NASA CR
147823

(NASA-CR-147823) THE USE OF THE WINOGRAD MATRIX MULTIPLICATION ALGORITHM IN DIGITAL MULTISPECTRAL PROCESSING (Rice Univ.) 18 p HC $3.50 CSCL 09B

N76-27903

Unclas

G3/61 44565

ICSA

INSTITUTE FOR COMPUTER SERVICES AND APPLICATIONS

RICE UNIVERSITY

275-025-027

The Use of the Winograd Matrix Multiplication Algorithm in Digital Multispectral Processing

by

D.L. Van Rooy, R.A. Schafer, and
M.S. Lynn
I C S A
Rice University

ABSTRACT:

The Winograd procedure for matrix multiplication [S. Winograd, Comm. on Pure and Applied Math., 23, 1970] provides a method whereby general matrix products may be computed more efficiently than the normal method. In this report, we describe the algorithm and the time savings that can be effected. A FORTRAN program is provided which performs a general matrix muitiply according to this algorithm.

Additionally, we describe a variation of this procedure that may be used to calculate Gaussian probability density functions. It is shown how a time savings can be effected in this calculation. The extension of this method to other similar calculations should yield similar savings.

Institute for Computer Services and Applications
Rice University
Houston, TX 77001
May, 1976

This work was supported under NASA contract NAS 9-12776

I. Introduction:

In this paper, we show the Winograd[(1)] procedure for computing matrix products can be applied to various calculations used in digital processing of remotely sensed data. The basic procedure is described, a FORTRAN program for general matrix multiplication is provided, and an example (computation of Gaussian probability density functions) is worked out showing the regions where it is computationally faster and the amount of time savings involved. Of course, there are many other calculations where this procedure can be effectively utilized.

Essentially, the Winograd procedure effects a time savings in computing matrix products by trading off some of the multiplications involved for additions (multiplies are usually slower operations on a computer than adds). A relatively small amount of additional storage is required, but a significant decrease in the mumber of multiplies (up to a factor of 2 for the case of multiplying two $n \times n$ matrices, if n is sufficiently large) can be gained. However, one must sacrifice some numerical stability in employing this procedure[(2)].

In the next section, we describe the general matrix multiplication procedure developed by Winograd and compare it with the standard method. In Section III, we describe a variation of this procedure for the computation of Gaussian probability density functions. An appendix containing a FORTRAN program for general matrix multiplication is provided.

II. The Winograd Algorithm:

Let $A=(a_{ij})$ be a $p \times q$ matrix, $B=(b_{ij})$, a $q \times s$ matrix, $C=(c_{ij})$, a $p \times s$ matrix, $x=(x_i)$, a q-vector, and $y=(y_i)$, a p-vector. The standard method for computing $y=Ax$ is

$$y_i = \sum_{j=1}^{q} a_{ij} x_j \qquad i=1,2, \ldots, p \qquad (1)$$

and for computing $C=AB$ is

$$c_{ij} = \sum_{\ell=1}^{q} a_{iq} b_{qj} \qquad \begin{matrix} i=1,2,\ldots,p \\ j=1,2,\ldots,s \end{matrix} \qquad (2)$$

Thus to compute y in this manner, pq multiplies and $p(q-1)$ adds are necessary; and to compute C, pqs multiplies and $ps(q-1)$ adds.

The Winograd procedure consists of rewriting eqs. (1) and (2) so that some quantities are precomputed. This procedure is based on the identity

$$a_{ik} b_k + a_{i,k+1} b_{k+1}$$

$$= (a_{ik} + b_{k+1})(a_{i,k+1} + b_k) - a_{ik} a_{i,k+1} - b_k b_{k+1}$$

Similar identities of a higher order may be used to construct other algorithms, but, for our purposes here, these are not of much interest. Following Winograd's notation, we let $\lfloor d \rfloor$ = largest integer $\le d$ and $\lceil d \rceil$ = smallest integer $\ge d$, with $\ell = \lfloor \frac{1}{2} q \rfloor$, the Winograd procedure, then, for computing y

