Algorithms and Applications on Vector and Parallel Computers
H.J.J. te Riele, Th.J. Dekker and H.A. van der Vorst (Editors)
© Elsevier Science Publishers B.V. (North-Holland), 1987

Optimization of the real Level 2 BLAS on the Cyber 205

W.M. Lioen, M. Louter-Nool and H.J.J. te Riele
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.

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 $\geqslant$ 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

$$y \leftarrow \alpha A x + \beta y, \quad y \leftarrow \alpha A^T x + \beta y, \quad y \leftarrow \alpha \bar{A}^T x + \beta y,$$

where α and β are scalars, x and y are vectors, A is a matrix, and of the form

$$x \leftarrow Tx, \quad x \leftarrow T^T x, \quad x \leftarrow \bar{T}^T x$$

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

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

$$A \leftarrow \alpha x y^T + A, \quad A \leftarrow \alpha x \bar{y}^T + A,$$
$$H \leftarrow \alpha x \bar{x}^T + H, \quad H \leftarrow \alpha x \bar{y}^T + \bar{\alpha} y \bar{x}^T + H,$$

where H is a Hermitian matrix.

c) *Solution of triangular equations* of the form

$$x \leftarrow T^{-1} x, \quad x \leftarrow T^{-T} x, \quad x \leftarrow \bar{T}^{-T} x,$$

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.

complex	real	MV	R	R2	SV
CGE	SGE	*	*		
CGB	SGB	*			
CHE	SSY	*	*	*	
CHP	SSP	*	*	*	
CHB	SSB	*			
CTR	STR	*			*
CTP	STP	*			*
CTB	STB	*			*

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).

name	options	dim	b-width	scalar	matrix	x-vector	scalar	y-vector
SGEMV	(TRANS,	M,N,		ALPHA,	A, LDA,	X, INCX,	BETA,	Y, INCY)
SGBMV	(TRANS,	M,N,	KL, KU,	ALPHA,	A, LDA,	X, INCX,	BETA,	Y, INCY)
SSYMV	(UPLO,	N,		ALPHA,	A, LDA,	X, INCX,	BETA,	Y, INCY)
SSBMV	(UPLO,	N,	K,	ALPHA,	A, LDA,	X, INCX,	BETA,	Y, INCY)
SSPMV	(UPLO,	N,		ALPHA,	AP,	X, INCX,	BETA,	Y, INCY)
STRMV	(UPLO, TRANS, DIAG,	N,			A, LDA,	X, INCX)		
STBMV	(UPLO, TRANS, DIAG,	N,	K,		A, LDA,	X, INCX)		
STPMV	(UPLO, TRANS, DIAG,	N,			AP,	X, INCX)		
STRSV	(UPLO, TRANS, DIAG,	N,			A, LDA,	X, INCX)		
STBSV	(UPLO, TRANS, DIAG,	N,	K,		A, LDA,	X, INCX)		
STPSV	(UPLO, TRANS, DIAG,	N,			AP,	X, INCX)		

name	options	dim	scalar	x-vector	y-vector	matrix
SGER	(	M,N,	ALPHA,	X, INCX,	Y, INCY,	A, LDA)
SSYR	(UPLO,	N,	ALPHA,	X, INCX,		A, LDA)
SSPR	(UPLO,	N,	ALPHA,	X, INCX,		AP)
SSYR2	(UPLO,	N,	ALPHA,	X, INCX,	Y, INCY,	A, LDA)
SSPR2	(UPLO,	N,	ALPHA,	X, INCX,	Y, INCY,	AP)

TABLE 2.1. The calling sequences for the real Level 2 BLAS.

The arguments that specify the options have the following meaning:

name	value	meaning
UPLO	'U'	Upper triangle
	'L'	Lower triangle
TRANS	'N'	Operate with the matrix
	'T' or 'C'	Operate with the transpose of the matrix
DIAG	'U'	The matrix is unit triangular
	'N'	The matrix is non-unit triangular

