Algorithm 784: GEMM-Based Level 3 BLAS: Portability and Optimization Issues

BO KÅGSTRÖM and PER LING
Umeå University
and
CHARLES VAN LOAN
Cornell University

This companion article discusses portability and optimization issues of the GEMM-based level 3 BLAS model implementations and the performance evaluation benchmark. All software comes in all four data types (single- and double-precision, real and complex) and are designed to be easy to implement and use on different platforms. Each of the GEMM-based routines has a few machine-dependent parameters that specify internal block sizes, cache characteristics, and branch points for alternative code sections. These parameters provide means for adjustment to the characteristics of a memory hierarchy.

Categories and Subject Descriptors: D.3.2 [**Programming Languages**]: Language Classifications—*Fortran 77*; F.2.1 [**Analysis of Algorithms and Problem Complexity**]: Numerical Algorithms and Problems—*computation on matrices;* G.1.3 [**Numerical Analysis**]: Numerical Linear Algebra—*linear systems (direct and iterative methods);* G4 [**Mathematics of Computing**]: Mathematical Software—*benchmarking; certification and testing; efficiency; portability; reliability and robustness; verification*

General Terms: Algorithms, Benchmarking, Measurements, Performance

Additional Key Words and Phrases: Blocked algorithms, GEMM-based level 3 BLAS, matrix-matrix kernels, memory hierarchy, parallelization, vectorization

1. INTRODUCTION

In Kågström et al. [1998] our high-performance model implementations of the GEMM-based level 3 BLAS are described. Performance results for several different computer systems are also presented. Moreover, we de-

This work was partially supported by the Swedish Board of Industrial an Technical Development under Grant NUTEK 89-02578P.

Authors' addresses: B. Kågström and P. Ling, Department of Computing Science, Umeå University, S-901, Umeå, Sweden; email: {bokg; per.ling}@cs.umu.se; C. Van Loan, Department of Computer Science, Cornell University, Ithaca, NY 14853-7501; email: cv@cs.cornell.edu.

scribed the GEMM-based level 3 benchmark, its purpose, and design, and presented some benchmark results for different vendor-manufactured level 3 BLAS implementations.

The present contribution focuses on portability and optimization issues so that correct results are produced and high uniform performance is achieved on different target architectures. In Section 2, we discuss the GEMM-based model implementations, and Section 3 is devoted to the performance evaluation benchmark.

Four data types are supported with standard naming conventions. The software is available via the Collected Algorithms from ACM (and Netlib). The naming standard is that all routine names have a prefix-character (_), which is s for single-precision real data, d for double-precision real data, c for single-precision complex data, and z for double-precision complex data. This holds for all GEMM-based level 3 BLAS routines, auxiliary routines, and all routines related to the GEMM-based benchmark. In the following we make illustrations only for the double-precision real data case. The necessary changes for the other data types and precisions are obvious from the context.

2. THE GEMM-BASED MODEL IMPLEMENTATIONS

The model implementations are primarily intended for single-processor use, on machines with local or global caches, and for microprocessors with on-chip caches. They can also be parallelized using a parallelizing compiler, or linked with underlying parallel BLAS routines. Most of the machine characteristics of a target architecture are hidden and utilized in the underlying BLAS routines (_GEMM and some level 1 and level 2 BLAS routines). However, we have found it necessary to provide means for specifying some characteristics in terms of machine-specific parameters which are used in two auxiliary routines (only cache parameters and branch points for alternative code sections) and in the GEMM-based routines (only blocking parameters).

2.1 Auxiliary Routines

The original Fortran 77 model implementations of the level 3 BLAS [Dongarra et al. 1990a; 1990b] include two auxiliary subprograms, LSAME and XERBLA. The GEMM-based level 3 BLAS model implementations have two additional auxiliary subprograms, _BIGP and _CLD.

