Basic Matrix Library (bml)

Release v2.4.0

bml authors

Apr 03, 2024

CONTENTS:

Issues	Pull Requests	CI	Conda	Documentation	Docker

CONTENTS:

ONE

INTRODUCTION

This website is intended to provide some guidance on how to get and install the bml library. LA-UR number LA-UR-**17-27373**.

The basic matrix library (bml) is a collection of various matrix data formats (for dense and sparse) and their associated algorithms for basic matrix operations. Application programming interfaces (API) are available for both C and FOR-TRAN. The current status of this library allows us to use two different formats for representing matrix data. Currently these formats are: dense, ELLPACK-R, ELLBLOCK, ELLSORT, and CSR. For information on how to use the BML library can be find in BML-API.

MAILING LIST

We are running the following mailing list for discussions on usage and features of the bml library:

- bml
- Subscribe
- Archives

SUPPORTED MATRIX FORMATS

The bml library supports the following matrix formats:

- dense
- ELLPACK-R
- ELLSORT
- ELLBLOCK
- CSR

BINARY PACKAGES

We offer binary packages of the bml library in RPM format thanks to SUSE's OpenBuild Service and for Ubuntu in DEB format.

TESTING IN OUR CI CONTAINER

We are switching our CI tests from Travis-CI to GitHub Actions because Travis-CI is limiting the number of builds for open source projects. Our workflow uses a custom Docker image which comes with the necessary compiler tool chain to build and test the `bml` library. Using `docker` is a convenient and quick way to develop, build, and test the `bml` library.

```
$ ./scripts/run-local-docker-container.sh
latest: Pulling from nicolasbock/bml
2f94e549220a: Already exists
8d8ab0ffcd5e: Pull complete
3fa4d3b6f5b4: Pull complete
4f4fb700ef54: Pull complete
Digest: sha256:18237f909f19896a57c658c93af5e8ed91c9fa596f15021be777a97444a3eaaf
Status: Downloaded newer image for nicolasbock/bml:latest
docker.io/nicolasbock/bml:latest
groups: cannot find name for group ID 1000
I have no name!@3a4ae718ba4f:/bml$
```

Inside the container:

```
I have no name!@6ea3f4937c0d:/bml$ ./build.sh compile
Writing output to /bml/build.log
Running command compile
mkdir: created directory '/bml/build'
mkdir: created directory '/bml/install'
-- CMake version 3.12.1
-- The C compiler identification is GNU 7.5.0
-- The CXX compiler identification is GNU 7.5.0
-- The Fortran compiler identification is GNU 7.5.0
-- Check for working C compiler: /usr/bin/gcc
-- Check for working C compiler: /usr/bin/gcc -- works
```

Alternatively, you can run one of the CI tests by executing e.g.

```
I have no name!@6ea3f4937c0d:/bml$ ./scripts/ci-gcc-10-C-single-real.sh
+++ dirname ./scripts/ci-gcc-10-C-single-real.sh
++ readlink --canonicalize ./scripts/..
+ basedir=/bml
+ export CC=gcc-10
+ CC=gcc-10
+ export CXX=g++-11
+ CXX=g++-11
```

(continues on next page)

```
+ export FC=gfortran-11
+ FC=gfortran-11
```

BUILD INSTRUCTIONS

The bml library is built with CMake. For convenience, we provide a shell script which goes through the necessary motions and builds the library, runs the tests, and installs it (in the `install` directory).

6.1 For a quick installation

We suggest to take a look at the `example_build.sh` script that sets the most important environmental variables needed by `build.sh` script. Change the Variables according to the compilers and architecture. The script can be run just by doing:

```
$ ./scripts/example_build.sh
Writing output to /bml/build.log
Running command configure
mkdir: created directory '/bml/build'
mkdir: created directory '/bml/install'
-- CMake version 3.12.1
-- The C compiler identification is GNU 7.5.0
-- The CXX compiler identification is GNU 7.5.0
-- The Fortran compiler identification is GNU 7.5.0
```

6.2 For a more involved installation

By running:

```
$ ./build.sh install
```

the library will be built in the `build` directory and installed in the `install` directory. In case you change any sources and simply want to rebuild the library, you don't have to run `build.sh` again, but rather

```
$ make -C build
```

The compiled library can be installed by running

```
$ make -C build install
```

The install directory can be modified by running

```
$ CMAKE_INSTALL_PREFIX=/some/path ./build.sh install
```

(which assumes that you are using the bash shell).

