



MAX PLANCK INSTITUTE
FOR DYNAMICS OF COMPLEX
TECHNICAL SYSTEMS
MAGDEBURG



COMPUTATIONAL METHODS IN
SYSTEMS AND CONTROL THEORY

FlexiBLAS

A BLAS and LAPACK wrapper library with runtime
exchangable backends

Martin Köhler

March 29, 2020

EasyBuild tech talks III: FlexiBLAS



- 2010: diploma in mathematics from TU Chemnitz

- 2010: diploma in mathematics from TU Chemnitz
- since 2011: MPI Magdeburg – HPC administrator and researcher
 - 106 node Westmere/Magny Cours Cluster from 2011 to 2018
 - 122 node Skylake Cluster from 2018 to today
 - various specialized system (IBM POWER8, POWER9) with GPUs
 - continuous integration services for scientific software
 - **research:** efficient algorithms for generalized eigenvalue problems and matrix equations
 - **software development:** production ready code for my research and whatever helps in my workflow

- 2010: diploma in mathematics from TU Chemnitz
- since 2011: MPI Magdeburg – HPC administrator and researcher
 - 106 node Westmere/Magny Cours Cluster from 2011 to 2018
 - 122 node Skylake Cluster from 2018 to today
 - various specialized system (IBM POWER8, POWER9) with GPUs
 - continuous integration services for scientific software
 - **research:** efficient algorithms for generalized eigenvalue problems and matrix equations
 - **software development:** production ready code for my research and whatever helps in my workflow
- 2021: submitted my PhD thesis at OvGU Magdeburg



- 2010: diploma in mathematics from TU Chemnitz
- since 2011: MPI Magdeburg – HPC administrator and researcher
 - 106 node Westmere/Magny Cours Cluster from 2011 to 2018
 - 122 node Skylake Cluster from 2018 to today
 - various specialized system (IBM POWER8, POWER9) with GPUs
 - continuous integration services for scientific software
 - **research:** efficient algorithms for generalized eigenvalue problems and matrix equations
 - **software development:** production ready code for my research and whatever helps in my workflow
- 2021: submitted my PhD thesis at OvGU Magdeburg
- (co-) author of several software packages: FlexiBLAS, M.E.S.S., C-M.E.S.S., libUFget, ...



- 2010: diploma in mathematics from TU Chemnitz
- since 2011: MPI Magdeburg – HPC administrator and researcher
 - 106 node Westmere/Magny Cours Cluster from 2011 to 2018
 - 122 node Skylake Cluster from 2018 to today
 - various specialized system (IBM POWER8, POWER9) with GPUs
 - continuous integration services for scientific software
 - **research:** efficient algorithms for generalized eigenvalue problems and matrix equations
 - **software development:** production ready code for my research and whatever helps in my workflow
- 2021: submitted my PhD thesis at OvGU Magdeburg
- (co-) author of several software packages: FlexiBLAS, M.E.S.S., C-M.E.S.S., libUFget, ...
- maintainer of grupdate-ng (maintenance fork of grupdate)



- 2010: diploma in mathematics from TU Chemnitz
- since 2011: MPI Magdeburg – HPC administrator and researcher
 - 106 node Westmere/Magny Cours Cluster from 2011 to 2018
 - 122 node Skylake Cluster from 2018 to today
 - various specialized system (IBM POWER8, POWER9) with GPUs
 - continuous integration services for scientific software
 - **research:** efficient algorithms for generalized eigenvalue problems and matrix equations
 - **software development:** production ready code for my research and whatever helps in my workflow
- 2021: submitted my PhD thesis at OvGU Magdeburg
- (co-) author of several software packages: FlexiBLAS, M.E.S.S, C-M.E.S.S, libUFget, ...
- maintainer of qrupdate-ng (maintenance fork of qrupdate)
- Github: **grisuthedragon**



- **Jens Saak** – testing, ideas, was annoyed by recompiling his whole Gentoo system when changing the system-wide BLAS library.
- **Christian Himpe** – testing, ideas, documentation, seeks the fastest Octave software stack.
Twitter: @modelreduction
- **Jörn Papenbroock** – student assistant, coding work, now in industry
- and of course **myself**

Basic Linear Algebra Subprograms (BLAS)

“The BLAS (Basic Linear Algebra Subprograms) are routines that provide standard building blocks for performing basic vector and 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.”¹

¹From: <http://www.netlib.org/blas/faq.html> – What and where are the BLAS?