The logical function _BIGP identifies the branch point for two alternative code sections. Parameters _IPx, where x is a number identifying a specific branch point in a particular GEMM-based level 3 BLAS routine, are specified in _BIGP. These are used to determine which of two alternative code sections that will be the fastest, depending on the problem size. In the present implementation _BIGP looks only at one of two input dimensions that specify the problem size.

The logical function _CLD is used to set up the cache policy. _CLD tests the size of a leading dimension of a two-dimensional array to see if it will

Table I. Characteristics of the Model Implementations

	_SYMM	_SYRK	_SYR2K	_TRMM	_TRSM
Auxiliary routines called	LSAME XERBLA	LSAME XERBLA _BIGP _CLD	LSAME XERBLA	LSAME XERBLA _BIGP _CLD	LSAME XERBLA _BIGP _CLD
Intrinsic functions called	MAX MIN	MAX MIN	MAX MIN	MAX MIN MOD	MAX MIN MOD

lead to excessive data movements in the memory hierarchy. For example, excessive cache traffic may result in severe performance degradation. The implementation of _CLD is designed for a multiway associative cache, and how this is done is explained in the next section.

The cache lines in a multiway associative cache are divided among a number of sets, each containing the same number of lines. The number of lines in a set equals the associativity of the cache. For example, in a four-way associative cache, each set contains four cache lines, and up to four data lines (with the same cache address determined by a mapping function) can be stored simultaneously in the cache. When data that is not in cache are requested, its data line replaces one of the four lines currently stored (at the address assigned by the mapping function). The least-recently-used (LRU) line is a common cache replacement policy [Bailey 1995]. Other policies are based on some random choice for the line to be replaced.

It may be rewarding to modify _CLD for a machine with a different cache policy, and _BIGP to involve both of the problem dimensions.

Auxiliary routines called in the GEMM-based model implementations are displayed in Table I. Moreover, the table shows the intrinsic functions used.

2.2 Machine-Specific Parameters

Each of the GEMM-based routines has system-dependent parameters that specify internal block sizes, cache characteristics, and branch points for alternative code sections. These values are given in PARAMETER statements, by the user, or the system manager. A simple program _SGPM that facilitates tuning of the parameters is included in the GEMM-based package (see Section 2.3).

The internal blocking parameters r, c, and rc used in the description of the model implementations (see Kågström et al. [1998, Section 5]) have the names RB, CB, and RCB, respectively, in the Fortran 77 routines. Local arrays of size RCB · RCB double-precision words are allocated in each of the GEMM-based routines to hold general, symmetric, or triangular matrix blocks temporarily. Moreover, a local array of size RB × CB is allocated in each of _SYRK, _TRMM, and _TRSM, and a local array of size CB × CB in _TRMM and _TRSM. Note that each routine has its own independent blocking values.

_CLD has parameters specifying characteristics of the cache memory to determine which sizes of the leading dimension of a two-dimensional array that should be considered critical:

LNSZ	Size of a cache line in number of bytes.
NPRT	Number of sets in the cache memory.
PRTSZ	The largest number of cache lines in each set, which can be used exclusively to hold array data during the execution of a GEMM-based level 3 routine. The remaining cache lines are occupied by scalars, vectors, and possibly program code, depending on the system characteristics.
LOLIM	Leading dimensions smaller than or equal to LOLIM are not regarded as critical.
_P	Size of the current precision word in number of bytes.

2.3 Assigning Values to the Machine-Specific Parameters

The machine-specific parameters for the GEMM-based level 3 BLAS provide means for adjustment to some of the characteristics of a memory hierarchy. Considering the diversity of memory systems, the following guidelines for assigning values to the machine-specific parameters must be viewed as heuristics rather than rules:

- The block dimensions RCB, RB, and CB, should all be multiples of the number of double-precision words that fit in a cache line (LNSZ/_P).
- For _SYRK, _TRMM, and _TRSM:
  - If the machine has vector registers, then RB should equal the number of double-precision words that fit in a vector register.
  - A block of size RB $\times$ CB double-precision words should safely fit in the cache memory, possibly together with scalars, two column vectors, of size RB and CB, and program code. Typically, RB $\cdot$ CB double-precision words correspond to 50–75% of the size of a cache.
  - A block of size RCB $\times$ RCB double-precision words should safely fit in cache and occupy, for instance, 50–75% of a cache.

