Miskolc Mathematical Notes
Vol. 25 (2024), No. 1, pp. 5–19

HU e-ISSN 1787-2413
DOI: 10.18514/MMN.2024.4418

TWICE IS ENOUGH METHOD FOR A^TA CONJUGATE DIRECTIONS AND FOR BICONJUGATE DIRECTIONS

JÓZSEF ABAFFY AND VITTORIO MORIGGIA

Received 22 September, 2022

Abstract. The aim of the article is to increase the accuracy of A^TA conjugate directions and biconjugate directions by applying the twice is enough method to them [14]. The twice is enough algorithm and analysis are due to W. Kahan, cf. Parlett's book [14], pp. 115-117. It was shown that while two consecutive orthogonalization steps improved the accuracy of the computation, further orthogonalization steps failed to provide additional benefit, establishing the principle of "twice is enough". In our previous works, we have introduced the "twice is enough" type algorithms for conjugate directions of positive definite symmetric matrices, cf. [1, 4, 6] and [3]. These results were also generalized for arbitrary symmetric matrices [2]. In the paper [3], we generalized this idea to the computation of conjugate directions. Now, we show that it can be generalized to any matrices; furthermore, we give the conjugate directions of the problem A^TA and the biconjugate directions of any square matrix A. With the help of intensive testing [7], we propose specialized algorithms for these problems. We compared our algorithms to four well-known biconjugate methods that we implemented to obtain the biconjugated directions as well [9]. Using the refined conjugate directions, they can be used to further refine the solution of systems of linear equations iteratively, and to solve $Ax = B$ where B contains all possible right-hand vectors b. We underline that in the computations of the A^TA conjugate directions and the biconjugate directions we do not need to compute the A^TA matrix directly. Another goal of the article, in addition to the applicability of the twice is enough idea to conjugate and biconjugate directions, is to determine the most accurate methods for producing conjugate and biconjugate directions. For this we will need the `vpa` option of MATLAB.

2010 *Mathematics Subject Classification:* 58F15; 58F17; 53C35

Keywords: linear algebra, biconjugate direction

1. INTRODUCTION

In this paper, for the sake of simplicity and brevity, suppose that A is an n by n square and nonsingular matrix, that is $rank(A) = n$. We will see that without these

The second author was supported in part by the MIUR-ex-60% Fund, Grant No. 2020 and 2021.

conditions all our statements remain true. That is consider the problem

$$Ax = b$$

where $A \in \mathbb{R}^{n\times n}$ is an arbitrary matrix, $b \in \mathbb{R}^n$ and $x \in \mathbb{R}^n$. In this article, conjugate directions for A^TA are defined as follows.

Definition 1. Let A be arbitrary nonsingular matrix. Then, we say that the vectors $p_1,\dots,p_n$ are A^TA conjugates, if

$$p_i^T A^T A p_j = \begin{cases} 0, & \text{if } i \neq j \\ \text{non zero}, & \text{otherwise.} \end{cases}$$

Furthermore, we define the biconjugate directions as follows.

Definition 2. Let A be arbitrary nonsingular matrix. Then, we say that the vectors $p_1,\dots,p_n$ and $t_1,\dots,t_n$ are A biconjugate if

$$t_i^T A p_j = \begin{cases} 0, & \text{if } i \neq j \\ \text{non zero}, & \text{otherwise.} \end{cases}$$

Note that if we have the $P = [p_1,\dots,p_n]$ conjugate directions to A^TA then we can compute the $T = [t_1,\dots,t_n]$ matrix from it as

$$t_i = Ap_i, \;\; i = 1,\dots,n$$

and therefore we also have the biconjugate directions of A.

Remark 1. We already noted in our earlier papers that to determine the most accurate algorithms, we need to have the exact conjugate directions. In double-precision arithmetic, we need the knowledge of exact conjugate directions in order to determine the best methods for calculating conjugate directions. To do this, we use MATLAB's VPA option for 50 digits, because we believe that its first 16 digits are accurate. We did not choose more than 50 digits because this calculation is very time-consuming and our personal computer is not very efficient. Matrix P contains the computed $p_i,\; i = 1,\dots,n$ vectors in columns and similarly S contains the same directions computed by `vpa` for 50 decimals. Then the differences B are

$$\begin{aligned} B &= S^T(A^TA)S - \text{vpa}(P^T)(A^TA)\text{vpa}(P) \\ yB &= \max_{i,j}(\text{abs}(\text{double}(B))) \end{aligned}$$

In the two formulas, we are talking about determining the difference matrix with the help of the matrix containing the precise S conjugate directions calculated with vpa and the matrices calculated with the conjugate directions computed in double precision. If the difference were zero in double precision, then we have the best method. This difference, i.e. deviation, is calculated using the formula in the next line. The log10 value of the max deviation gives the number of exact digits, see below.

Let us now move on to checking the accuracy of the biconjugate vectors t_i. Similarly, we have to calculate the accuracy of the biconjugate vectors t_i, i.e

$$\begin{aligned} TTP &= AP \qquad (1.1)\\ TT &= \text{vpa}(AS)\\ BB &= TT^T AS - \text{vpa}(TTP^T)A\,\text{vpa}(P)\\ yBB &= \max_{i,j}(\text{abs}(\text{double}(BB))) \end{aligned}$$

The minimum number of accurate digits of the conjugate vectors p_i and t_i, for $i = 1,\dots,n$, are computed by

$$\begin{aligned} ym &= -\log_{10}(yB)\\ tm &= -\log_{10}(yBB) \end{aligned}$$

Remark 2. Finally, we note that in the testing we do not apply any precondition method because we want to find the most accurate algorithms even for difficult problems and without manipulating them in any way before usage.

