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-syn'max 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.

Copyright © Stichting Mathematisch Centrum, Amsterdam
NUMVEC FORTRAN Library manual

Chapter: Basic Linear Algebra
Routine: MATMUL

Chapter: Simultaneous Linear Equations
Routines: BIDIAGL and BIDIAGU

J.J.F.M. Schlichting
Contro 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
reduction 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.
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-fol'd 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 >= 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 >= 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 >= 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 >= K.
       Unchanged on exit.

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

   LDC - INTEGER.
       On entry LDC specifies the first dimension of array C as declared in the calling (sub)program
       LDC >= 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.
MATMUL

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 For-
tran 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 20 J = 1, M
DO 20 I = 1, K
20 D(I,J) = 0.0
DO 30 J = 1, L
DO 30 I = 1, K
30 D(I,J) = D(I,J) + A(I,J)*B(J,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)) GOTO 123
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 ', 2F9.3, ' SECONDS MFL = ', F7.2, ';
2 TIME FOR MATMUL ROUTINE ', 2F9.3, ' 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
**Simultaneous Linear Equations**

**BIDIAGL and BIDIAGU**

**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:

   \[
   M(1,i) = 1.0, \quad i = 1, n \\
   M(i+1,1) = A(i), \quad i = 1, n-1 \quad \text{for BIDIAGL or} \\
   M(i,1) = A(i), \quad i = 1, n-1 \quad \text{for BIDIAGU} \\
   M(i,i) = 0.0, \quad \text{otherwise.}
   \]

   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 \( N \leq 100 \): algorithm M, described in [1], section IV.2.

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

   ```fortran
   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
   ```

   Page 1
BIDIAGL and BIDIAGU

\[ xx(i) = xx \\
1 = 1 + 1 \\
\ldots \\
3 \quad xx = b(i) - a(i) \cdot xx \\
   \quad x(i) = xx \\
   \quad 1 = 1 + 1 \\
2 \quad xx = b(n) - a(n) \cdot xx \\
   \quad x(n) = xx \\
1 \quad \text{CONTINUE} \\
C \quad \text{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

**Simultaneous Linear Equations**

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 12 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

- Bidigonal linear system.
- Cyber 205.
- Cycle 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/ DJM,X(0:65023),B(0:65023),DQ(5),A(0:65023)
COMMON/VY/ Y(0:65023)
CHARACTER*5 GB
CALL VRANF(A,65002)
A(0) = 0.0
A(1:65000) = VAIN(T(A(1:65000)*1E3;65000))/1024.0
CALL VRANF(Y,65002)
Y(0:65001) = VAIN(T(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 I = 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.E-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
```

***Simultaneous Linear Equations

BIDIAGL and BIDIAGU

STOP
1 FORMAT(18,11,3f10.3,a)
2 FORMAT(/,' TEST LENGTH',j4,/)
1' N A(I) B(I) X(I)
3 FORMAT(15,3f12.4)
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

<table>
<thead>
<tr>
<th>TEST LENGTH 1</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>N A(I) B(I) X(I)</td>
<td>1</td>
<td>0.0000</td>
<td>0.5166</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TEST LENGTH 2</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>N A(I) B(I) X(I)</td>
<td>1</td>
<td>0.0000</td>
<td>0.5166</td>
</tr>
<tr>
<td>2</td>
<td>0.3789</td>
<td>0.8657</td>
<td>0.6699</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TEST LENGTH 3</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>N A(I) B(I) X(I)</td>
<td>1</td>
<td>0.0000</td>
<td>0.5166</td>
</tr>
<tr>
<td>2</td>
<td>0.3789</td>
<td>0.8657</td>
<td>0.6699</td>
</tr>
<tr>
<td>3</td>
<td>0.8496</td>
<td>1.0194</td>
<td>0.4502</td>
</tr>
</tbody>
</table>

<p>| PERFORMANCE TEST | | | |</p>
<table>
<thead>
<tr>
<th>SIZE OF SYSTEM</th>
<th>REPEAT COUNT</th>
<th>TIME IN SECONDS</th>
<th>CYCLES PER EQ.</th>
<th>MFLOP RATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>500000</td>
<td>1.470</td>
<td>147.010</td>
<td>0.680</td>
</tr>
<tr>
<td>10</td>
<td>500000</td>
<td>2.920</td>
<td>29.202</td>
<td>3.424</td>
</tr>
<tr>
<td>19</td>
<td>263157</td>
<td>2.000</td>
<td>20.001</td>
<td>5.000</td>
</tr>
<tr>
<td>28</td>
<td>178571</td>
<td>1.679</td>
<td>16.787</td>
<td>5.957</td>
</tr>
<tr>
<td>37</td>
<td>135135</td>
<td>1.522</td>
<td>15.217</td>
<td>6.571</td>
</tr>
<tr>
<td>46</td>
<td>108695</td>
<td>1.422</td>
<td>14.219</td>
<td>7.033</td>
</tr>
<tr>
<td>55</td>
<td>90909</td>
<td>1.351</td>
<td>13.513</td>
<td>7.400</td>
</tr>
<tr>
<td>64</td>
<td>78125</td>
<td>1.304</td>
<td>13.037</td>
<td>7.671</td>
</tr>
<tr>
<td>73</td>
<td>68493</td>
<td>1.262</td>
<td>12.620</td>
<td>7.924</td>
</tr>
<tr>
<td>82</td>
<td>60975</td>
<td>1.237</td>
<td>12.374</td>
<td>8.082</td>
</tr>
<tr>
<td>91</td>
<td>54945</td>
<td>1.212</td>
<td>12.122</td>
<td>8.250</td>
</tr>
<tr>
<td>100</td>
<td>500000</td>
<td>1.180</td>
<td>11.801</td>
<td>8.474</td>
</tr>
<tr>
<td>159</td>
<td>31446</td>
<td>1.069</td>
<td>10.692</td>
<td>9.353</td>
</tr>
<tr>
<td>218</td>
<td>22935</td>
<td>0.969</td>
<td>9.692</td>
<td>10.318</td>
</tr>
<tr>
<td>277</td>
<td>18050</td>
<td>0.875</td>
<td>8.753</td>
<td>11.425</td>
</tr>
</tbody>
</table>

Page 5
**BIDIAGL and BIDIAGU**

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

**Simultaneous Linear Equations**