Sobre minhas experiências recentes com desenvolvimento de bibliotecas de software

Weslley da Silva Pereira

Seminários LNCC – Série Alumni

07/15/2024

Photo by Dennis Schroeder, NREL 55200

Trajetória professional

UFJF:

- Engenharia Computacional.
- IC em Elementos Finitos.
- Mestrado em Matemática.

LNCC:

- DSc em Modelagem Computacional.
  ↳ Elementos Finitos, **Software** e HPC.
- Experiências internacionais.

Photo from presenter's personal library.

Trajetória professional

UFJF:

- Posdoc: Física + Análise + **Software**.

University of Colorado:

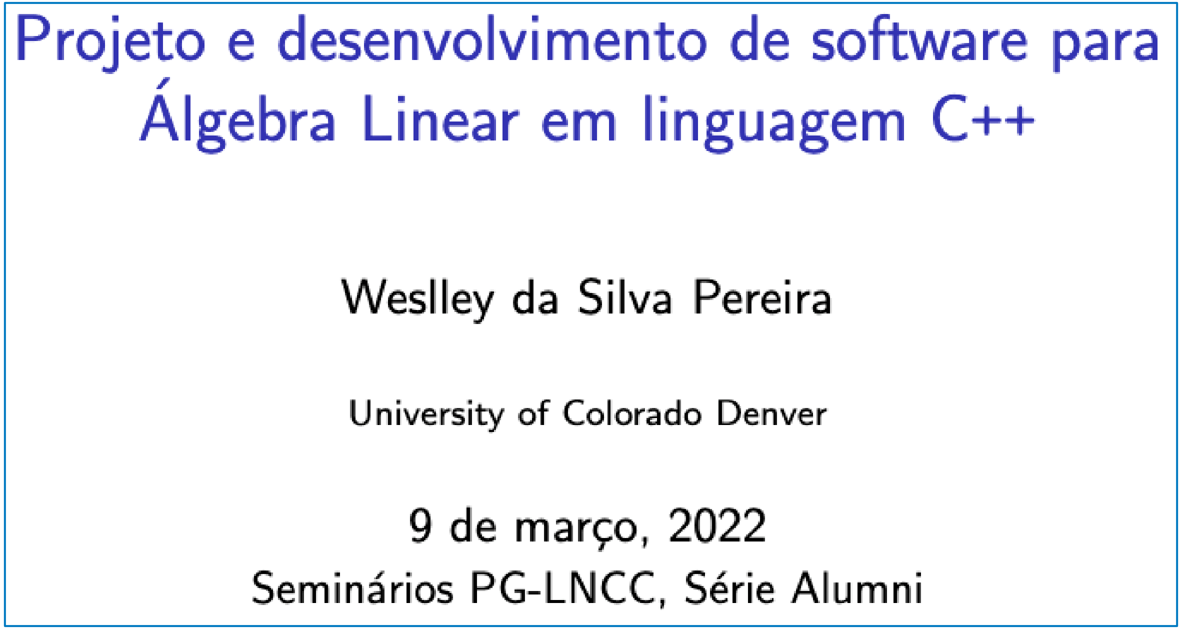
- **Software** para Álgebra Linear.

NREL:

- Research **Software** Engineer.

Photo from presenter's personal library.

2 anos atrás...



Projeto e desenvolvimento de software para Álgebra Linear em linguagem C++

Weslley da Silva Pereira

University of Colorado Denver

9 de março, 2022

Seminários PG-LNCC, Série Alumni

Obrigado pelo novo convite!

