Which C compiler and BLAS/LAPACK library should I use – gretl's numerical efficiency in different configurations¹

Marcin Błażejowski

University WSB Merito in Toruń, Poland

Gretl Conference 2023

¹This research is a part of Oracle for Research grant (agreement No. v040119) entitled: "Which C compiler and which BLAS/LAPACK library? – gretl numerical efficiency in different configurations".

Workplan

- 1 Environment
 - Hardware
 - Libraries
 - Compilers
- 2 Compilation
 - CFLAGS in use
 - LAPACK_LIBS in use
- ③ Simulation scenarios
 - dgemm efficiency (pure BLAS Level 3)
 - LAPACK/BLAS efficiency
 - Real econometrics efficiency

Results

- dgemm efficiency (pure BLAS Level 3)
- LAPACK/BLAS efficiency
- Real econometrics efficiency
- Overall performance

6 core paravirtualized machines available in Orace Cloud and run under Canonical-Ubuntu-22.04-2023.02.15-0:

AMD	Intel	
EPYC 7J13 64-Core Processor	Xeon Gold 6354 CPU @ 3.00GHz	
VM.Standard.E4.Flex	VM.Optimized3.Flex	

Hardware Libraries Compilers

Terminology

BLAS Basic Linear Algebra Subprograms

LAPACK Linear Algebra Package

Marcin Błażejowski Which C compiler and BLAS/LAPACK library...

Hardware Libraries Compilers

Terminology

BLAS Basic Linear Algebra Subprograms

$\bullet\,$ originated as a Fortran library in 1979

LAPACK Linear Algebra Package

• originated as a Fortran library in 1992

Hardware Libraries Compilers

Terminology

BLAS Basic Linear Algebra Subprograms

- originated as a Fortran library in 1979
- \bullet stable release: 3.11.0 (November 11, 2022)

LAPACK Linear Algebra Package

- originated as a Fortran library in 1992
- stable release: 3.11.0 (November 11, 2022)

Hardware Libraries Compilers

Terminology

BLAS Basic Linear Algebra Subprograms

- originated as a Fortran library in 1979
- stable release: 3.11.0 (November 11, 2022)
- provides so-called: Level 1, Level 2 and Level 3 operations

LAPACK Linear Algebra Package

- $\bullet\,$ originated as a Fortran library in 1992
- stable release: 3.11.0 (November 11, 2022)

Hardware Libraries Compilers

Terminology

BLAS Basic Linear Algebra Subprograms

- originated as a Fortran library in 1979
- stable release: 3.11.0 (November 11, 2022)
- provides so-called: Level 1, Level 2 and Level 3 operations

LAPACK Linear Algebra \mathbf{Pac} kage

- originated as a Fortran library in 1992
- stable release: 3.11.0 (November 11, 2022)
- provides routines for solving systems of simultaneous linear equations, least-squares solutions, eigenvalue and singular value problems, matrix factorizations (LU, Cholesky, QR, SVD, Schur, generalized Schur)

・ 同 ト ・ ヨ ト ・ ヨ ト

Hardware Libraries Compilers

Terminology

BLAS Basic Linear Algebra Subprograms

- originated as a Fortran library in 1979
- stable release: 3.11.0 (November 11, 2022)
- provides so-called: Level 1, Level 2 and Level 3 operations

LAPACK Linear Algebra \mathbf{Pac} kage

- originated as a Fortran library in 1992
- stable release: 3.11.0 (November 11, 2022)
- provides routines for solving systems of simultaneous linear equations, least-squares solutions, eigenvalue and singular value problems, matrix factorizations (LU, Cholesky, QR, SVD, Schur, generalized Schur)
- relies on an underlying BLAS implementation

Hardware Libraries Compilers

BLAS-only libraries

• BLIS:

• AOCL-BLIS:

Marcin Błażejowski Which C compiler and BLAS/LAPACK library...

