Centrum voor Wiskunde en Informatica
Centre for Mathematics and Computer Science


Optimization of the real level 2 BLAS on the CYBER 205

Department of Numerical Mathematics Report NM-R8715 July
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 Pure Research (Z.W.O.).
Optimization of the real Level 2 BLAS on the Cyber 205

Department of Numerical Mathematics
Centre for Mathematics and Computer Science
P.O. Box 4709, 1009 AB Amsterdam,
The Netherlands

The results of the implementation and optimization of the real Level 2 BLAS routines on the Cyber 205 vectorcomputer are presented. The Level 2 BLAS routines perform three types of matrix-vector operations, viz., matrix-vector multiplication, rank-1 and rank-2 updates, and solution of triangular systems of equations. The performance of the routines varies between 60% and 80% of the maximum Cyber 205 performance, for general matrices of order 500, and for band matrices of order 30000 with 6 non-zero diagonals.

1980 Mathematics Subject Classification (1985 Revision): 65V05, 65F05, 65F30
1980 CR Categories: 5.14, 4.6
Key Words & Phrases: matrix-vector operations, Level 2 BLAS, Cyber 205 optimization

INTRODUCTION
A good compromise between portability and efficiency of numerical software can be found in the LINPACK-Library ([1]), a library of numerical linear algebra routines, which is available on many large scientific computers. In order to optimize this library on a given machine, only a small core of routines which perform basic operations on vectors, the so-called BLAS, has to be adapted to the specific speed requirements of this machine. The remainder of this library, written in portable ANSI Fortran 77, can be installed unchanged on the given machine.

However, with the advent of supercomputers like the Cray 1 and the Cyber 205 one realized ([2]) that in order to get maximal performance on these machines it is necessary to optimize on the matrix-vector level, rather than on the vector-vector level. To that end, Dongarra et al. have proposed a set of Extended BLAS or Level 2 BLAS routines ([2, 3]). This set performs three types of matrix-vector operations, viz., matrix-vector multiplication, rank-1 and -2 updates and solution of triangular systems of equations. Four Fortran data types of the matrices involved are defined, viz., REAL, DOUBLE PRECISION, COMPLEX, and COMPLEX*16 or DOUBLE COMPLEX (if available).

We have optimized on a CDC Cyber 205 the real versions of the Extended BLAS. With these routines we have obtained nearly (between 60% and 80% of the) maximum performance of this machine with full matrices of order \( \geq 500 \). The Cyber 205 optimized routines are available in the NUMVEC-Library and the NUMVEC-Letter presents more information about this library ([5]).

In Section 2 of this paper we give a concise description of the Extended BLAS. Section 3 provides details of how we have optimized the real Level 2 BLAS on a Cyber 205 vectorcomputer. In Section 4 we present a selection of results obtained with a Master Test Program of DONGARRA ([4]) together with results of efficiency tests, for the 16 real Level 2 BLAS routines.
2. THE EXTENDED BLAS
The set of Extended BLAS is subdivided into three subsets which perform the following types of operations:

a) **Matrix-vector products** of the form

\[ y \leftarrow \alpha Ax + \beta y, \quad y \leftarrow \alpha A^T x + \beta y, \quad y \leftarrow \alpha A^T x + \beta y. \]

where \( \alpha \) and \( \beta \) are scalars, \( x \) and \( y \) are vectors, \( A \) is a matrix, and of the form

\[ x \leftarrow Tx, \quad x \leftarrow T^T x, \quad x \leftarrow \overline{T}^T x \]

where \( x \) is a vector and \( T \) is an upper or lower triangular matrix.

b) **Rank-one and rank-two updates** of the form

\[ A \leftarrow \alpha xy^T + A, \quad A \leftarrow \alpha yx^T + A, \]
\[ H \leftarrow \alpha xx^T + H, \quad H \leftarrow \alpha xy^T + \alpha yx^T + H, \]

where \( H \) is a Hermitian matrix.

c) **Solution of triangular equations** of the form

\[ x \leftarrow T^{-1} x, \quad x \leftarrow T^{-T} x, \quad x \leftarrow \overline{T}^T x, \]

where \( T \) is a non-singular upper or lower triangular matrix.