Contents

1 LAPACK, <T>LAPACK and BBOpt

2 A software besides the functionality

3 Software development tools

4 Research software, and research again

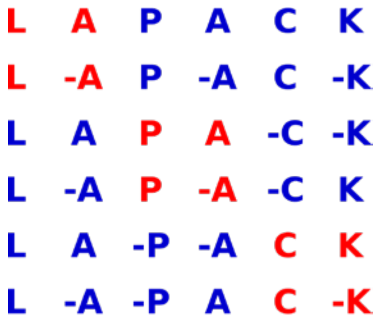
5 Takeaways

My examples of software libraries

LAPACK, <T>LAPACK and BBOpt

Netlib BLAS & LAPACK

https://github.com/Reference-LAPACK/lapack



L	A	P	A	C	K
L	-A	P	-A	C	-K
L	A	P	A	-C	-K
L	-A	P	-A	-C	K
L	A	-P	-A	C	K
L	-A	-P	A	C	-K

Cover image in the LAPACK Users' Guide 3rd edition.

Provide Fortran implementations for:

- $\alpha AB + \beta C$
- $\|A\|$, rot(A), amax(A), αA
- $A = LU$, $A = LL^t$, $A = QR$, …
- $Ax = b$
- $(A^tA)x = A^tb$
- $Av = \lambda v$, $A = V\Lambda V^{-1}$
- $(A^tA)v = \sigma^2 v$, $A = U\Sigma V^t$

- Reference implementations of state-of-the-art algorithms.
- API oriented to portability.
- Standard to other software, e.g.:
  - OpenBLAS, BLIS.
  - Nvidia cuBLAS, AMD rocBLAS, IBM ESSL, Cray LibSci, Arm PL, Intel MKL, Apple Accelerate.

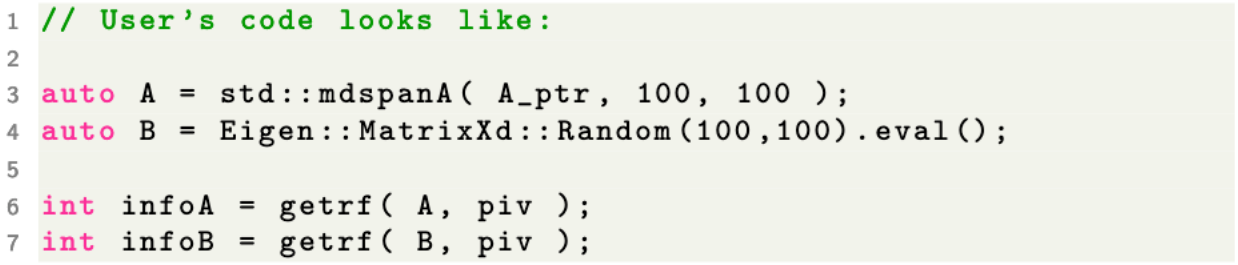
Some contributions to LAPACK

- Participation in the release processes from v3.9.1 to v3.12.0.
- Compiler “shame” tests during build.
- Migration Travis CI -> Github Actions.
- Make CI work on Windows.
- Add memory check to the CI.
- Tests with OpenMP.
- OpenSSF Scorecard badge.
- Givens rotations with lower accumulation error.
- Scaling by the reciprocal of a complex number.

<T>LAPACK

https://github.com/tlapack/tlapack

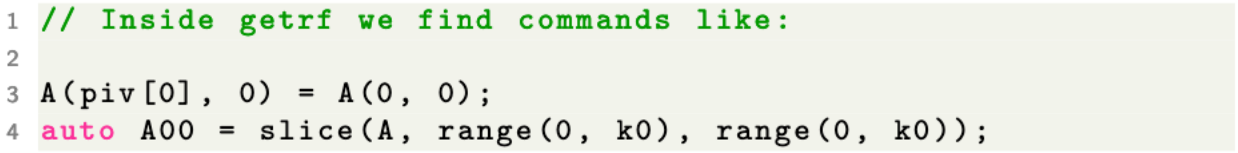
- C++ template-based Linear Algebra Package.
- High-level matrix and type abstraction, compatible to C++23.
- Interoperability with: mdspan, Eigen, StarPU, SLATE, etc.



```
// User's code looks like:

auto A = std::mdspanA( A_ptr, 100, 100 );
auto B = Eigen::MatrixXd::Random(100,100).eval();

int infoA = getrf( A, piv );
int infoB = getrf( B, piv );
```



```
// Inside getrf we find commands like:

A(piv[0], 0) = A(0, 0);
auto A00 = slice(A, range(0, k0), range(0, k0));
```

