Software Development Kit for Multicore Acceleration
Version 3.1

LAPACK - Linear Algebra Package Library Programmer's Guide and API Reference

SC33-8428-02

Software Development Kit for Multicore Acceleration
Version 3.1

LAPACK - Linear Algebra Package Library Programmer's Guide and API Reference

SC33-8428-02

Note
Before using this information and the product it supports, read the information in "Notices" on page 33.

Edition notice

This edition applies to version 3, release 1, modification 0 of the IBM Software Development Kit for Multicore Acceleration (Product number 5724-S84) and to all subsequent releases and modifications until otherwise indicated in new editions.

This edition replaces SC33-8428-01.

Information for reviewers: draft changes

This topic lists the changes in this revision of the document.

Note: This is a draft version. As such, it contains incomplete information and possible inaccuracies. It is provided for informational or review purposes only.

Note: This section is included in the draft copy only. It will not be published with the book.

Selected information that is new or changed in this edition since version 3.0 of the SDK are marked with the revision tag | (pipe symbol.) Online editions might use a contrasting color to mark revisions and insertions.

Here is a summary of changes in this draft release of this book:

- Update to 3.1 versioning
- Large set of content updates from Hong Bo Peng
- Added a ″Concepts″ page
- Restructure and title changes
- Updates from Mike Perks
- New glossary

Contents

About this publication

This publication describes how to configure the IBM® Linear Algebra Package (LAPACK) library and how to program applications that use it on the IBM Software Development Kit for Multicore Acceleration (SDK). It contains reference information about APIs for the library and sample applications showing usage of these APIs.

Audience

The target audience for this document is application programmers using the SDK. You should have a basic understanding of programming on the Cell Broadband Engine™ Architecture (Cell/B.E.) and common terminology associated with the Cell/B.E. platform.

Related documentation

Open source LAPACK documentation is available from the *netlib* Web site at: http://netlib.org/lapack/.

For a list of SDK documentation, see "Related documentation" on page 37.

How to send your comments

Your feedback is important in helping to provide the most accurate and highest quality information. If you have any comments about this publication, send your comments using Resource Link™ at http://www.ibm.com/servers/resourcelink. Click **Feedback** on the navigation pane. Be sure to include the name of the book, the form number of the book, and the specific location of the text you are commenting on (for example, a page number or table number).

What's new

What's new in this publication.

All LAPACK routines are available for use in the SDK. These routines include real single precision, real double precision, complex single precision, and complex double precision.

The following 5 routines have been optimized to use features of the Cell/B.E. Synergistic Processing Elements (SPEs):

- **DGETRS** - Linear equation solver.
- **DPOTRS** - Linear equation (symmetric positive definite) solver
- **DGELQF** - Computes the LQ factorization of a general matrix.
- **DGESVD** - Computes the singular value decomposition (SVD) of a real M-by-N matrix A using implicit zero-shift QR algorithm, optionally computing the left and/or right singular vectors.
- **DGESDD** - Computes the singular value decomposition (SVD) of a real M-by-N matrix A using a divide-and-conquer algorithm. It optionally computes the left and/or right singular vectors.

Concepts

These are the main concepts and terms used when describing LAPACK.

DMA Direct Memory Access. This is a technique for using a special purpose controller to generate the source and destination addresses for a memory or I/O transfer.

PPE IBM PowerPC® Processor Element. The general purpose processor in the Cell/B.E..

PPU IBM PowerPC Processor Unit. The part of the PPE that executes instructions from its main memory.

SPE Synergistic processor element. These extend the PowerPC 64 architecture by acting as cooperative offload processors (synergistic processors), with direct memory access (DMA) and synchronization mechanisms to communicate with them (memory flow control), and with enhancements for real time management. There are eight SPEs on each Cell/B.E. processor die.

SPU Synergistic processor unit. This is the part of an SPE that executes instructions from its local store (LS).

Part 1. Overview of LAPACK

The LAPACK (Linear Algebra Package) library is based upon a published standard interface for commonly used linear algebra operations in high performance computing (HPC) and other scientific domains. It provides routines for solving systems of simultaneous linear equations, least-squares solutions of linear systems of equations, eigenvalue problems, and singular value problems. The associated matrix factorizations LU decomposition (lower triangular matrix L and upper triangular matrix U), Cholesky decomposition, QR decomposition (orthogonal matrix Q and upper triangular matrix R), and SVD (Singular value decomposition) are provided. Dense and banded matrices are handled, but not general sparse matrices.