2. The ABS class and the "Twice is enough" algorithm

In the next subsection 2.1 we present the ABS class, then the ABS Conjugate Direction Parlett- Kahan type (*ABS_CD_PK*) in subsection 2.2.ű

2.1. *Description of the ABS class with the ABS_CD_PK*

We present the general ABS class where the matrix A can even be rectangular. Let us consider the following scaled system

$$V^T Ax = V^T b$$

where $A \in \mathbb{R}^{n\times n}$ is an arbitrary matrix, $V \in \mathbb{R}^{n\times n}$ is an arbitrary non-singular matrix, $b \in \mathbb{R}^n$ and $x \in \mathbb{R}^n$.

The class of the scaled ABS algorithm [5] is as follows.

ABS class with *ABS_CD_PK*

Step 1. Set $x_1 \in \mathbb{R}^n$, $H_1 = I \in \mathbb{R}^{n\times n}$ where I is the unit matrix, $i = 1$, $\kappa = 1.25$ and $iflag = 0$.

Step 2. Let $v_i \in \mathbb{R}^n$ be arbitrary, except that $v_1,\dots,v_i$ be linearly independent. Compute the residual error vector $r_i = Ax_i - b$. If $r_i = 0$, stop and x_i solves the system. Otherwise,compute the scalar $\tau_i = v_i^T r_i$ and the vector $s_i = H_i A^T v_i$.

Step 3. If $s_i \neq 0$, go to Step 4; if $s_i = 0$ and $\tau_i = 0$, set $x_{i+1} = x_i$, $H_{i+1} = H_i$, $iflag = iflag + 1$, and if $i < n$, go to Step 6; otherwise stop; if $s_i = 0$ and $\tau_i \neq 0$, set $iflag = -i$ and stop.

Step 4. Compute the search direction p_i by

$$p_i = H_i^T z_i$$

where $z_i \in \mathbb{R}^n$ is arbitrary saving for $z_i^T H_i A^T v_i \neq 0$.

$$p_i = ABS\_CD\_PK(H_i, A, \kappa, p_i, z_i).$$

Step 5. Update the approximation of the solution by

$$x_{i+1} = x_i - \alpha_i p_i$$

where the step size α_i is given by

$$\alpha_i = \frac{\tau_i}{v_i^T A p_i}$$

If $i = n$, stop and x_{n+1} is the solution of the equations.

Step 6. Update the matrix H_i by

$$H_{i+1} = H_i - \frac{H_i A^T v_i w_i^T H_i}{w_i^T H_i A^T v_i} \tag{2.1}$$

where w_i is arbitrary, but the denominator must be non-zero.

Step 7. Set $i = i + 1$ and go to Step 2.

Some important theorems are following, cf. [5] page 95pp.

Theorem 1. *The residual vector $r_{i+1} = Ax_{i+1} - b$ is orthogonal to the first i columns of V i.e. $V^{iT} r_{i+1} = 0$*

Theorem 2. *For $1 \leq i \leq n$ the vectors $A^T v_1, \ldots, A^T v_i$ are non zero, linearly independent and a basis of the null space of H_{i+1}. The vectors $w_1, \ldots w_i$ are non-zero, linearly independent, and a basis of the null space of H_{i+1}.*

The $ABS\_CD\_PK()$ function returns the new p_i vector in case projection is required, see below in Section 2.2. The properties of this algorithm can be found in [5]. Hereinafter, we use the index i only when needed. Before we turn to our "twice is enough" type reprojection algorithm for the general matrix, we remark that in an earlier papers we developed it for positive definite symmetric matrices (see [1] and [2]), for symmetric matrices in [8] as well as to the computation of conjugate directions in [3]. In the paper [8], we presented an intensive testing for symmetric but singular, indefinite and singular plus indefinite test matrices. In that work the twice is enough type algorithm *ABS_CD_PK* was defined, like in the original twice is enough algorithm [14], to control also the linear dependency. Since the symmetry of $A^T A$ is always guaranteed and it is not necessary to care about the symmetry of A, like in the Gaussian problem. In this paper we test some non-symmetric test problems from MATLAB R2021b Gallery.

2.2. *Twice is enough type reprojection in ABS class*

The twice is enough algorithm and analysis are due to W. Kahan, see pages 115-117 of Parlett's book [14]. In our previous works, we have introduced the "twice is enough" type algorithms for conjugate directions of positive definite symmetric matrices, cf. [1, 3, 4, 6]. These results were also generalized for arbitrary symmetric matrices in [2]. In the paper [3], we generalized this idea to the computation of conjugate directions. Now, we show that it can be generalized to any matrices; furthermore, we give the conjugate directions of the problem A^TA and the biconjugate directions of any square matrix A.

We first present the Parlett-Kahan algorithm [14]:

"Suppose that a simple subprogram *orthog* is available which, given $y \neq o$ and z, computes an approximation x' to $p \equiv z - y(y \cdot z/||y||^2)$. Let the error $e'(\equiv x' - p)$ satisfy $||e'|| \leq \varepsilon ||z||$ for some tiny positive ε independent of y and z. Let κ any fixed value in the range $[1/(0.83-\varepsilon), 0.83/\varepsilon]$.

Algorithm. First call orthog(y, z, x') to get x'.

