Centrum voor Wiskunde en Informatica
Centre for Mathematics and Computer Science

J.J.F.M. Schlichting

NUMVEC FORTRAN Library manual
Chapter: Basic linear algebra
Routine: MATMUL
Chapter: Simultaneaous linear equations
Routines: BIDIAGL and BIDIAGU

Department of Numerical Mathematics Report NM-R9017 September

---

The Centre for Mathematics and Computer Science is a research institute of the Stichting Mathematisch Centrum, which was founded on February 11, 1946, as a nonprofit institution aiming at the promotion of mathematics, computer science, and their applications. It is sponsored by the Dutch Government through the Netherlands Organization for the Advancement of Research (N.W.O.).

NUMVEC
is a library of NUMerical software for VECtor and parallel computers in FORTRAN.
The documentation conforms as much as possible to that of the NAG - library. A Subsection: *11.1 Vectorization information* mentions, a.o., whether the code is standard FORTRAN 77. If not, information is given about special vector-syntax used and about specific machine(s) to which the code is aimed.
The source code described can be obtained by writing to the NUMVEC Library Manager at the CWI.

NUMVEC FORTRAN Library manual

Chapter: Basic Linear Algebra
Routine: MATMUL

Chapter: Simultaneous Linear Equations
Routines: BIDIAGL and BIDIAGU

J.J.F.M. Schlichting
Control Data BV
P.O. Box 11, 2285 VL Rijswijk, The Netherlands

This document describes the following NUMVEC FORTRAN Library routines (developed for the Cyber 205): MATMUL performs matrix-matrix multiplication on full matrices, by means of 64-fold unrolling in the usual vector update method; BIDIAGL and BIDIAGU solve systems of linear equations where the matrix is lower and upper bidiagonal respectively.

In order to obtain maximal performance on the Cyber 205 (120 MFLOPS for MATMUL and 18 MFLOPS for BIDIAGL and BIDIAGU) these routines were written in the assembly language META.

1980 Mathematics Subject Classification (1985 revision): 65V05, 65F05.
1987 CR Categories: G.1.3.

Keywords & Phrases: software, matrix-matrix multiplication, bidiagonal systems of linear equations.

Note: This software has been developed while the author was a visiting scientist at CWI.

Report NM-R9017
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

MATMUL - NUMVEC FORTRAN Library Routine Document

1. Purpose

MATMUL performs matrix multiplication on full matrices.

2. Specification

```
      SUBROUTINE MATMUL(A,B,C,LDA,LDB,LDC,K,L,M)
C     INTEGER LDA,LDB,LDC,L,K,M
C     REAL A(LDA,K),B(LDB,M),C(LDC,M)
```

3. Description

MATMUL performs matrix-matrix multiplication. The algorithm uses 64-fold unrolling in the normal vector update method. The interference of scalar load instructions with the startup of vector instructions is avoided yielding a performance improvement.

4. References

None.

5. Parameters

A - REAL array of DIMENSION (LDA,p), where p $\geqslant$ L
On entry A must contain the L columns of K elements of the first matrix.
Unchanged on exit.

B - REAL array of DIMENSION (LDB,p), where p $\geqslant$ M.
On entry B must contain the M columns of L elements of the second matrix.
Unchanged on exit.

C - REAL array of DIMENSION (LDC,p), where p $\geqslant$ M.
On exit C contains the M columns of K elements of the product matrix.

LDA - INTEGER.
On entry LDA specifies the first dimension of array A as declared in the calling (sub)program LDA $\geqslant$ K.
Unchanged on exit.

LDB - INTEGER.
On entry LDB specifies the first dimension of array B as declared in the calling (sub)program LDB $\geqslant$ L.
Unchanged on exit.

LDC - INTEGER.
On entry LDC specifies the first dimension of array C as declared in the calling (sub)program LDC $\geqslant$ K.
Unchanged on exit.

K - INTEGER.
On input K must contain the number of rows of array A. By definition this is the number of rows of array C.
Unchanged on exit.

L - INTEGER.
On input L must contain the number of columns of array A. By definition this is the number of rows of array B.
Unchanged on exit.