The LAPACK API is available with standard ANSI C and standard FORTRAN 77 interfaces. LAPACK implementations are also available as open source from http://netlib.org.

Each LAPACK routine has up to four versions, as detailed in the following table:

Table 1. LAPACK routine precision

Precision	Routine name prefix
Real single precision	S
real double precision	D
complex single precision	C
complex double precision	Z

The LAPACK library in the IBM Software Development Kit for Multicore Acceleration (SDK) supports all C/S/D/Z LAPACK routines. All routines are available as PPE APIs and conform to the standard LAPACK FORTRAN 77 interface.

The following routines have been optimized to use features of the Synergistic Processing Elements (SPEs):

- DGETRF - Computes the LU factorization of a general matrix
- DGETRS – Linear equation solver
- DGETRI - Computes the inverse of a general matrix using an LU factorization
- DGEQRF - Computes the QR factorization of a general matrix
- DGELQF - Compute the LQ factorization of a general matrix
- DPOTRF - Computes the Cholesky factorization of a symmetric positive matrix
- DPOTRS – Linear equation (symmetric positive definite) solver
- DBDSQR - Computes the singular value decomposition of a real bi-diagonal matrix using an implicit zero-shift QR algorithm
- DSTEQR - Computes the singular value decomposition of a real symmetric tridiagonal matrix using an implicit QR algorithm
- DGESVD - Computes the singular value decomposition (SVD) of a real matrix using an implicit zero-shift QR algorithm, optionally computing the left and/or right singular vectors v

- DGESDD - Computes the singular value decomposition (SVD) of a real matrix using a divide-and-conquer algorithm, optionally computing the left and/or right singular vectors.

Chapter 1. Conventions

The following table explains the typographical conventions used in this documentation.

Table 2. Typographical conventions

Typeface	Indicates	Example
Bold	Used for specific emphasis with lowercase commands, executables, compiler options and directives.	If you specify **-O3**, the compiler assumes `-qhot=level=0`. To prevent all HOT optimizations with -O3, specify **-qnohot**.
Italics	Parameters or variables whose actual names or values are to be supplied by the user. Italics are also used to introduce new terms.	Make sure that you update the *size* parameter if you return more than the *size* requested.
`monospace`	Programming keywords and library functions, compiler built-in functions, examples of program code, command strings, or user-defined names.	If one or two cases of a `switch` statement are typically executed much more frequently than other cases, break out those cases by handling them separately before the `switch` statement.

Part 2. Installing and configuring LAPACK

Installation of the LAPACK library occurs during installation of the Software Development Kit for Multicore Acceleration. No additional configuration for LAPACK is necessary after installing the SDK.

For details on installing the SDK, see the "Installing the SDK" section of the *Software Development Kit for Multicore Acceleration Installation Guide* available at the Cell Broadband Engine Architecture Resource Center developerWorks® Web site: http://www-128.ibm.com/developerworks/power/cell.

Part 3. Programming with LAPACK

The following topics describe how to program applications to use the LAPACK library.

Chapter 2. Basic structure of the LAPACK library

The following tables show the location of the files installed with the LAPACK package:

Table 3. LAPACK library contents (X86)

Platform	X86 or X86_64 (Development)
PPE 32–bit library	/opt/cell/sysroot/usr/lib/liblapack.a /opt/cell/sysroot/usr/lib/liblapack.so /opt/cell/sysroot/usr/lib/liblapack.so.3.1.1
PPE 64–bit library	/opt/cell/sysroot/usr/lib64/liblapack.a /opt/cell/sysroot/usr/lib64/liblapack.so /opt/cell/sysroot/usr/lib64/liblapack.so.3.1.1
PPE header files	/opt/cell/sysroot/usr/include/lapack.h /opt/cell/sysroot/usr/include/lapack_errno.h
Example Code	/opt/cell/sdk/src/lapack-examples-source.tar

Table 4. LAPACK library contents (Cell/B.E. and PowerXCell 8i processors)

Platform	Cell/B.E. or Power Host (Development or execution including Full-System Simulator)
PPE 32–bit library	/usr/lib/liblapack.a /usr/lib/liblapack.so /usr/lib/liblapack.so.3.1.1
PPE 64–bit library	/usr/lib64/liblapack.a /usr/lib64/liblapack.so /usr/lib64/liblapack.so.3.1.1
PPE header files	/usr/include/lapack.h /usr/include/lapack_errno.h
Example Code	/opt/cell/sdk/src/lapack-examples-source.tar

The following table describes the important file components of the LAPACK library.

