ , the number of cyclic reduction steps to perform. To find an optimal value for  $l$ , we will compute the asymptotic operation count of the algorithm, under the assumption that  $A$  is tridiagonal and  $T$  is diagonal. Since  $A^{(r-1)}$  is a product of  $2^{r-1}$  tridiagonal matrices, every reduction step requires  $8m2^{r-1}$  flops for every  $r$  and  $j$ . We have to compute  $n/2^r - 1$  vectors  $\mathbf{q}_j$ , so the total count for the reduction stage is  $4lmn$  flops. A similar argument holds for the backsubstitution. Finally for solving the remaining system, in the beginning and at the end we have to perform  $m$  sine transforms of length  $n/2^l - 1$ , which requires in total  $5mn/2^l \log_2 n$  flops. In the middle we have to solve  $2^l$  tridiagonal systems of

order  $m$  for every  $j$ , so this gives  $8mn$  flops in total. Hence, the total operation count  $C_l$  for the algorithm is

$$C_l = 8lmn + 5mn/2^l \log_2 n \text{ flops.}$$

This expression is minimized at  $l = \log_2 \log_2 n + \log(5/8 \ln 2)$  or  $l \approx \log_2 \log_2 n - 1$ . With this value of  $l$ , we obtain an asymptotic operation count of  $8mn \log_2 \log_2 n$  flops for the algorithm, which is better than cyclic reduction or the Kronecker product method used independently.

We noticed already that the method is similar to the FACR( $l$ )-method, where first  $l$  steps of cyclic reduction are performed, followed by Fourier analysis to solve the remaining system. Swarztrauber [10] proved that the asymptotic operation count is  $8mnl + 5mn/2^l \log_2 m$ , which is minimized at  $l \approx \log_2 \log_2 m - 1$ . For this value of  $l$ , we obtain a total operation count of  $8mn \log_2 \log_2 m$  for the FACR( $l$ )-algorithm. So, for  $m = n$ , we expect that the KPCR( $l$ )-algorithm is as performant as the FACR( $l$ )-algorithm. However, the FACR( $l$ )-algorithm only reaches this performance for matrices  $A$  and  $T$  that can be diagonalized by a matrix which allows some kind of fast Fourier transform, where the KPCR( $l$ )-algorithm is fast for arbitrary matrices  $A$  and  $T$  with small bandwidth. We do not need knowledge of the eigenvalues or -vectors of  $A$  and  $T$ . Moreover, the KPCR( $l$ )-algorithm can be generalized to the case that  $A$  and  $T$  do not commute, as will be seen in the next section.

Since cyclic reduction can be generalized to the case where  $n$  is not anymore a power of 2 [11], the KPCR-method also does not have to be restricted to this case. Moreover, it can be easily generalized to the Poisson equation with other types of boundary conditions, like Neumann or periodic, by using other types of trigonometric transforms.

## 5 The KPCR( $l$ )-method for non-commutative $A$ and $T$

If we want to solve the system (1) in the case where  $A$  and  $T$  do not commute, we can still apply the KPCR-method by handling the equivalent system

$$\begin{bmatrix} T^{-1}A & I & & & \\ & I & T^{-1}A & & \\ & & & \ddots & \\ & & & & I & T^{-1}A \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_{n-1} \end{bmatrix} = \begin{bmatrix} T^{-1}\mathbf{y}_1 \\ T^{-1}\mathbf{y}_2 \\ \vdots \\ T^{-1}\mathbf{y}_{n-1} \end{bmatrix}. \quad (9)$$

Now, the matrices  $T^{-1}A$  and  $I$  do commute of course, and we can apply algorithm 2. For simplicity, let us denote  $\tilde{A} = T^{-1}A$ . The matrix  $\tilde{A}^{(r)}$  in algorithm 2 can be rewritten as

$$\tilde{A}^{(r)} = - \prod_{k=1}^{2^r} T^{-1} \left( A + 2T \cos \frac{(2k-1)\pi}{2^{r+1}} \right).$$

Similarly it is easy to see that for  $\tilde{\Lambda}_j^{(l)}$  holds

$$\tilde{\Lambda}_j^{(l)} = - \prod_{k=1}^{2^l} T^{-1} \left( A + 2T \cos \left( \frac{j\pi}{n} + k \frac{2\pi}{2^l} \right) \right).$$

This can be used to solve the systems with  $\tilde{A}^{(r)}$  or  $\tilde{\Lambda}_j^{(l)}$  as matrix.

Of course, by rewriting the system as (9), it is also possible to apply the FACR-method. However, in that case we will need the eigenvalues and -vectors of  $T^{-1}A$  and they are not always easy to compute. Moreover, even if we know the eigenvectors, in most cases they are not related to a fast transform, leading to a loss of performance of the FACR( $l$ )-algorithm.