Case 1. If $||x'|| \geq ||z||/\kappa$ accept $x = x'$ and $e = e'$.
Otherwise call orthog(y, x', x'') to get x'' with error $e'' \equiv x'' - (x' - yy \cdot x'/||y||^2)$ satisfying $||e''|| \leq \varepsilon ||x'||$ and proceed to Case 2.

Case 2. If $||x''|| \geq ||x'||/\kappa$ accept $x = x''$, $e = x'' - p$.

Case 3. If $||x''|| < ||x'||/\kappa$ accept $x = o$, $e = -p$."

together with its Lemma:

Lemma. The vector x computed by the algorithm ensure that $||e|| \leq (1 + 1/\kappa)\varepsilon ||z||$ and $||y \cdot x|| \leq \kappa\varepsilon ||y|| ||x||$."

Our ABS_CD_PK algorithm is as follows. Using the notation of the original Parlett–Kahan algorithm in [14], let $x = H^T z$ and κ is fixed in the range $[1/(0.83 eps), 0.83/eps]$. Furthermore, let e' represents the error vector between the computed x' and its accurate value, see [1] and [4, pp. 41-51].

Algorithm 1. ABS Conjugate Direction of Parlett–Kahan-type for arbitrary matrices ($p = ABS\_CD\_PK(H, A, \kappa, p, z)$.

Compute Let $x = p$ (because we use the Parlett-Kahan notations) and $x^{'} = Ax$.

Case 1. If $x'^T A^T A x^{'} > z^T A^T A z/\kappa$, accept $x = x'$ and $e = e'$, otherwise compute $x'' = H^T x'$ to get x'' with error e'' and go to Case 2.

Case 2. If $x''^T A^T A x'' \geq x'^T A^T A x''/\kappa$ accept $x = x''$ and $e = e''$.

Case 3. Otherwise, accept $x = 0$.

Remark 3. Here, the indefiniteness is also possible. Just as in the proof of the algorithm, the positivity of these values was supposed, therefore no any new proof is needed, because here the scalar products are ≥ 0. Furthermore, we note that in the ABS algorithms and in this *ABS_CD_PK*, the singularity is well-treated.

Remark 4. We underline again that this procedure works for any matrices.

The theorem for the "twice is enough" type reprojections is as follows

Theorem 3. *The vector x computed by the ABS_CD_PK algorithm ensures that for the error vector e*

$$(e^T A^T Ae) \leq eps(z^T A^T Az) + O(eps^2)$$

and

$$(p_0^T A^T Ax) \leq \kappa eps(p_0^T A^T Ap_0)(x^T A^T Ax) + O(eps^2).$$

where p_0 is any other $A^T A$ conjugate direction.

For the proof, see [1],[4, pp. 41-51]. In this work we run our experiments with $\kappa = 100$ and $\kappa = 1000$ without any relevant improvement, then we accept $\kappa = 1.25$ as suggested in Parlett's book [14]. We have to note that the optimal value of κ is unknown and we believe that it is problem dependent.

3. THE CONSIDERED ABS SUBCLASS

In this paper, we consider the subclass S3 of ABS. This is because we can easily derive the conjugate directions of the matrix $A^T A$ or the biconjugate directions of A from this subclass.

3.1. *The S3 subclass*

In the S3 subclass, vector v_i is defined as

$$v_i = Ap_i$$

Theorem 4. *Let A be square nonsingular. Then the scaled ABS class with $v_i = Ap_i$ generates $A^T A$-conjugate search vectors and x_{i+1} minimizes, over the linear variety $x_1 + Span(p_1, \dots, p_i)$ the quadratic function $F(x) = (x - x^+)A^T A(x - x^+)$ where x^+ is the solution of the linear system.*

For the proof see Theorem 8.11 in [5]. Again, we note that all our algorithms treat the singularity, therefore this condition of the theorem is just a formality. Also note that if the projection vector p_i is zero in a certain step (less than `eps`, where constant `eps` in MATLAB is the distance from 1.0 to the next larger double precision number) then all further p_i vectors are zeros, because in the matrix update H_i the rank one term becomes zero.

In [7] we tested 61 methods in S3, while in this article we present only three best special cases according to our tests. This means that the number of exact digits is the highest for the conjugate direction with the worst accuracy for the best algorithms.

1) $z_i = e_i, \quad w_i = A^T v_i$ (S3e) The projection matrix H_i is symmetric.

2) $z_i = e_i$, $w_i = e_i$ (S3ee) The projection matrix H_i is not symmetric.
3) $z_i = e_i$, $w_i = p_i$ (S3ep) The projection matrix H_i is not symmetric.

3.2. *Classical algorithms*

As we did not find algorithms in MATLAb R2021b that produce biconjugate directions for a matrix A, nor did we find them commercially, we implemented four methods to calculate the solution of linear systems of equations with biconjugate directions: BiCG in Broyden et al. [9,10,13], p. 52; BiCGL in Broyden et al. [9,10,13], p. 53; BiCR in Broyden et al. [9,12], p. 60 and BiCRL in Broyden et al. [9,12], p. 61. Algorithm descriptions can also be found in [7] where all the source codes written in MATLAB can also be found in the Appendix. Note that the bicgstab algorithm and its variants do not give biconjugate directions, see for example [15] and [17].