Table 5. File descriptions

File	Description
lapack.h	Contains the C function interface of LAPACK on PPE for DP
lapack_errno.h	Defines the error codes that LAPACK functions can return through a pointer. This file is included in the lapack.h header file.
liblapack.a	Contains the static library which contains the LAPACK library for Cell/B.E.
liblapack.so	Shared LAPACK library for Cell/B.E.
lapack-examples-source.tar	Contains two examples that demonstrate how to use the LAPACK library with the Software Development Kit for Multicore Acceleration

Chapter 3. Programming FORTRAN applications

This topic describes how to program FORTRAN applications using the LAPACK library.

Calling LAPACK routines

In Fortran programs, most LAPACK routines can be invoked with the CALL statement, such as *CALL Routine-name (argument_1, ..., argument_n)*. The following is an example:

```
CALL DGETRF(m, n, A, lda, ipiv, info);
```

Passing arguments for arrays

Arrays are declared in Fortran by specifying the array name, the number of dimensions, and the range of each dimension in a DIMENSION statement or an explicit data type statement, such as INTEGER, REAL, DOUBLE PRECISION, and so forth. For example, for a two-dimensional array in Fortran77, use `DATA_TYPE A(E1:E2, F1:F2)`, where A is the name of the array, E1 and F1 are the lower bounds of the first and second dimensions, respectively, and E2 and F2 are the upper bounds of the first and second dimensions, respectively. If either of the lower bounds is not specified, such as in A(E2,F1:F2), the value is assumed to be 1. The upper bounds are always required for each dimension.

Differences between dynamic linking and static linking

You do not need to modify your existing Fortran compilation procedures when using LAPACK. When linking and running your program, you must modify your existing Makefiles for LAPACK in order to set up the necessary libraries. When using dynamic linking, specify which libraries will be linked. When using static linking, add `–Wl,--export-dynamic` to your linker flags so that the symbols in LAPACK library are exported as dynamic symbols in the resulting binary. If you are accessing LAPACK from a Fortran program, you can compile and link using the commands shown in the table below. Or, you can follow the samples provided in LAPACK examples to use the build tool provided with the Software Development Kit for Multicore Acceleration.

Table 6. Compiler options for Fortran programs

Compiler	Environment	Fortran compiler command
ppu-gfortran	32 bit application, 32 bit integer, 32 bit pointer	`ppu-gfortran –m32 –ff2c abc.f –llapack –lblas` (Dynamic linking)
		`ppu-gfortran –m32 –ff2c abc.f -Wl,--export-dynamic -Wl,-Bstatic,-llapack,-lalf,-Bdynamic –lblas -lspe2 -lpthread -ldl` (Static linking)
	64 bit application, 32 bit integer, 64 bit pointer	`ppu-gfortran –m64 –ff2c abc.f –llapack –lblas` (Dynamic linking)

Table 6. Compiler options for Fortran programs (continued)

Compiler	Environment	Fortran compiler command
		`ppu-gfortran –m64 –ff2c abc.f -Wl,--export-dynamic -Wl,-Bstatic,-llapack,-lalf,-Bdynamic –lblas -lspe2 -lpthread -ldl` (Static linking)
ppuxlf	32 bit application, 32 bit integer, 32 bit pointer	`ppuxlf -qextname -q32 abc.f –llapack –lblas` (Dynamic linking)
		`ppuxlf –qextname –q32 abc.f -Wl,--export-dynamic -Wl,-Bstatic,-llapack,-lalf,-Bdynamic –lblas -lspe2 -lpthread -ldl` (Static linking)
	64 bit application, 32 bit integer, 64 bit pointer	`ppuxlf -qextname –q64 abc.f –llapack –lblas` (Dynamic linking)
		`ppuxlf –qextname –q64 abc.f -Wl,--export-dynamic -Wl,-Bstatic,-llapack,-lalf,-Bdynamic –lblas -lspe2 -lpthread -ldl` (Static linking)

Chapter 4. Programming C applications

This topic describes how to program C applications using the LAPACK library.

