Comp. & Maths. with Appls. Vol. 8, No. 4, pp. 329–337, 1982
Printed in Great Britain.

0097–4943/82/040329–09$03.00/0

FOUR DIMENSIONAL x-PROJECTION METHOD (WITH ACCELERATION TECHNIQUES) FOR SOLVING SYSTEMS OF LINEAR EQUATIONS

ROGER L. WAINWRIGHT
Division of Mathematical Sciences, The University of Tulsa, Tulsa, OK 74104, U.S.A.

Communicated by E. Y. Rodin

(*Received May* 1981; *revised December* 1981)

Abstract—The solution of linear systems of equations using a 4-dimensional x-projection method is presented. At each step of the iterative process the approximate solution vector is projected to a point in the intersection of four of the hyperplanes of the linear system. Nonsingularity of the coefficient matrix is the only requirement for convergence. An algorithm is presented to select four of the hyperplanes to project the approximate solution vector at each step. The algorithm is quasi-optimal since the hyperplanes, which are determined by the row vectors of the coefficient matrix, are selected *a priori*. This is shown to significantly reduce the number of cycles required for convergence. We observe that in some cases the ratio of the change vectors of the approximate solution vectors after some number of cycles becomes a constant. Thus, when this occurs a simple geometric acceleration can be applied to calculate the solution directly. Geometric acceleration can significantly reduce computation time and improve the accuracy of the solution by orders of magnitude. The 4-dimensional x-projection method was tested against the 3-dimensional x-projection method using random and Hilbert coefficient matrices and proved superior (less CPU time required) in nearly every case.

1. INTRODUCTION

The purpose of this paper is to develop a 4-dimensional x-projection method for solving the equation

$$Ax = b \tag{1.1}$$

where A is a nonsingular matrix of order n and x and b are n-vectors. In this paper we also develop an acceleration technique for a 4-dimensional x-projection method. The paper is divided into five sections. In Section 2, a 4-dimensional x-projection method is developed and in Section 3 a hyperplane selection criteria is developed to accelerate the rate of convergence of the 4-dimensional x-projection method. Test cases and comparisons are presented in Section 4. Concluding remarks are presented in Section 5.

Each equation of (1.1) (i.e. $(a^i, x) = b_i$ where a^i is the ith row of A) geometrically defines an $(n-1)$-dimensional hyperplane called the ith hyperplane which is orthogonal to a^i. (a^i, x) denotes the inner product the two vectors, a^i and x. At each iterative step of an m-dimensional x-projection method the appxoximate solution vector, x, is projected onto the intersection of $m(n-m)$-dimensional hyperplanes in a direction orthogonal to the m hyperplanes. The m hyperplanes are determined by m of the rows of A. Since the approximate solution vector, x, is projected at each iterative step these methods are called x-projection methods. In order to accomplish the projection at each step, a symmetric system of m equations and m unknowns must be solved. Once the hyperplanes have been selected the coefficient matrix for the system is determined and does not change. Thus it is calculated once.

In the 1-dimensional case Pizer [2] proves this process always converges for arbitrary initial approximate solution vector, assuming the non-singularity of A. Wainwright [4, 5] gives a detailed development of the 1-dimensional, 2-dimensional and 3-dimensional x-projection methods for solving systems of linear equations including acceleration techniques. This paper is an extension of that work to four dimensions. A discussion of projection methods in general is given in [3].

2. 4-DIMENSIONAL X-PROJECTION METHOD

Consider the following nonstationary iterative scheme for solving (1.1) that makes successive changes to the components of a starting vector, x^0

$$x^{k+1} = x^k + \alpha_k \omega^k + \beta_k \gamma^k + \lambda_k \sigma^k + \mu_k \phi^k$$