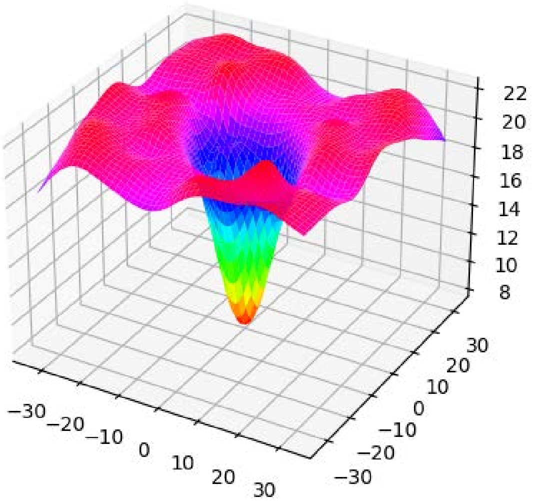
Highlights in the short history of <T>LAPACK

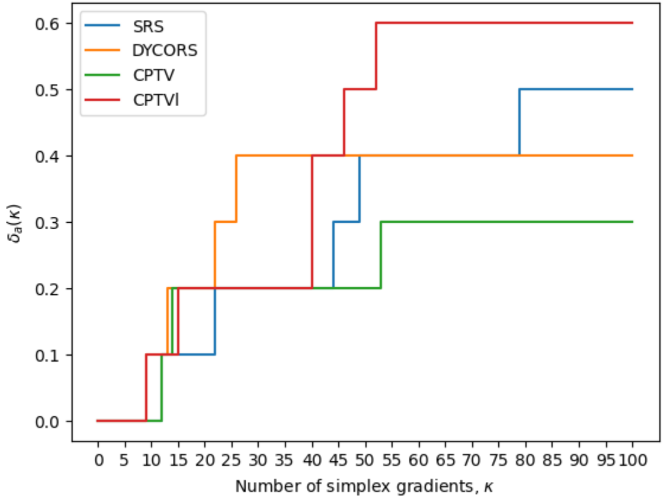
- 04/2021: Templated BLAS aligned to BLAS++.
- 06/2021: testBLAS moved to separate repository.
- 02/2022: Functions templated by matrix class.
- 03/2022: Thijs Steel starts collaborating (AED multishift QR).
- 06/2022: Undergrad students contribute to the software.
- 03/2023: Task-based parallelism with StarPU.
- 06/2023: Concepts helps understanding of the design.
- 11/2023: Undergrad students emulate 8-bit precision in <T>LAPACK.
- 03/2024: Testing mixed precision algorithms using <T>LAPACK.

Black-Box Opt

https://github.com/NREL/bbopt

- Black-box optimization algorithms using Active Learning with surrogates.
- WIP with: RBF models, constrained opt, multi-objective opt.





A software besides the functionality

Examples and Tests

Examples:

- First thing users usually look at.
- Show use cases of the software.

Good practices:

- Easy to find, easy to run.
- Well documented.
- Each example is a separate subprogram.

Tests:

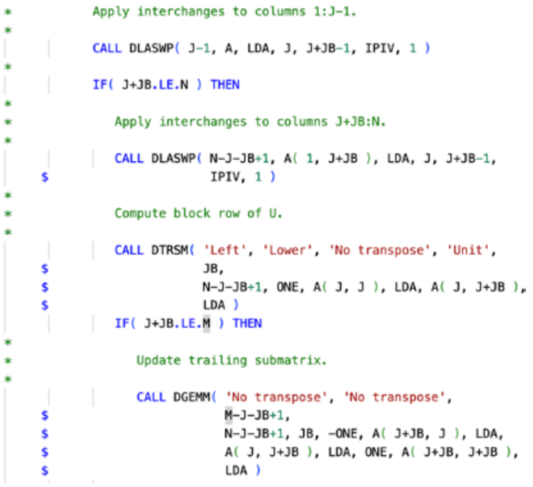
- Verifies multiples ways each piece of software can be used.