Calling LAPACK routines

Before calling LAPACK routines from your C program, add the following statement to your program to include the LAPACK header file:

```
#include <lapack.h>
```

This file contains entries for all of the LAPACK routines. You can invoke the LAPACK routines using the following type of statement: *Routine-name (argument_1, ..., argument_n)* Here is an example:

```
dgetrf_(&m, &n, A, &lda, &ipiv, &info);
```

Passing arguments for arrays

In C programs, arrays are arranged in storage in row-major order. This means that the last subscript expression increases most rapidly, the next-to-the-last subscript expression increases less rapidly, and so forth, with the first subscript expression increasing the least rapidly. LAPACK routines require that arrays passed as arguments must be in column-major order. This is the array storage convention used by Fortran programs. To pass an array from your C program to LAPACK, to have LAPACK process the data correctly, and to get a result that is in the proper form for your C program, choose one of the following methods:

- Build and process the matrix, logically transposed from the outset, and transpose the results as necessary.
- Before the LAPACK call, transpose the input arrays. Then, following the LAPACK call, transpose any arrays updated as output.
- If there are arguments in the LAPACK calling sequence indicating whether the arrays are to be processed in normal or transposed form, such as the *trans* arguments in the DGETRS_ routines, use these arguments in combination with the matrix equivalence rules to avoid transposing your data in separate operations.

Differences between dynamic linking and static linking

To compile C programs calling LAPACK routines, specify the –I option to identify the location of the LAPACK header file. For X86 machines, the location is /opt/cell/sysroot/usr/include. For PowerPC machines and CBEA BladeCenter® servers, the location is already included in the default search path, /usr/include.

When linking and running your program, modify your existing Makefiles for LAPACK in order to set up the necessary libraries. When using dynamic linking, specify which libraries will be linked. When using static linking, add `–Wl,--export-dynamic` to your linker flags so that the symbols in the LAPACK library are exported as dynamic symbols in the output binary.

If you are accessing LAPACK from a C program, you can compile and link using the commands shown in the table below. Or you can follow the procedure listed in “Building the LAPACK example code” on page 21 to use the build tool included in the Software Development Kit for Multicore Acceleration.

Table 7. Compiler options for C programs

Compiler	Environment	Fortran compiler command
ppu-gcc	32 bit application, 32 bit integer, 32 bit pointer	`ppu-gcc –m32 abc.c –llapack –lblas` (Dynamic linking)
		`ppu-gcc –m32 abc.c -Wl,--export-dynamic -Wl,-Bstatic,-llapack,-lalf,-Bdynamic –lblas -lspe2 -lpthread -ldl` (Static linking)
	64 bit application, 32 bit integer, 64 bit pointer	`ppu-gcc –m64 abc.c –llapack –lblas` (Dynamic linking)
		`ppu-gcc –m64 abc.c -Wl,--export-dynamic -Wl,-Bstatic,-llapack,-lalf,-Bdynamic –lblas -lspe2 -lpthread -ldl` (Static linking)
ppuxlc	32 bit application, 32 bit integer, 32 bit pointer	`ppuxlc -q32 abc.c –llapack –lblas` (Dynamic linking)
		`ppuxlc –q32 abc.c -Wl,--export-dynamic -Wl,-Bstatic,-llapack,-lalf,-Bdynamic –lblas -lspe2 -lpthread -ldl` (Static linking)
	64 bit application, 32 bit integer, 64 bit pointer	`ppuxlc –q64 abc.c –llapack –lblas` (Dynamic linking)
		`ppuxlc –q64 abc.c -Wl,--export-dynamic -Wl,-Bstatic,-llapack,-lalf,-Bdynamic –lblas -lspe2 -lpthread -ldl` (Static linking)

Chapter 5. Tuning the LAPACK library for performance

The LAPACK library provides additional features to help you optimize available resources and create high performance code.

The LAPACK routines use algorithms tailored to efficiently use specific characteristics of the Cell Broadband Engine Architecture, such as double buffering and dual issue, to achieve high performance.

When writing your code to use the LAPACK library, apply the following guidelines to optimize performance.

1. Use 128–byte aligned data. Memory access is more efficient when data is 128–byte aligned.
2. Supply a matrix whose number of rows and columns is greater than 1024.
3. Use the NUMA APIs to interleave the memory of the data on different nodes or bind the memory and processor to *node0*. You can try different modes to find out which gives the best performance. For example, if you want to interleave memory on both processors, you can run your application by typing the following command:

   ```
   numactl --interleave=all <Your_Application> <Options>
   ```

4. When operating on a matrix whose number of rows and columns is less than 1024, you can specify that LAPACK use fewer SPEs. LAPACK routines can do automatic adjustment for smaller size matrices. Interactive determination of the most efficient matrix size improves performance.

Environment variables

The following table lists the environment variables you can modify.

Table 8. Environment variables

Variable name	Purpose	Default value
LAPACK_NUMSPES	Specifies the number of SPEs to use.	16

Chapter 6. Catching failures

At run time, you can encounter different types of errors that are specifically related to the use of the LAPACK routines. It is good practice to check for a nonzero value of *INFO* on return from a LAPACK routine.