The matrices can be general, general band, Hermitian, Hermitian band, triangular and triangular band (where appropriate); the operations can be done in real and complex arithmetic, in single and double precision (In appendix B to [2] some additional routines are proposed which allow extended precision matrix-vector operations to be performed).

For the names of the various routines the following convention is adopted: the first character indicates the Fortran data type of the matrix:

- \( S \) REAL
- \( D \) DOUBLE PRECISION
- \( C \) COMPLEX
- \( Z \) COMPLEX*16 OR DOUBLE COMPLEX

characters **two** and **three** denote the kind of matrix:

- \( GE \) General matrix
- \( GB \) General band matrix
- \( SY \) Symmetric matrix
- \( SP \) Symmetric matrix stored in packed form
- \( SB \) Symmetric band matrix
- \( HE \) Hermitian matrix
- \( HP \) Hermitian matrix stored in packed form
- \( HB \) Hermitian band matrix
- \( TR \) Triangular matrix
- \( TP \) Triangular matrix in packed form
- \( TB \) Triangular band matrix
characters four and five denote the type of operation:

- **MV** Matrix-vector product
- **R** Rank-one update
- **R2** Rank-two update
- **SV** Solve a system of equations

The available combinations are indicated in the following table. In the first column (headed complex) the initial C may be replaced by Z. In the second column (headed real) the initial S may be replaced by D. We have implemented on the Cyber 205 the real Level 2 BLAS routines. The complex routines will be finished shortly. We have not implemented the D-routines since on the Cyber 205 the single-precision wordlength (47 bits mantissa) is sufficient for most applications.

<table>
<thead>
<tr>
<th>complex</th>
<th>real</th>
<th>MV</th>
<th>R</th>
<th>R2</th>
<th>SV</th>
</tr>
</thead>
<tbody>
<tr>
<td>CGE</td>
<td>SGE</td>
<td>*</td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CGB</td>
<td>SGB</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CHE</td>
<td>SSY</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>CHP</td>
<td>SSP</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>CHB</td>
<td>SSB</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CTR</td>
<td>STR</td>
<td>*</td>
<td></td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>CTP</td>
<td>STP</td>
<td>*</td>
<td></td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>CTB</td>
<td>STB</td>
<td>*</td>
<td></td>
<td></td>
<td>*</td>
</tr>
</tbody>
</table>

For details of the Extended BLAS (Parameter Conventions, Storage Conventions, Subroutine Specifications, Calling Sequences, Applicability of the Extended BLAS in the LINPACK and EISPACK-libraries) the reader is referred to [2]. Here, we present only the calling sequences for all the real Level 2 Blas routines (Table 2.1).

<table>
<thead>
<tr>
<th>name</th>
<th>options</th>
<th>dim</th>
<th>b-width</th>
<th>scalar</th>
<th>matrix</th>
<th>x-vector</th>
<th>scalar</th>
<th>y-vector</th>
</tr>
</thead>
<tbody>
<tr>
<td>SGEMV</td>
<td>(TRANS, M,N)</td>
<td>ALPHA, A, LDA, X, INCX, BETA, Y, INCY</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SGBMV</td>
<td>(TRANS, M,N, KL, KU)</td>
<td>ALPHA, A, LDA, X, INCX, BETA, Y, INCY</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSYMV</td>
<td>(UPLO, N)</td>
<td>ALPHA, A, LDA, X, INCX, BETA, Y, INCY</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSBMV</td>
<td>(UPLO, N, K)</td>
<td>ALPHA, A, LDA, X, INCX, BETA, Y, INCY</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSPMV</td>
<td>(UPLO, N)</td>
<td>ALPHA, AP, X, INCX, BETA, Y, INCY</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STRMV</td>
<td>(UPLO,TRANS, DIAG, N)</td>
<td>A, LDA, X, INCX</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STBMV</td>
<td>(UPLO,TRANS, DIAG, N, K)</td>
<td>A, LDA, X, INCX</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STPMV</td>
<td>(UPLO,TRANS, DIAG, N)</td>
<td>AP, X, INCX</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STRSV</td>
<td>(UPLO,TRANS, DIAG, N)</td>
<td>A, LDA, X, INCX</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STBSV</td>
<td>(UPLO,TRANS, DIAG, N, K)</td>
<td>A, LDA, X, INCX</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STPSV</td>
<td>(UPLO,TRANS, DIAG, N)</td>
<td>AP, X, INCX</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The calling sequences for the real Level 2 BLAS.