$$\xi_i = \sum_{u=1}^{\ell} a_{i,2u-1} \cdot a_{i,2u} \qquad i=1,2,\ldots,p \qquad (3a)$$

$$\eta = \sum_{u=1}^{\ell} x_{2u-1} \cdot x_{2u} \qquad (3b)$$

and

$$y_i = \begin{cases} \sum_{u=1}^{\ell} (a_{i,2u-1} + x_{2u})(a_{i,2u} + x_{2u-1}) - \\ \xi_i - \eta \quad \text{if } q = 2\ell \\ \sum_{u=1}^{\ell} (a_{i,2u-1} + x_{2u})(a_{i,2u} + x_{2u-1}) - \\ \xi_i - \eta - a_{iq}\, x_q \quad \text{if } q = 2\ell + 1 \end{cases} \qquad (3c)$$

This algorithm requires $p \lceil \frac{1}{2} q \rceil + \ell(p+1)$ multiplications and $p(\lceil \frac{1}{2} q \rceil + \ell - 1$ adds. If we have t y's to compute using the same A and t x's, the operation counts become $\ell p + t \cdot (p \lceil \frac{1}{2} q \rceil + \ell)$ multiplies and $p(\ell - 1) + t \cdot \left(\ell - 1 + p(2\ell + 1 + \lceil \frac{1}{2} q \rceil)\right)$ adds, since the ξ_i's need not be recomputed. Table 1 shows the approximate (ignoring indexing, etc.) time savings to be expected for computing 100 y's for various values of p and q for a ratio of the machine multiply time T_m to the add time T_a of 2.7 (the approximate value for an IBM 370/155). We see from this table that for low values

p \ q	2	4	6	8	10	12	14	16
2	1.26	1.14	1.1	1.08	1.07	1.06	1.06	1.05
4	1.16	1.02	0.978	0.958	0.946	0.938	0.933	0.928
6	1.12	0.98	0.938	0.917	0.905	0.897	0.892	0.888
8	1.1	0.961	0.918	0.897	0.885	0.877	0.871	0.867
10	1.09	0.95	0.906	0.885	0.873	0.865	0.859	0.855
12	1.09	0.942	0.898	0.877	0.865	0.857	0.851	0.847
14	1.08	0.936	0.893	0.871	0.859	0.851	0.845	0.841
16	1.08	0.932	0.888	0.867	0.855	0.846	0.841	0.836

Table 1

Ratio of the time to compute 100 y's of eq. 1 by the Winograd procedure vs the standard method, where T_m/T_a is assumed to be 2.7.

(Note: Net savings always occur if p, q ≥ 4) of p and q, there is a net loss ...e for large values, up to a 16% savings results.

The extension of this concept to full matrix multiplication is (for $C = AB$)

$$\xi_i = \sum_{u=1}^{\ell} a_{i,2u-1}\, a_{i,2u} \tag{4a}$$

$$\eta_j = \sum_{u=1}^{\ell} b_{2u-1,j}\, b_{2u,j} \tag{4b}$$

$$c_{ij} = \begin{cases} \sum_{u=1}^{\ell} (a_{i,2u-1} + b_{2u,j})(a_{i,2u} + b_{2u-1,j}) - \\ \xi_i - \eta_j \quad \text{if } q = 2\ell \\ \sum_{u=1}^{\ell} (a_{i,2u-1} + b_{2u,j})(a_{i,2u} + b_{2u-1,j}) - \\ \xi_i - \eta_j + a_{iq}\, b_{qj} \quad \text{if } q = 2\ell + 1 \\ \text{for } i = 1, 2, \ldots, p \text{ and } j = 1, 2, \ldots, s \end{cases} \tag{4c}$$

This requires $ps[\frac{1}{2}q] + \ell(p+s)$ multiplies and $(\ell-1)(p+s) + ps([\frac{1}{2}q] + 2 + 1)$ adds. For $p = q = s = n$ and n large, this reduces to $\sim \frac{1}{2}n^3 + n^2$ multiplies and $\frac{3}{2}n^3 + n^2$ adds, which can effect a savings in computation time over the standard procedure which requires n^3 multiplies and $n^3 - n^2$ adds; using the ratio of $2.7:1$ for multiplies to adds, in fact a savings always occurs for $n > 5$.