method. Thus

$$x^{k+1} = x^k + \alpha a^i + \beta a^j + \lambda a^p + \mu a^q$$

and

$$(r^{k+1}, r^{k+1}) = (b - Ax^{k+1}, b - Ax^{k+1})$$

or equivalently

$$\begin{aligned}
&(r^k, r^k) - (r^{k+1}, r^{k+1}) \\
&= 2\alpha(r^k, Aa^i) + 2\beta(r^k, Aa^j) + 2\lambda(r^k, Aa^p) + 2\mu(r^k, Aa^q) \\
&\quad - \alpha^2(Aa^i, Aa^i) - \beta^2(Aa^j, Aa^j) - \lambda^2(Aa^p, Aa^p) - \mu^2(Aa^q, Aa^q) \\
&\quad - 2\alpha\beta(Aa^i, Aa^j) - 2\alpha\lambda(Aa^i, Aa^p) - 2\alpha\mu(Aa^i, Aa^q) \\
&\quad - 2\beta\lambda(Aa^j, Aa^p) - 2\beta\mu(Aa^j, Aa^q) - 2\lambda\mu(Aa^p, Aa^q).
\end{aligned}$$

Substituting (2.9) for α, β, λ and μ and $\cos\theta_{ij}$ for (a^i, a^j) (since $(a^i, a^i) = (a^j, a^j) = 1$), we get

$$(r^k, r^k) - (r^{k+1}, r^{k+1}) = \frac{1}{D^{ijpq}}\left[R^{ijpq} - \frac{S^{ijpq}}{D^{ijpq}}\right]$$

where

$$\begin{aligned}
D^{ijpq} = {} & 1 + \cos^2\theta_{ij}\cos^2\theta_{pq} + \cos^2\theta_{ip}\cos^2\theta_{jq} + \cos^2\theta_{iq}\cos^2\theta_{jp} \\
& - \cos^2\theta_{ij} - \cos^2\theta_{ip} - \cos^2\theta_{iq} \\
& - \cos^2\theta_{jp} - \cos^2\theta_{jq} - \cos^2\theta_{pq} \\
& + 2\cos\theta_{jp}[\cos\theta_{jq}\cos\theta_{pq} + \cos\theta_{ij}\cos\theta_{ip}] \\
& + 2\cos\theta_{iq}[\cos\theta_{ij}\cos\theta_{jq} + \cos\theta_{ip}\cos\theta_{pq}] \\
& - 2\cos\theta_{ij}\cos\theta_{pq}[\cos\theta_{iq}\cos\theta_{jp} + \cos\theta_{ip}\cos\theta_{jp}] \\
& - 2\cos\theta_{ip}\cos\theta_{iq}\cos\theta_{jp}\cos\theta_{jq}
\end{aligned}$$

which is the same as (A1),

$$R^{ijpq} = \mathbf{0}(r_i^k + r_j^k + r_p{}^k + r_q{}^k)$$

and

$$S^{ijpq} = \mathbf{0}(r_i^k + r_j^k + r_p{}^k + r_q{}^k)^2.$$

To select a single step optimal method (i.e. one which would give a minimum value for (r^{k+1}, r^{k+1})), i, j, p and q are chosen such that $(1/D^{ijpq})[R^{ijpq} - (S^{ijpq}/D^{ijpq})]$ is a maximum. S^{ijpq} will approach zero at a faster rate than R^{ijpq} as $r^k \to \emptyset$. ($\emptyset$ is the zero vector). In fact since $(1/D^{ijpq})$ is a constant for the choice of i, j, p and q it dominates the product $(1/D^{ijpq})[R^{ijpq} - (S^{ijpq}/D^{ijpq})]$ as $r^k \to \emptyset$. Therefore a quasi-optimal selection for the 4-dimensional x-projection method is to select quadruples of rows of A in decending order of the values of $(1/D^{ijpq})$ (i, j, p, $q = 1,2,\ldots,n$). All $(1/D^{ijpq})$ ($i, j, p, q = 1, 2, \ldots, n$) are computed initially and a stationary algorithm is obtained by selecting *a priori* quadruples of rows of A based on the values of $(1/D^{ijpq})$ such that each row is used at least once and at most two rows are used twice in each cycle. However, calculating all $(1/D^{ijpq})$ values *a priori* results in $\mathbf{0}\binom{n}{4}n$ operations which is too costly. The most coplanar quadruples of rows of A results in the largest $(1/D^{ijpq})$ value.