Let α, β be scalars, x, y be vectors, A, B, C be matrices.

level	included operations	data	flops
1	$\alpha x, \alpha x + y, x^*y, \ x\ _2, \ x\ _1, \ x\ _\infty$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
2	$\alpha Ax + \beta y, \alpha A^*x + \beta y,$ $A + \alpha xy^*, A + \alpha xx^*,$ $A + \alpha xy^* + \beta yx^*$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$
3	$\alpha AB + \beta C, \alpha AB^* + \beta C, \alpha A^*B^* + \beta C, \alpha AA^* + \beta C,$ rank k updates $\alpha A^*B + \beta C,$ $\alpha B^*A + \beta C$ rank $2k$ updates	$\mathcal{O}(n^2)$	$\mathcal{O}(n^3)$



What is BLAS?

Some important BLAS implementations

Open Source

- NetLib BLAS: <http://www.netlib.org/blas/>
- OpenBLAS: <http://www.openblas.net/>
- Automatically Tuned Linear Algebra Software (ATLAS):
<http://math-atlas.sourceforge.net/>
- BLIS (BLAS-like Library Instantiation Software Framework):
<https://github.com/flame/blis>



What is BLAS?

Some important BLAS implementations

Open Source

- NetLib BLAS: <http://www.netlib.org/blas/>
- OpenBLAS: <http://www.openblas.net/>
- Automatically Tuned Linear Algebra Software (ATLAS):
<http://math-atlas.sourceforge.net/>
- BLIS (BLAS-like Library Instantiation Software Framework):
<https://github.com/flame/blis>

Hardware Vendor Implementations

- Intel[®] Math kernel library (MKL):
<http://software.intel.com/en-us/intel-mkl/>
- AMD Core Math Library (ACML): ... discontinued
- Apple Accelerate, IBM ESSL, ARM Performance Libraries ...

Open Source

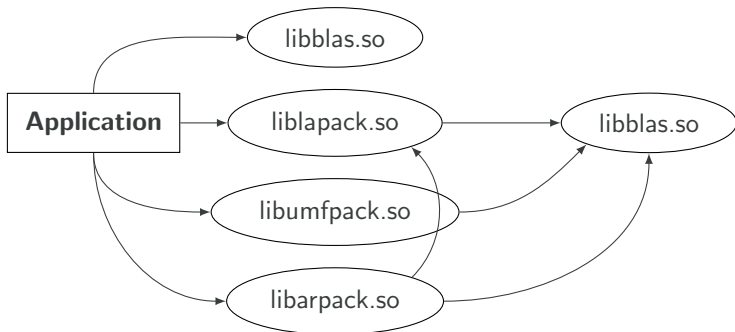
- NetLib BLAS: <http://www.netlib.org/blas/>
- OpenBLAS: <http://www.openblas.net/>
- Auto
- BLIS

Why do we need yet another BLAS library?

Hardware

- Intel
<http://software.intel.com/en-us/intel-mkl/>
- AMD Core Math Library (ACML): ... discontinued
- Apple Accelerate, IBM ESSL, ARM Performance Libraries ...

Example application:



Compiled with:

```
$ gcc -o application app.o -larpack -lumfpack ↔
-llapack -lblas
```

