J Math Chem (2013) 51:881–889
DOI 10.1007/s10910-012-0122-7

ORIGINAL PAPER

A novel algorithm for solving quasi penta-diagonal linear systems

Ji-Teng Jia · Tomohiro Sogabe

Received: 22 June 2012 / Accepted: 24 November 2012 / Published online: 11 December 2012

Abstract In this paper, a novel numerical algorithm for solving quasi penta-diagonal linear systems is presented. The computational costs of the algorithm is less than those of three successful algorithms given by El-Mikkawy and Rahmo (Comput Math Appl 59:1386–1396, 2010), by Lv and Le (Appl Math Comput 204:707–712, 2008), and by Jia et al. (Int J Comput Math 89:851–860, 2012). In addition, a new recursive method for inverting the quasi penta-diagonal matrices is also discussed. The implementation of the algorithm using Computer Algebra Systems (CASs) such as MATLAB and MAPLE is straightforward. Two numerical examples are given in order to demonstrate the performance and efficiency of our algorithm.

Keywords Quasi penta-diagonal matrices · LU factorization · Linear systems · Inverse · Computer Algebra Systems (CASs)

Mathematics Subject Classification (2000) 15A09 · 33F30 · 15A23 · 65F30

1 Introduction

Quasi penta-diagonal linear systems frequently arise in computational physics and mathematical chemistry [1,2], especially because the discretization of differential equations, transforming them into finite-difference equations, often results in quasi penta-diagonal matrices [3,4]. Several examples of this can be found in boundary value

J.-T. Jia (✉)
Department of Mathematical Sciences, Xi'an Jiaotong University, Xi'an, 710049, Shaanxi, China
e-mail: lavenderjjt@163.com

T. Sogabe
Graduate School of Information Science and Technology, Aichi Prefectural University, 1522-3 Kumabari, Nagakute, Aichi 480-1198, Japan
e-mail: sogabe@ist.aichi-pu.ac.jp

problems (BVP) [5–7], numerical solution of ordinary and partial differential equations (ODE and PDE), fluid mechanics [8], parallel computing [9,10], etc. [11–14]. In quantum chemistry, finite difference methods using quasi penta-diagonal matrices are used both in the density functional theory [15] and wavefunction formalism [16].

The $n \times n$ quasi penta-diagonal linear system takes the form

$$A\mathbf{x} = \mathbf{f}, \tag{1}$$

where

$$A = \begin{pmatrix} d_1 & a_1 & \tilde{a}_1 & 0 & \cdots & 0 & \tilde{b}_1 & b_1 \\ b_2 & d_2 & a_2 & \tilde{a}_2 & 0 & \cdots & 0 & \tilde{b}_2 \\ \tilde{b}_3 & b_3 & d_3 & a_3 & \tilde{a}_3 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \tilde{b}_{n-2} & b_{n-2} & d_{n-2} & a_{n-2} & \tilde{a}_{n-2} \\ \tilde{a}_{n-1} & 0 & \cdots & 0 & \tilde{b}_{n-1} & b_{n-1} & d_{n-1} & a_{n-1} \\ a_n & \tilde{a}_n & 0 & \cdots & 0 & \tilde{b}_n & b_n & d_n \end{pmatrix}, \tag{2}$$

$\mathbf{x} = (x_1, x_2, \ldots, x_n)^T$, $\mathbf{f} = (f_1, f_2, \ldots, f_n)^T$ and $n \geq 6$. The superscript T corresponds to the transpose operation. The above matrix can be written by the following 2×2 block matrix

$$A = \begin{pmatrix} P & V \\ U & D \end{pmatrix}, \tag{3}$$

where

$$\begin{aligned} V &= (\mathbf{v}_1, \mathbf{v}_2) = \begin{pmatrix} \tilde{b}_1 & 0 & \cdots\cdots & \tilde{a}_{n-3} & a_{n-2} \\ b_1 & \tilde{b}_2 & 0 & \cdots 0 & \tilde{a}_{n-2} \end{pmatrix}^T \in \mathbb{R}^{(n-2)\times 2}, \\ U &= \begin{pmatrix} \tilde{a}_{n-1} & 0 & \cdots\cdots & \tilde{b}_{n-1} & b_{n-1} \\ a_n & \tilde{a}_n & 0 & \cdots 0 & \tilde{b}_n \end{pmatrix} \in \mathbb{R}^{2\times(n-2)}, \\ D &= \begin{pmatrix} d_{n-1} & a_{n-1} \\ b_n & d_n \end{pmatrix} \in \mathbb{R}^{2\times 2}, \end{aligned}$$

and P is the $(n-2)$th leading principal submatrix of A.