A FORTRAN subroutine WNØMUL which performs matrix multiplication according to this algorithm is listed in the appendix.

III. Calculation of Gaussian Probability Density Functions:

The n-dimensional probability density for a normal population with mean u and covariance K is given by

$$f(x) = (2\pi)^{-\frac{n}{2}} \; |K|^{-\frac{1}{2}} \exp\left[(x-u)^T \; K^{-1}(x-u)\right]$$

In this section, we shall concern ourselves only with the calculation of the quadratic form (the argument of exp), given u and L and D where L and D are the modified Cholesky decomposition[3] of K, i.e. $K = LDL^T$, with L being unit lower triangular and D, diagonal with positive diagonal entries. Then we can write

$$\begin{aligned}(x-u)^T \; K^{-1}(x-u) &= (x-u)^T \; L^{-1^T} \; D^{-1} \; L^{-1}(x-u) \\ &= y^T \; D^{-1} \; y \\ &= \sum_{i=1}^{n} y_i^2 \,/\, d_i \qquad (5)\end{aligned}$$

where

$$y = L^{-1}(x-u) \qquad (6)$$

Eq. (6) can be solved by forward substitution, i.e.

$$y_i = (x_i - u_i) - \sum_{j=1}^{i-1} \ell_{ij} \, y_j \qquad (7)$$

where $L = (\ell_{ij})$. We then can use the Winograd procedure on the summation term in eq. (7). (The standard method requires $\frac{n^2 - n}{2}$ multiplies and $\frac{(n-2)(n-1)}{2}$ adds to evaluate this term). We note that for evaluating this expression for more than one x, we can use a variation of the algorithm for full matrix products.

Taking special note of the structure of L, we then use

$$\left.\begin{aligned} \xi_i &= \sum_{s=1}^{r_i} \ell_{i,2s-1}\, \ell_{i,2s} \\ \eta_i &= \sum_{s=1}^{r_i} y_{2s-1}\, y_{2s} \end{aligned}\right\} \quad i=3,4,\ldots,n \qquad (8)$$

with $r_i = \left\lfloor \frac{i-1}{2} \right\rfloor$. This is equivalent to

$$\eta_3 = y_1\, y_2 \qquad j=2,3,\ldots,\left\lfloor \frac{n}{2} \right\rfloor \qquad (9)$$

$$\eta_{2j} = \eta_{2j-1}$$

$$\eta_{2j+1} = \eta_{2j-1} + y_{2j-1}\, y_{2j} \quad j=2,3,\ldots,\left\lfloor \frac{n-1}{2} \right\rfloor$$

We then have, with

$$a_i = \sum_{j=1}^{i-1} \ell_{ij}\, y_j$$

$$a_1 = 0$$

$$a_2 = \ell_{21}\, y_1$$

$$a_i = \begin{cases} \sum_{j=1}^{r_i} (\ell_{i,2j-1} + y_{2j})(\ell_{i,2j} + y_{2j-1}) - \\ \xi_i - \eta_i & \text{if } i \text{ odd} \\ \sum_{j=1}^{r_i} (\ell_{i,2j-1} + y_{2j})(\ell_{i,2j} + y_{2j-1}) \\ \xi_i - \eta_i + \ell_{i,i-1} y_{i-1} & \text{if } i \text{ even} \\ \qquad i=3,4,\ldots,n \end{cases} \quad (10)$$

Since ξ depends only on L, it can be used in evaluating eq. (7) for various x. Table 2 shows the number of operations necessary for computing each of the various parts of the quadratic form. Note that the Winograd algorithm is actually slower when evaluating this expression for just one x, but for more than one x, the precomputed values of ξ may be used. Table 3 shows the ratio of times for the two methods for the case of $T_m/T_a = 2.7$ for computing m of the quadratic terms for various values of n. Note that a net savings occurs only in the lower right region. Also shown in Table 4 is the minimum ratio of T_m/T_a for the Winograd precedure to be faster for computing m quadratic terms.