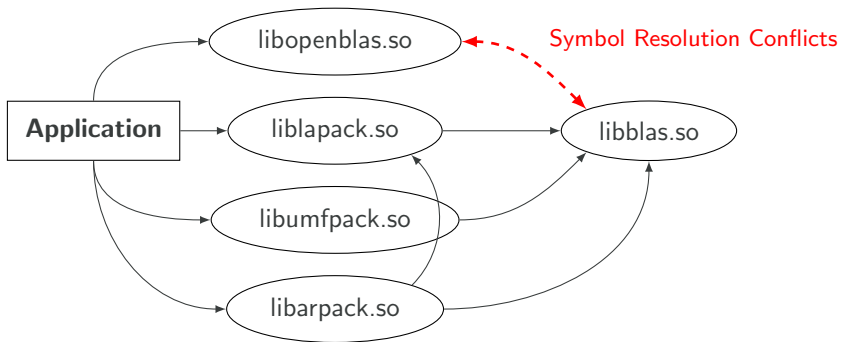
Example application:

```
$ ldd ./application
linux-vdso.so.1 => (0x00007ffc2d1de000)
libarpack.so.2.1.0 => /.../libarpack.so.2.1.0
libumfpack.so.5.7.1 => /.../libumfpack.so.5.7.1
liblapack.so.3 => /.../liblapack.so.3
libblas.so.3 => /.../libblas.so.3
libc.so.6 => /.../libc.so.6
...
```

Compiled with:

```
$ gcc -o application app.o -larpack -lumfpack ↔
    -llapack -lblas
```

Example application:



Compiled with:

```
$ gcc -o application app.o -larpack -lumfpack ←
    -llapack -lopenblas
```


Example application:

```
$ ldd ./application
linux-vdso.so.1 => (0x00007ffc2d1de000)
libarpack.so.2.1.0 => /.../libarpack.so.2.1.0
libumfpack.so.5.7.1 => /.../libumfpack.so.5.7.1
liblapack.so.3 => /.../liblapack.so.3
libopenblas.so.0 => /.../libopenblas.so.0
libc.so.6 => /.../libc.so.6
libm.so.6 => /.../libm.so.6
libblas.so.3 => /.../libblas.so.3
...
```

Compiled with:

```
$ gcc -o application app.o -larpack -lumfpack ←
    -llapack -lopenblas
```

ts



Why do we need yet another BLAS library?

Linker Problems: Existing Solutions

- `LD_LIBRARY_PATH / LD_PRELOAD`
only applicable for single file implementations
(i.e. **NOT** Intel[®] MKL, or ATLAS)
- **static libraries**
drastically increased binary sizes, often complicated linking, painful in large projects
- `update-alternatives` (Debian/Ubuntu/Suse)
requires super-user privileges and has similar restrictions as `LD_LIBRARY_PATH / LD_PRELOAD`
- `eselect / pkg-config` (Gentoo)
requires super-user privileges and switches at **build-time only**
- ***BSD ports/pkgsrc/dports**
Links against `libblas.so` if already installed otherwise installs some BLAS implementation depending on the maintainer.

gfortran vs g77/intel interface style

- **different calling sequences:**

gfortran and g77/f2c/intel return complex numbers as additional function parameters.

- **affected routines:** `zdotc`, `zdotu`, `cdotc`, `cdotu` (level 1)

gfortran vs g77/intel interface style

- **different calling sequences:**

gfortran and g77/f2c/intel return complex numbers as additional function parameters.

- **affected routines:** zdotc, zdotu, cdotc, cdotu (level 1)

missing routines

- Routines `sc/dzabs1` are missing in ATLAS and derived implementations, such as Apple Accelerate / AMD ACML.
- IBM ESSL make only a few LAPACK symbols available.

gfortran vs g77/intel interface style

- **different calling sequences:**

gfortran and g77/f2c/intel return complex numbers as additional function parameters.

- **affected routines:** zdotc, zdotu, cdotc, cdotu (level 1)

missing routines

- Routines `sc/dzabs1` are missing in ATLAS and derived implementations, such as Apple Accelerate / AMD ACML.
- IBM ESSL make only a few LAPACK symbols available.

auxiliary routines

- Intel[®] MKL and OpenBLAS extend the BLAS routine set by:
`xAXPBY`, `xOMATCOPY`, ...



dependency detection problems

Correct/reliable detection of alternative BLAS implementations not guaranteed for many software packages:

- faulty autotools scripts,
- old CMake versions,
- hard-coded library names,
- non-standard library locations.