  Notice that only one of the local blocks resides in cache at a given time.

- For _SYMM and _SYR2K:
  - A local array of size RCB $\times$ RCB is allocated. The array does not need to fit in cache and should be fairly large but still reasonable in size. If vector registers are present, a good choice for RCB is the number of words that fit in a vector register, or possibly a multiple of that number.
  - In some cases, rows of length CB are referenced. CB cache lines should safely fit in the cache in order to become reused efficiently. RCB is an upper limit for CB in this case. If CB $>$ RCB, then RCB is used instead of CB.

—The branch points _IPx used in _SYRK, _TRMM, and _TRSM, are assigned values in the routine _BIGP. If the problem size is larger than or equal to _IPx, then _GEMV is invoked. The values for _IPx may differ between the alternative code sections depending on the values of RCB, RB, and CB. Values returned from _BIGP should effectively reflect the performance characteristics of the underlying level 2 BLAS routines. If _GEMV is carefully optimized, a proper value for _IPx may be found in the range 0–10 (0 corresponds to having all calls to _GEMV). Timing experiments with small matrix dimensions are recommended.

—If the cache is shared for program instructions and data, a suitable value for PRTSZ is the total number of cache lines in a set minus one, or two, leaving one or two cache lines in each set for vectors, scalars, and program code.

—LOLIM is a lower limit for the size of the leading dimension that may be considered critical. We use LOLIM = max(RCB, RB), where RCB and RB are the block dimensions given for _SYRK, _TRMM, and _TRSM.

The parameter values for internal block sizes and branch points are adjusted separately for each routine. The software comes with a default input file that provides parameter settings for IBM RS/6000 530H (see Figure 1 and Section 2.4, where parameter settings are discussed for different architectures). The sensitivity of performance to the choice of parameter values is dependent on the characteristics of the target architecture. We have seen that for many RISC-based architectures, the default parameter settings often reach more than 90% of the best performance achievable. However, there is no guarantee that the default settings always will work that well. So, it can be rewarding to experiment with a range of different values for the parameters to improve on the performance of the routines.

If you are unable to make the function _CLD safely predict which leading dimensions that will cause substantial performance degradation, you may consider modifying _CLD so that the value .TRUE. is always returned for values greater than LOLIM. In this case, all leading dimensions greater than LOLIM are regarded as critical. The risk of severe performance degradation for the GEMM-based routines becomes significantly reduced, at the small expense of copying matrix blocks to local arrays a few more times.

The GEMM-based level 3 BLAS benchmark can be used to fine-tune the machine-specific parameters for high performance. Another useful timing program is distributed with the original level 3 BLAS [Dongarra et al. 1990a; 1990b]. For example, the double-precision real version can be obtained by sending the email message "send dblas3time from blas" to netlib@ornl.gov.

2.4 Sample Values for the Machine-Specific Parameters

The values for the machine-specific parameters presented here are the best values we have found experimentally. But there is no guarantee that they are optimal, and they should merely be viewed as starting-values for

```
dsymm.f
      PARAMETER        ( RCB = 128, CB = 64 )
dsyr2k.f
      PARAMETER        ( RCB = 128, CB = 64 )
dsyrk.f
      PARAMETER        ( RCB = 64, RB = 64, CB = 64 )
dtrmm.f
      PARAMETER        ( RCB = 64, RB = 64, CB = 64 )
dtrsm.f
      PARAMETER        ( RCB = 64, RB = 64, CB = 64 )
dbigp.f
      PARAMETER        ( DIP41 = 4, DIP42 = 3,
     $                   DIP81 = 4, DIP82 = 3, DIP83 = 4,
     $                   DIP91 = 4, DIP92 = 3, DIP93 = 4 )
dcld.f
      PARAMETER        ( LNSZ = 128, NPRT = 128, PRTSZ = 3,
     $                   LOLIM = 128, DP = 8 )
```

Fig. 1. Input file for IBM RS/6000 530H.

further refinement. They have been used during the development of the GEMM-based level 3 BLAS model implementations and for benchmarking on several different computer systems.