Therefore another quasi-optimal selection is to select quadruples of A which are most coplanar. Specifically, the quasi-optimal selection process that was used in all of the test problems and comparisons and which appears in the program listing is described as follows:

First, the minimum angle is determined among the pairs of rows of A. From among the remaining pairs of rows we determine which pair makes the best coplanar fit. This process is repeated again with the unused rows of A until all rows are used. Calculating the angles *a priori* results in $\mathbf{0}\ 2n^3$ operations which is considerably more efficient than calculating all of the $(1/D^{ijpq})$ values.

B. *Acceleration using geometric series*

Probably the most significant characteristic of the x-projection class of methods is that the ratio of the change vectors of the approximate solution vector as $r^k \to \emptyset$ converges to a constant [4]. That is, at some cycle, m, the solution can be written

$$x = x^m + \Delta x^{m+1} + \Delta x^{m+2} + \ldots, \qquad (\Delta x^{i+1} = x^{i+1} - x^i)$$

where $(\Delta x^{i+1}/\Delta x^i) = \rho$ for all $i > m$, $(\rho < 1)$ and m, $m+1$, $m+2, \ldots$ represents successive cycles of an x-projection method. Then the solution can be calculated directly:

$$x = x^m + \frac{\Delta x^{m+1}}{1-\rho}.$$

This result will not be proven. For some linear systems the iterative process converges before the mth cycle has been reached (i.e. before the value of ρ converges). Thus the geometric acceleration may not always be able to be applied. The computational cost to determine if geometric acceleratijon can be applied is $\mathbf{0}\ n$ while the computational cost of each cycle is $\mathbf{0}\ n^2$. It is best not to check after each cycle to see if geometric acceleration can be performed since the change vectors are so small. In practice, we have found that checking anwhere from every 10 to every 150 cycles works quite well. Thus the additional cost required to determine if geometric acceleration can be applied is negligible.

4. TEST PROBLEMS AND COMPARISONS

In this section, four programs are compared. Each program is identified in the comparison tables by the following notation:

(1) 3-X is the 3-dimensional x-projection method with quasi-optimal hyperplane selection as described in [5].

(2) 3-XA is the 3-dimensional x-projection method with quasi-optimal hyperplane selection and if possible geometric acceleration as described in [5].

(3) 4-X is the 4-dimensional x-projection method with quasi-optimal hyperplane selection as described in Sections 2 and 3.

(4) 4-XA is the 4-dimensional x-projection method with quasi-optimal hyperplane selection and if possible, geometric acceleration as described in Sections 2 and 3.

Each method was programmed in FORTRAN and executed from a library using a Honeywell Sigma 6. All calculations were performed in double precision and each test case used an initial starting vector of zero. Any I/0 that was performed was not included in the CPU time.

A. *Random coefficient matrix*

In test case 1 we tested the methods in the following manner. For a given dimension n, we generated a random matrix (a matrix consisting of random numbers between 0 and 1). We assumed a solution vector with components all 1's and generated an appropriate righthand side. The resulting systems were solved and the CPU times and row vector groupings (BEST and WORST) were recorded. This procedure was repeated for each method for the same matrix. BEST indicates the quasi-optimal hyperplane selection was used to select the best groupings of the row vectors of A. For comparison, WORST indicates the quasi-optimal selection was used in reverse to select the worst groupings of the row vectors of A. This shows the importance of

Table 1. Results for test case 1 (random matrices)