Recently, some authors have developed fast numerical or symbolic algorithms for solving quasi penta-diagonal linear systems. For example, El-Mikkawy and Rahmo [17], Jia et al. [18], Lv and Le [19]. All these algorithms solve the linear systems in linear time. In this study, our main objective is to establish a more efficient algorithm for solving linear system (1).

The rest of this paper is organized as follows. A novel numerical algorithm that will not suffer from breakdown is constructed in Sect. 2. In addition, we give an algorithm for inverting the quasi penta-diagonal matrices. In Sect. 3, two numerical examples are provided to demonstrate the performance and efficiency of our algorithm. Finally, some conclusions of the work are given in Sect. 4.

2 Main results

In this section, we are going to formulate an efficient algorithm for solving a quasi penta-diagonal linear system of the form (1). To do this, we begin by reviewing a method [20], designed for serial implementation, for the solution of a penta-diagonal linear system.

2.1 The symbolic method for solving penta-diagonal linear systems

Let P be the $m \times m$ penta-diagonal matrix of the form

$$P = \begin{pmatrix} d_1 & a_1 & \tilde{a}_1 & 0 & \cdots & 0 & 0 \\ b_2 & d_2 & a_2 & \tilde{a}_2 & 0 & \cdots & 0 \\ \tilde{b}_3 & b_3 & d_3 & a_3 & \tilde{a}_3 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \tilde{a}_{m-2} \\ 0 & \cdots & 0 & \tilde{b}_{m-1} & b_{m-1} & d_{m-1} & a_{m-1} \\ 0 & 0 & \cdots & 0 & \tilde{b}_m & b_m & d_m \end{pmatrix}.$$

Consider the LU factorization of P in the form

$$P = LU,$$

where

$$L = \begin{pmatrix} 1 & 0 & \cdots & \cdots & \cdots & 0 \\ g_2 & 1 & 0 & \cdots & \cdots & 0 \\ h_3 & g_3 & 1 & 0 & \ddots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & h_{m-1} & g_{m-1} & 1 & 0 \\ 0 & \cdots & 0 & h_m & g_m & 1 \end{pmatrix},$$

and

$$U = \begin{pmatrix} c_1 & e_1 & \tilde{a}_1 & 0 & \cdots & 0 \\ 0 & c_2 & e_2 & \tilde{a}_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & c_{m-2} & e_{m-2} & \tilde{a}_{m-2} \\ 0 & \cdots & \cdots & 0 & c_{m-1} & e_{m-1} \\ 0 & \cdots & \cdots & \cdots & 0 & c_m \end{pmatrix}.$$

By using the above matrices, the penta-diagonal linear system

$$P\mathbf{y} = \tilde{\mathbf{f}},$$

where $\mathbf{y} = (y_1, y_2, \ldots, y_m)^T$ and $\tilde{\mathbf{f}} = (\tilde{f}_1, \tilde{f}_2, \ldots, \tilde{f}_m)^T$, can be efficiently solved by forward substitution and back substitution.

We may now formulate the following results:

Algorithm 2.1

```
Step 1. Set $c_1 = d_1$. If $c_1 = 0$ then $c_1 = t$ end if;
Step 2. Compute and simplify $c_2 = d_2 - \frac{b_2}{c_1}a_1$, If $c_2 = 0$ then $c_2 = t$ end if;
Step 3. Set $e_1 = a_1$ and $g_2 = \frac{b_2}{c_1}$, then compute and simplify;
        For $k = 3, 4, \ldots, m$ do
            $h_k = \frac{\tilde{b}_k}{c_{k-2}}$,
            $e_{k-1} = a_{k-1} - g_{k-1}\tilde{a}_{k-2}$,
            $g_k = \frac{b_k - h_k e_{k-2}}{c_{k-1}}$,
            $c_k = d_k - g_k e_{k-1} - h_k \tilde{a}_{k-2}$,
            If $c_k = 0$ then $c_k = t$ end if,
        End do.
Step 4. Compute $\tilde{y}_1 = \tilde{f}_1$ and $\tilde{y}_2 = \tilde{f}_2 - g_2\tilde{y}_1$;
        For $k = 3, 4, \ldots, m$ do
            $\tilde{y}_k = \tilde{f}_k - h_k\tilde{y}_{k-2} - g_k\tilde{y}_{k-1}$,
        End do.
Step 5. Compute $y_m = \frac{\tilde{y}_m}{c_m}$ and $y_m = \frac{\tilde{y}_{m-1} - e_{m-1}y_m}{c_{m-1}}$;
        For $k = m-2, m-3, \ldots, 1$ do
            $y_k = \frac{\tilde{y}_k - \tilde{a}_k y_{k+2} - e_k y_{k+1}}{c_k}$,
        End do.
```

As stated in [20], Step 3 of Algorithm 2.1 is the main part of the DETGTRI algortihm [21], Algorithm 2.1, therefore, does not suffer from breakdown.