<table>
<thead>
<tr>
<th>name</th>
<th>options</th>
<th>dim</th>
<th>scalar</th>
<th>x-vector</th>
<th>y-vector</th>
<th>matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>SGER</td>
<td>(M,N, ALPHA, X, INCX, Y, INCY, A, LDA)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSYR</td>
<td>(UPLO, N, ALPHA, X, INCX, A, LDA)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSPR</td>
<td>(UPLO, N, ALPHA, X, INCX, AP)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSYR2</td>
<td>(UPLO, N, ALPHA, X, INCX, Y, INCY, A, LDA)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSPR2</td>
<td>(UPLO, N, ALPHA, X, INCX, Y, INCY, AP)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The arguments that specify the options have the following meaning:

<table>
<thead>
<tr>
<th>name</th>
<th>value</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>UPLO</td>
<td>'U'</td>
<td>Upper triangle</td>
</tr>
<tr>
<td></td>
<td>'L'</td>
<td>Lower triangle</td>
</tr>
<tr>
<td>TRANS</td>
<td>'N'</td>
<td>Operate with the matrix</td>
</tr>
<tr>
<td></td>
<td>'T' or 'C'</td>
<td>Operate with the transpose of the matrix</td>
</tr>
<tr>
<td>DIAG</td>
<td>'U'</td>
<td>The matrix is unit triangular</td>
</tr>
<tr>
<td></td>
<td>'N'</td>
<td>The matrix is non-unit triangular</td>
</tr>
</tbody>
</table>

The size of the matrix is determined by the arguments M and N for an m by n rectangular matrix, and by the argument N for an n by n symmetric or triangular matrix. The bandwidth of a matrix is determined by the arguments KL and KU for a rectangular matrix with kl sub-diagonals and ku super-diagonals, and by the argument K for a symmetric or triangular matrix with k sub-diagonals and/or super-diagonals. The matrix can be described by the array name (A) followed by the leading dimension (LDA) of the array as declared in the calling (sub) program, when the matrix is stored in a two-dimensional array. When the matrix is being stored as a (packed) vector, it is described by the array name (AP) alone. The scalars a and b are described by the arguments ALPHA and BETA, respectively. The vectors x and y are described by the arguments X and Y, respectively, each followed by the storage spacing in the array of the vector elements, INCX and INCY, respectively. For example, a call to SGEMV looks as follows:

```
CALL SGEMV(TRANS, M, N, ALPHA, A, LDA,
+ X, INCX, BETA, Y, INCY)
```

This activates one of the following two real matrix-vector operations:

\[
y \leftarrow \alpha A x + \beta y \quad (\text{TRANS}='N'), \quad \text{or} \quad y \leftarrow \alpha A^T x + \beta y \quad (\text{TRANS}='T' \text{ or } 'C').
\]

A is a general M×N matrix, ALPHA and BETA are scalars, X and Y are vectors. LDA is the leading dimension of A (declared in the calling (sub) program), INCX and INCY specify the increments for the elements of x and y, respectively.
3. IMPLEMENTATION OF THE REAL LEVEL 2 BLAS ON THE CYBER 205

3.1. Matrix-vector multiplication on the Cyber 205

On the Cyber 205, operations on arrays should be carried out as much as possible on elements which
are stored in contiguous memory locations. Since in Fortran elements of two-dimensional arrays are
stored columnwise, the ordinary matrix-vector multiplication should be organized in terms of opera­tions
on columns (which could be rows of the matrix in the transposed case). We shall discuss here
the matrix-vector multiplication \( y := Ax, A = (a_{ij}), x = (x_i), y = (y_j), i = 1, \ldots, m; j = 1, \ldots, n \), where
\( A \) is a general matrix, a symmetric matrix, or a band matrix (the triangular case runs similar to the
symmetric case).

