Parallel & Distributed Computing: Lecture 32

Alberto Paoluzzi

December 16, 2019

Basic Linear Algebra Subprograms (BLAS)

Introduction

The BLAS (Basic Linear Algebra Subprograms) are routines that provide standard building blocks for performing basic vector and matrix operations.

Introduction

The BLAS (Basic Linear Algebra Subprograms) are routines that provide standard building blocks for performing basic vector and matrix operations.

Level 1 BLAS performs scalar, vector and vector-vector operations,
Level 2 BLAS performs matrix-vector operations, and
Level 3 BLAS performs matrix-matrix operations.

Introduction

The BLAS (Basic Linear Algebra Subprograms) are routines that provide standard building blocks for performing basic vector and matrix operations.

Level 1 BLAS performs scalar, vector and vector-vector operations,
Level 2 BLAS performs matrix-vector operations, and
Level 3 BLAS performs matrix-matrix operations.

Because the BLAS are efficient, portable, and widely available, they are commonly used in the development of high quality linear algebra software, LAPACK for example.

History

They are the de facto standard low-level routines for linear algebra libraries; the routines have bindings for both C and Fortran.

It originated as a Fortran library in 1979 and its interface was standardized by the BLAS Technical (BLAST) Forum, whose latest BLAS report can be found on the netlib website. This Fortran library is known as the reference implementation

History

They are the de facto standard low-level routines for linear algebra libraries; the routines have bindings for both C and Fortran.

It originated as a Fortran library in 1979 and its interface was standardized by the BLAS Technical (BLAST) Forum, whose latest BLAS report can be found on the netlib website. This Fortran library is known as the reference implementation

Most libraries that offer linear algebra routines conform to the BLAS interface, allowing library users to develop programs that are agnostic of the BLAS library being used.

OpenBLAS is an open-source library that is hand-optimized for many of the popular architectures.

The LINPACK benchmarks rely heavily on the BLAS routine `gemm` for its performance measurements.

Level 1: SINGLE

- SROTG - setup Givens rotation
- SROTMG - setup modified Givens rotation
- SROT - apply Givens rotation
- SROTM - apply modified Givens rotation
- SSWAP - swap x and y
- SSCAL - x = a*x
- SCOPY - copy x into y
- SAXPY - y = a*x + y
- SDOT - dot product
- SDSDOT - dot product with extended precision accumulation
- SNRM2 - Euclidean norm
- SCNRM2- Euclidean norm
- SASUM - sum of absolute values
- ISAMAX - index of max abs value

Level 1: DOUBLE

- DROTG - setup Givens rotation
- DROTMG - setup modified Givens rotation
- DROT - apply Givens rotation
- DROTM - apply modified Givens rotation
- DSWAP - swap x and y
- DSCAL - x = a*x
- DCOPY - copy x into y
- DAXPY - y = a*x + y
- DDOT - dot product
- DSDOT - dot product with extended precision accumulation
- DNRM2 - Euclidean norm
- DZNRM2 - Euclidean norm
- DASUM - sum of absolute values
- IDAMAX - index of max abs value

Level 1: Complex

- CROTG - setup Givens rotation
- CSROT - apply Givens rotation
- CSWAP - swap x and y
- CSCAL - x = a*x
- CSSCAL - x = a*x
- CCOPY - copy x into y
- CAXPY - y = a*x + y
- CDOTU - dot product
- CDOTC - dot product, conjugating the first vector
- SCASUM - sum of absolute values
- ICAMAX - index of max abs value

Level 1: Double Complex

- ZROTG - setup Givens rotation
- ZDROTF - apply Givens rotation
- ZSWAP - swap x and y
- ZSCAL - x = a*x
- ZDSCAL - x = a*x
- ZCOPY - copy x into y
- ZAXPY - y = a*x + y
- ZDOTU - dot product
- ZDOTC - dot product, conjugating the first vector
- DZASUM - sum of absolute values
- IZAMAX - index of max abs value

Level 2: DOUBLE

- DGEMV - matrix vector multiply
- DGBMV - banded matrix vector multiply
- DSYMV - symmetric matrix vector multiply
- DSBMV - symmetric banded matrix vector multiply
- DSPMV - symmetric packed matrix vector multiply
- DTRMV - triangular matrix vector multiply
- DTBMV - triangular banded matrix vector multiply
- DTPMV - triangular packed matrix vector multiply
- DTRSV - solving triangular matrix problems
- DTBSV - solving triangular banded matrix problems
- DTPSV - solving triangular packed matrix problems
- DGER - performs the rank 1 operation A := alpha*x*y' + A
- DSYR - performs the symmetric rank 1 operation A := alpha*x*x' + A
- DSPR - symmetric packed rank 1 operation A := alpha*x*x' + A
- DSYR2 - performs the symmetric rank 2 operation, A := alpha*x*y' + alpha*y*x' + A
- DSPR2 - performs the symmetric packed rank 2 operation, A := alpha*x*y' + alpha*y*x' + A

Level 3:

- Single
  - SGEMM - matrix matrix multiply
  - SSYMM - symmetric matrix matrix multiply
  - SSYRK - symmetric rank-k update to a matrix
  - SSYR2K - symmetric rank-2k update to a matrix
  - STRMM - triangular matrix matrix multiply
  - STRSM - solving triangular matrix with multiple right hand sides
- Double
  - DGEMM - matrix matrix multiply
  - DSYMM - symmetric matrix matrix multiply
  - DSYRK - symmetric rank-k update to a matrix
  - DSYR2K - symmetric rank-2k update to a matrix
  - DTRMM - triangular matrix matrix multiply
  - DTRSM - solving triangular matrix with multiple right hand sides

Level 3:

- Complex
  - CGEMM - matrix matrix multiply
  - CSYMM - symmetric matrix matrix multiply
  - CHEMM - hermitian matrix matrix multiply
  - CSYRK - symmetric rank-k update to a matrix
  - CHERK - hermitian rank-k update to a matrix
  - CSYR2K - symmetric rank-2k update to a matrix
  - CHER2K - hermitian rank-2k update to a matrix
  - CTRMM - triangular matrix matrix multiply
  - CTRSM - solving triangular matrix with multiple right hand sides

Linear Algebra Package (LAPACK)

Introduction (from http://www.netlib.org/lapack/)

LAPACK is written in Fortran 90 and 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, Cholesky, QR, SVD, Schur, generalized Schur) are also provided, as are related computations such as reordering of the Schur factorizations and estimating condition numbers.

Dense and banded matrices are handled, but not general sparse matrices.

In all areas, similar functionality is provided for real and complex matrices, in both single and double precision.

LAPACK routines are written so that as much as possible of the computation is performed by calls to the Basic Linear Algebra Subprograms (BLAS).

LAPACK is designed at the outset to exploit the Level 3 BLAS

LAPACK

Release history

Related projects

LAPACK function naming scheme

The name of each LAPACK routine is a coded specification of its function (within the very tight limits of standard Fortran 77 6-character names).

All driver and computational routines have names of the form XYYZZZ, where for some driver routines the 6th character is blank.

The first letter, X, indicates the data type as follows:

S	REAL
D	DOUBLE PRECISION
C	COMPLEX
Z	COMPLEX*16 or DOUBLE COMPLEX

When we wish to refer to an LAPACK routine generically, regardless of data type, we replace the first letter by x. Thus xGESV refers to any or all of the routines SGESV, CGESV, DGESV and ZGESV.

The next two letters, YY, indicate the type of matrix (or of the most significant matrix).

Most of these two-letter codes apply to both real and complex matrices; a few apply specifically to one or the other

LAPACK matrix naming scheme

- BD bidiagonal
- DI diagonal
- GB general band
- GE general (i.e., unsymmetric, in some cases rectangular)
- GG general matrices, generalized problem (i.e., a pair of general matrices)
- GT general tridiagonal
- HB (complex) Hermitian band
- HE (complex) Hermitian
- HG upper Hessenberg matrix, generalized problem (i.e a Hessenberg and a triangular matrix)
- HP (complex) Hermitian, packed storage
- HS upper Hessenberg
- OP (real) orthogonal, packed storage
- OR (real) orthogonal
- PB symmetric or Hermitian positive definite band
- PO symmetric or Hermitian positive definite
- PP symmetric or Hermitian positive definite, packed storage
- PT symmetric or Hermitian positive definite tridiagonal
- SB (real) symmetric band
- SP symmetric, packed storage
- ST (real) symmetric tridiagonal
- SY symmetric
- TB triangular band
- TG triangular matrices, generalized problem (i.e., a pair of triangular matrices)
- TP triangular, packed storage
- TR triangular (or in some cases quasi-triangular)
- TZ trapezoidal
- UN (complex) unitary
- UP (complex) unitary, packed storage

Julia's Linear Algebra

Standard Library

https://docs.julialang.org/en/v1/stdlib/LinearAlgebra/

Low-level matrix operations

https://docs.julialang.org/en/stable/stdlib/linalg/#Low-level-matrix-operations-1

BLAS functions

How to Check If Julia Is Using OpenBLAS or Intel MKL

BLAS Functions

LinearAlgebra.BLAS provides wrappers for some of the BLAS functions. Those BLAS functions that overwrite one of the input arrays have names ending in '!'. Usually, a BLAS function has four methods defined, for Float64, Float32, ComplexF64, and ComplexF32 arrays.

LAPACK functions

LAPACK Functions

LinearAlgebra.LAPACK provides wrappers for some of the LAPACK functions for linear algebra. Those functions that overwrite one of the input arrays have names ending in '!'.
Usually a function has 4 methods defined, one each for Float64, Float32, ComplexF64 and ComplexF32 arrays.

Intel MKL linear algebra in Julia.

MKL.jl is a package that makes Julia's linear algebra use Intel MKL BLAS and LAPACK instead of OpenBLAS. The build step of the package will automatically download Intel MKL and rebuild Julia's system image against Intel MKL

Intel MKL linear algebra in Julia.