Dimension	Methods							
	3-X (BEST) TIME	3-XA (BEST) TIME	4-X (BEST) TIME	4-XA (BEST) TIME	AVE $\frac{1}{D}$ USED	4-X (WORST) TIME	4-XA (WORST) TIME	AVE $\frac{1}{D}$ USED
8	9.89	1.73	3.55	1.13	374	29.28	.94	153
12	3.03	2.71	7.88	2.68	345	32.39	5.87	51
16	29.50	13.96	10.45	4.90	193	42.83	21.75	77
20	17.52	13.19	22.63	13.65	57	53.58	30.83	25
30	155.81	97.69	208.82	81.26	59	437.77	313.08	21
40	290.62	198.02	274.76	241.78	43	404.91	296.19	19
50	236.44	237.28	351.29	308.04	35	757.48	758.90	15

hyperplane selections. Results are displayed in Table 1. The interative process terminated when every corresponding component of two successive approximate solution vectors was within 0.000005. AVE $(1/D)$ USED indicates the average value of $(1/D^{ijpq})$ for the selected quadruples. This is indicated in both the BEST and WORST cases for comparison. TIME indicates the CPU time required for converges in seconds.

B. *Ill-conditioned matrices*

A classical example of ill-conditioned matrices is the set of *Hilbert matrices* H_n of order n with elements $H_n(i, j) = 1/(i+j-1)$[1]. In test case 2 we compare methods 3-X, 3-XA, 4-X and 4-XA for various orders of Hilbert coefficient matrices. In each case, a solution vector with components all 1's was assumed and the appropriate righthand side was generated. Table 2 gives the results of test case 2 where CPU times in seconds for various dimensions of Hilbert matrices are given. In addition, the maximum absolute difference of any component of the final approximate solution vector from 1 is given. This is used as a measure of accuracy rather than the inner produce of the final residual vector. In each case, the best quasi-optimal hyperplane selection is to group the rows consecutively (i.e. (1,2,3,4), (5,6,7,8), ...). However, in most cases this proved unsatisfactory. This grouping produces extremely high $(1/D)$ values which is beyond the precision capabilities of the computer (using 64 bit double precision arithmetic). Instead a mixed grouping of rows was used. For example, for $n = 40$ the rows were grouped (1,11,21,31), (2,12,22,32), ..., (10,20,30,40). The same pattern of grouping was used for other values of n. The nature of Hilbert matrices is such that any grouping of rows produces large $(1/D)$ values. The mixed grouping produces extremely large $(1/D)$ values but within the precision limitations of the computers. The consecutive grouping of rows was used for $n = 12$ and $n = 16$ and the mixed groupings in all of the other cases. Worst case quasi-optimal hyperplanee selections are not included; as one might suspect in each case the rate of convergence was extremely slow. In all cases, a tolerance of 0.000005 was used to terminate the iterative process.

Table 2. Results for test case 2 (Hilbert matrices)

Dimension	Methods							
	3-X		3-XA		4-X		4-XA	
	Max Dif	Time	Max Dif	Time	Max Dif	Time	Max Dif	Time
8	.0078	.78	.0078	.38	.0014	1.45	.0014	.27
12	.0255	1.98	.0256	.50	.0036	1.67	.0037	.45
16	.0311	51.93	.0691	5.06	.0053	1.96	.0055	.65
20	.0107	19.49	.0100	2.57	.0147	2.02	.0144	1.37
30	.0147	39.57	.0162	6.87	.0085	43.91	.0100	5.16
40	.0361	183.26	.0223	43.84	.0044	56.03	.0050	13.28
50	.0187	170.30	.0093	56.03	.0066	67.46	.0071	28.18

5. SUMMARY AND CONCLUSIONS

One characteristic of all x-projection methods is that each equation of (1.1) is scaled according to the Euclidean norm so that $(a^i, a^i) = 1$, $1 \le i \le n$. As a result it is possible to introduce rounding errors. Pizer ([2], p. 165) in his discussion of the one dimensional case suggests we scale each equation according to the Euclidean norm to save time in the iteration. This savings in time is one of the contributing factors for the x-projection methods being competitive. The 1-dimensional x-projection method is depicted by

$$x^{k+1} = x^k + r_i^k a^i$$

where a^i has already been scaled by $\sqrt{(a^i, a^i)}$. Without scaling the equation becomes:

$$x^{k+1} = x^k + r_i^k \frac{a^i}{(a^i, a^i)}.$$

The only difference is the square root. This is true for higher dimensions as well. Scaling according to the Euclidean norm greatly simplifies the mathematical notation especially equations (2.9) and (A1). Also since $(a^i, a^i) = (a^i, a^i) = 1$ and $\cos \theta_{ij} = (a^i, a^j)$ the notation in Section 3 is much simpler and the conclusions are more easily obtained. In addition, the proof of the theorem which shows that the 2-dimensional x-projection method is based on repeated applications of the 1-dimensional x-projectijon method and thus always convergent, assumes each equation of (1.1) has been scaled according to the Euclidean norm ([4], pp. 63–64). The proof of Theorem 2.1 is based on this. All of these advantages of course are irrelevant if the scaling perturbs the problem. Table 2 gives the maximum difference of any component of the final approximate solution vector from the true solution of one for Hilbert matrices of various dimensions. The maximum difference values are within acceptable limits and do not increase with the dimension of the matrix. In fact the maximum difference values seem to be independent of the dimension of the matrix. Thus rounding errors due to scaling according to the Euclidean norm are not a problem.

We feel the test cases provide fairly clear-cut conclusions. The 4-dimensional x-projection method with hyperplane selection and geometric acceleration, 4-XA, proved to be superior to the best known x-projection method to date, 3-XA. In test case 1 4-XA, on an overall basis, performed slightly better than 3-XA (less CPU time). In some instances, however, the 3-XA method was computationally superior to the 4-XA method. In these cases it is possible the grouping of the hyperplanes in triples rather than quadruples yielded an overall superior (more coplanar) collection of hyperplanes. This would yield convergence at a faster rate. This is the exception, however, rather than the rule. Another possibility is that while the 4-X method is superior to the 3-X method, the 3-X method may be generating a sequence of Δx^i vectors more conducive to geometric acceleration (see $n = 40$ where 4-X is superior to 3-X, but 3-XA is superior to 4-XA). There is certainly room for future research in this area. For example, is it possible *a priori* to determine which x-projection method is most likely to prove superior for the given linear system? Perhaps one method should be used up to a certain point and then a switch made to another. Still another possibility is to perform the methods in cyclic order alternating every 10 cycles or so. In test case 2, however, 4-XA provied to be significantly superior to 3-XA (less CPU time for convergence) and in every case. In both test cases the accuracy of the solution using 3-XA and 4-XA was the same. Wainwright [4, 5] has previously shown the 3-XA method to be superior in CPU time and accuracy over the 2-XA method. In addition he has shown the 2-XA method to be superior in CPU time and accuracy over a wide range of test cases to Gauss–Seidel, the 1-dimensional x-projection method with acceleration, 1-XA, the best known r-projection method, 2-R as well as Gaussian elimination (in the case of ill-conditioned matrices).

The quasi-optimal hyperplane selection criteria presented in Section 3A proved in practice to be extremely successful for increasing the rate of convergence (see Table 1, 4-X BEST vs 4-X WORST comparing CPU TIME and AVE (1/D) USED). The geometric acceleration technique presented in section 3B also proved extremely successful in practice. For the cases where it could be applied, the CPU time was reduced significantly and in the Random matrices

accuracy improved by orders of magnitude (see 4-X BEST vs 4-XA BEST in Table 1, 4-X WORST vs 4-XA WORST in Table 1 and 4-X vs 4-XA in Table 2.) For the single example where it could not be applied (Table 1 $n = 50$, the added cost in CPU time) from 757.48 to 758.90 to test for the geometric acceleration condition was insignificant. In addition, Table 2 clearly shows one of the most important characteristics of the class of x-projection methods: the ability to solve ill-conditioned systems! The nature of most ill-conditioned matrices is that they produce very large $(1/D)$ values thus making the hyperplane selection extremely effective in increasing the rate of convergence (see 3-X and 4-X in Table 2). For example, for 4-X for $n = 12$ the AVE $(1/D)$ USED is 3.86×10^{10}; for $n = 20$ the AVE $(1/D)$ USED $= 2.95 \times 10^{10}$; for $n = 40$ the AVE $(1/D)$ USED $= 2.62 \times 10^{10}$ and for $n = 50$ the AVE $(1/D)$ USED $= 2.03 \times 10^{10}$. Compare this to the AVE $(1/D)$ USED values in Table 1. Furthermore, in addition to large $(1/D)$ values we were able in each instance to apply geometric acceleration (see 4-XA in Table 2).