4. TEST PROBLEMS

Before we list the problems that are being tested, we have to note two things.
For all problems, we computed the quantity

$$q = \text{cond}_\infty(A^T A) \times eps \tag{4.1}$$

where *cond* is the condition number, ∞ means infinite norm and `eps` is MATLAB eps. If the value of q reaches 1 for a matrix A, then the algorithm that solves the system of linear equations cannot be criticized if it does not provide an acceptable solution. See Wilkinson [16] or Golub and Van Loan [11], Therefore, in the sequel we compute q for different dimensions of A and the dimension where q reaches 1 defines the upper bound of the feasible interval in which we test our methods.

In the followings, the vector $x_1 = 0$ is chosen for each test problem and for each method. The right side of the system of equations b is determined by generating a solution with the built-in `rand()` function, which is valid for all methods, including the known ones. We did this for a fair comparison.

4.1. *Chosen test problems from the MATLAB Gallery*

In this subsection we report each problem followed by the interval $[a,b]$, where a is the smallest tested dimension and b is the largest one. In the case this value becomes larger than 3000, then we give the test interval $[100,110]$ because we observed that q is practically the same for all of these dimensions and the memory and computation capabilities of the computer we used is limited. Detailed description about the test problems can be found in MATLAB Gallery.

We have chosen the following non-symmetric matrices, most of them from the gallery of MATLAB R2021b: *binomial* [6,27], *chebspec* [4,10], *chebvand* [5,10], *chow* [5,10], *circul* [100,110], *clement* [5,50], *cycol* [5,10], *dramadah* [5,28], *forsythe* [100,110], *frank* [5,10], *grcar* [100,110], *invhess* [100,110], *invol* [2,5], *jordbloc* [100,110], *kahan* [5,41], *krylov* [5,12], *leslie* [100,110], *lesp* [100,110], *lotkin* [3,6], *parter* [100,110], *qmult* [100,110], *rand* [100,110], *randcolu* [100,110], *randhess* [100,110], *randjorth* [5,10], *rando* [100,110], *randsvd* [5,10], *riemann* [100,110], *toeppen* [100,110], *triw* [5,22]. Two more test problems are not in the Gallery of MATLAB: *gfpp* [100,110], and *makejcf* [100,107].

5. Test results

In this section we report our test results of the above described problems for some special case only, because of the limited scope of this article, detailed tables can be found in [7], p. 13. In that paper, we present the accuracy of the projection vectors for P and T matrices for all the three cases: without any reprojection, with the *ABS_CD_PK* method and with reprojections in all steps. The linear dependency is threshold by $2eps$.

First, we present the results of the small-dimension problems, then those where the dimension is between 100-110. We notice again that $\kappa = 1.25$, after a short testing with others κ, can be accepted as well as proposed by Parlett and Kahan. The experiments were performed on a personal computer with Intel(R) Core(TM) i7 2.67 GHz CPU with integrated graphics, 6 GB RAM 64-bit Operating System running Microsoft Windows 10 Professional and MATLAB version R2021b.

5.1. *Test results of the small dimensions problems*

Here, we present test results of 4 classical methods defined above that is the so called BiCG, BiCGL, BiCR and the BiCRL methods. The accuracy of the biconjugate directions is computed by (1.1). The BiCGL method seems to be the best among them, but also this method gives worse results (see Table 1 below) than many our methods. Therefore we did not make any experiment with "high" dimension.

In Tables 1 instead of zero correct digits we often find a negative number of accurate digits. Let see an example to better understand the reason that the worst number of accurate digits can be negative in the diagonal.

Suppose that the worst number of accurate digits is in the diagonal. Furthermore, suppose that all the elements in the diagonal are accurate for all digits except one element. Let the accurate diagonal element in this case 555555555555555 be of 15 digits and the computed one differs from it in the last digits, say with a value of 1.5. that is 555555555555556.5. That is the deviation is 1.5. In this case the minus logarithm in base 10 is $-\log_{10}(1.5) = -0.1761$. On the other hand, the number of accurate digits is 14, which is more than acceptable. Therefore, it is obvious that if the positions of the negative worst-case values are in the diagonals, we need to examine the number of exact digits. In our cases, with the negative worst accuracy in the diagonal, the number of the accurate digits are not good.

In Table 1 below, only the diagonal negative numbers were left and the rest was replaced by zero. We can see that all 4 classical methods usually give a negative or small positive number for the number of exact digits while we show the same

TABLE 1. Worst number of accurate digits of the projection vector and the number of reprojections in maximum dimensions by ABS_CD_PK

Methods	**binomial**	**chebspec**	**chebvand**	**chow**	**clement**	**cycol**	**dramadah**	**frank**
	dim = 6-27	*dim = 4-10*	*dim = 5-10*	*dim = 5-10*	*dim = 5-50*	*dim = 5-10*	*dim = 5-28*	*dim = 5-10*
BiCG	0.00	0.00	0.00	-1.06	9.36	11.04	0.00	1.77
BiCGL	-inf	0.15	6.60	9.18	-115.33	crash d.	-8.77	0.50
BiCR	0.00	12.59	4.50	5.96	1.41	crash d.	1.05	4.22
BiCRL	-inf	0.00	9.07	4.45	-124.36	crash d.	-10.52	0.00
S3e	3.01 (22)	14.07 (6)	14.07 (9)	16.00 (8)	12.46 (32)	16.00 (4)	14.64 (26)	14.59 (9)
S3ee	7.07 (25)	14.13 (9)	13.49 (9)	16.00 (8)	0.00 (28)	16.00 (7)	15.28 (25)	14.87 (9)
S3ep	7.11 (25)	14.03 (8)	13.45 (9)	16.00 (8)	0.00 (28)	16.00 (7)	15.24 (25)	14.81.(9)