The program _SGPM modifies the GEMM-based level 3 BLAS source files replacing lines containing old PARAMETER statements for machine-specific parameters, with lines containing new PARAMETER statements given in the input file. The user (or system manager) can conveniently assign new values to the PARAMETER statements in the input file, and then run _SGPM to distribute the values among the GEMM-based routines. The files dsgpm.f and dgpm.in contain the program DSGPM and an example input file, respectively. An input file to _SGPM consists of three different types of lines, apart from empty lines.

—Comment lines starting with an asterisk (*).

—Lines containing single filenames for GEMM-based source files.

—Lines containing PARAMETER statements that replace the corresponding lines in the GEMM-based routines.

A line containing a file name is followed by lines containing the new PARAMETER statements for the particular file.

Input file for IBM RS/6000 530H is displayed in Figure 1. The following machine characteristics are used to determine values for the parameters. This machine has separate caches for data and instructions, where the size of the data cache is 64KB with 128-bytes cache lines. The cache scheme is four-way associative (mapping), and the size of a double word is eight bytes.

In Tables II–IV we show sample values of the machine-specific parameters for some different architectures. Notably, the best values on the branch points in Table III are all the same for the architectures we have

Table II. Sample Data Values for Internal Blocking Parameters

Computer system:		DSYMM	DSYRK	DSYR2K	DTRMM	DTRSM
Alliant FX/2816	RB	-	32	-	32	32
	CB	32	32	32	32	32
	RCB	128	32	128	32	32
IBM 3090J-VF	RB	-	256	-	256	256
	CB	96	96	96	96	96
	RCB	256	144	256	144	144
IBM RS6000 250	RB	-	56	-	56	56
	CB	56	56	56	56	56
	RCB	128	56	128	56	56
IBM RS6000 530H	RB	-	64	-	64	64
	CB	64	64	64	64	64
	RCB	128	64	128	64	64
IBM SP2 thin	RB	-	96	-	96	96
	CB	96	96	96	96	96
	RCB	256	96	256	96	96
IBM SP2 wide	RB	-	144	-	144	144
	CB	144	144	144	144	144
	RCB	256	144	256	144	144
Intel Paragon	RB	-	40	-	40	40
	CB	40	40	40	40	40
	RCB	128	40	128	40	40
Parsytec(80MHz)	RB	-	48	-	48	48
	CB	48	48	48	48	48
	RCB	144	48	144	48	48
SGI Indy R4000	RB	-	32	-	32	32
	CB	24	24	24	24	24
	RCB	128	24	128	24	24
SGI Indy R4400	RB	-	48	-	48	48
	CB	32	32	32	32	32
	RCB	128	32	128	32	32

Table III. Sample Values for Branch Points Used in DBIGP

Computer system:	DIP41	DIP42	DIP81	DIP82	DIP83	DIP91	DIP92	DIP93
IBM SP2 thin	4	3	4	3	4	4	3	4

considered, except for Alliant FX/2800 for which we used all values equal to 3. Table IV also displays some cache memory characteristics, namely the associativity of the cache, cache size (in kilobytes), and cache line size (in bytes). Notice that the SGI machines have a direct-mapped cache and that some of the machines have a separate data cache (D in the table), while others have a common instruction and data cache (C in the table).