M - INTEGER.
On input M must contain the number of columns of array B. By definition this is the number of columns of array C.
Unchanged on exit.

6. Error indicators and warnings

None.

7. Auxiliary routines

None.

8. Timing

For K = L = M = 300 the execution time is 0.45 seconds corresponding to a megaflop rate of 120. This is a 20 percent improvement over the usual Fortran code.

9. Storage

None.

10. Accuracy

Same as the usual algorithm.

11. Further comments

None.

11.2. Vectorization information

MATMUL is written in META, the assembly language for the Cyber 205.

12. Keywords

Matrix multiplication.
Unrolling.

13. Example

The example program compares the result and performance of the usual algorithm coded in Fortran and the routine MATMUL.

13.1. Program text

```
      PROGRAM TMAT
      PARAMETER (K=301,L=302,M=303)
      PARAMETER (LDA=311,LDB=312,LDC=313)
      REAL A(LDA,L),B(LDB,M),C(LDC,M),D(LDC,M)
      DO 10 J=1,L
      DO 10 I=1,K
      A(I,J) = RANF()+J
   10 CONTINUE
      DO 15 J=1,M
      DO 15 I=1,L
      B(I,J) = RANF()
   15 CONTINUE
      T0 = SECOND()
      DO 30 J=1,M
      DO 23 I=1,K
   23 D(I,J) = 0.0
      DO 30 JJ =1,L
      DO 30 I=1,K
   30 D(I,J) = D(I,J)+A(I,JJ)*B(JJ,J)
      T1 = SECOND()
      CALL MATMUL(A,B,C,LDA,LDB,LDC,K,L,M)
      T2 = SECOND()-T1
      T1 = T1-T0
      DO 50 J=1,M
      DO 50 I=1,K
      IF(C(I,J).NE.D(I,J)) GOTO123
   50 CONTINUE
      NOMF = K*L*M*2/1000000
      PRINT 1,K,L,M,T1,NOMF/T1,T2,NOMF/T2
    1 FORMAT(' DIMENSIONS: ', 3I6,/,
     1' TIME FOR FORTRAN CODE ', F9.3, ' SECONDS MFL = ', F7.2,/,
     2' TIME FOR MATMUL ROUTINE', F9.3, ' SECONDS MFL = ', F7.2)
      GOTO 125
  123 PRINT *,'FOUT ',J,I
  125 CONTINUE
      END
```

13.2. Program data

None.

13.3. Program results

```
THE OUTPUT OF THE PROGRAM IS:

 DIMENSIONS:    301   302   303
 TIME FOR FORTRAN CODE      0.548 SECONDS MFL =  100.34
 TIME FOR MATMUL ROUTINE    0.456 SECONDS MFL =  120.68
```

BIDIAGL and BIDIAGU - NUMVEC FORTRAN Library Routine Document

1. Purpose

BIDIAGL and BIDIAGU solve a bidiagonal system of linear equations. BIDIAGL is used for an upper bidiagonal system and BIDIAGU for a lower bidiagonal system.

2. Specification

```
      SUBROUTINE BIDIAGL(X,A,B,N)
C     INTEGER N
C     REAL X(N),A(N-1),B(N)

      SUBROUTINE BIDIAGU(X,A,B,N)
C     INTEGER N
C     REAL X(N),A(N-1),B(N)
```

3. Description

BIDIAGL and BIDIAGU solve a bidiagonal system of linear equations:

$$MX = B$$

where M is a square matrix ($n \times n$) with unit main diagonal and of which the first lower diagonal for BIDIAGL or the first upper diagonal for BIDIAGU are non-zero, and X and B are vectors (n). The size n of the matrix is supplied through parameter N. The user must supply the non-zero lower or upper diagonal through the parameter A. The elements of M are given by:

```
M(I,I)     = 1.0,   I=1,N
M(I+1,I)   = A(I),  I=1,N-1   for BIDIAGL or
M(I,I+1)   = A(I),  I=1,N-1   for BIDIAGU
M(I,J)     = 0.0,   otherwise.
```

The right hand side elements are passed through the parameter B. The solution vector is stored in the array passed as parameter X.