Methods	**invol**	**kahan**	**krylov**	**lotkin**	**randjorth**	**randsvd**	**triw**
	dim = 2-5	*dim = 5-41*	*dim = 5-12*	*dim = 3-6*	*dim = 2-10*	*dim = 2-10*	*dim = 5-22*
BiCG	9.28	0.00	0.00	3.03	0.00	-0.74	5.64
BiCGL	7.76	3.56	-70.05	13.01	-41.99	16.00	6.26
BiCR	0.00	0.00	0.00	7.08	0.00	1.31	0.01
BiCRL	0.00	0.56	-127.39	13.05	-49.68	16.00	0.00
S3e	12.28 (4)	15.34 (34)	11.37 (9)	14.52 (5)	5.21 (8)	10.66 (9)	15.03 (19)
S3ee	12.03 (4)	15.18 (37)	12.13 (10)	14.07 (5)	5.30 (7)	11.66 (9)	16.00 (21)
S3ep	11.75 (4)	15.21 (37)	12.71 (1)	14.68 (5)	5.64 (8)	9.65 (9)	16.00 (21)

worst number of accurate digits to the projection vectors P using the ABS_CD_PK algorithm. In parentheses, we give the number of reprojections for the maximum dimensions.

In Table 1 we present the number of worst deviation in two decimal digit precision. The values are the minimum value in the worst number of accurate digits considering all dimensions of the projection vectors.

Note that in case of problem *clement* at least the solutions are not very bad for S3ee and S3ep, $1.377\mathrm{e}{-008}$ and $1.174\mathrm{e}{-008}$ respectively in maximum dimensions.

Moreover, the methods BiCG, BiCR and BiCRL in case *cycol* crash down by divison by zero in the MATLAB function `mrdivide`, which did not happen in MATLAB R2007b with the same code, see our working paper [7].

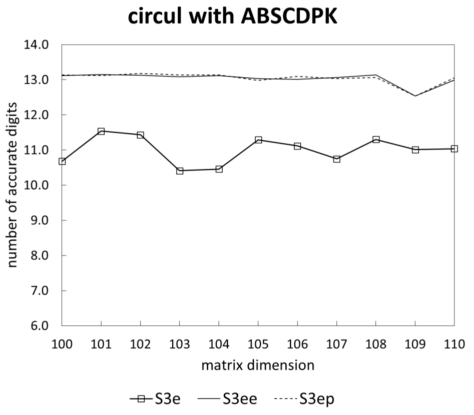
Furthermore, in problem *krylov* not only the p vectors are very bad, cf. methods BiCGL, BICRL, but even the solutions of the linear systems are not good. In the method BiCG of case *randsvd* the maximum norm of the worst residuals equals 0.06452. The *randsvd* is the only problem where the BiCGL and BiCRL are definitely better than our chosen methods.

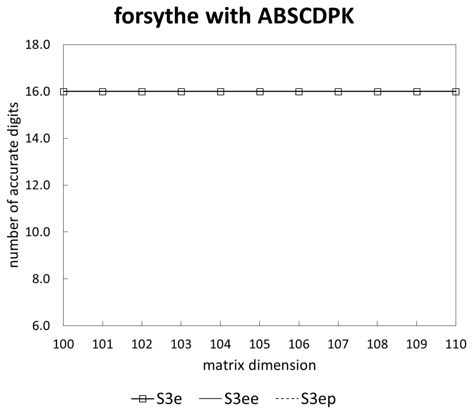
Note that in the method BiCGL of case *randjorth* also the maximum norm of the residuals are bad, while in the method BiCRL it is 0.0007351.

We conclude from Table 1 that the three ABS methods are definitely much better than the classical ones. Therefore, we skipped to test all the classical methods for large-scale problems.

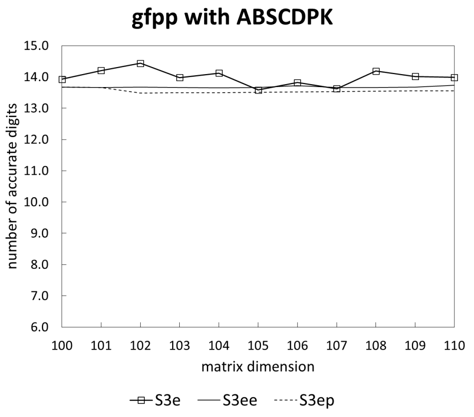
5.2. *Test results of problems for dimensions 100-110*

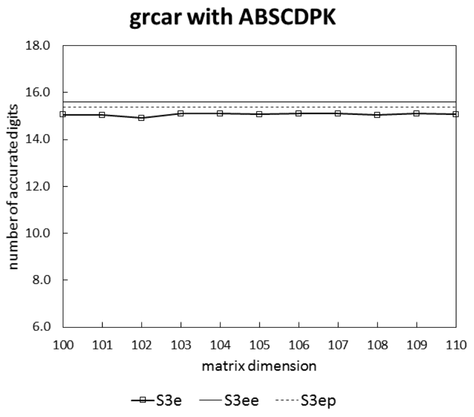
The ”big” dimensional nonsymmetric problems are: *circul*, *forsythe*, *grcar*, *invhess*, *jordbloc*, *leslie*, *lesp*, *parter*, *qmult*, *rand*, *rndcolu*, *randhess*, *rando*, *riemann*, *gfpp*, *makejcf* and *toeppd*.