Often in remote sensing calculations (e.g. maximum likelihood classification), many probability density functions must be evaluated over the same set of data vectors. In this case, one may precompute both the ξ's and the η's. Table 5 shows the asymptotic (ξ precomputed and η computed only for the first class) time savings to be expected for $T_m/T_a = 2.7$ for k classes for various values of n. Note that a net savings results for $n > 4$.

Term	Adds / Subtracts	For Large n*	Multiplies / Divides	For Large n*
ξ	$\frac{p(p-1)}{2} + \frac{(q-2)(q-1)}{2}$	$\frac{n^2}{4} - 2n$	$\frac{p(p+1)}{2} + \frac{q(q-1)}{2}$	$\frac{n^2}{4}$
η	$\max(p-1,0)$	$\frac{n}{2}$	p	$\frac{n}{2}$
a	$\frac{3}{2}\left(p(p+1) + q(q-1)\right) + 2p + 3q - n - 1$	$\frac{3n^2}{4} + \frac{3n}{2}$	$\frac{p(p+1) + q(q-1)}{2} + q$	$\frac{n^2}{4} + \frac{n}{2}$
$\lambda - \mu$	n	n	0	0
y	$n - 1$	n	0	0
$\sum y_i^2/d_i$	$n - 1$	n	$2n$	$2n$

Table 2

Number of operations necessary for computing each of the various terms in eqs. 5-10

where $p = \left\lfloor \frac{n-1}{2} \right\rfloor$ and $q = \left\lfloor \frac{n}{2} \right\rfloor$ for $n \geq 2$

*Asymptotic value for large n

m \ n	2	4	6	8	10	12	14	16	18	20	22	24
2	1	1.12	1.13	1.13	1.12	1.11	1.1	[illegible]	1.09	1.09	1.08	1.08
10	1	1.08	1.06	1.03	1.01	.99	.973	.959	.947	.937	.929	.921
50	1	1.07	1.05	1.02	.988	.965	.947	.932	.919	.908	.898	.89
100	1	1.07	1.04	1.01	.985	.962	.944	.928	.915	.904	.895	.886
1000	1	1.07	1.04	1.01	.983	.96	.941	.925	.912	.901	.891	.883

Table 3

Ratio of the time to compute m quadratic terms of dimension n of eq. 5 by the Winograd procedure vs. the standard procedure for $T_m / T_a = 2.7$

m \ n	2	4	6	8	10	12	14	16	18	20	22	24
5	-	10	5.5	4.12	3.45	3.05	2.79	2.6	2.46	2.36	2.27	2.2
10	-	7.5	4.47	3.41	2.87	2.55	2.33	2.17	2.05	1.96	1.89	1.83
25	-	6.52	4.01	3.08	2.6	2.3	2.1	1.96	1.85	1.77	1.7	1.65
100	-	6.12	3.81	2.93	2.48	2.19	2	1.87	1.76	1.68	1.62	1.56
1000	-	6.01	3.76	2.89	2.44	2.16	1.98	1.84	1.74	1.66	1.59	1.54

Table 4

Minimum value of T_m/T_a for Winograd procedure to be faster in calculating m quadratic terms of dimension n

k \ n	2	4	6	8	10	12	14	16	18	20	22	24
5	1	1.02	.991	.961	.936	.917	.902	.889	.879	.87	.862	.856
10	1	1.02	.984	.954	.931	.912	.897	.885	.875	.866	.859	.853
20	1	1.01	.981	.951	.928	.909	.895	.883	.873	.864	.857	.851
50	1	1.01	.979	.949	.926	.908	.893	.881	.871	.863	.856	.85