2.5 Verification of the Correctness of the Installed Programs

This section gives a brief description on how to install the GEMM-based level 3 BLAS model implementations on a Unix-based system, and how to

Table IV. Cache Characteristics and Sample Values for Parameters Used in DCLD

Computer system:	Associ-ativity	Cache size Kbytes	Line size Bytes	LNSZ	NPRT	PRTSZ	LOLIM	DP
Alliant FX/2816	2-way	8 (D)	32	128	128	2	32	8
IBM 3090J-VF	4-way	256 (C)	128	128	512	3	128	8
IBM RS6000 250	8-way	32 (C)	64	64	64	6	64	8
IBM RS6000 530H	4-way	64 (D)	128	128	128	3	128	8
IBM SP2 thin	4-way	128 (D)	128	128	256	3	128	8
IBM SP2 wide	4-way	256 (D)	256	256	256	3	256	8
Intel Paragon-XP	2-way	16 (D)	32	32	512	2	40	8
Parsytec(80MHz)	8-way	32 (C)	64	64	64	6	64	8
SGI Indy R4000	direct	8 (D)	32(64)	32	128	2	32	8
SGI Indy R4400	direct	16 (D)	32(64)	32	256	2	32	8

check the correctness of these routines. A makefile is included in the GEMM-based package to facilitate the installation.

The machine-specific parameters come with default values. These values need to be optimized for different target machines according to the guidelines in Section 2.3.

The program _SGPM together with an input file can be used to assign values to the machine-specific parameters. Compile and link the program _SGPM:

```
% make dsgpm
```

Create a copy, newdgpm.in, of the enclosed input file dgpm.in. Assign new values to the machine-specific parameters in newdgpm.in (see the guidelines in Section 2.3 and the examples in Section 2.4). Run the program which rewrites the GEMM-based routines with the new parameter values given in newdgpm.in:

```
% dsgpm < newdgpm.in
```

Create a GEMM-based level 3 BLAS library:

```
% make
```

If everything worked out well, a library named libgbl3b.a has been created in the top-level directory of the directory hierarchy. Notice, you may also specify compiler flags in makefile.

Be sure to verify the correctness of the compiled routines thoroughly, before production use. Do not trust the underlying BLAS, or the compiler used, especially if compiler options for code optimization, inlining, etc., were used. We recommend the test program DBLAT3 for verification of double-precision level 3 BLAS [Dongarra et al. 1990a; 1990b]. (These testing programs are included in the package.) The test should be specified to ensure that all calls to underlying BLAS routines that can be invoked due to values of the user-specified parameters in fact are invoked. If the default values of the user-specified parameters are used, then use the enclosed input file for DBLAT3. Otherwise, specify the largest matrix dimensions of the tests to be larger than the parameter RCB of _SYMM, _HEMM, _SYR2K, and _HER2K, and at least twice as large as the largest of all remaining block parameters (in all routines).

3. PERFORMANCE EVALUATION BENCHMARK

We describe the input file for the performance evaluation benchmark and the benchmark results produced, and finally we briefly discuss the software and benchmarking.

3.1 The Input File

The user supplies an input file for the benchmark specifying tests to be made and results to be presented. The following parameters need to be specified in the input file:

LBL An arbitrary label which identifies the test to be performed (max 50 characters). The label is printed together with the output results.

TAB One or more numbers specifying tests to be made and results to be presented.

RUNS All results presented are based on the fastest of RUNS executions for each problem configuration.

At least one of the numbers 1–6 need to be specified for the parameter TAB. The numbers are interpreted as follows:

(1) The collected benchmark result.
(2) Performance of the built-in GEMM-based level 3 BLAS library in Mflops.
(3) Performance of the user-specified level 3 BLAS library in Mflops.
(4) Performance of the user-specified _GEMM routine in Mflops. Problem configurations for _GEMM are chosen to “correspond” to those in 2 and 3 for timing purposes (see Section 3.2).
(5) GEMM-efficiency of the user-specified level 3 routines.
(6) GEMM-ratio.

Both GEMM-efficiency and GEMM-ratio are defined in Kågström et al. [1998] (Section 7).