In conclusion, 4-XA like all projection methods is guaranteed to converge for arbitrary starting vector and nonsingular coefficient matrix. Also, 4-XA has been shown to be a superior method over the best projection method known to date, 3-XA. 3-XA has previously been shown to be competitive and in most cases superior to other well known iterative algorithms both in rate of convergence and accuracy [5]. Furthermore, with ill-conditioned matrices 4-XA proved very effective and was shown to be a superior to the 3-XA method. A program listing of method 4-XA in the form of a FORTRAN subroutine can be obtained from the author if desired.

REFERENCES

1. R. T. Gregory and D. L. Karney, *A Collection of Matrices for Testing Computational Algorithms.* Wiley, New York (1969).
2. S. M. Pizer, *Numerical Computing and Mathematical Analysis.* Science Research Associates, Inc., Chicago (1975).
3. R. L. Wainwright, and R. F. Keller, Algorithms for projection methods for solving linear systems of equations. *Comput. Math. Applics* **3**, 235–245 (1977).
4. R. L. Wainwright, Acceleration techniques for a class of X-projection methods for solving systems of linear equations. *Comput Math. Applics* **5**, 59–73 (1979).
5. R. L. Wainwright, Three dimensional X-projection methods (with acceleration techniques) for solving systems of linear equations. *Comput. Math. Applics* **7**, 211–223 (1981).

APPENDIX A

$$W_1 = 1 + 2(a^{i+5}, a^{i+6})(a^{i+5}, a^{i+7})(a^{i+6}, a^{i+7}) - (a^{i+6}, a^{i+7})^2 - (a^{i+5}, a^{i+6})^2 - (a^{i+5}, a^{i+7})^2$$

$$W_2 = 1 + 2(a^{i+4}, a^{i+6})(a^{i+4}, a^{i+7})(a^{i+6}, a^{i+7}) - (a^{i+6}, a^{i+7})^2 - (a^{i+4}, a^{i+6})^2 - (a^{i+4}, a^{i+7})^2$$

$$W_3 = 1 + 2(a^{i+4}, a^{i+5})(a^{i+4}, a^{i+7})(a^{i+5}, a^{i+7}) - (a^{i+5}, a^{i+7})^2 - (a^{i+4}, a^{i+5})^2 - (a^{i+4}, a^{i+7})^2$$

$$W_4 = 1 + 2(a^{i+4}, a^{i+5})(a^{i+4}, a^{i+6})(a^{i+5}, a^{i+6}) - (a^{i+5}, a^{i+6})^2 - (a^{i+4}, a^{i+5})^2 - (a^{i+4}, a^{i+6})^2$$

$$U_1 = (a^{i+6}, a^{i+7})[(a^{i+4}, a^{i+6})(a^{i+5}, a^{i+7}) + (a^{i+4}, a^{i+7})(a^{i+5}, a^{i+6})] + (a^{i+4}, a^{i+5})[1 - (a^{i+6}, a^{i+7})^2] - (a^{i+4}, a^{i+6})(a^{i+5}, a^{i+6}) - (a^{i+4}, a^{i+7})(a^{i+5}, a^{i+7})$$

$$U_2 = (a^{i+5}, a^{i+7})[(a^{i+4}, a^{i+5})(a^{i+6}, a^{i+7}) + (a^{i+4}, a^{i+7})(a^{i+5}, a^{i+6})] + (a^{i+4}, a^{i+6})[1 - (a^{i+5}, a^{i+7})^2] - (a^{i+4}, a^{i+7})(a^{i+6}, a^{i+7}) - (a^{i+4}, a^{i+5})(a^{i+5}, a^{i+6})$$