Invalid arguments

If you supply an illegal value as an input argument to a LAPACK routine, the routine will set *INFO* to a negative value (greater than *LAPACK_ERR_BASE*) and cause the error handler XERBLA to write a message to the standard output. For example,

```
** On entry to SGESV parameter number 4 had an illegal value
```

This message is caused by passing a value of *LDA* to SGESV that is less than the value of the argument N. To remove this error, check the input arguments, recompile, re-link, and then run your program again.

Computational failures

A positive value of *INFO* returned by a LAPACK routine results from a failure in the algorithm process. The following are common causes of this failure:

- A matrix is singular (to working precision)
- Asymmetric matrix is not positive definite
- An iterative algorithm for computing eigenvalues or eigenvectors fails to converge in the permitted number of iterations.

For example, if you call SGESVX to solve a system of equations with a coefficient matrix that is approximately singular, the routine might detect exact singularity at the *i*-th stage of the LU factorization. If this happens, the routine returns `INFO = i`, or in most cases it will compute an estimate of the reciprocal condition number that is less than machine precision. In this case it returns `INFO = n+1`. A failure with a return value of `INFO > 0` causes control to return to the calling program. Check for a nonzero value of *INFO* on return from a LAPACK to catch these failures.

Program exceptions

A failure in the runtime environment might cause a LAPACK routine to return a negative value for *INFO* (less than *LAPACK_ERR_BASE*). Common causes of this failure are:

- Out of memory
- A Cell/B.E. SPE resource is unavailable or not functioning correctly

For example, the return value *E_LAPACK_INIT* indicates that there is no accelerator available and the LAPACK library cannot initialize the environment to do the calculation. To remove this error, check the runtime environment, ensure that there is enough memory, and that the accelerator is functioning correctly. Finally, run your program again.

Chapter 7. Memory size limitations

The Cell/B.E. memory size and lack of swap partition limits the matrix size processed by the LAPACK library.

The following tables provides an approximation of how much memory will be used by the DGESVD and DGESDD routines. Use these tables to determine how large the matrix will be after one processing cycle. These limits assume a `N*N` square matrix containing double precision floating point numbers

Table 9. Memory limitation when calling DGESVD from LAPACK library

JOBU	JOBVT	Additional Memory required by the LAPACK library	Matrix size limitation for BladeCenter QS21 (2GB)	Matrix size limitation for BladeCenter QS22 (8GB)	Matrix size limitation for BladeCenter QS22 (32GB)
A/S	A/S	5*N*N	7000 x 7000	14000 x 14000	28000 x 28000
A/S	N/O	4*N*N	7900 x 7900	15800 x 15800	31600 x 31600
N/O	A/S	4*N*N	7900 x 7900	15800 x 15800	31600 x 31600
N/O	N/O	3*N*N	9100 x 9100	18200 x 18200	36400 x 36400

Table 10. Memory limitation when calling DGESDD from LAPACK library

JOBZ	Additional Memory required by the LAPACK library	Matrix size limitation for BladeCenter QS21 (2GB)	Matrix size limitation for BladeCenter QS22 (8GB)	Matrix size limitation for BladeCenter QS22 (32GB)
N	5*N*N + 17*N	7300 x 7300	14600 x 14600	29000 x 29000
O	13*N*N + 11*N	4500 x 4500	9000 x 9000	17408 x 17408
S/A	12*N*N + 11*N	4700 x 4700	9400 x 9400	17408 x 17408

Chapter 8. LAPACK programming examples

To build the examples listed in this document, follow this procedure:

Building the LAPACK example code

1. Cut and paste the Makefile source from an online or PDF copy of this document into an editor and save it as "Makefile".
2. Cut and paste the example source from an online or PDF copy of this document into an editor and save the file with a name such as doc_example.c
3. Edit the Makefile to use the name of the source file that you chose in the previous step. If you did not use "doc_example.c" then substitute the name you chose into the lines that contain `doc_example` and `doc_example.a`.
4. Copy the Makefile and the example source file into your development source directory (for example into /opt/sandbox/).
5. From a shell prompt, type the following commands:

```
$ cd /opt/sandbox
$ export CELL_TOP=/opt/cell/sdk
$ make
```