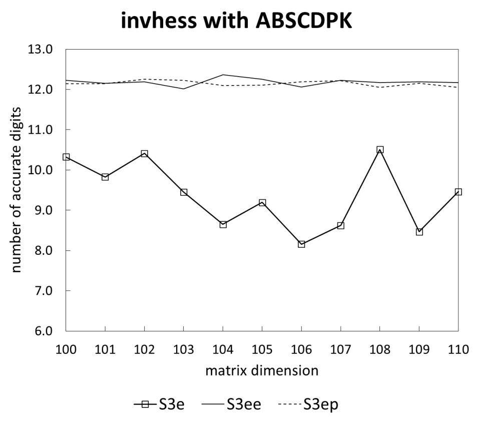


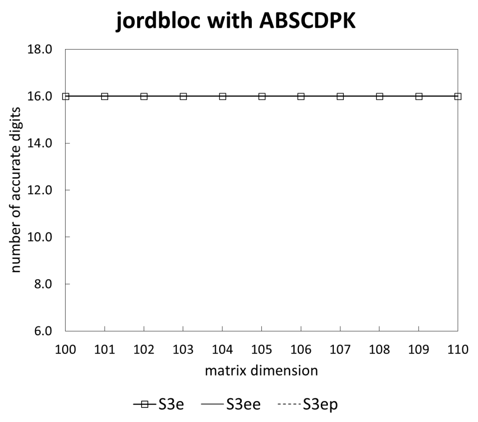
The number of reprojections in maximal dimension in S3e, S3ee, S3ep are 109, 109, 109 respectively for case *circul*, which means reprojections in all steps, and 0, 0, 0 for case *forsythe*, which means no reprojection at all, in fact all the digits are correct since the problem is too easy. For details see [7].



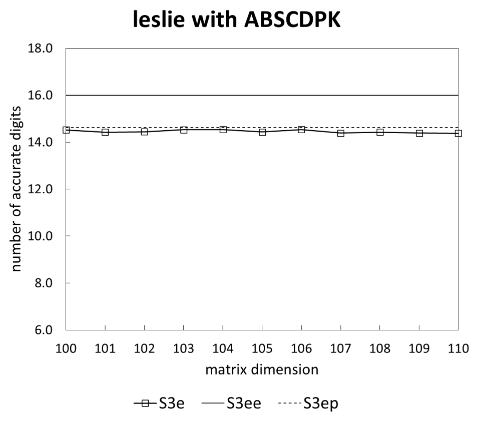


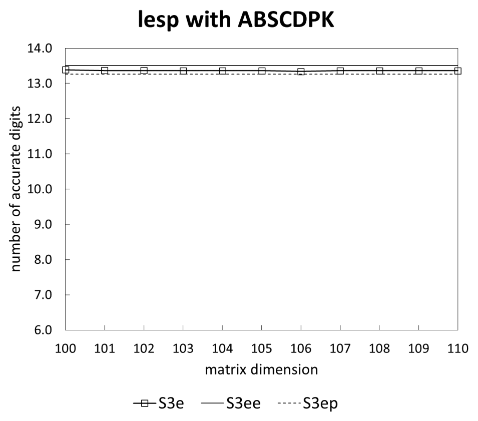
The number of reprojections in maximal dimension in S3e, S3ee, S3ep are 108, 106, 106 respectively for case *gfpp* and 107, 0, 0 for *grcar*.



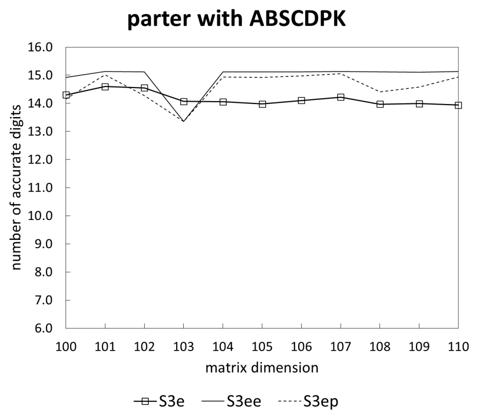


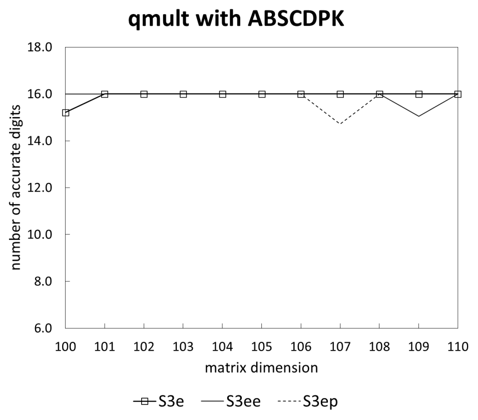
The number of reprojections in maximal dimension in S3e, S3ee, S3ep are 104, 109, 109 respectively for case *invhess* and 109, 109, 109 for case *jordbloc*.