Good practices:

- Add new functionality with tests.
- Transform solved issues into tests.
- Make it clear what is and isn’t tested.

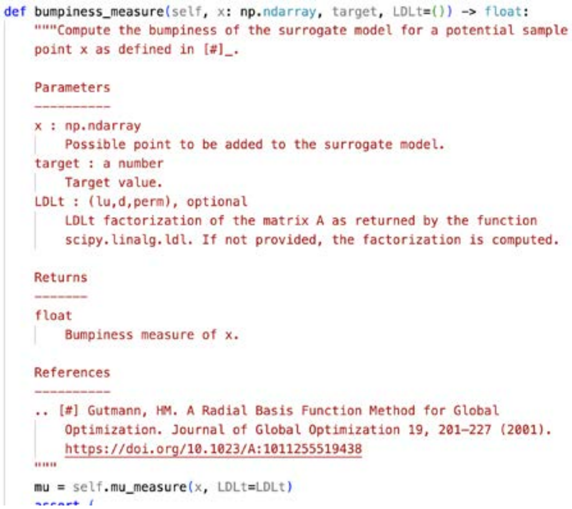
Documentation

- Inside code: comments.
- With code for: functions, classes, objects, files. (Tools: Doxygen, Sphinx)
- Outside code: how to use, how to contribute, description, license, etc.



```fortran
*           Apply interchanges to columns 1:J-1.
*
            CALL DLASWP( J-1, A, LDA, J, J+JB-1, IPIV, 1 )
*
            IF( J+JB.LE.N ) THEN
*
*              Apply interchanges to columns J+JB:N.
*
               CALL DLASWP( N-J-JB+1, A( 1, J+JB ), LDA, J, J+JB-1,
     $                      IPIV, 1 )
*
*              Compute block row of U.
*
               CALL DTRSM( 'Left', 'Lower', 'No transpose', 'Unit',
     $                     JB,
     $                     N-J-JB+1, ONE, A( J, J ), LDA, A( J, J+JB ),
     $                     LDA )
               IF( J+JB.LE.M ) THEN
*
*                 Update trailing submatrix.
*
                  CALL DGEMM( 'No transpose', 'No transpose',
     $                        M-J-JB+1,
     $                        N-J-JB+1, JB, -ONE, A( J+JB, J ), LDA,
     $                        A( J, J+JB ), LDA, ONE, A( J+JB, J+JB ),
     $                        LDA )
```

LAPACK v3.12.0



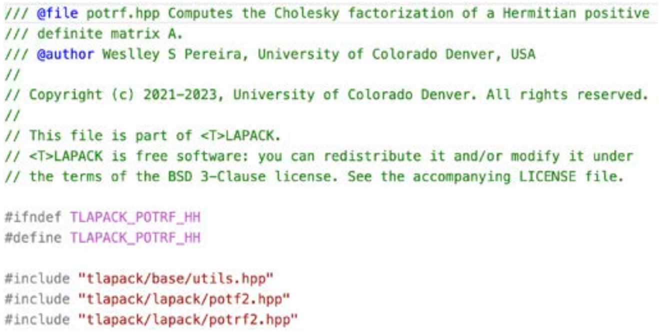
```python
def bumpiness_measure(self, x: np.ndarray, target, LDLt=()) -> float:
    """Compute the bumpiness of the surrogate model for a potential sample
    point x as defined in [#]_.

    Parameters
    ----------
    x : np.ndarray
        Possible point to be added to the surrogate model.
    target : a number
        Target value.
    LDLt : (lu,d,perm), optional
        LDLt factorization of the matrix A as returned by the function
        scipy.linalg.ldl. If not provided, the factorization is computed.

    Returns
    -------
    float
        Bumpiness measure of x.

    References
    ----------
    .. [#] Gutmann, HM. A Radial Basis Function Method for Global
        Optimization. Journal of Global Optimization 19, 201-227 (2001).
        https://doi.org/10.1023/A:1011255519438
    """
    mu = self.mu_measure(x, LDLt=LDLt)
    assert (
```