The following Makefile is similar to other SDK library Makefiles; however the `IMPORTS` line is different.

```
# ----------------------------------------------------------------
# (C)Copyright 2007,2008
# International Business Machines Corporation
# All Rights Reserved.
#
# Redistribution and use in source and binary forms, with or
# without modification, are permitted provided that the
# following conditions are met:
#
# Redistributions of source code must retain the above copyright
# notice, this list of conditions and the following disclaimer.
#
# Redistributions in binary form must reproduce the above
# copyright notice, this list of conditions and the following
# disclaimer in the documentation and/or other materials
# provided with the distribution.
#
# Neither the name of IBM Corporation nor the names of its
# contributors may be used to endorse or promote products
# derived from this software without specific prior written
# permission.
#
# THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND
# CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,
# INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
# MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
# DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
# CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
# SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
# NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
# LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
# HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
# CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
# OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,
# EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
# ----------------------------------------------------------------
# PROLOG END TAG zYx
```

```
############################################################################
#                       Target
############################################################################

   PROGRAM_spu    := doc_example
   LIBRARY_embed  := doc_example.a

############################################################################
#                       Local Defines
############################################################################

   IMPORTS        := -llapack –lblas -lm

############################################################################
#                       make.footer
############################################################################

# If necessary, change this to the location of make.footer:
   include /opt/cell/sdk/buildutils/make.footer

############################################################################
```

Example: Inverse matrix application

The following sample application shows you how to use the LAPACK library. It invokes the DGETRF and DGETRI routines to get an inverse matrix.

```
#include <stdlib.h>
#include <stdio.h>
#include <lapack.h>

int main( int argc, char *argv[] )
{
   int info = 0;
   double *a;
   int *ipiv;
   int n;
   int i,j;

   if( argc < 2 )
   {
      fprintf(stderr, "%s N\n", argv[0]);
      return 0;
   }
   n = atoi(argv[1]);

   posix_memalign((void **)(&ipiv), 128, sizeof(int)*n);
   posix_memalign((void **)(&a), 128, sizeof(double)*n*n);
   if( a == NULL || ipiv == NULL )
   {
      fprintf(stderr, "a/ipiv malloc error\n");
      return 0;
   }

   for( i = 0; i < n; i++ )
   {
      for( j = 0; j < n; j++ )
      {
         a[i+j*n]= (drand48() - 0.5f)*4;
      }
   }

   /*---------Call Cell LAPACK library---------*/
   dgetrf_(&n, &n, a, &n, ipiv, &info);
   if( info != 0 )
```

```
   {
      fprintf(stderr, "Call dgetrf error\n");
      goto end;
   }

   /*---------Query workspace-------*/
   double workspace;
   int tmp=-1;
   int lwork;
   double *work;
   dgetri_(&n, a, &n, ipiv, &workspace, &tmp, &info);
   lwork = (int)workspace;
   work = malloc(sizeof(double)*lwork);
   if(work == NULL)
   {
      printf("work malloc error\n");
      goto end;
   }

   /*---------Call Cell LAPACK library---------*/
   dgetri_(&n, a, &n, ipiv, work, &lwork, &info);
   if( info != 0 )
   {
      fprintf(stderr, "Call dgetri error\n");
      free(work);
      goto end;
   }
   printf("Inverse matrix completed!\n");

   end:
   free(ipiv);
   free(a);
   return 0;
}
```

Part 4. LAPACK PPE APIs

The LAPACK package provides optimized APIs.

The LAPACK package included with the Software Development Kit for Multicore Acceleration includes optimized LAPACK routines.

The following routines have been optimized to use features of the Synergistic Processing Elements (SPEs):

- DGETRF - Computes the LU factorization of a general matrix
- DGETRS – Linear equation solver
- DGETRI - Computes the inverse of a general matrix using an LU factorization
- DGEQRF - Computes the QR factorization of a general matrix
- DGELQF - Compute the LQ factorization of a general matrix
- DPOTRF - Computes the Cholesky factorization of a symmetric positive matrix
- DPOTRS – Linear equation (symmetric positive definite) solver
- DBDSQR - Computes the singular value decomposition of a real bi-diagonal matrix using an implicit zero-shift QR algorithm
- DSTEQR - Computes the singular value decomposition of a real symmetric tridiagonal matrix using an implicit QR algorithm
- DGESVD - Computes the singular value decomposition (SVD) of a real matrix using an implicit zero-shift QR algorithm, optionally computing the left and/or right singular vectors v
- DGESDD - Computes the singular value decomposition (SVD) of a real matrix using a divide-and-conquer algorithm, optionally computing the left and/or right singular vectors.