The input parameters for the level 3 BLAS routines are specified as follows:

side Characters. L(eft) and/or R(ight).

uplo. Characters. U(pper) and/or L(ower) triangular part.

trans Characters. N(o transpose) and/or T(ranspose).

diag Characters. N(o unit) and/or U(nit) triangular.

dim1 Integer values for the first of the two dimensions.

dim2 Integer values for the second of the two dimensions.

lda Integer values for leading dimension of the matrices.

```
LBL    Example 1, double precision.
***  Benchmark results to be presented  ***
TAB    1 2 3 4 5 6
***  RUNS executions of each problem configuration  ***
RUNS   2
***  Values of input parameters for the level 3 BLAS routines  ***
SIDE   L R
UPLO   U L
TRANS  N T
DIAG   N
DIM1    32  64 256 256
DIM2   256 256  32  64
LDA    256
***  Routines to be timed  ***
DSYMM  T
DSYRK  T
DSYR2K T
DTRMM  T
DTRSM  T
```

Fig. 2. Sample input file for GEMM-based benchmark.

See Dongarra et al. [1990a; 1990b] for further explanations of the input parameters side, uplo, trans, and diag. The parameters dim1 and dim2 are used to specify the first and second dimensions in the calling sequence of the level 3 BLAS routines, respectively. The values for dim1 and dim2 come in pairs. lda (= ldb = ldc) specifies the leading dimension of the matrices A, B, and C in calls to the level 3 BLAS routines.

Specify for each routine whether it should be timed or not. Put T after the routine name if the routine should be timed, otherwise F. An example of an input file is given in the file example.in, which can be used as a template for user-constructed tests (see Figure 2).

3.2 Benchmark Results

The output from the benchmark optionally includes a "collected mean value" statistic of the user-specified level 3 routines, and tables showing detailed performance results and comparisons between the user-specified and the built-in GEMM-based level 3 BLAS routines (see Section 7 in Kågström et al. [1998]). Problem configurations, routines to be timed, and results to be presented are selected according to specifications in the input file.

3.2.1 *The Table Results.* The performance of the level 3 routines to be benchmarked is compared with the performance of _GEMM with the input parameters given in Table V. We use alpha = 0.9, beta = 1.1, and lda = ldb = ldc. Notice the parameters of _GEMM that are not displayed are equal to the parameters of the GEMM-based routine it is compared with.

The number of floating-point operations (flops) performed by a level 3 BLAS routine is divided by the execution time in seconds, times 10^6, to

Table V. Input Parameters for _GEMM

GEMM-based routines			Input parameters for _GEMM								
routine	side	trans	transa	transb	m	n	k	A	B	C	beta
_SYMM	'L'		'N'	'N'	m	n	m	A	B	C	1.1
	'R'		'N'	'N'	m	n	n	B	A	C	1.1
_SYRK		'N'	'N'	'T'	n	n	k	A	A	C	1.1
		'T'	'T'	'N'	n	n	k	A	A	C	1.1
_SYR2K		'N'	'N'	'T'	n	n	k	A	B	C	1.1
		'T'	'T'	'N'	n	n	k	A	B	C	1.1
_TRMM,	'L'		trans	'N'	m	n	m	A	B	C	1.0
_TRSM	'R'		'N'	trans	m	n	n	B	A	C	1.0

obtain the performance in Mflops. The number of flops of the level 3 BLAS operations (nops) and of the corresponding _GEMM operation (gops) is displayed in Table VI.

3.2.2 *The Collected Benchmark Result.* The purpose of the collected benchmark result is to expose the capacity of the target machine for level 3 kernels and to show how well the routines utilize the machine. Furthermore, the collected result is intended to be easy to compare between different computer systems.

We propose two standard test suites for the collected benchmark result, _MARK01 and _MARK02 (see the files dmark01.in and dmark02.in). These tests are designed to show performance of the user-specified level 3 library for problem sizes that often are likely to be requested by a calling routine. For example, LAPACK implements blocked algorithms which are based on calls (with varying problem configurations, e.g., size and operation) to the level 3 BLAS [Anderson et al. 1992].

The problems in the two tests are similar. However, some of the matrix dimensions are larger in _MARK02 than in _MARK01. This corresponds to larger matrix blocks in the calling routine. The tests are expected to match various target machines differently. Since performance results depend strongly on sizes of different storage units in the memory hierarchy, we propose two standard tests instead of one.