2.2 A numerical algorithm for solving quasi penta-diagonal linear systems

From (3), we can see that the linear system (1) can be written in the form

$$\begin{pmatrix} P & V \\ U & D \end{pmatrix} \begin{pmatrix} \hat{\mathbf{x}} \\ \tilde{\mathbf{x}} \end{pmatrix} = \begin{pmatrix} \hat{\mathbf{f}} \\ \tilde{\mathbf{f}} \end{pmatrix}, \tag{4}$$

where

$$\hat{\mathbf{x}} = (x_1, x_2, \ldots, x_{n-2})^T, \ \tilde{\mathbf{x}} = (x_{n-1}, x_n)^T,$$
$$\hat{\mathbf{f}} = (f_1, f_2, \ldots, f_{n-2})^T, \ \tilde{\mathbf{f}} = (f_{n-1}, f_n)^T.$$

Thus (4) is equivalent to

$$\begin{cases} P\hat{\mathbf{x}} + V\tilde{\mathbf{x}} = \hat{\mathbf{f}}, \\ U\hat{\mathbf{x}} + D\tilde{\mathbf{x}} = \tilde{\mathbf{f}}. \end{cases} \tag{5}$$

It is easy to deduce that

$$\hat{\mathbf{x}} = P^{-1}\hat{\mathbf{f}} - x_{n-1} \cdot P^{-1}\mathbf{v}_1 - x_n \cdot P^{-1}\mathbf{v}_2. \tag{6}$$

Let $\mathbf{y} = (y_1, y_2, \ldots, y_{n-2})^T$, $\mathbf{z} = (z_1, z_2, \ldots, z_{n-2})^T$, and $\mathbf{w} = (w_1, w_2, \ldots, w_{n-2})^T$ be solutions of the following equations

$$P\mathbf{y} = \hat{\mathbf{f}}, \quad P\mathbf{z} = \mathbf{v}_1, \quad P\mathbf{w} = \mathbf{v}_2, \tag{7}$$

respectively. Then, we have

$$\begin{cases} x_1 = y_1 - x_{n-1}z_1 - x_n w_1, \\ x_2 = y_2 - x_{n-1}z_2 - x_n w_2, \\ x_{n-3} = y_{n-3} - x_{n-1}z_{n-3} - x_n w_{n-3}, \\ x_{n-2} = y_{n-2} - x_{n-1}z_{n-2} - x_n w_{n-2}. \end{cases} \tag{8}$$

By substituting (8) into the second equation of (5), we can also deduce that

$$\tilde{\mathbf{x}} = D_1^{-1}\tilde{\mathbf{f}}_1, \tag{9}$$

where

$$D_1 = \begin{pmatrix} d_{n-1} - \tilde{a}_{n-1}z_1 - \tilde{b}_{n-1}z_{n-3} - b_{n-1}z_{n-2} & a_{n-1} - \tilde{a}_{n-1}w_1 - \tilde{b}_{n-1}w_{n-3} - b_{n-1}w_{n-2} \\ b_n - a_n z_1 - \tilde{a}_n z_2 - \tilde{b}_n z_{n-2} & d_n - a_n w_1 - \tilde{a}_n w_2 - \tilde{b}_n w_{n-2} \end{pmatrix},$$

$$\tilde{\mathbf{f}}_1 = \begin{pmatrix} f_{n-1} - \tilde{a}_{n-1}y_1 - \tilde{b}_{n-1}y_{n-3} - b_{n-1}y_{n-2} \\ f_n - a_n y_1 - \tilde{a}_n y_2 - \tilde{b}_n y_{n-2} \end{pmatrix}.$$

After determining $\tilde{\mathbf{x}}$, we can obtain $\hat{\mathbf{x}}$ by using (6), thus

$$\hat{\mathbf{x}} = \mathbf{y} - x_{n-1} \cdot \mathbf{z} - x_n \cdot \mathbf{w}. \tag{10}$$