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 α and β 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:

$$y \leftarrow \alpha A x + \beta y \quad \text{(TRANS='N'), or}$$

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

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 $y:=Ax$, $A=(a_{ij})$, $x=(x_i)$, $y=(y_j)$, $i=1,\cdots,m$; $j=1,\cdots,n$, 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:

$$y_i := \sum_{j=1}^{n} a_{ij}x_j = (a_{i.},x), \quad i=1,\cdots,m,$$

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

$$y := \sum_{j=1}^{n} x_j a_{.j},$$

i.e., a linear combination of the columns of A with coefficients $x_1,x_2,\cdots,x_n$. This can be done with the well-known linked-triad (or SAXPY) construction (vector 1:= 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 $(a_{1i}, a_{2i}, \cdots, a_{ii}, 0, \cdots, 0)^T$, $i = 1,2, \cdots, n$. Multiplication of the *lower* part of A by x and addition to y proceeds as follows:

$$y_i := y_i + \sum_{j=1}^{i-1} a_{ji} x_j, \quad i = 2, \cdots, n.$$

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 linked-triad 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:

$$\begin{array}{cccc} a_{11}, & a_{12}, a_{22}, & a_{13}, a_{23}, a_{33}, \quad \cdots, & a_{1n}, a_{2n}, \cdots, a_{nn}. \\ \text{col.1} & \text{col.2} & \text{col.3} & \text{col.n} \end{array}$$

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, $i = 1,...,n$, the subdiagonal by b_i, $i = 1, \cdots, n-1$ and the superdiagonal by c_i, $i = 2, \cdots, n$. Then the diagonal-based multiplication scheme for $y := Ax$ reads as follows:

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_{n-1} \\ y_n \end{bmatrix} := \underset{\substack{\uparrow \\ \text{subdiagonal}}}{\begin{bmatrix} 0 \\ b_1 \\ b_2 \\ \vdots \\ b_{n-2} \\ b_{n-1} \end{bmatrix}} * \begin{bmatrix} 0 \\ x_1 \\ x_2 \\ \vdots \\ x_{n-2} \\ x_{n-1} \end{bmatrix} + \underset{\substack{\uparrow \\ \text{main diagonal}}}{\begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_{n-1} \\ a_n \end{bmatrix}} * \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} + \underset{\substack{\uparrow \\ \text{superdiagonal.}}}{\begin{bmatrix} c_2 \\ c_3 \\ c_4 \\ \vdots \\ c_n \\ 0 \end{bmatrix}} * \begin{bmatrix} x_2 \\ x_3 \\ x_4 \\ \vdots \\ x_n \\ 0 \end{bmatrix}$$

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

$$y := \beta y + \alpha A x = \beta y + \alpha \sum_{j=1}^{n} x_j a_{.j}$$

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.

```
      REAL  A(1000)
      REAL  AD, BD
      DESCRIPTOR  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:

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

Changing the length to L ($\leqslant$ 65535) can be done as follows:

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

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

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

and the reverse operation, storing, by

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 (**M**illion **FLO**ating **P**oint 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:

Construct (v = vector, s = scalar)		Maximum MFLOP/s on a Cyber 205		
		1-pipe	2-pipe	4-pipe
$v \leftarrow v + s * v$	(linked triad)	100	200	400
$s \leftarrow (v, v)$	(dot product)	100	100	100
$v \leftarrow v + v * v$		50	100	200

Routine	Number of operations	Choice of arguments
SGEMV	2MN + M +N	$\alpha \neq 0$, $\beta \neq 0$
SGBMV	2N(KL+KU+2) -KL(KL+1)-KU(KU+1)	M=N , $\alpha \neq 0$, $\beta \neq 0$
SSYMV SSPMV	$2N^2+2N$	$\alpha \neq 0$, $\beta \neq 0$
SSBMV	4N(K+1)-2K(K+1)	$\alpha \neq 0$, $\beta \neq 0$
STRMV STPMV	N^2	DIAG = 'N'
STBMV	N(2K+1)-K(K+1)	DIAG = 'N'
SGER	2MN+*MIN*(M,N)	$\alpha \neq 0$
SSYR SSPR	N^2+2N	$\alpha \neq 0$
SSYR2 SSPR2	$2N^2+2N$	$\alpha \neq 0$
STRSV STPSV	$2N^2$	DIAG = 'N'
STBSV	N(2K+1)-K(K+1)	DIAG = 'N'

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).

Routine	INCX (=INCY)=1			INCX (=INCY)=−2		
	TIMES IN MSEC.		MFLOP/s	TIMES IN MSEC.		MFLOP/s
	I	II	of II*)	I	II	of II*)
SGEMV	7.1	6.6	76(124)	131	6.5	77(124)
SGBMV	133	16.0	26(38)	174	17.4	24(32)
SSYMV	8.1	8.1	62(74)	66	8.1	62(73)
SSPMV	66	8.3	60(71)	66	8.3	60(71)
SSBMV	190	11.6	31(48)	120	13.4	27(37)
STRMV	4.9	4.0	62(90)	66	4.0	62(90)
STPMV	66	4.2	60(85)	66	4.2	60(85)
STBMV	149	13.7	24(33)	160	14.6	23(29)
SGER	7.0	6.4	78(128)	131	6.4	78(127)
SSYR	4.6	3.9	64(93)	66	4.0	63(92)
SSPR	66	3.9	64(93)	66	4.0	63(93)
SSYR2	8.1	7.2	70(107)	81	7.2	70(106)
SSPR2	81	7.2	70(107)	81	7.2	70(106)
STRSV	66	4.3	58(82)	79	4.3	58(81)
STPSV	66	4.6	54(75)	79	4.6	54(74)
STBSV	184	109	3(3)	190	110	3(3)

I: Fortran 77 model implementation from [4], compiled with automatic vectorization option

II: Cyber 205 implementation (run on the 1-pipe Cyber 205 of SARA: The Academic Computing Centre Amsterdam)

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

TABLE 4.2. Selection of results of efficiency tests of Level 2 BLAS routines.

The results presented in Table 4.2 were obtained with the following values of the arguments of the routines:

UPLO = 'U' , TRANS = 'N' , 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