- Initial idea: Summer 2013 after struggling with the linking issue
- First release: December 2013 (BLAS and CBLAS only)
- 2017: Version 2.x wraps LAPACK, switching the BLAS library from the inside of an application
- 2020: Version 3.0.x hooks can be installed around BLAS calls
- October 2020: default BLAS in Fedora 33+ (thanks to Iñaki Ucar)
- Provides interfaces for BLAS, CBLAS, and LAPACK.
- Automatic code generation for the wrappers
- API interface for GNU Octave
- API interface for R (thanks to Iñaki Ucar²)
- **Latest version: 3.0.4 – October 22nd, 2020**
- **License:** GPLv3+ with linking exception

²Twitter/Github: @Enchufa2



How does it work?

General Approach

Long Story Short

We employ a plugin-like framework on top of the POSIX features for dynamic loading of shared libraries at runtime.

Long Story Short

We employ a plugin-like framework on top of the POSIX features for dynamic loading of shared libraries at runtime.

POSIX.1 2001 `dl*`-family

`dlopen` add a shared library and its dynamic dependencies to the current address space.

`dlsym` search for symbols in the current address space beginning in the handle retrieved by `dlopen`.

`dlclose` close a previously opened shared library if no other references to the library exist.

`dlerror` provide human readable error messages.

dlopen based issues to solve

1. dlopen only integrates selected parts of the library:
Each required BLAS call needs to be initialized separately.
2. Dynamically (runtime) loaded symbols can not be resolved while linking a program.
3. dlopen only loads a single file:
Multi-file implementations require additional treatment.

`__attribute__((constructor))`

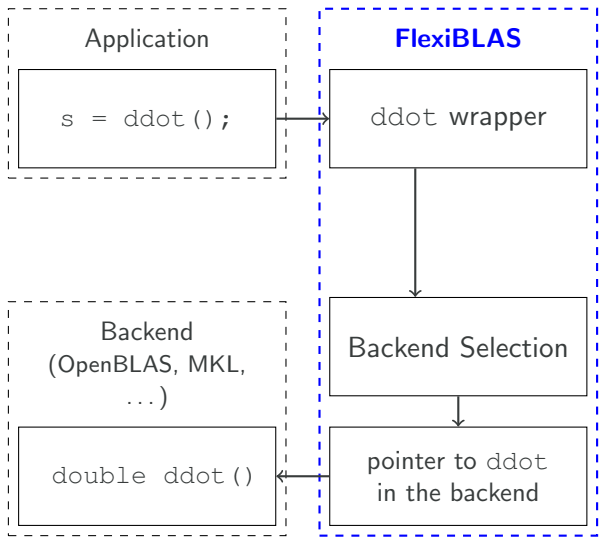
- automatically executed before the program starts.
- replaces deprecated `_init()`.
- Here used to read configuration and explicitly resolve all BLAS-routines to make sure they get loaded by `dlopen` as an initialization stage.