Hardware Libraries Compilers

BLAS-only libraries

• BLIS:

 $\bullet\,$ based on GotoBLAS2 written by Kazushige Goto $\bar{\rm o}$

• AOCL-BLIS:

Hardware Libraries Compilers

BLAS-only libraries

• BLIS:

- based on GotoBLAS2 written by Kazushige Got $\bar{\mathrm{o}}$
- implements both: new and classic BLAS API, but no LAPACK API
- AOCL-BLIS:

Hardware Libraries Compilers

BLAS-only libraries

• BLIS:

- based on GotoBLAS2 written by Kazushige Got $\bar{\mathrm{o}}$
- implements both: new and classic BLAS API, but no LAPACK API
- version: 0.9.0
- package used: libblis4-openmp 0.9.0-1

• AOCL-BLIS:

Hardware Libraries Compilers

BLAS-only libraries

• BLIS:

- based on GotoBLAS2 written by Kazushige Got $\bar{\mathrm{o}}$
- implements both: new and classic BLAS API, but no LAPACK API
- version: 0.9.0
- package used: libblis4-openmp 0.9.0-1
- AOCL-BLIS:
 - based on BLIS

Hardware Libraries Compilers

BLAS-only libraries

• BLIS:

- based on GotoBLAS2 written by Kazushige Got $\bar{\mathrm{o}}$
- implements both: new and classic BLAS API, but no LAPACK API
- version: 0.9.0
- package used: libblis4-openmp 0.9.0-1
- AOCL-BLIS:
 - based on BLIS
 - part of AMD Optimizing CPU Libraries (AOCL)

Hardware Libraries Compilers

BLAS-only libraries

• BLIS:

- based on GotoBLAS2 written by Kazushige Got $\bar{\mathrm{o}}$
- implements both: new and classic BLAS API, but no LAPACK API
- version: 0.9.0
- package used: libblis4-openmp 0.9.0-1
- AOCL-BLIS:
 - based on BLIS
 - part of AMD Optimizing CPU Libraries (AOCL)
 - version: 4.0.0 Build 20221031
 - package used: aocl-linux-aocc-4.0

Hardware Libraries Compilers

LAPACK-only libraries

• libFLAME:

• AOCL-libFLAME:

Marcin Błażejowski Which C compiler and BLAS/LAPACK library...

Hardware Libraries Compilers

LAPACK-only libraries

- libFLAME:
 - C-only implementation (no Fortran libraries dependency)

• AOCL-libFLAME:

Hardware Libraries Compilers

LAPACK-only libraries

• libFLAME:

- C-only implementation (no Fortran libraries dependency)
- a complete dense linear algebra framework with LAPACK compatible API through lapack2flame layer

• AOCL-libFLAME:

Hardware Libraries Compilers

LAPACK-only libraries

- C-only implementation (no Fortran libraries dependency)
- a complete dense linear algebra framework with LAPACK compatible API through lapack2flame layer
- version: 5.2.0
- package used: libflame1 5.2.0-3ubuntu3
- AOCL-libFLAME:

Hardware Libraries Compilers

LAPACK-only libraries

• libFLAME:

- C-only implementation (no Fortran libraries dependency)
- a complete dense linear algebra framework with LAPACK compatible API through lapack2flame layer
- version: 5.2.0
- package used: libflame1 5.2.0-3ubuntu3
- abbreviation: **bflame** (as BLIS and libFLAME)

• AOCL-libFLAME:

Hardware Libraries Compilers

LAPACK-only libraries

- C-only implementation (no Fortran libraries dependency)
- a complete dense linear algebra framework with LAPACK compatible API through lapack2flame layer
- version: 5.2.0
- package used: libflame1 5.2.0-3ubuntu3
- abbreviation: **bflame** (as BLIS and libFLAME)
- AOCL-libFLAME:
 - based on libFLAME

Hardware Libraries Compilers

LAPACK-only libraries

- C-only implementation (no Fortran libraries dependency)
- a complete dense linear algebra framework with LAPACK compatible API through lapack2flame layer
- version: 5.2.0
- package used: libflame1 5.2.0-3ubuntu3
- abbreviation: **bflame** (as BLIS and libFLAME)
- AOCL-libFLAME:
 - based on libFLAME
 - part of AMD Optimizing CPU Libraries (AOCL)

Hardware Libraries Compilers

LAPACK-only libraries

- C-only implementation (no Fortran libraries dependency)
- a complete dense linear algebra framework with LAPACK compatible API through lapack2flame layer
- version: 5.2.0
- package used: libflame1 5.2.0-3ubuntu3
- abbreviation: **bflame** (as BLIS and libFLAME)
- AOCL-libFLAME:
 - based on libFLAME
 - part of AMD Optimizing CPU Libraries (AOCL)
 - version: 4.0.0 Build 20221031
 - package used: aocl-linux-aocc-4.0

Hardware Libraries Compilers

LAPACK-only libraries

• libFLAME:

- C-only implementation (no Fortran libraries dependency)
- a complete dense linear algebra framework with LAPACK compatible API through lapack2flame layer
- version: 5.2.0
- package used: libflame1 5.2.0-3ubuntu3
- abbreviation: **bflame** (as BLIS and libFLAME)
- AOCL-libFLAME:
 - based on libFLAME
 - part of AMD Optimizing CPU Libraries (AOCL)
 - version: 4.0.0 Build 20221031
 - package used: aocl-linux-aocc-4.0
 - abbreviation: aocl (as AOCL-BLIS with AOCL-libFLAME)

Hardware Libraries Compilers

BLAS/LAPACK combo libraries

• OpenBLAS:

• Intel oneAPI Math Kernel Library (oneMKL):

Marcin Błażejowski Which C compiler and BLAS/LAPACK library...

Hardware Libraries Compilers

BLAS/LAPACK combo libraries

- OpenBLAS:
 - $\bullet\,$ based on GotoBLAS2 written by Kazushige Goto $\bar{\rm o}$

• Intel oneAPI Math Kernel Library (oneMKL):

Hardware Libraries Compilers

BLAS/LAPACK combo libraries

- based on GotoBLAS2 written by Kazushige Got $\bar{\mathrm{o}}$
- version: 0.3.20
- package used: libopenblas0-openmp 0.3.20+ds-1
- Intel oneAPI Math Kernel Library (oneMKL):

Hardware Libraries Compilers

BLAS/LAPACK combo libraries

- based on GotoBLAS2 written by Kazushige Got $\bar{\mathrm{o}}$
- version: 0.3.20
- package used: libopenblas0-openmp 0.3.20+ds-1
- abbreviation: **openblas** or **oblas**
- Intel oneAPI Math Kernel Library (oneMKL):

Hardware Libraries Compilers

BLAS/LAPACK combo libraries

- based on GotoBLAS2 written by Kazushige Got $\bar{\mathrm{o}}$
- version: 0.3.20
- package used: libopenblas0-openmp 0.3.20+ds-1
- abbreviation: **openblas** or **oblas**
- Intel oneAPI Math Kernel Library (oneMKL):
 - version: 2023.1 build 20230303
 - package used: intel-oneapi-mkl-2023.1.0

Hardware Libraries Compilers

BLAS/LAPACK combo libraries

- based on GotoBLAS2 written by Kazushige Got $\bar{\mathrm{o}}$
- version: 0.3.20
- package used: libopenblas0-openmp 0.3.20+ds-1
- abbreviation: **openblas** or **oblas**
- Intel oneAPI Math Kernel Library (oneMKL):
 - version: 2023.1 build 20230303
 - package used: intel-oneapi-mkl-2023.1.0
 - abbreviation: **mkl**

Kernels used

Values reported by **\$sysinfo["blascore"]**

library	AMD	Intel
OpenBLAS	Zen	SkylakeX
BLIS	zen3	haswell
AOCL-BLIS	zen3	
oneMKL		AVX-512

御 ト イヨト イヨト

Hardware Libraries Compilers

Open Source

• GCC

• Clang/LLVM

Marcin Błażejowski Which C compiler and BLAS/LAPACK library...

Hardware Libraries Compilers

Open Source

• GCC

- version: 12.1.0
- package used: 12.1.0-2ubuntu1~22.04
- Clang/LLVM

э

Hardware Libraries Compilers

Open Source

• GCC

- \bullet version: 12.1.0
- package used: 12.1.0-2ubuntu1~22.04
- \bullet abbreviation: \mathbf{gcc}

• Clang/LLVM

Hardware Libraries Compilers

Open Source

• GCC

- \bullet version: 12.1.0
- package used: 12.1.0-2ubuntu1~22.04
- $\bullet\,$ abbreviation: \mathbf{gcc}

• Clang/LLVM

 $\bullet\,$ based upon LLVM 15.0.7
Hardware Libraries Compilers

Open Source

• GCC

- \bullet version: 12.1.0
- package used: 12.1.0-2ubuntu1~22.04
- abbreviation: gcc
- Clang/LLVM
 - based upon LLVM 15.0.7
 - version: Ubuntu Clang 15.0.7
 - package used: 1:15.0.7-0ubuntu0.22.04.1

Hardware Libraries Compilers

Open Source

• GCC

- \bullet version: 12.1.0
- package used: 12.1.0-2ubuntu1~22.04
- abbreviation: gcc
- Clang/LLVM
 - based upon LLVM 15.0.7
 - version: Ubuntu Clang 15.0.7
 - package used: 1:15.0.7-0ubuntu0.22.04.1
 - abbreviation: llvm