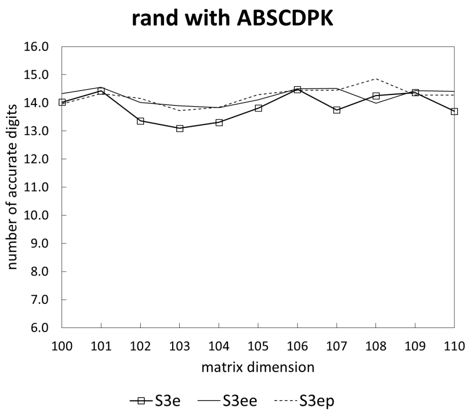


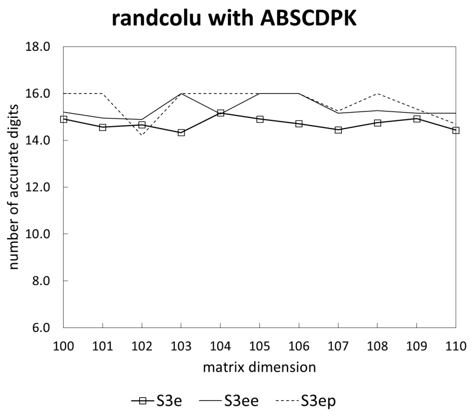
The number of reprojections in maximal dimension in S3e, S3ee, S3ep are 109, 107, 107 respectively for case *leslie* and 109, 0, 0 for case *lesp*.



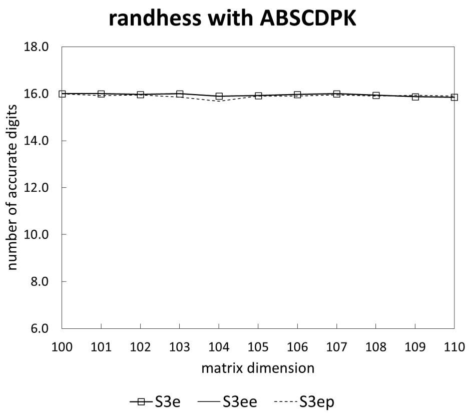


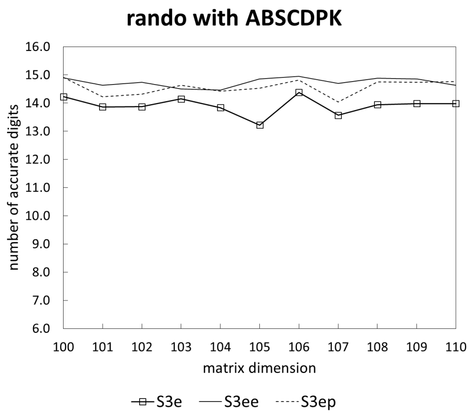
The number of reprojections in maximal dimension in S3e, S3ee, S3ep are 5, 0, 0 respectively for case *parter* and 0, 0, 0 for case *qmult*.



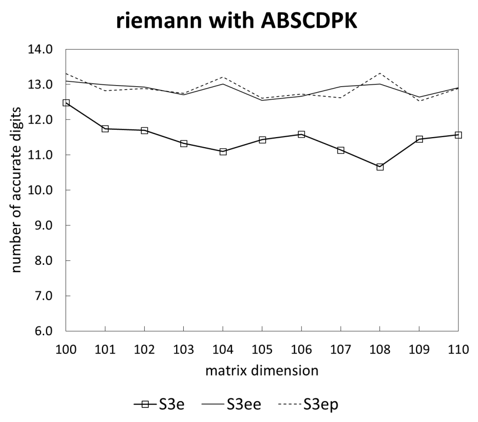


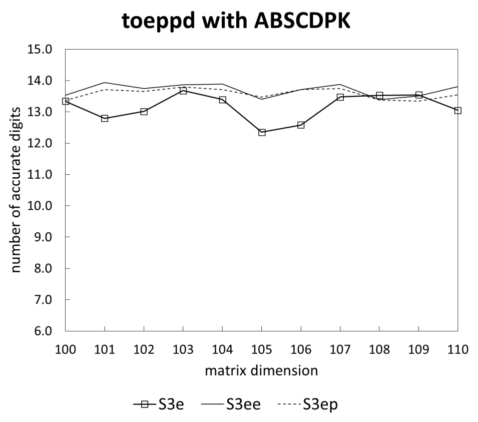
The number of reprojections in maximal dimension in S3e, S3ee, S3ep are 109, 109, 109 respectively for case *rand* and 79, 57, 55 for case *randcolu*.



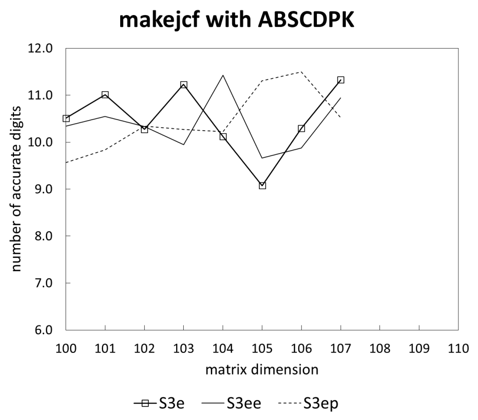


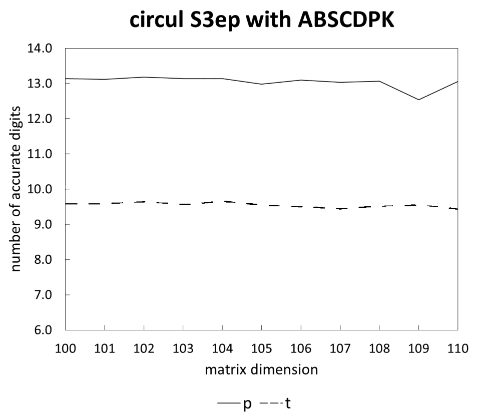
The number of reprojections in maximal dimension in S3e, S3ee, S3ep are 0, 0, 0 respectively for case *randhess* and 109, 108, 108 for case *rando*.