To build with GNU compilers, OpenMP, and Intel MKL do the following.

```
$ CC=gcc FC=gfortran \
  BLAS_VENDOR=Intel CMAKE_BUILD_TYPE=Release \
  BML_OPENMP=yes CMAKE_INSTALL_PREFIX=/some/path \
  ./build.sh install
```

To build with MPI, OpenMP, and use Intel MKL do the following.

```
$ CC=mpicc FC=mpif90 \
  BLAS_VENDOR=Intel CMAKE_BUILD_TYPE=Release \
  BML_OPENMP=yes BML_MPI=yes CMAKE_INSTALL_PREFIX=/some/path \
  ./build.sh install
```

6.3 Prerequisites

In order to build the library, the following tools need to be installed:

- `gcc` with Fortran support
- `>=cmake-2.8.8`
- `>=python-2.7`
- `>=OpenMP-3.1` (i.e. `>=gcc-4.7`)

6.4 If the build fails

In case the build fails for some reason, please contact the developers by opening an issue on GitHub (https://github.com/lanl/bml/issues) and attach the files

```
build/CMakeFiles/CMakeOutput.log
build/CMakeFiles/CMakeError.log
```

DEVELOPER SUGGESTED WORKFLOW

Our main development happens on the `master` branch and is continuously verified for correctness. If you would like to contribute with your work to the bml project, please follow the instructions at the GitHub help page "About pull requests". To summarize:

- Fork the project on github

- Clone that forked repository

- Create a branch in it

- Commit any changes to the branch

- Push the branch to your forked repository

- Go to https://github.com/lanl/bml and click on 'Create Pull Request'

During the review process you might want to update your pull request. Please add commits or `amend` your existing commits as necessary. If you amend any commits you need to add the `--force-with-lease` option to the `git push` command. Please make sure that your pull request contains only one logical change (see "Structural split of change" for further details.

CODING STYLE

Please indent your C code using

```
$ indent -gnu -nut -i4 -bli0 -cli4 -ppi0 -cbi0 -npcs -bfda
```

You can use the script `indent.sh` to indent all C code.

HELPFUL DEVELOPER RESOURCES

9.1 Optimizations

For low level optimization work it is useful to understand what assembly code the compiler generates. For example, to verify that the compiler vectorizes the loop in the following example:

```c
void double_array(float a[8]) {
  a = __builtin_assume_aligned(a, 64);
  for (int i = 0; i < 8; i++) {
    a[i] *= 2;
  }
}
```

we can build the source with

```
gcc -S -O3 -fverbose-asm test.c
```

and analyze the generated assembly code,

```
# test.c:8:     a[i] *= 2;
  movaps      (%rdi), %xmm0    # MEM <vector(4) float> [(float *)a_9], vect__5.8
  addps       %xmm0, %xmm0     #, vect__5.8
  movaps      %xmm0, (%rdi)    # vect__5.8, MEM <vector(4) float> [(float *)a_9]
  movaps      16(%rdi), %xmm0  # MEM <vector(4) float> [(float *)a_9 + 16B], vect__5.8
  addps       %xmm0, %xmm0     #, vect__5.8
  movaps      %xmm0, 16(%rdi)  # vect__5.8, MEM <vector(4) float> [(float *)a_9 + 16B]
```

The aligned memory access, *movaps*, moving 4 (aligned packed single-precision) *float* values into *%xmm0*, and the subsequent *addps* instruction show that the compiler fully vectorized the loop.

Note that the Compiler Explorer provides an alternative that does not require local compilations, see https://godbolt.org/z/ejEdqKa6Y.