3.1.1. \( A \) is a general matrix. The usual mathematical formulation for \( y := Ax \) is:

\[
y_i := \sum_{j=1}^{n} a_{ij}x_j = (a_{i1}, x)_{1\leq i \leq m},
\]
i.e., \( y_i \) is the inner product of the \( i \)-th row of \( A \) (denoted by \( a_i \)) and the vector \( x \). On the Cyber 205
we compute

\[
y := \sum_{j=1}^{n} x_j a_{1j},
\]
i.e., a linear combination of the columns of \( A \) with coefficients \( x_1, x_2, \ldots, x_n \). This can be done with
the well-known linked-triad (or SAXPY) construction (vector \( 1 := \text{vector 1 + scalar*vector 2} \)), hence
each column addition requires \( m \) clock cycles (apart from start-up time).

3.1.2. \( A \) is a symmetric matrix. We have \( a_{ji} = a_{ij} \) and \( m = n \), and we suppose that UPLO = 'U', so that
the matrix \( A \) is available in the upper part (including the main diagonal) of the array \( A \). Multiplica­tion of the upper part of \( A \) by \( x \) and storage into \( y \) is carried out as follows:

\[
y := x_1 \bar{a}_1 + x_2 \bar{a}_2 + \cdots + x_n \bar{a}_n,
\]
where \( \bar{a}_i \) is the \( n \)-vector \((a_{i1}, a_{i2}, \ldots, a_{in}, 0, \cdots, 0)^T, i = 1, 2, \ldots, n \). Multiplication of the lower
part of \( A \) by \( x \) and addition to \( y \) proceeds as follows:

\[
y_i = y_i + \sum_{j=1}^{i-1} a_{ij}x_j, \quad i = 2, \ldots, n.
\]

In both cases, columns of \( A \) are referenced, i.e., elements of \( A \) which are stored in contiguous memory
locations. On the Cyber 205, the upper part computation is carried out with the aid of the linked­triad
construction, and the lower part computation with the dot-product construction (cf. Section 4
for maximum obtainable performances). We emphasize that here, due to the separate treatment of
the upper and the lower part of the matrix \( A \), the total number of vector operations is \( 2n - 1 \),
compared with \( n \) in the general case. Consequently, the performance in the symmetric case will be worse
than in the general case (because of \( n - 1 \) more vector start-up times needed). When the symmetric
matrix \( A \) is stored in packed form, the elements of \( A \) are stored in a linear, rather than a 2-
dimensional, array in the following order:

\[
a_{11}, \quad a_{12}, a_{22}, \quad a_{13}, a_{23}, a_{33}, \ldots, a_{1n}, a_{2n}, \ldots, a_{nn},
\]
col.1 \quad col.2 \quad col.3 \quad col.n

Hence, the algorithm for the packed symmetric case runs similar to the non-packed symmetric case.
3.1.3. A is a band matrix. In the convention of the Level 2 BLAS, band matrices are supposed to be stored in rectangular arrays such that diagonals of the matrix are stored in rows, and columns of the matrix in corresponding columns of the array. For the computation of $Ax$ we use a multiplication scheme based on diagonals of $A$, rather than on rows or columns. This allows us to write reasonably efficient vector code for band matrix multiplication, albeit that on the Cyber 205 the diagonals of the matrix $A$ have to be gathered from the rows of the array, before the necessary vector operations can be carried out. For example, suppose that $A$ is a tridiagonal matrix with $n$ rows and columns, the main diagonal being given by $a_i$, $i = 1, \ldots, n$, the subdiagonal by $b_i$, $i = 1, \ldots, n-1$ and the superdiagonal by $c_i$, $i = 2, \ldots, n$. Then the diagonal-based multiplication scheme for $y := Ax$ reads as follows:

$$
\begin{bmatrix}
Y_1 \\
Y_2 \\
Y_3 \\
\vdots \\
Y_{n-1} \\
Y_n
\end{bmatrix} = 
\begin{bmatrix}
0 \\
b_1 \\
x_1 \\
0 \\
b_{n-2} \\
x_{n-2} \\
0 \\
b_{n-1} \\
x_{n-1} \\
0
\end{bmatrix} A
\begin{bmatrix}
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0
\end{bmatrix} X
$$