Three different algorithms are used:

A. For N $\leqslant$ 100: algorithm M, described in [1], section IV.2.

This algorithm may be illustrated by the following piece of Fortran code:

```
      XX = B(1)
      X(1) = XX
      I = 2
      GOTO (1,2,...100), N
  100 XX = B(I) - A(I) * XX
      X(I) = XX
      I = I+1
   99 XX = B(I) - A(I) * XX
      X(I) = XX
      I = I+1
   98 XX = B(I) - A(I) * XX
```

```
      XX(I) = XX
      I = I+1
      .
      .
      .
    3 XX = B(I) - A(I) * XX
      X(I) = XX
      I = I+1
    2 XX = B(N) - A(N) * XX
      X(N) = XX
    1 CONTINUE
C     THERE IS ONLY ONE BRANCH IN THIS CODE.
```

B. For 100<N<223: algorithm P, described in [1], section IV.4.

The algorithm is divided into three steps:

1. The array A is swapped into the registers.
2. The system is solved; the elements of the right hand side are read from the B array; the matrix elements in the registers are overwritten by the elements of the solution vector. This step uses cyclic reduction on the fly, where one reduction is performed for every two equations.
3. The solution vector is swapped from the registers into the X array.

C. For N>222: algorithm R, described in [1], section IV.5.

This algorithm is divided in 3 phases:

1. One step of cyclic reduction in vector mode is applied; the matrix elements and the right hand side elements of the reduced system are stored alternatingly in a single array BC.
2. The new system is divided into blocks of 110 or less equations; for each block the following steps are executed:

   1. The appropriate section of the BC array is swapped into the registers.
   2. The equations of this block are solved; all input data is already in the registers and the results are left in the registers; this step uses cyclic reduction on the fly, where three reductions are performed for every five equations.
   3. The block of results is swapped into the X array.

   Step 1 and step 3 are implemented by a single hardware instruction.

3. The results of the reduced system are used in a single back-substitution step in vector mode to compute the remaining elements of the result vector.

4. References

[1] Schlichting, J.J.F.M., and Van der Vorst, H.A., Solving bidiagonal systems of linear equations on the CDC Cyber 205, CWI, Report NM-R8725, November 1987.

5. Parameters

X - REAL array of DIMENSION at least (N).
On exit X contains the elements of the solution vector.

A - REAL array of DIMENSION at least (N−1).
On entry A must contain the elements of the lower (BIDIAGL) or upper (BIDIAGU) diagonal of the matrix.
Unchanged on exit.

B - REAL array of DIMENSION at least (N).
On entry B must contain the right hand side elements.
Unchanged on exit, but see Section 11.

N - INTEGER.
On entry N must contain the number of equations to be solved. N must not exceed 65534.

6. Error indicators and warnings

None.

7. Auxiliary routines

None.

8. Timing

The time required per equation decreases with the size of the system. For a system of 60000 equations 5.5 cycles of 10 nanoseconds per equation are needed. The example given under 13 in this report shows the performance for different sizes of the system of equations.

9. Storage

Internally declared arrays occupy 2 words of storage per equation.

10. Accuracy

Accuracy may be different from straightforward code, either better or worse. For diagonal dominant systems the accuracy is about the same as for standard recursion.

11. Further comments

If the routine is called with the same name for parameters X and B then the solution vector will overwrite the right hand side.

11.1. Vectorization information

For maximal performance these routines are written in META, the assembler language for the Cyber 205.

12. Keywords

Bidiagonal linear system.
Cyber 205.
Cyclic reduction.
Scalar optimization.
Vectorization.

13. Example

This program tests the correctness and measures the performance of BIDIAGL for systems of different sizes in cycles of 10 nanoseconds per equations and in megaflops, i.e., millions of floating point operations per second, where one equation requires two such operations.