$$U_3 = (a^{i+5}, a^{i+6})[(a^{i+4}, a^{i+6})(a^{i+5}, a^{i+7}) + (a^{i+4}, a^{i+5})(a^{i+6}, a^{i+7})] + (a^{i+4}, a^{i+7})[1 - (a^{i+5}, a^{i+6})^2] - (a^{i+4}, a^{i+5})(a^{i+5}, a^{i+7}) - (a^{i+4}, a^{i+6})(a^{i+6}, a^{i+7})$$

$$V_1 = (a^{i+4}, a^{i+7})[(a^{i+4}, a^{i+6})(a^{i+5}, a^{i+7}) + (a^{i+4}, a^{i+5})(a^{i+6}, a^{i+7})] + (a^{i+5}, a^{i+6})[1 - (a^{i+4}, a^{i+7})^2] - (a^{i+5}, a^{i+7})(a^{i+6}, a^{i+7}) - (a^{i+4}, a^{i+5})(a^{i+4}, a^{i+6})$$

$$V_2 = (a^{i+4}, a^{i+6})[(a^{i+4}, a^{i+5})(a^{i+6}, a^{i+7}) + (a^{i+4}, a^{i+7})(a^{i+5}, a^{i+6})] + (a^{i+5}, a^{i+7})[1 - (a^{i+4}, a^{i+6})^2] - (a^{i+4}, a^{i+5})(a^{i+4}, a^{i+7}) - (a^{i+6}, a^{i+7})(a^{i+5}, a^{i+7})$$

$$V_3 = (a^{i+4}, a^{i+5})[(a^{i+4}, a^{i+6})(a^{i+5}, a^{i+7}) + (a^{i+4}, a^{i+7})(a^{i+5}, a^{i+6})]$$
$$+ (a^{i+6}, a^{i+7})[1 - (a^{i+4}, a^{i+5})^2]$$
$$- (a^{i+5}, a^{i+6})(a^{i+5}, a^{i+7}) - (a^{i+4}, a^{i+6})(a^{i+4}, a^{i+7}).$$

D is the determinant of the coefficient matrix in (2.8):

$$\begin{aligned} D = {} & 1 + (a^{i+4}, a^{i+5})^2(a^{i+6}, a^{i+7})^2 + (a^{i+4}, a^{i+6})^2(a^{i+5}, a^{i+7})^2 \\ & + (a^{i+4}, a^{i+7})^2(a^{i+5}, a^{i+6})^2 \\ & - (a^{i+4}, a^{i+5})^2 - (a^{i+4}, a^{i+6})^2 - (a^{i+4}, a^{i+7})^2 \\ & - (a^{i+5}, a^{i+6})^2 - (a^{i+5}, a^{i+7})^2 - (a^{i+6}, a^{i+7})^2 \\ & + 2(a^{i+5}, a^{i+6})[(a^{i+5}, a^{i+7})(a^{i+6}, a^{i+7}) + (a^{i+4}, a^{i+5})(a^{i+4}, a^{i+6})] \\ & + 2(a^{i+4}, a^{i+7})[(a^{i+4}, a^{i+5})(a^{i+5}, a^{i+7}) + (a^{i+4}, a^{i+6})(a^{i+6}, a^{i+7})] \\ & - 2(a^{i+4}, a^{i+5})(a^{i+6}, a^{i+7}) \\ & [(a^{i+4}, a^{i+7})(a^{i+5}, a^{i+6}) + (a^{i+4}, a^{i+6})(a^{i+5}, a^{i+7})] \\ & - 2(a^{i+4}, a^{i+6})(a^{i+4}, a^{i+7})(a^{i+5}, a^{i+6})(a^{i+5}, a^{i+7}). \end{aligned} \tag{A1}$$