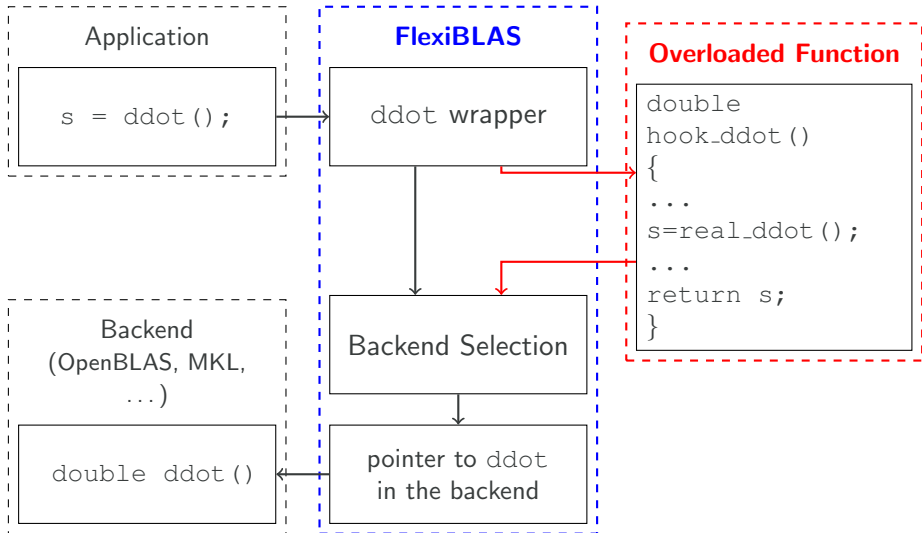
`__attribute__((constructor))`

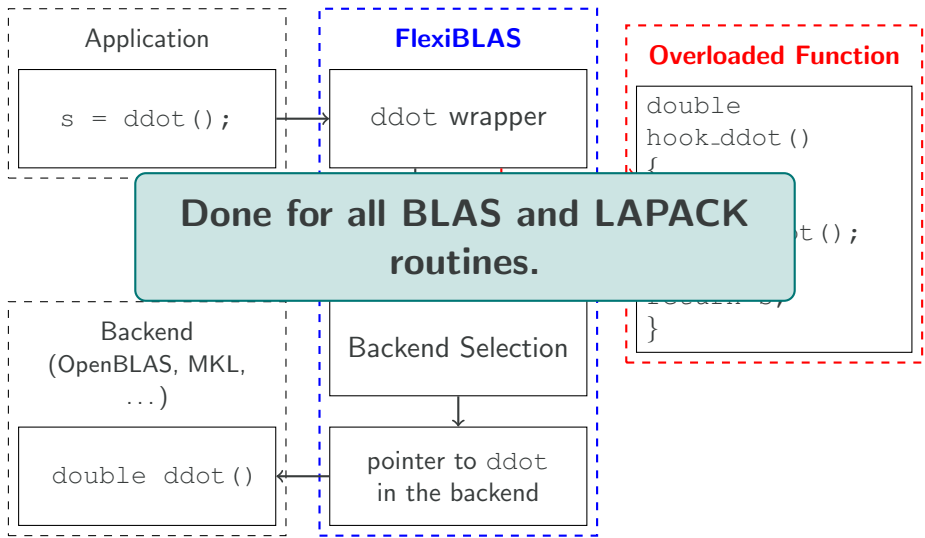
- automatically executed before the program starts.
- replaces deprecated `_init()`.
- Here used to read configuration and explicitly resolve all BLAS-routines to make sure they get loaded by `dlopen` as an initialization stage.

`__attribute__((destructor))`

- automatically executed after the main program exits.
- replaces deprecated `_fini()`.
- Here used to cleanly close the loaded shared library and potentially print profiling data.









How does it work?

Wrapper Functions

Python based code-gen

- NumPy's `f2py` module allows to parse f77/f90 function headers.
- Extracted function headers are translated into Fortran-ABI compatible C functions containing the wrapper.

Python based code-gen

- NumPy's `f2py` module allows to parse f77/f90 function headers.
- Extracted function headers are translated into Fortran-ABI compatible C functions containing the wrapper.

From

```
SUBROUTINE DAXPY(N, ALPHA, X, INCX, Y, INCY)
```

we obtain

```
void daxpy_(Int *N, double *ALPHA, double *X,  
            Int *INCX, double *Y, Int * INCY) {  
    ...  
    fncall_daxpy = backend->daxpy.ffunction;  
    fncall_daxpy(N, ALPHA, X, INCX, Y, INCY);  
    ... }  
}
```



How does it work?

Multi-file BLAS treatment

Remaining Question

How do we treat BLAS libraries consisting of multiple files (e.g. MKL and some versions of ATLAS), when the `dl*`-family can only use single file shared object libraries?

Remaining Question

How do we treat BLAS libraries consisting of multiple files (e.g. MKL and some versions of ATLAS), when the `dl*`-family can only use single file shared object libraries?

Simple trick

Place an additional surrogate library between FlexiBLAS and, e.g., MKL that references all necessary symbols in MKL and behaves like a Netlib-BLAS interface from the view of the dynamic linker.

Intel MKL provides the *mkl-builder* makefile to create such dummy libraries containing arbitrary BLAS symbols.



Remaining Question

How do we treat BLAS libraries consisting of multiple files (e.g. MKL and some versions of ATLAS), when the `dl*`-family can only use single file shared object libraries?

Simple trick

Place an additional surrogate library between FlexiBLAS and, e.g., MKL that references all necessary symbols in MKL and behaves like a Netlib-BLAS interface from the view of the dynamic linker.