Hardware Libraries Compilers

Provided by CPU vendor

 $\bullet\,$ AMD Optimizing C/C++ and Fortran Compilers (AOCC)

• Intel oneAPI DPC++/C++ Compiler

伺 ト イヨト イヨト

Hardware Libraries Compilers

Provided by CPU vendor

AMD Optimizing C/C++ and Fortran Compilers (AOCC)
based on LLVM Mirror.Version.14.0.6

• Intel oneAPI DPC++/C++ Compiler

御 と く き と く き と

Hardware Libraries Compilers

Provided by CPU vendor

- based on LLVM Mirror.Version.14.0.6
- version: AMD Clang 14.0.6 (CLANG: AOCC_4.0.0-Build#434 2022_10_28)
- package used: aocc-compiler-4.0.0
- Intel oneAPI DPC++/C++ Compiler

Hardware Libraries Compilers

Provided by CPU vendor

- based on LLVM Mirror.Version.14.0.6
- version: AMD Clang 14.0.6 (CLANG: AOCC_4.0.0-Build#434 2022_10_28)
- package used: aocc-compiler-4.0.0
- abbreviation: **aocc**
- Intel oneAPI DPC++/C++ Compiler

Hardware Libraries Compilers

Provided by CPU vendor

- based on LLVM Mirror.Version.14.0.6
- version: AMD Clang 14.0.6 (CLANG: AOCC_4.0.0-Build#434 2022_10_28)
- package used: aocc-compiler-4.0.0
- abbreviation: **aocc**
- Intel oneAPI DPC++/C++ Compiler
 - based upon LLVM 16.0.0git

Hardware Libraries Compilers

Provided by CPU vendor

- based on LLVM Mirror.Version.14.0.6
- version: AMD Clang 14.0.6 (CLANG: AOCC_4.0.0-Build#434 2022_10_28)
- package used: aocc-compiler-4.0.0
- abbreviation: **aocc**
- Intel oneAPI DPC++/C++ Compiler
 - based upon LLVM 16.0.0git
 - version: Intel oneAPI DPC++/C++ Compiler 2023.1.0 (2023.1.0.20230320)
 - package used: intel-oneapi-dpcpp-cpp-2023.1.0

Hardware Libraries Compilers

Provided by CPU vendor

• AMD Optimizing C/C++ and Fortran Compilers (AOCC)

- based on LLVM Mirror.Version.14.0.6
- version: AMD Clang 14.0.6 (CLANG: AOCC_4.0.0-Build#434 2022_10_28)
- package used: aocc-compiler-4.0.0
- abbreviation: **aocc**
- Intel oneAPI DPC++/C++ Compiler
 - based upon LLVM 16.0.0git
 - version: Intel oneAPI DPC++/C++ Compiler 2023.1.0 (2023.1.0.20230320)
 - package used: intel-oneapi-dpcpp-cpp-2023.1.0
 - abbreviation: icx

伺 ト イヨト イヨト

We use gret1 version 2023a compiled with the following flags:

-march=native -mtune=native -03.

Marcin Błażejowski Which C compiler and BLAS/LAPACK library...

• • = • • = •



We use gret1 version 2023a compiled with the following flags:

-march=native -mtune=native -O3.

In case of compilers provided by CPU vendor we also use:

	aocl	mkl	oblas	bflame
aocc	-ffast-math			
icx		-fp-model=precise		

	Environment Compilation Simulation scenarios Results	CFLAGS in use LAPACK_LIBS in use	
aocl			

We always use: -l:libflame.so.4.0 -lblis-mt.

Marcin Błażejowski Which C compiler and BLAS/LAPACK library...



We always use: -1:libflame.so.4.0 -lblis-mt.

In case of different compilers we also use:

gcc -lm -ldl
llvm -fopenmp=libomp
aocc -fopenmp=libomp -lalm

FLAGS in use APACK_LIBS in use

We always use:

- -L/opt/intel/oneapi/mkl/2023.1.0/lib/intel64
- -lmkl_intel_lp64 -lmkl_intel_thread -lmkl_core
- -liomp5 -lpthread -lm -ldl.

 Environment
 Compilation

 Compilation
 CFLAGS in use

 Simulation scenarios
 LAPACK_LIBS in use

 Results
 Coblas

We always use: -lopenblas.