BBOpt v0.4.1



```cpp
/// @file potrf.hpp Computes the Cholesky factorization of a Hermitian positive
/// definite matrix A.
/// @author Weslley S Pereira, University of Colorado Denver, USA
//
// Copyright (c) 2021-2023, University of Colorado Denver. All rights reserved.
//
// This file is part of <T>LAPACK.
// <T>LAPACK is free software: you can redistribute it and/or modify it under
// the terms of the BSD 3-Clause license. See the accompanying LICENSE file.

#ifndef TLAPACK_POTRF_HH
#define TLAPACK_POTRF_HH

#include "tlapack/base/utils.hpp"
#include "tlapack/lapack/potf2.hpp"
#include "tlapack/lapack/potrf2.hpp"
```

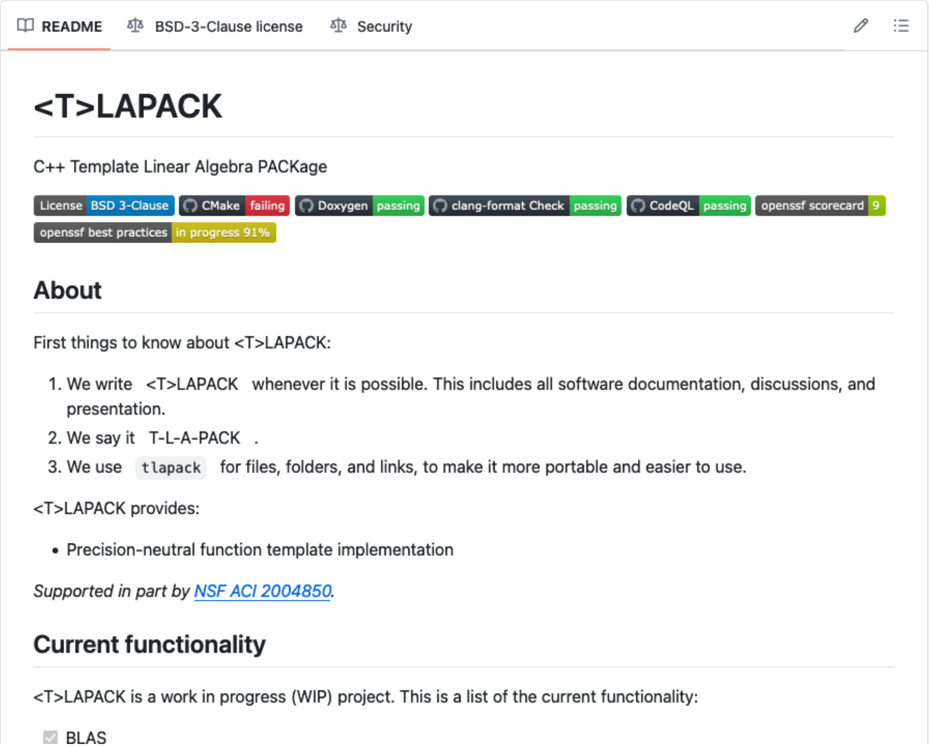
<T>LAPACK master branch in 07/13/2024

Documentation outside the code

Where to put?

- README files
- CONTRIBUTING file
- Wiki
- Papers

Let’s look at https://github.com/tlapack/tlapack and https://github.com/NREL/bbopt



<T>LAPACK

C++ Template Linear Algebra PACKage

About

First things to know about <T>LAPACK:

1. We write <T>LAPACK whenever it is possible. This includes all software documentation, discussions, and presentation.
2. We say it T-L-A-PACK .
3. We use `tlapack` for files, folders, and links, to make it more portable and easier to use.

<T>LAPACK provides:

- Precision-neutral function template implementation

Supported in part by NSF ACI 2004850.