3.3 The Built-in GEMM-Based Level 3 BLAS

The GEMM-based level 3 BLAS model implementations are included in the benchmark and are used in the evaluation of another set of implementations. In order to keep the benchmark general, and make it possible to test variants of our model implementations, we have renamed the level 3 routines in the GEMM-based benchmark.

The new names are _GB02 (for _SYMM), _GB03 (for _HEMM), _GB04 (for _SYRK), _GB05 (for _HERK), _GB06 (for _SYR2K), _GB07 (for _HER2K), _GB08 (for _TRMM), _GB09 (for _TRSM), _GB90 (for _BIGP), and finally _GB91 (for _CLD). The reason for renaming the two new auxiliary routines is that it makes it possible to compare the same

Table VI. Number of Flops for Level 3 BLAS

GEMM-based routines					
			nops	:	number of flops for a level 3 BLAS problem
routine	side	diag	gops	:	number of flops for the corresponding _GEMM problem
_SYMM	'L'		nops	=	$(2m+1)mn + \min(mn, m(m+1)/2)$
			gops	=	$(2m+1)mn + \min(mn, mm)$
	'R'		nops	=	$(2n+1)mn + min(mn, n(n+1)/2)$
			gops	=	$(2n+1)mn + min(mn, nn)$
_SYRK			nops	=	$(2k+1)(n(n+1)/2) + min(nk, n(n+1)/2)$
			gops	=	$(2k+1)nn + \min(nk, nn)$
_SYR2K			nops	=	$(4k+1)(n(n+1)/2) + \min(2nk, n(n+1))$
			gops	=	$(2k+1)nn + \min(nk, nn)$
_TRMM,	'L'	'N'	nops	=	$mmn + \min(mn, m(m+1)/2)$
_TRSM	'L'	'U'	nops	=	$mmn - mn + \min(mn, m(m+1)/2)$
	'L'		gops	=	$(2m-1)mn + \min(mn, mm)$
	'R'	'N'	nops	=	$mnn + \min(mn, n(n+1)/2)$
	'R'	'U'	nops	=	$mnn - mn + \min(mn, n(n+1)/2)$
	'R'		gops	=	$(2n-1)mn + \min(mn, nn)$

GEMM-based model implementations, but with two different sets of auxiliary routines.

3.4 The Benchmark Software

All routines are written in Fortran 77 for portability. No changes to the code should be necessary in order to run the programs correctly on different target machines. Indeed, we strongly recommend the user to avoid changes, except for the machine-specific parameters and for unit numbers for input and output communication. This will ensure that performance results from different target machines are comparable. unit numbers are set in the main program _GBTIM and the machine-specific parameters exist only in the built-in GEMM-based level 3 BLAS routines.

The benchmark program consists of the following routines apart from the built-in GEMM-based level 3 BLAS routines:

- _GBTIM is the main program which reads the input file and calls the routines described below.
- _GBT01 times the user-specified _GEMM routine.
- _GBT02 times the built-in GEMM-based level 3 BLAS routines and the user-specified level 3 BLAS routines except _GEMM.
- _GBTP1 calculates and prints the collected benchmark result.
- _GBTP2 calculates and prints the table results.

The following is a description of how to install the GEMM-based level 3 BLAS benchmark on machines with Unix-based operating systems. A makefile is enclosed to facilitate the installation. The user-specified parameters of the built-in GEMM-based level 3 BLAS routines come with default values, which might need to be optimized for the target machine (see Sections 2.3 and 2.4).

The program _SBPM assigns values to the machine-specific parameters, and corresponds to the program _SGPM for the GEMM-based level 3 BLAS model implementations. Input files for _SGPM may also be used with _SBPM. To compile and link _SBPM give the command

```
% make dsbpm
```

Run _SBPM which updates the built-in GEMM-based level 3 BLAS routines with the new parameters given in the input file, newdgpm.in:

```
% dsbpm < newdgpm.in
```

The benchmark program calls a function SECOND (or DSECND in double precision) with no arguments. This function is assumed to return the CPU time in seconds from some fixed starting time. Create this function if it does not already exist on your system. The enclosed Fortran 77 function in the file dsecnd.f can be used as a template. This routine is based on calls to the timing function etime under Unix.