For general information about LAPACK, or for specific syntax information about these APIs, refer to the netlib documentation at: http://netlib.org/lapack/. In addition to the features documented in this link, the Cell/B.E. LAPACK library provides extended error information in *INFO*. For example, if the output of *info is less than *LAPACK_ERR_BASE* (defined in the lapack_errno.h header file), then an error specific to the Cell Broadband Engine Architecture occurred during the call to that routine.

Part 5. Appendixes

Appendix A. Accessibility features

Accessibility features help users who have a physical disability, such as restricted mobility or limited vision, to use information technology products successfully.

The following list includes the major accessibility features:

- Keyboard-only operation
- Interfaces that are commonly used by screen readers
- Keys that are tactilely discernible and do not activate just by touching them
- Industry-standard devices for ports and connectors
- The attachment of alternative input and output devices

IBM and accessibility

See the IBM Accessibility Center at http://www.ibm.com/able/ for more information about the commitment that IBM has to accessibility.

Appendix B. Getting help or technical assistance

If you need help, service, or technical assistance or just want more information about IBM products, you will find a wide variety of sources available from IBM to assist you.

This appendix contains information about where to go for additional information about IBM and IBM products and whom to call for service, if it is necessary.

Using the documentation

Information about your IBM hardware or software is available in the documentation that comes with the product. That documentation can include printed documents, online documents, readme files, and help files. See the troubleshooting information in your documentation for instructions for using diagnostic programs. The troubleshooting information or the diagnostic programs might tell you that you need additional or updated device drivers or other software. IBM maintains pages on the World Wide Web where you can get the latest technical information and download device drivers and updates. To access these pages, go to http://www.ibm.com/bladecenter/, click Support, and follow the instructions. Also, some documents are available through the IBM Publications Center at http://www.ibm.com/shop/publications/order/.

Getting help and information from the World Wide Web

You can locate documentation and other resources on the World Wide Web. Refer to the following web sites:

- IBM BladeCenter systems, optional devices, services, and support information at http://www.ibm.com/bladecenter/. For service information, select Support.
- developerWorks Cell Broadband Engine Resource Center at http://www.ibm.com/developerworks/power/cell/. To access the Cell/B.E. forum on developerWorks, select **Community**.
- The Barcelona Supercomputing Center (BSC) Web site at http://www.bsc.es/projects/deepcomputing/linuxoncell.
- There is also support for the Full-System Simulator and XL C/C++ Compiler through their individual alphaWorks® forums. If in doubt, start with the Cell/B.E. architecture forum.
- The GNU Project debugger, GDB, is supported through many different forums on the Web, but primarily at the GDB Web site http://www.gnu.org/software/gdb/gdb.html.

Contacting IBM Support

To obtain telephone assistance, for a fee or on a support contract, contact IBM Support. In the U.S. and Canada, call 1-800-IBM-SERV (1-800-426-7378), or see http://www.ibm.com/planetwide/ for support telephone numbers.

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing 2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Software Interoperability Coordinator, Department 49XA
3605 Highway 52 N
Rochester, MN 55901
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating

platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_. All rights reserved.

If you are viewing this information in softcopy, the photographs and color illustrations may not appear.

Trademarks

The following terms are trademarks of International Business Machines Corporation in the United States, other countries, or both:

alphaWorks
BladeCenter
developerWorks
IBM
PowerPC
Resource Link

Adobe®, Acrobat, Portable Document Format (PDF), and PostScript® are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, other countries, or both.

Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.

Linux® is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product or service names may be trademarks or service marks of others.

Terms and conditions

Permission for the use of these publications is granted subject to the following terms and conditions.

Personal Use: You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative works of these publications, or any portion thereof, without the express consent of the manufacturer.

Commercial Use: You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of the manufacturer.

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any data, software or other intellectual property contained therein.

The manufacturer reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by the manufacturer, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

THE MANUFACTURER MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THESE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

Related documentation

This topic helps you find related information.

Document location

Links to documentation for the SDK are provided on the IBM developerWorks Web site located at:

`http://www.ibm.com/developerworks/power/cell/`

Click the **Docs** tab.