```
     K = KK
     DO 10, I=1,J
        AP(K) = AP(K) + X(I) * TEMP
        K = K + 1
  10 CONTINUE
```

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

	INCX=1		INCX=−2	
Vector Operation	1-pipe Cyber 205	2-pipe	1-pipe	2-pipe
gather x			5/4	5/4
gather main diagonal	5/4	5/4	5/4	5/4
multiply by x and store into h	1	1/2	1	1/2
gather other diagonal	5/4	5/4	5/4	5/4
multiply by x } K times	1	1/2	1	1/2
add result to h	1	1/2	1	1/2
store h into x	1	1/2		
scatter x			5/4	5/4
Total number of clock cycles per vector element	13(K+1)/4	9(K+1)/4	(19+13K)/4	(17+9K)/4
Corresponding theoretical CPU-times for K=5 and N=30,000	11.7 msec.	8.1 msec.	12.6 msec.	9.3 msec.
Measured CPU-times	13.7 msec.	10.0 msec.	14.6 msec.	11.5 msec.

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

```
   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.

		Speed of model implementation with automatic vectorization		Speed of Cyber 205 optimized implementation
Routine	TRANS	on CRAY-1S	on Cyber 205*)	*)
SGEMV	'N'	39	55(75)	62(90)
	'T'	31	55(56)	58(58)
SSYMV		31	45(51)	46(51)
STRMV	'N'	33	34(41)	43(56)
	'T'	20	6(6)	38(39)
SGER		39	56(78)	64(95)
SSYR		36	38(47)	47(61)
SSYR2		47	45(58)	54(73)

*) 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.

1. J.J. DONGARRA, J.R. BUNCH, C.B. MOLER, G.W. STEWART, (1979). *LINPACK Users' Guide*, SIAM Publications, Philadelphia.
2. J.J. DONGARRA, J. DU CROZ, S. HAMMARLING, R.J. HANSON, (Nov. 1986). *An Extended Set of Fortran Basic Linear Algebra Subprograms*, Tech. Memo. No. 41 (Revision 3), Math. and Comp. Science Division, Argonne National Laboratory, Argonne, Illinois, USA.
3. J.J. DONGARRA, J. DU CROZ, S. HAMMARLING, R.J. HANSON, (1986). *An Update Notice on the Extended BLAS*, SIGNUM, *21*, No. 4, pp. 2-4.
4. J.J. DONGARRA, J. DU CROZ, S. HAMMARLING, R.J. HANSON, (Jan. 1987). *An extended set of Basic Linear Algebra Subprograms: Model Implementation and Test Programs*, Tech. Memo. No. 81, Math. and Comp. Science Division, Argonne National Laboratory, Argonne, Illinois, USA.
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 athe address see the title page of this paper).
6. (April 1986). *Control Data FTN200 Reference Manual*, Publication Number 60480200, Revision G.