Marcin Błażejowski Which C compiler and BLAS/LAPACK library...

- - E

CFLAGS in use LAPACK_LIBS in use

bflame

We always use: -lflame -lblis -ldl.

Marcin Błażejowski Which C compiler and BLAS/LAPACK library...

• • = • • = •

dgemm efficiency (pure BLAS Level 3) LAPACK/BLAS efficiency Real econometrics efficiency

gretl's packages in use

In simulations we use:

- **•** matrix_perf (ver. 1.11) by Allin Cottrell
- OCC (ver. 0.1) by Riccardo "Jack" Lucchetti, Giulio Palomba and Luca Pedini
- StrucTiSM (ver. 0.7) by Riccardo "Jack" Lucchetti and Sven Schreiber
- BMA (ver. 3.1) by Marcin Błażejowski and Jacek Kwiatkowski

dgemm efficiency (pure BLAS Level 3) LAPACK/BLAS efficiency Real econometrics efficiency

General assumptions

Marcin Błażejowski Which C compiler and BLAS/LAPACK library...

dgemm efficiency (pure BLAS Level 3) LAPACK/BLAS efficiency Real econometrics efficiency

General assumptions

• Each simulation was repeated 10 times.

dgemm efficiency (pure BLAS Level 3) LAPACK/BLAS efficiency Real econometrics efficiency

General assumptions

- Each simulation was repeated 10 times.
- Before each simulation (but right after setting) we put it to sleep for 3 seconds.

dgemm efficiency (pure BLAS Level 3) LAPACK/BLAS efficiency Real econometrics efficiency

General assumptions

- Each simulation was repeated 10 times.
- Before each simulation (but right after setting) we put it to sleep for 3 seconds.
- Each scenario is run in non-threaded and threaded mode. For the later we use:

set omp_num_threads \$sysinfo["ncores"].

dgemm efficiency (pure BLAS Level 3) LAPACK/BLAS efficiency Real econometrics efficiency

General assumptions

- Each simulation was repeated 10 times.
- Before each simulation (but right after setting) we put it to sleep for 3 seconds.
- Each scenario is run in non-threaded and threaded mode. For the later we use:

set omp_num_threads \$sysinfo["ncores"].

• We have scenarios for smaller and bigger cases which is controlled via different mnk values.

dgemn efficiency (pure BLAS Level 3) LAPACK/BLAS efficiency Real econometrics efficiency

dgemm test

Marcin Błażejowski Which C compiler and BLAS/LAPACK library...

<ロト <問ト < 臣ト < 臣ト

э

dgemm efficiency (pure BLAS Level 3 LAPACK/BLAS efficiency Real econometrics efficiency

dgemm test

• We use matrix_perf with Allin's setup.

Marcin Błażejowski Which C compiler and BLAS/LAPACK library...

• • = • • = •

dgemn efficiency (pure BLAS Level 3 LAPACK/BLAS efficiency Real econometrics efficiency

dgemm test

- We use matrix_perf with Allin's setup.
- The packages was slightly modified to return results as **bundle** of bundles of matrixes.

dgemm efficiency (pure BLAS Level 3) LAPACK/BLAS efficiency Real econometrics efficiency

LAPACK/BLAS test

Marcin Błażejowski Which C compiler and BLAS/LAPACK library...

• • = • • = •

dgemm efficiency (pure BLAS Level 3) LAPACK/BLAS efficiency Real econometrics efficiency

LAPACK/BLAS test

• We use the following build-in functions: mols, qrdecomp, cholesky, inv, invpd, eigen, det, psdroot, svd

dgemm efficiency (pure BLAS Level 3) LAPACK/BLAS efficiency Real econometrics efficiency

LAPACK/BLAS test

- We use the following build-in functions: mols, qrdecomp, cholesky, inv, invpd, eigen, det, psdroot, svd
- Dimensions for smaller cases:

• Dimensions for bigger cases:

dgemm efficiency (pure BLAS Level 3) LAPACK/BLAS efficiency Real econometrics efficiency

LAPACK/BLAS test

- We use the following build-in functions: mols, qrdecomp, cholesky, inv, invpd, eigen, det, psdroot, svd
- Dimensions for smaller cases:
 - non-square: $5000 \times 40 \ (m = k = 5000, n = 40)$
 - square: $500 \times 500 \ (m = n = k = 500)$
- Dimensions for bigger cases:

dgemm efficiency (pure BLAS Level 3) LAPACK/BLAS efficiency Real econometrics efficiency