The number of reprojections in maximal dimension in S3e, S3ee, S3ep are, respectively, 66, 14, 14 for case *riemann* and 93, 78, 78 for case *toeppd*,





For case *makejcf* the number of reprojections in maximal dimension in S3e, S3ee, S3ep are 105, 106, 106, respectively.

Figure named "circul S3ep with ABSCDPK" shows that the number of accurate digits in worst t vector can be more than 3 digits less than the p_i vectors, $i = 1, \ldots, n$. When the accuracy of the t conjugate directions are important, we suggest to use our algorithms for the matrix A^T. In fact, using S3e, S3ee, S3ep the p_i vectors are more accurate and are also the t_i vectors of the original problem, $i = 1, \ldots, n$. Note further that these t_i vectors are not used in the computations of the solutions of the linear system of equations.

As it can be seen, some figures show strange drop in the curves. We explained it describing a case in our working paper [7].

5.3. *Conclusion*

In this paper we tested three ABS algorthms (S3e, S3ee and S3ep) using 32 test-problems to decide which is the best using the twice is enough theorem for biconjugate directions. These methods were compared by 4 well-known methods (BiCG, BiCGL, BiCR and BiCRL). Based on these test problems, we found that the best two methods are S3ee and S3ep to compute the p_i, for $i = 1, \dots, n$, conjugate and biconjugate directions accurately.

References

[1] J. Abaffy, "Reprojection of the conjugate directions in abs classes part i," *Acta Polytechnica Hungarica*, vol. 13, no. 3, pp. 7–24, 2016.

[2] J. Abaffy, "Reprojection of the conjugate directions in abs classes part ii," *Working papers MEQ. Quantitative Methods Series*, pp. 1–21, 2017.

[3] J. Abaffy, "A new reprojection of the conjugate directions," *Numerical Algebra Control and Optimization*, vol. 9, no. 2, pp. 157–171, 2019, doi: 10.3934/naco2019012.

[4] J. Abaffy, E. Feng, and A. Galántai, *Introduction to ABS methods for equations and optimization PART I*. GlobeEdit, (Ominiscriptum Publishing Group) Germany, 2017.

[5] J. Abaffy and E. Spedicato, *ABS Projections Algorithms Mathematical Techniques for Linear and Nonlinear Algebraic Equations*. Ellis Horwood Ltd, Chichester, England, 1989.

[6] J. Abaffy, Z. Xia, and L. Zhang, *Introduction to ABS methods for equations and optimization PART II*. GlobeEdit, (Ominiscriptum Publishing Group) Germany, 2017.

[7] J. Abaffy, D. Kiss, and V. Moriggia, "Twice is enough method for $A^T A$ conjugate directions and for biconjugate directions," *http://users.nik.uni-obuda.hu/kissdani/publications//Abaffy_ATA_ABS_CD_2019.pdf*, pp. 1–167, 2019.

[8] J. Abaffy, L. Mohácsi, and V. Moriggia, "Testing different abs algorithms for computation of conjugate directions," *QDSAEMQ012018*, vol. 1, pp. 1–603, 2018.

[9] C. Broyden and M. T. Vespucci, *Krylov Solvers for Linear Algebraic Systems*. Elsevier., 2004.

[10] R. Fletcher, *Conjugate gradients methods for indefinite system In G. A. Watson, editor, Numerical Analysis Dundee*. Berlin Heidelberg Springer, 1976, vol. 506.

[11] G. H. Golub and C. F. Van Loan, *Matrix computation 1983*. North Oxford Academic Press, Oxford, 1983.

[12] C. J. Hegedűs, "Generating conjugate directions for arbitrary matrices by matrix equations.-i," *Comput. Math. Appl.*, vol. 21, no. 1, pp. 73–86, 1991, doi: 10.1016/0898-1221(91)90233-T.

[13] C. Lanczos, "Solution of systems of linear equations by minimized iterations," *J. Res.Natlr. Bur. Stand.*, vol. 49, pp. 33–53, 1952, doi: 10.6028/jres.049.006.

[14] B. Parlett, *The symmetric eigenvalue problem*. Englewood Cliffs, N. J. Prentice-Hal, 1980. doi: org/10.1002/zamm.19810610714.

[15] H. H. A. Van der Vorst, "Bi-cgstab: A fast and smoothly converging variant of bi-cg for the solution of nonsymmetric linear systems," *SIAM. J. Sci Stat Comput.*, vol. 13, no. 2, pp. 631–644, 1992, doi: rg/10.1137/0913035.

[16] J. Wilkinson, *Rounding Errors in Algebraic Processes*. Prentice-Hall Inc., Englewood Cliffs (NJ), 1963.

[17] S.-L. Zhang, "Gpbi-cg: Generalized product-type methods based on bi-cg for solving nonsymmetric linear systems," *SIAM J. Sci. Comput.*, vol. 18, no. 2, pp. 537–551, 1997.

Authors' addresses

József Abaffy
(**Corresponding author**) Óbuda University, Institute of Applied Mathematics, Bécsi út 96/b, 1034 Budapest, Hungary
E-mail address: `abaffy.jozsef@nik.uni-obuda.hu`

Vittorio Moriggia
University of Bergamo, Department of Economics, via dei Caniana, 2, 24127 Bergamo, Italy
E-mail address: `vittorio.moriggia@unibg.it`