Intel MKL provides the *mkl-builder* makefile to create such dummy libraries containing arbitrary BLAS symbols.

Never use the `libmkl_rt.so` library!



How is it used?

We provide a tool that closely follows Gentoo's `eselect` syntax.
To check for backends, do

```
flexiblas list
```

To select the active backend, use

```
flexiblas default BLAS_BACKEND_NAME
```

We provide a tool that closely follows Gentoo's `eselect` syntax.
To check for backends, do

```
flexiblas list
```

To select the active backend, use

```
flexiblas default BLAS_BACKEND_NAME
```

Alternatively we use an environment variable as in:

```
export FLEXIBLAS=/usr/lib/libopenblas.so
```

or

```
export FLEXIBLAS=ATLAS
```

We provide a tool that closely follows Gentoo's `eselect` syntax.
To check for backends, do

```
flexiblas list
```

To select the active backend, use

```
flexiblas default BLAS_BACKEND_NAME
```

Alternatively we use an environment variable as in:

```
export FLEXIBLAS=/usr/lib/libopenblas.so
```

or

```
export FLEXIBLAS=ATLAS
```

config files: `/etc/flexiblasrc`, `~/.flexiblasrc`, and
`~/.flexiblasrc.$(hostname)`

Other environment variables to control the behavior:

`FLEXIBLAS_VERBOSE` Turn on additional debug outputs.

`FLEXIBLAS_NOLAPACK` Do not load LAPACK from the backend. Only the internal NETLIB version is used.

`FLEXIBLAS_COLOR_OUTPUT` Switch the color output in verbose mode.

`FLEXIBLAS_CONFIG` specify a different `flexiblasrc` file

`FLEXIBLAS_LIBRARY_PATH` specify additional library search paths

Test Setup

- Ubuntu 20.04, gcc 9.3, Intel Core i5-8500, OpenBLAS/OpenMP
- Measure the shortest successful return path of a BLAS routine, i.e. size zero inputs, with `rdtsc`:

```
__asm__ __volatile__ ("rdtsc" : "=a"(los), "=d"(his));  
daxpy(&(int){0}, &(double){1.0}, NULL, &(int){1},  
      NULL, &(int){1});  
__asm__ __volatile__ ("rdtsc" : "=a"(loe), "=d"(hie));
```

- average over 100 000 000 runs

Test Setup

- Ubuntu 20.04, gcc 9.3, Intel Core i5-8500, OpenBLAS/OpenMP
- Measure the shortest successful return path of a BLAS routine, i.e. size zero inputs, with `rdtsc`:

```
__asm__ __volatile__ ("rdtsc" : "=a"(los), "=d"(his));
daxpy(&(int){0}, &(double){1.0}, NULL, &(int){1},
      NULL, &(int){1});
__asm__ __volatile__ ("rdtsc" : "=a"(loe), "=d"(hie));
```

- average over 100 000 000 runs

RDTSC ticks until return	Linked with		Difference
	OpenBLAS/OpenMP	FlexiBLAS	
DAXPY	19.03	24.19	5.16
DGEMV	22.92	37.03	14.11
DGEMM	28.40	44.47	16.07

All BLAS and LAPACK routines can be overloaded:

- build profiling frameworks,
- dynamically offload them to accelerators, (first experiments done)
- introduce faulty behavior for debugging purpose,
- several hooks can be chained,
- original BLAS implementation is callable by a separate pointer.

Example - `DASUM` with perturbed output

```
double hook_dasum(Int *N, double *X, Int *INCX) {  
    double res = flexiblas_chain_dasum(N, X, INCX);  
    return res + ((*N)*2.2e-16);  
}
```

Functionality:

- measures cumulative the runtime of each BLAS call
- counts the number of calls to each BLAS routine