On the Cyber 205, gathering of a vector from memory with constant stride requires at least $5/4$ clock cycles per element (independent of the number of pipes). Hence the time needed to compute one element of $y$ is at least $3*5/4$ clock cycles for gathering, $3$ clock cycles for multiplication and $2$ clock cycles for addition of the elements. If the gathering were not needed (namely, when the diagonals of the matrix were stored in columns of the rectangular array) then on a $p$-pipe Cyber 205 a speed-up were possible with a factor of

$$\frac{5/p + 3*5/4}{5/p} = 
\begin{cases}
1.75 & \text{for } p = 1, \\
2.5 & \text{for } p = 2, \\
4 & \text{for } p = 4.
\end{cases}$$

3.2. Cyber 205 optimization

We have made extensive use of the following Cyber 205 optimization tools. Details may be found in [6].

3.2.1. Vectorsyntax. We explain this by an example. The computation of

$$y := \beta y + \alpha Ax = \beta y + \alpha \sum_{j=1}^{n} x_j a_j$$

may be expressed in vector-syntax as follows (suppose $\beta \neq 0$ and INCX = INCY = 1):

```
Y(1:M) = BETA * Y(1:M)
DO 10 J = 1, N
    Y(1:M) = Y(1:M) + ALPHA*X(J)*A(1,J:M)
10 CONTINUE
```

Here, $Y(1:M)$ is the vector of $M$ elements starting at $Y(1)$ and $A(1,J:M)$ the vector of $M$ elements starting at $A(1,J)$ (which is the $J$-th column of $A$).
3.2.2. Descriptors. A descriptor is a pointer to a vector (of array-elements), consisting of the starting location and the length of that vector; moreover, it has a certain data type which corresponds to the data type of the vector to which it points. A descriptor may either point to a vector of already existing array-elements, explicitly declared in some DIMENSION-statement, or to a dynamically allocated vector (of array-elements). The link between a descriptor and a vector is realized by a descriptor ASSIGN-statement. Examples of both cases occur in the following piece of code.

```plaintext
REAL A(1000)
REAL AD, BD
DESCRIP TOR AD, BD
.
.
ASSIGN AD, A(1; 1000)
.
.
* INITIALIZE A(1), . . . , A(1000) TO ZERO
AD = 0.0 E-0
.
.
* ALLOCATE A DYNAMIC VECTOR OF LENGTH 1000
* TO WHICH BD POINTS
ASSIGN BD, .DYN. 1000
* COPY A(1), . . . , A(1000) TO THIS VECTOR
BD = AD
```

The starting location and the length of a vector connected to a descriptor may be changed by so-called special calls (inline machine instructions). If the descriptor BD points to a vector starting at the J-th element of an array, and if this should be changed to element K, then this is accomplished by the code:

```plaintext
IDIF = (K-J)*64
CALL Q8ADDX (BD, IDIF, BD)
```

Changing the length to L (≤ 65535) can be done as follows:

```plaintext
CALL Q8PACK (L, BD, BD)
```

Loading the 1 + 1-th element of the vector to which BD points into the REAL R is accomplished by:

```plaintext
CALL Q8LOD (BD, I, R)
```

and the reverse operation, storing, by

```plaintext
CALL Q8STO (BD, I, R).
```
3.2.3. Vector intrinsic functions. On the Cyber 205 there are many vector intrinsic functions for operating on vectors ([6, pp. 10-20/10-29]). In our implementation of the Level 2 BLAS we have made use of the following:

Q8SDOT : computes the dot-product of two vectors
Q8VREV : reverts the elements of a vector
Q8VGATHP : periodic gather of the elements of a vector
Q8VSCATP : periodic scatter of the elements of a vector.