In the following, we state the algorithm for solving the linear system (1):

```
Algorithm 2.2
Step 1. Input $P, D, U, V, \hat{\mathbf{f}}, \tilde{\mathbf{f}}, n$.
Step 2. Using Algorithm 2.1, solve the equations:
            $P\mathbf{y} = \hat{\mathbf{f}}, \quad P\mathbf{z} = \mathbf{v}_1, \quad P\mathbf{w} = \mathbf{v}_2.$
Step 3. Compute $\tilde{\mathbf{x}} = (x_{n-1}, x_n)^T$ by using (9) (with symbolic computation
        if necessary).
Step 4. Compute $\hat{\mathbf{x}} = \mathbf{y} - x_{n-1} \cdot \mathbf{z} - x_n \cdot \mathbf{w}$.
Step 5. Output the solution of system: $\mathbf{x} = (\hat{\mathbf{x}}, \tilde{\mathbf{x}})^T$.
```

The computational costs for Algorithm 2.2 are $39n - 75$ ($n \geq 6$), since costs for the steps 2, 3 and 4 are $35n - 116$, 49, and $4n - 8$ respectively. Now, we compare computational costs among Lv and Le's algorithm [19], SYMBNPENTA algorithm [17], Jia, Kong and Sogabe's algorithm [18] and our algorithm in Table 1.

Table 1 Total operations for solving the quasi penta-diagonal linear systems

	Algorithm 1 [19]	Algorithm 2 [17]	Algorithm 3 [18]	Our algorithm
Operations	$58n - 151$	$53n - 142$	$49n + 187$	$39n - 75$

Remark 2.1 Since the processes of solving $P\mathbf{y} = \hat{\mathbf{f}}$, $P\mathbf{z} = \mathbf{v}_1$, and $P\mathbf{w} = \mathbf{v}_2$ are independent of each other in the Step 2 of Algorithm 2.2, they can be solved in parallel.

Remark 2.2 The method in this paper can be directly applied to periodic tridiagonal linear systems [20] by setting $\tilde{a}_i = \tilde{b}_i = 0$ for $i = 1, 2, \ldots, n$.

2.3 Inverse of a quasi penta-diagonal matrix

In this section, we are interested in obtaining the inverse matrix A^{-1}. It should be mentioned that the inverse of a quasi penta-diagonal matrix usually can not be obtained in $O(n)$ operations, since the inverse matrix has n^2 elements and it is not, in general, a quasi penta-diagonal matrix itself.

From the partition of matrix A in (3), it is readily verified that A has the following decomposition

$$A = A_1 A_2 = \begin{pmatrix} P & 0 \\ U & \tilde{D} \end{pmatrix} \begin{pmatrix} I_{n-2} & W \\ 0 & I_2 \end{pmatrix},$$

where $W = P^{-1}V$, $\tilde{D} = D - UW$. Hence,

$$A^{-1} = A_2^{-1} A_1^{-1} = \begin{pmatrix} I_{n-2} & -W \\ 0 & I_2 \end{pmatrix} \begin{pmatrix} P^{-1} & 0 \\ -\tilde{D}^{-1}UP^{-1} & \tilde{D}^{-1} \end{pmatrix}. \tag{11}$$

Let $W = \begin{pmatrix} w_1 & w_2 & \ldots & w_{n-2} \\ w'_1 & w'_2 & \ldots & w'_{n-2} \end{pmatrix}^T$, then we can obtain the following recursive relations between the rows of A^{-1} and A_1^{-1},

$$\begin{cases} R_n = R'_n, \\ R_{n-1} = R'_{n-1}, \\ R_j = R'_j - w_j R'_{n-1} - w'_j R'_n, \end{cases} \tag{12}$$

where $j = n-2, n-3, \ldots, 1$, R_j and R'_j are the j-th row of A^{-1} and A_1^{-1}, respectively. Therefore, we note that the inverse of a quasi penta-diagonal matrix reduces to the inverse of a penta-diagonal matrix. Very recently, Kanal, Baykara and Demiralp have presented an efficient algorithm for inverting a penta-diagonal matrix. The total computational cost of the algorithm is $3n^2 + 54n - 36$. However, we are not going to give the algorithm in this paper. The interested reader may refer to [22] and the references therein.