CITING

If you find this library useful, we encourage you to cite us. Our project has a citable DOI:

with the following `bibtex` snipped:

```
@misc{bml,
  author        = {Nicolas Bock and
                   Susan Mniszewski and
                   Bálint Aradi and
                   Michael Wall and
                   Christian F. A. Negre
                   Jamal Mohd-Yusof and
                   Anders N. M. Niklasson},
  title         = {qmmd/bml v2.1.2},
  month         = feb,
  year          = 2022,
  doi           = {10.5281/zenodo.5570404},
  url           = {https://doi.org/10.5281/zenodo.5570404}
}
```

Another citation source is the following journal article (DOI: 10.1007/s11227-018-2533-0):

```
@article{bock2018basic,
  title     = {The basic matrix library (BML) for quantum chemistry},
  author    = {Bock, Nicolas and
               Negre, Christian FA and
               Mniszewski, Susan M and
               Mohd-Yusof, Jamaludin and
               Aradi, B{\'a}lint and
               Fattebert, Jean-Luc and
               Osei-Kuffuor, Daniel and
               Germann, Timothy C and
               Niklasson, Anders MN},
  journal   = {The Journal of Supercomputing},
  volume    = {74},
  number    = {11},
  pages     = {6201--6219},
  year      = {2018},
  publisher = {Springer}
}
```

AUTHORS

The core developers of the bml in alphabetical order:

- Christian Negre <cnegre@lanl.gov>

- Nicolas Bock <nicolasbock@gmail.com>

- Susan M. Mniszewski <smm@lanl.gov>

CONTRIBUTORS

- Adedoyin Adetokunbo <aadedoyin@lanl.gov>

- Bálint Aradi <aradi@uni-bremen.de>

- Daniel Osei-Kuffuor <oseikuffuor1@llnl.gov>

- Jamaludin Mohd-Yusof <jamal@lanl.gov>

- Jean-Luc Fattebert <fattebertj@ornl.gov>

- Mike Wall <mewall@lanl.gov>

LICENSE

The bml library is licensed under the BSD 3-clause license.

LA-CC

NOTICE OF OSS COPYRIGHT ASSERTION:

LANS has asserted copyright on the software package entitled *Basic Matrix Library (bml), Version 0.x (C16006)*.

14.1 ABSTRACT

The basic matrix library (bml) is a collection of various matrix data formats (for dense and sparse) and their associated algorithms for basic matrix operations.

This code is unclassified and has been assigned LA-CC-**15-093**. Los Alamos National Laboratory's Export Control Team made an in-house determination that this software is controlled under Department of Commerce regulations and the Export Control Classification Number (ECCN) **EAR99**. The export control review is attached.

The developers intend to distribute this software package under the OSI Certified **BSD 3-Clause License** (http://www.opensource.org/licenses/BSD-3-Clause)

This code was developed using funding from:

- Basic Energy Sciences (LANL2014E8AN) and the Laboratory Directed Research and Development Program of Los Alamos National Laboratory. To tests these developments we used resources provided by the Los Alamos National Laboratory Institutional Computing Program, which is supported by the U.S. Department of Energy National Nuclear Security Administration

- Exascale Computing Project (17-SC-20-SC), a collaborative effort of two U.S. Department of Energy organizations (Office of Science and the National Nuclear Security Administration) responsible for the planning and preparation of a capable exascale ecosystem, including software, applications, hardware, advanced system engineering, and early testbed platforms, in support of the nation's exascale computing imperative.

Larry Kwei, LAFO Program Manager, has granted his concurrence to asserting copyright and then distributing the **Basic Matrix Library (bml), Version 0.x** code using an open source software license. See attached memo.

LANS acknowledges that it will comply with the DOE OSS policy as follows:

1. submit form DOE F 241.4 to the Energy Science and Technology Software Center (ESTSC),

2. provide the unique URL on the form for ESTSC to distribute, and

3. maintain an OSS Record available for inspection by DOE.

Following is a table briefly summarizes information for this software package:

CODE NAME	**Basic Matrix Library (bml), Version 0.x (C16006)**
Classification Review Number	**LA-CC-15-093**
Export Control Classification Number (ECCN)	**EAR99**
B&R Code	**YN0100000**

FIFTEEN

INDICES AND TABLES

- genindex
- modindex
- search