4. Test Results with the Cyber 205 Level 2 BLAS
The Master Test Program of Dongarra et al. ([4]) was used as a debugging aid and a validation tool of our Cyber 205 implementation of the Level 2 BLAS. We have tested the 16 real Level 2 BLAS routines with the data-set as given on p. 15 of [4]. This data-set induces a total of 30,409 test-calls of the Level 2 BLAS routines and the lengths of the test-vectors vary between 0 and 9. The CPU-time on the 1-pipe Cyber 205 of SARA (Academic Computing Centre, Amsterdam) was 10.6 sec. The time to test the corresponding Fortran 77 model implementation of the Level 2 BLAS was 10.3 sec.

We have measured the efficiency of our Cyber 205 Level 2 BLAS routines by means of numerous test-calls with large vectors and matrices as input data. Below we present a selection of the results obtained. The MFLOP/s-rates (Million FLOating Point operations per second) presented there should be compared with the maximum obtainable MFLOP/s-rates for the various vector constructs employed in the Cyber 205 implementation:

<table>
<thead>
<tr>
<th>Construct (v = vector, s = scalar)</th>
<th>Maximum MFLOP/s on a Cyber 205</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1-pipe</td>
</tr>
<tr>
<td>v ← v + s*v</td>
<td>100</td>
</tr>
<tr>
<td>s ← (v, v)</td>
<td>100</td>
</tr>
<tr>
<td>v ← v + v*v</td>
<td>50</td>
</tr>
<tr>
<td>Routine</td>
<td>Number of operations</td>
</tr>
<tr>
<td>------------</td>
<td>----------------------</td>
</tr>
<tr>
<td>SGEMV</td>
<td>2MN + M + N</td>
</tr>
<tr>
<td>SGBMV</td>
<td>2N(KL+KU+2) -KL(KL+1)-KU(KU+1)</td>
</tr>
<tr>
<td>SSYMV</td>
<td>2N² + 2N</td>
</tr>
<tr>
<td>SSPMV</td>
<td></td>
</tr>
<tr>
<td>SSBMV</td>
<td>4N(K+1)-2K(K+1)</td>
</tr>
<tr>
<td>STRMV</td>
<td>N²</td>
</tr>
<tr>
<td>STPMV</td>
<td></td>
</tr>
<tr>
<td>STBMV</td>
<td>N(2K+1)-K(K+1)</td>
</tr>
<tr>
<td>SGER</td>
<td>2MN + MIN(M,N)</td>
</tr>
<tr>
<td>SSYR</td>
<td>N² + 2N</td>
</tr>
<tr>
<td>SSPR</td>
<td></td>
</tr>
<tr>
<td>SSYR2</td>
<td>2N² + 2N</td>
</tr>
<tr>
<td>SSPR2</td>
<td></td>
</tr>
<tr>
<td>STRSV</td>
<td>2N²</td>
</tr>
<tr>
<td>STPSV</td>
<td></td>
</tr>
<tr>
<td>STBSV</td>
<td>N(2K+1)-K(K+1)</td>
</tr>
</tbody>
</table>

Table 4.1. Numbers of operations for the various Level 2 BLAS routines.

The last construct is being used in the band matrix routines (except for STBSV), so that, e.g. on a 1-pipe the maximum performance of these routines is 50 MFLOP/s (if the diagonals were stored columnwise). For the other routines this maximum is 100 MFLOP/s, and for STBSV only scalar speed can be obtained.