In the following, we show how to use their algorithm for inverting a quasi penta-diagonal matrix.

```
Algorithm 2.3
Step 1. Compute $P^{-1}$ by using the algorithm in [22].
Step 2. Compute $A_1^{-1}$ and $A_2^{-1}$ by using (11).
Step 3. Compute each row of $A^{-1}$ by using (12).
```

Remark 2.3 Since costs for the steps 1, 2, and 3 are $3n^2 + 54n - 36$, $26n - 21$, and $4n^2 - 10n + 4$, respectively. The total computational operations for Algorithm 2.3 are $7n^2 + 70n - 53$, less than those of the CPINV algorithm [17] whose complexity is $8n^2 + 37n - 235$, when $n \geq 38$. Moreover, Algorithm 2.3 can be regarded as a natural generalization of the algorithm proposed in [22].

Remark 2.4 The method proposed in this section can be very useful for the case that users have the codes of an efficient algorithm for inverting the penta-diagonal matrices.

3 Illustrative examples

In this section, two examples are given for the sake of illustration. All tests were performed in MATLAB 7.12.0.635 (R2011a).

Example 3.1 The first example originating from [17] and is given by

$$\begin{pmatrix} 1&1&1&0&0&1&2\\ 1&1&1&0&0&0&1\\ 1&1&2&1&1&0&0\\ 0&0&1&2&1&1&0\\ 0&0&1&1&2&1&1\\ 1&0&0&1&1&2&2\\ 2&1&0&0&1&2&2 \end{pmatrix} \begin{pmatrix} x_1\\ x_2\\ x_3\\ x_4\\ x_5\\ x_6\\ x_7 \end{pmatrix} = \begin{pmatrix} 6\\ 4\\ 6\\ 5\\ 6\\ 7\\ 8 \end{pmatrix}.$$

The results for Algorithm 2.2 are as follows.

Step 1.

$$P = \begin{pmatrix} 1&1&1&0&0\\ 1&1&1&0&0\\ 1&1&2&1&1\\ 0&0&1&2&1\\ 0&0&1&1&2 \end{pmatrix}, D = \begin{pmatrix} 2&2\\ 2&2 \end{pmatrix},$$

$$U = \begin{pmatrix} 1&0&0&1&1\\ 2&1&0&0&1 \end{pmatrix}, V = \begin{pmatrix} 1&0&0&1&1\\ 2&1&0&0&1 \end{pmatrix}^T,$$

$$\hat{\mathbf{f}} = (6, 4, 6, 5, 6)^T,\ \tilde{\mathbf{f}} = (7, 8)^T.$$

Table 2 Absolute errors and CPU time for Example 3.2

	Algorithms	$n = 500$	$n = 1000$	$n = 2000$	$n = 5000$
$\|\mathbf{x} - \mathbf{x}^*\|$	Algorithm 1 [19]	8.5242e−015	1.1588e−014	1.6048e−014	2.5047e−014
	Algorithm 2 [17]	1.1534e−014	1.2056e−014	1.3038e−014	1.5619e−014
	Algorithm 3 [18]	8.6796e−015	1.0764e−014	1.4941e−014	2.2783e−014
	Our algorithm	4.7701e−015	5.9228e−015	7.7286e−015	1.1562e−014
CPU time (s)	Algorithm 1 [19]	0.077	0.108	0.213	1.042
	Algorithm 2 [17]	0.073	0.077	0.083	0.107
	Algorithm 3 [18]	0.071	0.074	0.079	0.088
	Our algorithm	0.058	0.060	0.063	0.076

Step 2. Solve $P\mathbf{y} = \hat{\mathbf{f}}$, $P\mathbf{z} = \mathbf{v}_1$ and $P\mathbf{w} = \mathbf{v}_2$ by using Algorithm 2.1, then we have:

$$\mathbf{y} = \left(\frac{17t+2}{t}, \frac{-2}{t}, -11, 5, 6\right)^T,$$
$$\mathbf{z} = \left(\frac{6t+1}{t}, \frac{-1}{t}, -5, 2, 2\right)^T,$$
$$\mathbf{w} = \left(\frac{9t+1}{t}, \frac{-1}{t}, -7, 2, 3\right)^T.$$

Step 3–Step 5. Compute $\tilde{\mathbf{x}} = (x_6, x_7)^T$, $\hat{\mathbf{x}} = (x_1, x_2, x_3, x_4, x_5)^T$ by (9) and (10) respectively, then we obtain

$$\mathbf{x} = \left(\frac{10t+3}{8t+3}, \frac{3}{8t+3}, \frac{15t+3}{8t+3}, \frac{2t+3}{8t+3}, \frac{6t+3}{8t+3}, \frac{15t+3}{8t+3}, \frac{4t+3}{8t+3}\right)^T\Bigg|_{t=0}$$
$$= (1, 1, 1, 1, 1, 1, 1)^T.$$