Usage:

```
FLEXIBLAS_HOOK=libflexiblas_hook_profile.so ./yourapp
```

or

```
FLEXIBLAS_HOOK=PROFILE ./yourapp
```

```

function [x] = conjgrad(A, b, x)
    r = b - A * x;
    p = r;
    rsold = r' * r;
    for i = 1:length(b)
        Ap = A * p;
        alpha = rsold / (p' * Ap);
        x = x + alpha * p;
        r = r - alpha * Ap;
        rsnew = r' * r;
        if sqrt(rsnew) < 1e-10
            break;
        end;
        p = r + (rsnew / rsold) * p;
        rsold = rsnew;
    end
end

A = full(sprandsym(1000,1.0));
b = A*ones(1000,1);
x = conjgrad(A,b,zeros(1000,1));
norm(A*x-b)/norm(b)

```

```
function [x] = conjgrad(A, b, x)
    r = b - A * x;
    p = r;
    rsold = r' * r;
    for i = 1:length(b)
        Ap = A * p;
        alpha = rsold / (p' * Ap);
        x = x + alpha * p;
        r = r - alpha * Ap;
        rsnew = r' * r;
        if sqrt(rsnew) < 1e-10
            break;
        end;
        p = r + (rsnew / rsold) * p;
        rsold = rsnew;
    end
end

A = full(sprandsym(1000, 1.0));
b = A*ones(1000, 1);
x = conjgrad(A, b, zeros(1000, 1));
norm(A*x-b) / norm(b)
```

Profiling:

Subroutine	# Calls	acc. Time
ddot	1000	1.11e-03s
dgemv	1003	4.04e-01s
dsyrk	1001	5.61e-03s
dlamch	5	2.69e-05s

Observations

- Vector addition/scaling/norms not mapped to BLAS.
- Where does the symmetric rank- k update come from?

```
function [x] = conjgrad(A, b, x)
    r = b - A * x;
    p = r;
    rsold = r' * r;
    for i = 1:length(b)
        Ap = A * p;
        alpha = rsold / (p' * Ap);
        x = x + alpha * p;
        r = r - alpha * Ap;
        rsnew = r' * r;
        if sqrt(rsnew) < 1e-10
            break;
        end;
        p = r + (rsnew / rsold) * p;
        rsold = rsnew;
    end
end

A = full(sprandsym(1000,1.0));
b = A*ones(1000,1);
x = conjgrad(A,b,zeros(1000,1));
norm(A*x-b)/norm(b)
```

Inspecting:

DSYRK computes

$$C := \alpha A \cdot A^T + \beta C \quad \text{if trans='N'}$$

or

$$C := \alpha A^T A + \beta C, \quad \text{if trans='T'}$$

with $A \in \mathbb{R}^{n \times k}$ or $A \in \mathbb{R}^{k \times n}$ and $C \in \mathbb{R}^{n \times n}$.

All 1001 DSYRK calls use:

- trans = 'T'
- $n = 1, k = 1000,$
- $\alpha = 1.0,$ and $\beta = 0$

```
function [x] = conjgrad(A, b, x)
    r = b - A * x;
    p = r;
    rsold = r' * r;
    for i = 1:length(b)
        Ap = A * p;
        alpha = rsold / (p' * Ap);
        x = x + alpha * p;
        r = r - alpha * Ap;
        rsnew = r' * r;
        if sqrt(rsnew) < 1e-10
            break;
        end;
        p = r + (rsnew / rsold) * p;
        rsold = rsnew;
    end
end

A = full(sprandsym(1000,1.0));
b = A*ones(1000,1);
x = conjgrad(A,b,zeros(1000,1));
norm(A*x-b)/norm(b)
```

Inspecting:

DSYRK computes

$$C := \alpha A \cdot A^T + \beta C \quad \text{if trans='N'}$$

or

$$C := \alpha A^T A + \beta C, \quad \text{if trans='T'}$$

with $A \in \mathbb{R}^{n \times k}$ or $A \in \mathbb{R}^{k \times n}$ and $C \in \mathbb{R}^{n \times n}$.

All 1001 DSYRK calls use:

- trans = 'T'
- $n = 1, k = 1000,$
- $\alpha = 1.0,$ and $\beta = 0$

→ **Misuse of DSYRK to compute the squared 2-norm of a vector.**

Bug Description:

(OpenBLAS up to version 0.2.20)

The DTRMV routine computes

$$x := \alpha T x$$

with $\alpha \in \mathbb{R}$, $x \in \mathbb{R}^n$, and $T \in \mathbb{R}^{n \times n}$ upper or lower triangular.