In order to compute the MFLOP/s-rates of the Cyber 205 implementation we have calculated, for some choices of the arguments of the various routines, the number of operations. (Table 4.1).
The results presented in Table 4.2 were obtained with the following values of the arguments of the routines:

- \( \text{UPLO} = 'U', \text{TRANS} = 'N', \text{DIAG} = 'N'; \)
- \( M=N=500 \) for full matrices, \( M=N=30,000 \) for band matrices;
- \( KL = 3 \), \( KU = 2 \), for SGBMV;
- \( K=2 \) for SSBMV, \( K=5 \) for STBMV and for STBSV;
- \( \alpha=0.7, \beta=0.9. \)

For the model implementation the differences between packed and non-packed versions are due to the fact that the loop

\[
\begin{align*}
\text{K} &= \text{KK} \\
\text{DO} \ 10, \ I=1,J \\
\text{AP}(K) &= \text{AP}(K) + X(I) \times \text{TEMP} \\
\text{K} &= \text{K} + 1 \\
10 & \text{ CONTINUE}
\end{align*}
\]

is not vectorized automatically by the FTN200-compiler.
The MFLOP/s-rates show that an efficiency of more than 60% of the maximal performance is achieved. For band matrix routines, the $v \leftarrow v + v^*v$ constructs would limit the maximal performance to 50 MFLOP/s if the diagonals of the matrix were stored columnwise. Because of the rowwise storage, prescribed for the Level 2 BLAS this maximum is reduced further to a maximal performance not greater than about 31 MFLOP/s for SGBMV and STBMV, and about 38 MFLOP/s for SSBMV (on a 1-pipe Cyber 205).

In order to get an impression of which part of the timings given in Table 4.2 is spent to vectorprocessing and which part to overhead (like subroutine call, loop administration and start-up times), we have tabulated in Table 4.3, for STBMV, the vector operations carried out in this subroutine and the theoretical number of clock cycles (of 20 nsec.) needed in each vector operation. From the total number of clock cycles we have computed the corresponding theoretical CPU-times for the example of Table 4.2 (with $N = 30,000$ and $K = 5$). A comparison with the measured CPU-times shows that the overhead time is not too large compared to the real processing time in subroutine STBMV, for this example. A similar conclusion may be drawn for the other routines.

The results of Table 4.2 show that, in most cases, the Cyber 205 implementation gives a better performance than the Fortran 77 model implementation. In [4], the Fortran 77 model implementation of the Level 2 BLAS was shown to reach a moderate efficiency on the CRAY-1S, by a test with $M = N = 256$, $INCX = INCY = 1$, $UPLO = 'U'$ and $DIAG = 'N'$. We have carried out the same test on a 1-pipe Cyber 205 (with $\alpha = 0.7$ and $\beta = 0.9$), both with the model implementation and with our Cyber 205 implementation. Table 4.4 gives the observed speeds in MFLOP/s, and for comparison, those on the CRAY-1S. Apart from the case STRMV, TRANS = 'T', the results show a better performance on the Cyber 205 than on the CRAY-1S. One should realize here that automatic vectorization works because of the simplest possible choices of INCX (= 1) and INCY (= 1) involved. In all other cases one should, in order to obtain vector speed on the Cyber 205, resort to the Cyber 205 optimized routines. The poor result for STRMV, TRANS = 'T' is due to the fact that the loop

```
DO 90, I = J-1, 1, -1
  TEMP = TEMP + A(I,J)*X(I)
90 CONTINUE
```

is not vectorized by the FTN200-compiler. In the other cases it appears that on the Cyber 205 the model implementation is at least as efficient as on the CRAY-1S. However, it is to be expected that the CRAY-1S-results can be improved considerably with the help of Cray Assembly Language. We are not acquainted with the existence of such a Level 2 BLAS implementation.