Example 3.2 Next, we consider an $n \times n$ quasi penta-diagonal linear system originating from [18] and is given by

$$\begin{pmatrix} -1 & -1 & 2 & & & 1 & -1 \\ 1 & -1 & -1 & 2 & & & 1 \\ -1 & \ddots & \ddots & \ddots & \ddots & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \ddots & \ddots & 2 \\ -1 & & & -1 & 1 & -1 & -1 \\ 1 & -1 & & & -1 & 1 & -1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_{n-2} \\ x_{n-1} \\ x_n \end{pmatrix} = \begin{pmatrix} 0 \\ 2 \\ 0 \\ \vdots \\ 0 \\ -3 \\ -1 \end{pmatrix}$$

It can be verified that the exact solution is $\mathbf{x}^* = (1, 1, \ldots, 1)^T$. We used Algorithm 2.2 and other three algorithms in [17–19] to compute $\mathbf{x}$. Each absolute error $\|\mathbf{x} - \mathbf{x}^*\|$ and CPU time are provided in Table 2. Here, $\|\cdot\|$ denotes the Euclidean vector norm.

From the Table 2, we know that the CPU time of our algorithm becomes longer with the increase of the order n. As we mentioned in Remark 2.1, the equations (7) can be solved by parallel computation, therefore, the CPU time can be shortened. The specific details of parallel implementation can refer to paper [12,14].

4 Concluding remarks

In this paper, we gave a novel numerical computational algorithm for solving the quasi penta-diagonal linear systems and showed that the computational costs is less than those of three algorithms in [17–19]. The algorithm is reliable, computationally efficient and competitive with other algorithms. Numerical examples are given to illustrate the performance and efficiency of our algorithm. Moreover, Algorithm 2.3 is a new recursive method for computing the inverse of the quasi penta-diagonal matrices.

References

1. M. Znojil, J. Math. Chem. **28**, 140 (2000)
2. J.M. Sanz-Serna, I. Christie, J. Comput. Phys. **67**, 348 (1986)
3. J. Dongarra, S.L. Johnson, Parallel Comput. **5**, 219 (1987)
4. J. Hyman, J. Morel, M. Shashkov, S. Steinberg, Comput. Geosci. **6**, 333 (2002)
5. R.E. Shaw, L.E. Garey, Int. J. Comput. Math. **65**, 121 (1997)
6. S.S. Nemani, L.E. Garey, Int. J. Comput. Math. **79**, 1001 (2002)
7. S.J. Wright, SIAM J. Sci. Stat. Comput. **13**, 742 (1992)
8. W.R. Briley, H. McDonald, J. Community Psychol. **24**, 372 (1977)
9. M.E. Kanal, J. Supercomput. **59**, 1071 (2012)
10. M. Paprzycki, I. Gladwell, Parallel Comput. **17**, 133 (1991)
11. M. Chen, SIAM J. Numer. Anal. **24**, 668 (1987)
12. C.P. Katti, R. Kumari, Appl. Math. Comput. **163**, 1215 (2005)
13. G.H. Golub, C.F. Van Loan, *Matrix Computations*, 3rd edn. (Johns Hopkins University Press, Baltimore, 1996)
14. S.S. Nemani, L.E. Garey, Appl. Math. Comput. **130**, 285 (2002)
15. M.A.L. Marques, A. Castro, G.F. Bertsch, A. Rubio, Comput. Phys. Commun. **151**, 60 (2003)
16. R. Guardiola, J. Ros, J. Comput. Phys. **111**, 374 (1999)
17. M. El-Mikkawy, E. Rahmo, Comput. Math. Appl. **59**, 1386 (2010)
18. J. Jia, Q. Kong, T. Sogabe, Int. J. Comput. Math. **89**, 851 (2012)
19. X.G. Lv, J. Le, Appl. Math. Comput. **204**, 707 (2008)
20. T. Sogabe, Appl. Math. Comput. **202**, 850 (2008)
21. M. El-Mikkawy, Appl. Math. Comput. **202**, 210 (2008)
22. M.E. Kanal, N.A. Baykara, M. Demiralp, J. Math. Chem. **50**, 289 (2012)