Specify the level 3 BLAS library to be evaluated and compiler flags in makefile. You may change the unit numbers for I/O communication nin, nout, and nerr in the main program _GBTIM, if necessary. Create the executable benchmark program by giving the command

```
% make
```

If everything worked out well, you will now have a useful performance evaluation tool for level 3 BLAS kernels.

3.5 Benchmarking

The GEMM-based level 3 BLAS benchmark can be used in different ways to evaluate performance of level 3 BLAS routines. The user controls which tests to be made and which results to be presented through specifications in the input file.

The following Unix command runs the benchmark program with the input file example.in and writes the result to the output file example.out:

```
% dgbtim < example.in > example.out
```

Notice that this benchmark may be quite time consuming to run. Obviously, the “size” of the test, specified in the input file, is decisive for the execution time. Moreover, the performance of the target machine and of the different level 3 BLAS libraries also affect the total execution time.

3.6 Collecting Benchmark Results

We encourage users to help us collect performance results from different target machines. Please, send results obtained with the proposed standard tests _MARK01 and _MARK02 to the second author at email address per.ling@cs.umu.se. Contributors that provide interesting results from the

GEMM-based level 3 BLAS benchmark will be acknowledged in a future collection of benchmark results. We also encourage users to send comments on the model implementations and benchmark to any of the authors.

In order to be able to interpret the results we also need to have as much as possible of the following system characteristics specified:

—Machine: name and version, number of processors, sizes of cache(s), and main memory, etc.

—Operating system: name, version, and release.

—Fortran compiler: name, version, release, and options used.

—User-specified underlying BLAS: name of library, version, and release.

—Machine configuration used in the benchmark.

—Precision tested (S, D, Z, or C), double precision: x-bit words.

—Timing function: describe the implementation of SECOND or DSECND. Specify which local timing function it is based on (e.g., etime, dclock, mclock), and which time it measures (e.g., real time, CPU-time, user-time) and to which resolution.

If the GEMM-based level 3 BLAS model implementations are the user-specified routines in the tests, please enclose values for the machine-specific parameters and describe the underlying BLAS implementations (_GEMM, level 1 and level 2 BLAS). If more than one processor are used, please explain how the parallelism is invoked. For example, whether the GEMM-based routines are automatically parallelized by the compiler and/or you are using parallel versions of the underlying BLAS routines.

REFERENCES

ANDERSON, E., BAI, Z., BISCHOF, C., DEMMEL, J., DONGARRA, J., DUCROZ, J., GREENBAUM, A., HAMMARLING, S., MCKENNY, A., OSTROUCHOV, S., AND SORENSEN, D. 1992 *LAPACK Users Guide.* SIAM Publications.

BAILEY, D. 1995. Unfavorable strides in cache memory systems. *Sci. Program. 4,* 53–58.

DONGARRA, J., DUCROZ, J., DUFF, I., AND HAMMARLING, S. 1990a. A set of level 3 Basic Linear Algebra Subprograms. *ACM Trans. Math. Softw. 16,* 1 (Mar.), 1–17.

DONGARRA, J., DUCROZ, J., DUFF, I., AND HAMMARLING, S. 1990b. Algorithm 679: A set of level 3 Basic Linear Algebra Subprograms: Model implementation and test programs. *ACM Trans. Math. Softw. 16,* 1 (Mar.), 18–28.

KÅGSTRÖM, B., LING, P., AND VAN LOAN, C. 1998. GEMM-based level 3 BLAS: High-performance model implementations and performance evaluation benchmark. *ACM Trans. Math. Softw.* This issue.

Received November 1995; revised March 1996, June 1996, and August 1997; accepted August 1997