**ACKNOWLEDGEMENT.**

We are grateful to Dr. G. v.d. Velde of Control Data Rijswijk for running the experiments of Tables 4.2 and 4.4 on a 2-pipe Cyber 205.
<table>
<thead>
<tr>
<th>Vector Operation</th>
<th>INCX = 1</th>
<th></th>
<th>INCX = -2</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1-pipe</td>
<td>2-pipe</td>
<td>1-pipe</td>
<td>2-pipe</td>
</tr>
<tr>
<td></td>
<td>Cyber 205</td>
<td></td>
<td>Cyber 205</td>
<td></td>
</tr>
<tr>
<td>gather x</td>
<td></td>
<td>5/4</td>
<td>5/4</td>
<td></td>
</tr>
<tr>
<td>gather main diagonal</td>
<td>5/4</td>
<td>5/4</td>
<td>5/4</td>
<td>5/4</td>
</tr>
<tr>
<td>multiply by x and store into h</td>
<td>1</td>
<td>1/2</td>
<td>1</td>
<td>1/2</td>
</tr>
<tr>
<td>gather other diagonal</td>
<td>5/4</td>
<td>5/4</td>
<td>5/4</td>
<td>5/4</td>
</tr>
<tr>
<td>multiply by x K times</td>
<td>1</td>
<td>1/2</td>
<td>1</td>
<td>1/2</td>
</tr>
<tr>
<td>add result to h</td>
<td>1</td>
<td>1/2</td>
<td>1</td>
<td>1/2</td>
</tr>
<tr>
<td>store h into x</td>
<td>1</td>
<td>1/2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>scatter x</td>
<td></td>
<td>5/4</td>
<td>5/4</td>
<td></td>
</tr>
<tr>
<td>Total number of clock cycles per vector element</td>
<td>13(K+1)/4</td>
<td>9(K+1)/4</td>
<td>(19+13K)/4</td>
<td>(17+9K)/</td>
</tr>
<tr>
<td>Corresponding theoretical CPU-times for K = 5 and N = 30,000</td>
<td>11.7 msec.</td>
<td>8.1 msec.</td>
<td>12.6 msec.</td>
<td>9.3 msec.</td>
</tr>
<tr>
<td>Measured CPU-times</td>
<td>13.7 msec.</td>
<td>10.0 msec.</td>
<td>14.6 msec.</td>
<td>11.5 msec.</td>
</tr>
</tbody>
</table>

**Table 4.3.** Theoretical numbers of clock cycles per vector element needed in the vector operations in subroutine STBMV.

<table>
<thead>
<tr>
<th>Routine</th>
<th>TRANS on CRAY-1S</th>
<th>TRANS on Cyber 205</th>
<th>Speed of model implementation</th>
<th>Speed of Cyber 205</th>
</tr>
</thead>
<tbody>
<tr>
<td>SGEMV</td>
<td>'N'</td>
<td>39</td>
<td>55(75)</td>
<td>62(90)</td>
</tr>
<tr>
<td></td>
<td>'T'</td>
<td>31</td>
<td>55(56)</td>
<td>58(58)</td>
</tr>
<tr>
<td>SSYMV</td>
<td>31</td>
<td>45(51)</td>
<td>58(51)</td>
<td></td>
</tr>
<tr>
<td>STRMV</td>
<td>'N'</td>
<td>33</td>
<td>34(41)</td>
<td>43(56)</td>
</tr>
<tr>
<td></td>
<td>'T'</td>
<td>20</td>
<td>6(6)</td>
<td>38(39)</td>
</tr>
<tr>
<td>SGER</td>
<td>39</td>
<td>56(78)</td>
<td>64(95)</td>
<td></td>
</tr>
<tr>
<td>SSYR</td>
<td>36</td>
<td>38(47)</td>
<td>47(61)</td>
<td></td>
</tr>
<tr>
<td>SSYR2</td>
<td>47</td>
<td>45(58)</td>
<td>54(73)</td>
<td></td>
</tr>
</tbody>
</table>

*) The numbers in parentheses give the corresponding speeds on a 2-pipe Cyber 205.

**Table 4.4.** Comparison of the efficiency of some Level 2 BLAS routines on the CRAY-1S and the Cyber 205.
REFERENCES.
5. NUMVEC, A Library of NUMerical software for VECtor and Parallel Processors. This library is being developed since 1984. Contributions are documented in CWI-Reports. A survey is issued in the NUMVEC-Letter, which can be obtained from the NUMVEC-Library Manager (H.J.J. te Riele) at the CWI (for the address see the title page of this paper).