Table 5

Asymptotic (ξ precomputed and η computed only for the first class) ratio of times for the standard vs. Winograd method to compute k quadratic terms for each of a large set of data vectors of dimension n for $T_m / T_a = 2.7$

IV. Conclusions:

The Winograd matrix multiplication scheme can produce a time savings in various computations using large order matrices in the digital processing of remotely sensed data. We have shown how a modification of this procedure may be applied to the calculation of Gaussian probability density functions, and indicated how this may be extended to other computations. For large dimensions, or a large number of points, there can be some time savings, but the user should determine the expected time difference using the value of T_m/T_a of the computer to be used. A further study should be undertaken to investigate the effects of decreased numerical stability of this algorithm in various applications.

APPENDIX

```
      SUBROUTINE WNOMUL(A,P,Q,IA,B,S,IB,C,IC,TMP)
C
C     THIS ROUTINE USES THE WINOGRAD PROCEDURE TO MULTIPLY TWO MATRICES
C     I.E. C=A*B WHERE A IS P * Q; B, Q * S; & C, P * S.
C     IA IS THE LENGTH OF THE COLS OF A AS DIMENSIONED IN THE MAIN PROG.
C     IB & IC ARE SIMILAR QUANTITIES FOR B & C
C     C MUST NOT OCCUPY THE SAME STORAGE AS A OR B
C     TMP IS WORKING STORAGE OF LENGTH .GE.S+P
C
      INTEGER P,Q,S,U,U1,U2
      REAL A(IA,Q),B(IB,S),C(IC,S),TMP(1)
      DOUBLE PRECISION SS,S1,S2
      LOGICAL ODD
      IETA=P
      L=Q/2
C
C     COMPUTE THE XI'S
C
      DO 10 I=1,P
         SS=-A(I,1)*A(I,2)
         IF (L.LT.2) GO TO 10
         DO 15 U=2,L
            U1=2*U
            U2=U1-1
   15    SS=SS-A(I,U2)*A(I,U1)
   10 TMP(I)=SS
C
C     COMPUTE THE ETA'S
C
      DO 20 J=1,S
         SS=-B(1,J)*B(2,J)
         IF (L.LT.2) GO TO 20
         DO 25 U=2,L
            U1=2*U
            U2=U1-1
   25    SS=SS-B(U2,J)*B(U1,J)
   20 TMP(IETA+J)=SS
      ODD=.FALSE.
      IF (2*L.NE.Q) ODD=.TRUE.
C
C     COMPUTE THE C(I,J)'S
C
      DO 30 I=1,P
         S1=TMP(I)
         S3=A(I,Q)
         DO 30 J=1,S
            SS=S1+TMP(IETA+J)
            IF (.NOT.ODD) GO TO 37
               SS=SS+S3*B(Q,J)
   37       DO 35 U=1,L
               U1=2*U
               U2=U1-1
   35       SS=SS+(A(I,U2)+B(U1,J))*(A(I,U1)+B(U2,J))
   30 C(I,J)=SS
      RETURN
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

REFERENCES

[1] Shmuel Winograd, "On the Number of Multiplications Necessary to Compute Certain Functions," Comm. on Pure and Applied Math., 23, 1970, pp. 165-179.

[2] W. Miller, "Computational Complexity and Numerical Stability," SIAM J. Comput., 4, June, 1975, pp. 97-107.

[3] D. L. Van Rooy, M. S. Lynn, and C. H. Snyder, "The Use of the Modified Cholesky Decomposition in Divergence and Classification Calculations," Conf. Proc. Machine Processing of Remotely Sensed Data, LARS, Purdue, Oct. 16-18, 1973, pp. 3B-35—3B-47.