LAPACK/BLAS test

- We use the following build-in functions: mols, qrdecomp, cholesky, inv, invpd, eigen, det, psdroot, svd
- Dimensions for smaller cases:
 - non-square: $5000 \times 40 \ (m = k = 5000, n = 40)$
 - square: $500 \times 500 \ (m = n = k = 500)$
- Dimensions for bigger cases:
 - non-square: $5000 \times 200 \ (m = k = 5000, n = 200)$
 - square: $1500 \times 1500 \ (m = n = k = 1500)$

dgemm efficiency (pure BLAS Level 3) LAPACK/BLAS efficiency Real econometrics efficiency

gretl's packages test

Marcin Błażejowski Which C compiler and BLAS/LAPACK library...

• • = • • = •

dgemm efficiency (pure BLAS Level 3) LAPACK/BLAS efficiency Real econometrics efficiency

gretl's packages test

• Dimensions for smaller cases:

• Dimensions for bigger cases:

dgemm efficiency (pure BLAS Level 3) LAPACK/BLAS efficiency Real econometrics efficiency

gretl's packages test

- Dimensions for smaller cases:
 - DCC: $528 \times 3 \ (m = k = 528, n = 3)$
 - StrucTiSM: $731 \times 4 \ (m = k = 731, n = 4)$
 - BMA: $72 \times 14 \ (m = k = 72, n = 14)$
- Dimensions for bigger cases:

dgemm efficiency (pure BLAS Level 3) LAPACK/BLAS efficiency Real econometrics efficiency

gretl's packages test

- Dimensions for smaller cases:
 - DCC: $528 \times 3 \ (m = k = 528, n = 3)$
 - StrucTiSM: $731 \times 4 \; (m=k=731, n=4)$
 - BMA: $72 \times 14 \ (m = k = 72, n = 14)$
- Dimensions for bigger cases:
 - DCC: $528 \times 4 \ (m = k = 528, n = 4)$
 - StrucTiSM: $731 \times 8 \ (m = k = 731, n = 8)$
 - BMA: $72 \times 41 \ (m = k = 72, n = 41)$





(a) AMD

(b) Intel

イロト イヨト イヨト イヨト

Figure: Results of matrix_perf_1_1 experiment: overall

setup: m = 128, n = 128, k = 128...2014





m=128, n=128, k=128. 2014 - general blas efficiency in Glops. Ibrary specific



setup: m = 128, n = 128, k = 128...2014

イロト イヨト イヨト イヨト

э




Figure: Results of matrix_perf_1_1 experiment: compiler specific

setup: m = 128, n = 128, k = 128...2014





(a) AMD

(b) Intel

< ロ > < 同 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ >

Figure: Results of matrix_perf_1_2 experiment: overall

setup: $m = 128 \dots 2048, n = 128 \dots 2048, k = 128$





m=128.2048, n=128.2048, k=128 - general bias efficiency in Gilops: library specific

イロト イヨト イヨト イヨト

Figure: Results of matrix_perf_1_2 experiment: library specific

setup: $m = 128 \dots 2048, n = 128 \dots 2048, k = 128$





Figure: Results of matrix_perf_1_2 experiment: compiler specific

setup: $m = 128 \dots 2048, n = 128 \dots 2048, k = 128$

(日) (周) (日) (日)





(a) AMD

(b) Intel

イロト イヨト イヨト イヨト

Figure: Results of matrix_perf_1_3 experiment: overall

setup: $m = 128 \dots 2048, n = 128 \dots 2048, k = 128 \dots 2048$





m=128.2048, n=128.2048, k=128.2048 - general bias efficiency in Gflops: library specific

(b) Inter

イロト イヨト イヨト

m=128.2048, n=128.2048, k=128.2048 - general blas efficiency in Gflops: library specific

Figure: Results of matrix_perf_1_3 experiment: library specific

setup: $m = 128 \dots 2048, n = 128 \dots 2048, k = 128 \dots 2048$





Figure: Results of matrix_perf_1_3 experiment: compiler specific

setup: $m = 128 \dots 2048, n = 128 \dots 2048, k = 128 \dots 2048$

- 4 回 ト - 4 回 ト





Figure: Results of matrix_perf_2_1 experiment: overall

setup:
$$m = 8 \dots 4096, n = 8, k = 8$$

Marcin Błażejowski Which C compiler and BLAS/LAPACK library...