In an application we store T with leading dimension 64 and increase n from 1 to 64 during an iterative process:

- if $n > 16$ the result x gets perturbed,
- and if $n > 32$ the result x is completely wrong.

Bug Description:

(OpenBLAS up to version 0.2.20)

The DTRMV routine computes

$$x := \alpha T x$$

with $\alpha \in \mathbb{R}$, $x \in \mathbb{R}^n$, and $T \in \mathbb{R}^{n \times n}$ upper or lower triangular.

In an application we store T with leading dimension 64 and increase n from 1 to 64 during an iterative process:

- if $n > 16$ the result x gets perturbed,
- and if $n > 32$ the result x is completely wrong.

Hook for DTRSV – Compute with OpenBLAS + Netlib and compare:

Cor. RESULT: DTRMV(U,N,N, 16, A, 64, X, 1) MAXERR = 0.00D+00

Pert. RESULT: DTRMV(U,N,N, 17, A, 64, X, 1) MAXERR = 0.56D-13

Pert. RESULT: DTRMV(U,N,N, 32, A, 64, X, 1) MAXERR = 0.58D-10

Wrong RESULT: DTRMV(U,N,N, 33, A, 64, X, 1) MAXERR = 0.59D+06



Bu

The

wit

In a
64

Ho

Cor

Perf

Perf

Wrong

Race-Condition: The error only appears if OpenBLAS uses multi-threading on highly optimized platforms.

First Workaround: Threading for `xTRMV` is deactivated.

Current Situation:

- Seems to exist more than 10 years.
- Longish discussion (more than 60 comments).
- Still not clear where the race condition comes from.
- Affects also a other race condition on the OpenPOWER 8 platform.
- The DAXPY operation seems to be involved as well.

20)

to

:

= 0.00D+00

= 0.56D-13

= 0.56D-10

= 0.59D+06



Bu

The

wit

In a

64

Ho

Cor

Perf

Perf.

Wrong

Race-Condition: The error only appears if OpenBLAS uses multi-threading on highly optimized platforms.

First Workaround: Threading for `xTRMV` is deactivated.

Current Situation:

- Seems to exist more than 10 years.
- Longish discussion (more than 60 comments).
- Still not clear where the race condition comes from.
- Affects also a other race condition on the OpenPOWER 8 platform.
- The DAXPY operation seems to be involved as well.

Finding such errors is easier with FlexiBLAS.

Outlook:

- meta-data logging hook ready, but no cool analysis tools yet
 - **planned:** replay tool for correctness and accuracy checking
 - **planned:** replay tool for application-driven performance optimization
- MacOS X support in testing
- **planned:** suffixed symbols (in a consistent way)
- **planned:** per application BLAS selection
- **planned:** per routine BLAS selection
- **planned:** LAPACK support
- **planned:** proper way to handle XERBLA and `cbblas_xerbla`
- New release: autumn 2021

Outlook:

- meta-data logging hook ready, but no cool analysis tools yet
 - **planned:** replay tool for correctness and accuracy checking
 - **planned:** replay tool for application-driven performance optimization
- MacOS X support in testing
- **planned:** suffixed symbols (in a consistent way)
- **planned:** per application BLAS selection
- **planned:** per routine BLAS selection
- **planned:** LAPACK support
- **planned:** proper way to handle XERBLA and `cblas_xerbla`
- New release: autumn 2021

Details



M. KÖHLER AND J. SAAK, *FlexiBLAS - A flexible BLAS library with runtime exchangeable backends*, Tech. Rep. 284, LAPACK Working Note, Jan. 2014.

Outlook:

- meta-data logging hook ready, but no cool analysis tools yet

Thank you very much for your attention!

for the software package visit:

<http://www.mpi-magdeburg.mpg.de/projects/flexiblas>

<https://github.com/mpimd-csc/flexiblas/>



Details



M. KÖHLER AND J. SAAK, *FlexiBLAS - A flexible BLAS library with runtime exchangeable backends*, Tech. Rep. 284, LAPACK Working Note, Jan. 2014.