13.1. Program text

```
      PROGRAM BID
      COMMON /DD/ DUM,X(0:65023),B(0:65023),DQ(5),A(0:65023)
      COMMON/YY/ Y(0:65023)
      CHARACTER*5 GB
      CALL VRANF(A,65002)
      A(0) = 0.0
      A(1;65000) = VAINT(A(1;65000)*1E3;65000)/1024.0
      CALL VRANF(Y,65002)
      Y(0;65001) = VAINT(Y(0;65001)*1E3;65001)/1024.0
      B(0) = Y(0)
      B(1;65000) = Y(1;65000)+A(1;65000)*Y(0;65000)
C
      DO 50 K = 1,3
      X(0;K+1) = -777
      PRINT 2,K
      CALL BIDIAGL(X,A(1),B,K)
      PRINT 3,(I+1,A(I),B(I),X(I),I=0,K-1)
   50 CONTINUE
      PRINT 4
      N = 1
      NT = 500000
   10 CONTINUE
      X(0;N) = -7777
      CALL BIDIAGL(X,A(1),B,N)
      S = SECOND()
      DO 100 J = 1,NT
      CALL BIDIAGL(X,A(1),B,N)
  100 CONTINUE
      S = SECOND()-S
      GB = ' BAD '
      DO 22 I =0,N-1
      IF(ABS(X(I)-Y(I)) .GE. 1.0E-6) GOTO 24
   22 CONTINUE
      GB = '     '
   24 C = S/(NT*(N))*5E7
      F = 100.0/C
      PRINT 1,N,NT,S,C,F,GB
      N = N+9
      IF (N.GT.100) N = N+50
      IF (N.GT.500) N=N+250
      IF(N.GT.1500) N = 2*N
      NT = 5E6/N
      IF(N.LT.65000) GOTO 10
```

```
      STOP
    1 FORMAT(I8,I11,3F10.3,A)
    2 FORMAT(/,' TEST LENGTH',I4,/,
     1' N      A(I)       B(I)       X(I)')
    3 FORMAT(I5,3F12.4)
    4 FORMAT(/,' PERFORMANCE TEST',/,/,
     1' SIZE OF   REPEAT    TIME IN  CYCLES    MFLOP',/,
     2' SYSTEM    COUNT     SECONDS  PER EQ.   RATE',/)
      END
```

13.2. Program data

None.

13.3. Program results

```
 TEST LENGTH   1
   N       A(I)        B(I)        X(I)
   1       0.0000      0.5166      0.5166

 TEST LENGTH   2
   N       A(I)        B(I)        X(I)
   1       0.0000      0.5166      0.5166
   2       0.3789      0.8657      0.6699

 TEST LENGTH   3
   N       A(I)        B(I)        X(I)
   1       0.0000      0.5166      0.5166
   2       0.3789      0.8657      0.6699
   3       0.8496      1.0194      0.4502

PERFORMANCE TEST

 SIZE OF     REPEAT      TIME IN   CYCLES      MFLOP
 SYSTEM      COUNT       SECONDS   PER EQ.     RATE

      1      500000      1.470    147.010      0.680
     10      500000      2.920     29.202      3.424
     19      263157      2.000     20.001      5.000
     28      178571      1.679     16.787      5.957
     37      135135      1.522     15.217      6.571
     46      108695      1.422     14.219      7.033
     55       90909      1.351     13.513      7.400
     64       78125      1.304     13.037      7.671
     73       68493      1.262     12.620      7.924
     82       60975      1.237     12.374      8.082
     91       54945      1.212     12.122      8.250
    100       50000      1.180     11.801      8.474
    159       31446      1.069     10.692      9.353
    218       22935      0.969      9.692     10.318
    277       18050      0.875      8.753     11.425
```

336	14880	0.810	8.102	12.342
395	12658	0.766	7.664	13.048
454	11013	0.745	7.451	13.421
763	6553	0.662	6.618	15.110
1072	4664	0.624	6.242	16.021
1381	3620	0.607	6.072	16.468
3380	1479	0.567	5.674	17.625
7378	677	0.552	5.528	18.088
15374	325	0.546	5.459	18.317
31366	159	0.542	5.431	18.412
63350	78	0.549	5.556	18.000