イロト イヨト イヨト イヨト





Figure: Results of matrix_perf_2_1 experiment: library specific

setup: $m = 8 \dots 4096, n = 8, k = 8$

・ロト ・ 同 ト ・ ヨ ト ・ ヨ ト



m=8.4096, n=8, k=8 - general blas efficiency in Glops: compiler specific



m=8.4096, n=8, k=8 - general blas efficiency in Gflops: compiler specific

Figure: Results of matrix_perf_2_1 experiment: compiler specific

setup: $m = 8 \dots 4096, n = 8, k = 8$

(日) (周) (日) (日)





(a) AMD

(b) Intel

< ロ > < 同 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ >

Figure: Results of matrix_perf_2_2 experiment: overall

setup: $m = 10 \dots 5120, n = 2, k = 1000$





Figure: Results of matrix_perf_2_2 experiment: library specific

setup: $m = 10 \dots 5120, n = 2, k = 1000$

・ロト ・ 母 ト ・ ヨ ト ・ ヨ ト





Figure: Results of matrix_perf_2_2 experiment: compiler specific

setup: $m = 10 \dots 5120, n = 2, k = 1000$

・ロト ・ 同 ト ・ ヨ ト ・ ヨ ト





Figure: Results of matrix_perf_2_3 experiment: overall

setup: $m = 10 \dots 320, n = 10, k = 1000$

< ロ > < 同 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ >





(b) Intel

イロト イヨト イヨト

Figure: Results of matrix perf 2 3 experiment: library specific

setup: $m = 10 \dots 320, n = 10, k = 1000$





Figure: Results of matrix_perf_2_3 experiment: compiler specific

setup: $m = 10 \dots 320, n = 10, k = 1000$

Image: A = A = A

→





Figure: Results of $algebra_perf_0_1$ experiment: overall

setup: m = 5000, n = 40 or m = n = 500 (non-threaded)

- 4 回 ト - 4 回 ト - 4 回





bigger matrixes, non-threaded case - lapack/blas efficiency in seconds: library specific

bigger matrixes, non-threaded case - lapack/blas efficiency in seconds: library specific

Figure: Results of algebra_perf_0_1 experiment: library specific

setup: m = 5000, n = 40 or m = n = 500 (non-threaded)





sigger matrixes, non-threaded case - lapack/blas efficiency in seconds: compiler specific

bigger matrixes, non-threaded case - lapack/blas efficiency in seconds: compiler specific

Figure: Results of algebra_perf_0_1 experiment: compiler specific

setup: m = 5000, n = 40 or m = n = 500 (non-threaded)

► < Ξ ►</p>

- - E





Figure: Results of algebra_perf_0_2 experiment: overall setup: m = 5000, n = 200 or m = n = 1500 (non-threaded)

(人間) トイヨト イヨト





matrixes, non-threaded case - lapack/blas efficiency in seconds: library specific

bigger matrixes, non-threaded case - lapack/blas efficiency in seconds: library specific

Figure: Results of algebra_perf_0_2 experiment: library specific

setup: m = 5000, n = 200 or m = n = 1500 (non-threaded)





bigger matrixes, non-threaded case - lapack/blas efficiency in seconds: compiler specific

bigger matrixes, non-threaded case - lapack/blas efficiency in seconds: compiler specific

Figure: Results of algebra_perf_0_2 experiment: compiler specific

setup: m = 5000, n = 200 or m = n = 1500 (non-threaded)

 Environment
 dgemm efficiency (pure BLAS Level 3)

 Compilation
 LAPACK/BLAS efficiency

 Simulation scenarios
 Real econometrics efficiency

 Results
 Overall performance



Figure: Results of algebra_perf_1_1 experiment: overall

setup: m = 5000, n = 40 or m = n = 500 (threaded)

- 4 回 ト - 4 回 ト





Figure: Results of algebra_perf_1_1 experiment: library specific

setup: m = 5000, n = 40 or m = n = 500 (threaded)

(4月) (1日) (日)





Figure: Results of algebra_perf_1_1 experiment: compiler specific

setup: m = 5000, n = 40 or m = n = 500 (threaded)

- - E





Figure: Results of algebra_perf_1_2 experiment: overall

setup: m = 5000, n = 200 or m = n = 1500 (threaded)

(人間) トイヨト イヨト





Figure: Results of algebra_perf_1_2 experiment: library specific

setup: m = 5000, n = 200 or m = n = 1500 (threaded)





smaller matrixes, threaded case - lapack/blas efficiency in seconds: compiler specific

smaller matrixes, threaded case - lapack/blas efficiency in seconds: compiler specific