## 6 Experimental results

We have implemented the KPCR-algorithm and the FACR-algorithm in Fortran 90. The programs were executed on an IBM SP2 machine in double precision. The FFTs were calculated via FFTPACK. We present two examples. The first example is the Poisson equation  $\nabla^2 \phi = f$  on the square  $[0, 1]^2$  with Dirichlet boundary conditions on all sides. We choose  $\phi(x, y) = 3e^{x+y}(x-x^2)(y-y^2)$  as the exact solution of the Poisson equation and we use a  $2048 \times 2048$  grid to discretize the equation, such that  $n = m = 2048 = 2^{11}$ . In table 1 you will find the execution times (in sec.) for the FACR- and the KPCR-method for different values of  $l$ , the number of cyclic reduction steps to perform. Notice that the case  $l = 0$  corresponds to the Fourier analysis method, respectively Kronecker product method separately, while  $l = 10$  corresponds to cyclic reduction separately. It is clear that the combined methods are faster than the individual methods. The minimum is not attained at  $l \approx \log_2 \log_2 n - 1$ , but rather at  $l \approx \log_2 \log_2 n$ . The KPCR-method and FACR-method have comparable execution times, although the FACR-method seems to be a little bit faster.

Our second example is the Poisson equation in polar coordinates

$$\frac{1}{r} \frac{\partial}{\partial r} \left( r \frac{\partial \phi}{\partial r} \right) + \frac{1}{r^2} \frac{\partial^2 \phi}{\partial \theta^2} = 16r^2,$$

$l$	KPCR	FACR
0	22.11	17.08
1	12.65	9.66
2	8.42	6.14
3	5.95	4.52
4	5.59	4.11
5	5.87	4.44
6	6.12	4.62
7	6.57	5.13
8	7.06	5.74
9	7.61	7.68
10	8.27	7.53

Table 1

Computation times in seconds for the Poisson equation on a square

on the circle segment  $0 \leq r \leq 1$ ,  $0 \leq \theta \leq \pi/2$ . We assume Dirichlet boundary conditions on all sides and  $\phi(r, \theta) = r^4(1 - \cos 4\theta)$  as the exact solution. In table 2 we take  $n = m = 1024$  for the grid and we list the execution times for different values for  $l$ . In this example the FACR-method is not possible, since the matrices  $A$  and  $T$  will not commute. With this value for  $n$  we can perform at most 9 steps of cyclic reduction. Again, the combination of cyclic reduction and Kronecker product methods is faster then the individual methods, and the minimum is attained at  $l = 4 \approx \log_2 \log_2 1024$ .

## References

- [1] D. Bini and M. Capovani, Spectral and computational properties of band symmetric Toeplitz matrices, *Linear Algebra Appl.* **52/53** (1983) 99–126.
- [2] B.L. Buzbee, G.H. Golub and C.W. Nielson, On direct methods for solving Poisson’s equation, *SIAM J. Numer. Anal.* **7** (1970) 627–656.
- [3] O. Buneman, A compact non-iterative Poisson solver, Technical Report 294, Stanford University Institute for Plasma Research, Stanford, California, 1969.
- [4] J. Hendrickx, Veralgemeende  $S$ -matrices en hun aanwending bij de eindigedifferentiebenaderingen van differentiaalvergelijkingen en het oplossen van symmetrische band-Toeplitzstelsels, Ph.D. Thesis, Katholieke Universiteit Leuven, 2000 (in dutch).
- [5] R.W. Hockney, A fast direct solution of Poisson’s equation using Fourier analysis, *J. Assoc. Comput. Mach.* **8** (1965) 95–113.

$l$	KPCR
0	11.69
1	8.73
2	7.05
3	6.79
4	7.05
5	7.62
6	8.21
7	8.94
8	9.57
9	10.10

Table 2

Computation times in seconds for the Poisson equation in polar coordinates

- [6] R.W. Hockney, The potential calculation and some applications, in: B. Adler, S. Fernback and M. Rotenberg, eds., *Methods of Computational Physics, Vol. 9* (Academic Press, New York and London, 1969) 136–211.
- [7] R.E. Lynch, J.R. Rice and D.H. Thomas, Tensor product analysis of partial difference equations, *Bull. Amer. Math. Soc.* **70** (1964) 378–384.
- [8] R.E. Lynch, J.R. Rice and D.H. Thomas, Direct solution of partial difference equations by tensor product methods, *Numer. Math.* **6** (1964) 185–199.
- [9] L. Mertens and H. Van de Vel, A special class of structured matrices constructed with the Kronecker product and its use for difference equations, *Linear Algebra Appl.* **106** (1988) 117–147.
- [10] P.N. Swarztrauber, The methods of cyclic reduction, Fourier analysis and the FACR algorithm for the discrete solution of Poisson’s equation on a rectangle, *SIAM Rev.* **19** (1977) 490–501.
- [11] R.A. Sweet, A cyclic reduction algorithm for solving block tridiagonal systems of arbitrary dimensions, *SIAM J. Numer. Anal.* **14** (1977) 706–720.