The following documents are available, organized by category:

Architecture

- *Cell Broadband Engine Architecture*
- *Cell Broadband Engine Registers*
- *SPU Instruction Set Architecture*

Standards

- *C/C++ Language Extensions for Cell Broadband Engine Architecture*
- *Cell Broadband Engine Linux Reference Implementation Application Binary Interface Specification*
- *SIMD Math Library Specification for Cell Broadband Engine Architecture*
- *SPU Application Binary Interface Specification*
- *SPU Assembly Language Specification*

Programming

- *Cell Broadband Engine Programmer's Guide*
- *Cell Broadband Engine Programming Handbook*
- *Cell Broadband Engine Programming Tutorial*

Library

- *Accelerated Library Framework for Cell Broadband Engine Programmer's Guide and API Reference*
- *Basic Linear Algebra Subprograms Programmer's Guide and API Reference*
- *Data Communication and Synchronization for Cell Broadband Engine Programmer's Guide and API Reference*
- *Example Library API Reference*
- *Fast Fourier Transform Library Programmer's Guide and API Reference*
- *LAPACK (Linear Algebra Package) Programmer's Guide and API Reference*
- *Mathematical Acceleration Subsystem (MASS)*
- *Monte Carlo Library Programmer's Guide and API Reference*
- *SDK 3.0 SIMD Math Library API Reference*
- *SPE Runtime Management Library*
- *SPE Runtime Management Library Version 1 to Version 2 Migration Guide*
- *SPU Runtime Extensions Library Programmer's Guide and API Reference*

- *Three dimensional FFT Prototype Library Programmer's Guide and API Reference*

Installation

- *SDK for Multicore Acceleration Version 3.1 Installation Guide*

Tools

- *Getting Started - XL C/C++ for Multicore Acceleration for Linux*
- *Compiler Reference - XL C/C++ for Multicore Acceleration for Linux*
- *Language Reference - XL C/C++ for Multicore Acceleration for Linux*
- *Programming Guide - XL C/C++ for Multicore Acceleration for Linux*
- *Installation Guide - XL C/C++ for Multicore Acceleration for Linux*
- *Getting Started - XL Fortran for Multicore Acceleration for Linux*
- *Compiler Reference - XL Fortran for Multicore Acceleration for Linux*
- *Language Reference - XL Fortran for Multicore Acceleration for Linux*
- *Optimization and Programming Guide - XL Fortran for Multicore Acceleration for Linux*
- *Installation Guide - XL Fortran for Multicore Acceleration for Linux*
- *Performance Analysis with the IBM Full-System Simulator*
- *IBM Full-System Simulator User's Guide*
- *IBM Visual Performance Analyzer User's Guide*

IBM PowerPC Base

- *IBM PowerPC Architecture Book*
  - *Book I: PowerPC User Instruction Set Architecture*
  - *Book II: PowerPC Virtual Environment Architecture*
  - *Book III: PowerPC Operating Environment Architecture*
- *IBM PowerPC Microprocessor Family: Vector/SIMD Multimedia Extension Technology Programming Environments Manual*

Glossary

This glossary provides definitions for terms included in this publication.

ALF. Accelerated Library Framework. This an API that provides a set of services to help programmers solving data parallel problems on a hybrid system. ALF supports the multiple-program-multiple-data (MPMD) programming style where multiple programs can be scheduled to run on multiple accelerator elements at the same time. ALF offers programmers an interface to partition data across a set of parallel processes without requiring architecturally-dependent code.

Barcelona Supercomputing Center. Spanish National Supercomputing Center, supporting IBM BladeCenter servers and Linux on Cell/B.E.

Broadband Engine. See *CBEA*.

CBEA. Cell Broadband Engine Architecture. A new architecture that extends the 64-bit PowerPC Architecture™. The CBEA and the Cell Broadband Engine are the result of a collaboration between Sony, Toshiba, and IBM, known as STI, formally started in early 2001.

Cell BE processor. The Cell BE processor is a multi-core broadband processor based on IBM's Power Architecture.

Cell Broadband Engine processor. See *Cell BE*.

GNU. GNU is Not Unix. A project to develop free Unix-like operating systems such as Linux.

GPL. GNU General Public License. Guarantees freedom to share, change and distribute free software.

HTTP. Hypertext Transfer Protocol. A method used to transfer or convey information on the World Wide Web.

Makefile. A descriptive file used by the `make`command in which the user specifies: (a) target program or library, (b) rules about how the target is to be built, (c) dependencies which, if updated, require that the target be rebuilt.

NUMA. Non-uniform memory access. In a multiprocessing system such as the Cell/B.E., memory is configured so that it can be shared locally, thus giving performance benefits.

PDF. Portable document format.

RPM. Originally an acronym for Red Hat Package Manager, and RPM file is a packaging format for one or more files used by many Linux systems when installing software programs.

SDK. Software development toolkit for Multicore Acceleration. A complete package of tools for application development.

Index

Printed in USA

SC33-8428-02