Figure: Results of algebra_perf_1_2 experiment: compiler specific

setup: m = 5000, n = 200 or m = n = 1500 (threaded)

通 ト イ ヨ ト イ ヨ ト





bigger matrixes, non-threaded case - general lapack/blas efficiency in seconds



- 4 回 ト - 4 回 ト

bigger matrixes, non-threaded case - general lapack/blas efficiency in seconds

Figure: Results for selected packages experiment: overall setup: smaller cases (non-threaded)





bigger matrixes, non-threaded case - lapack/blas efficiency in seconds: library specific

bigger matrixes, non-threaded case - lapack/blas efficiency in seconds: library specific



- 4 回 ト - 4 回 ト





Figure: Results for selected packages experiment: compiler specific setup: smaller cases (non-threaded)

- 4 回 ト - 4 回 ト





bigger matrixes, non-threaded case - general lapack/blas efficiency in seconds



(b) Intel

- 4 回 ト - 4 回 ト

bigger matrixes, non-threaded case - general lapack/blas efficiency in seconds

Figure: Results for selected packages experiment: overall setup: bigger cases (non-threaded)





bigger matrixes, non-threaded case - lapack/blas efficiency in seconds: library specific

bigger matrixes, non-threaded case - lapack/blas efficiency in seconds: library specific

Figure: Results for selected packages experiment: library specific setup: bigger cases (non-threaded)

- 4 回 ト - 4 回 ト





Figure: Results for selected packages experiment: compiler specific setup: bigger cases (non-threaded)

- 4 回 ト - 4 回 ト - 4 回 ト





(a) AMD

smaller matrixes, threaded case - general lapack/blas efficiency in seconds

(b) Intel

・ロト ・ 同 ト ・ ヨ ト ・ ヨ ト

smaller matrixes, threaded case - general lapack/blas efficiency in seconds

Figure: Results for selected packages experiment: overall setup: smaller cases (threaded)





Figure: Results for selected packages experiment: library specific setup: smaller cases (threaded)

・ロト ・ 同 ト ・ ヨ ト ・ ヨ ト




Figure: Results for selected packages experiment: compiler specific setup: smaller cases (threaded)





smaller matrixes, threaded case - general lapack/blas efficiency in seconds



(b) Intel

- 4 回 ト - 4 回 ト

smaller matrixes, threaded case - general lapack/blas efficiency in seconds

Figure: Results for selected packages experiment: overall setup: bigger cases (threaded)





smaller matrixes, threaded case - lapack/blas efficiency in seconds. Ibrary specific



< ロト < 同ト < ヨト < ヨト

matrixes, threaded case - lapack/blas efficiency in seconds: library specific

Figure: Results for selected packages experiment: library specific

setup: bigger cases (threaded)





smaller matrixes, threaded case - lapack/blas efficiency in seconds: compiler specific

smaller matrixes, threaded case - lapack/blas efficiency in seconds: compiler specific

Figure: Results for selected packages experiment: compiler specific setup: bigger cases (threaded)

イロト イ理ト イヨト イヨト

	dgemm efficiency (pure BLAS Level 3)
Compilation	LAPACK/BLAS efficiency

To calculate overall performance (across all three scenarios) we use *Min-Max Feature Scaling*:

	dgemm efficiency (pure BLAS Level 3)
Compilation	LAPACK/BLAS efficiency
	Real econometrics efficiency
	Overall performance

To calculate overall performance (across all three scenarios) we use *Min-Max Feature Scaling*:

• for results in Gflops:

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}}$$

Environment dgemm efficiency (pure BLAS Level 3) Compilation LAPACK/BLAS efficiency Simulation scenario Results Overall performance

To calculate overall performance (across all three scenarios) we use *Min-Max Feature Scaling*:

• for results in Gflops:

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}}$$

• for results in seconds:

$$x' = \frac{x_{max} - x}{x_{max} - x_{min}}$$





Figure: Overall score based on three experiments: compiler specific





Figure: Overall score based on three experiments: library specific

Marcin Błażejowski Which C compiler and BLAS/LAPACK library...

-

Environment	dgemm efficiency (pure BLAS Level 3)
Compilation	LAPACK/BLAS efficiency
Simulation scenarios Results	

The end

Marcin Błażejowski Which C compiler and BLAS/LAPACK library...