Current functionality

<T>LAPACK is a work in progress (WIP) project. This is a list of the current functionality:

☑ BLAS

Build and install configurations

Only for compiled languages (e.g., C, C++ and Fortran):

- Recipes for building and installation.
- Build systems, e.g., Make and Ninja.
- Build script generator, e.g., Cmake and Meson.

In all cases:

- Package dependencies, e.g., Numpy, Scipy, libBLAS.
- Programming language, e.g., ISO C++14, C++ extensions, Python 3, Python PEP.
- List of artifacts to be installed, i.e., files, libraries, etc.

Software development tools

Integrated Development Environment (IDE)

Aims to improve developer productivity by combining

- code editor with syntax highlighting, code completion, code refactoring, with
- testing, debugging and building tools.

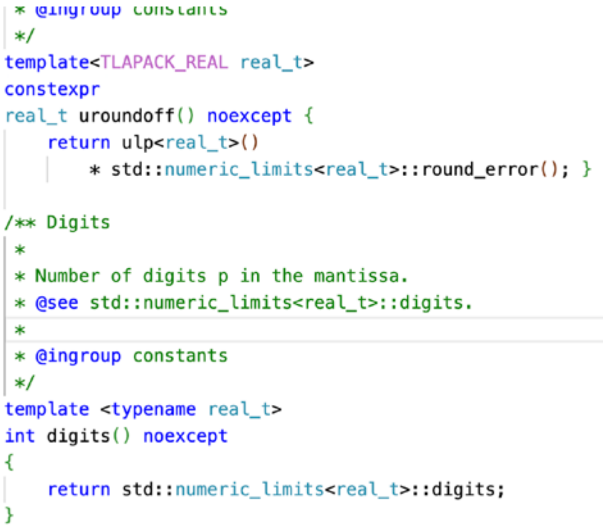
Recommendation: ~~VS Code!~~ Use an IDE that satisfies your needs.

My metrics:

- Actively maintained.
- Large number of users.
- Low memory usage.
- Can run on remote servers.

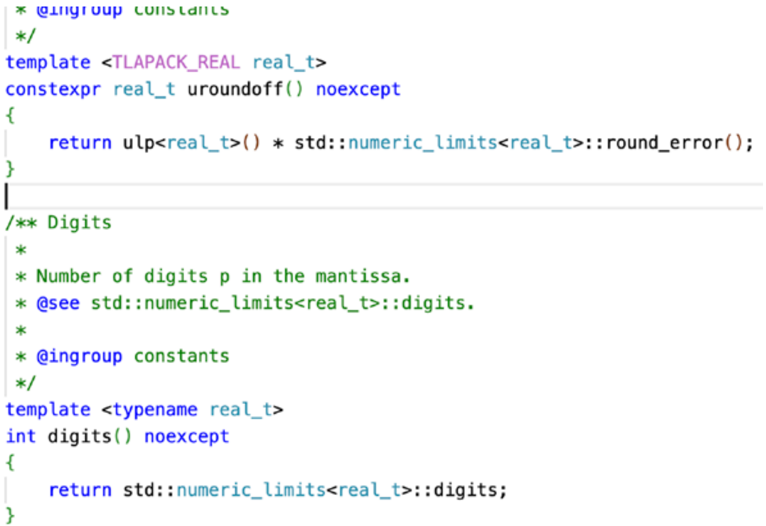
Code formatters

- Keeps code readable and consistent in style.
- Examples: ClangFormat, Ruff.



```
 * @ingroup constants
 */
template<TLAPACK_REAL real_t>
constexpr
real_t uroundoff() noexcept {
    return ulp<real_t>()
        * std::numeric_limits<real_t>::round_error(); }

/** Digits
 *
 * Number of digits p in the mantissa.
 * @see std::numeric_limits<real_t>::digits.
 *
 * @ingroup constants
 */
template <typename real_t>
int digits() noexcept
{
    return std::numeric_limits<real_t>::digits;
}
```



```
 * @ingroup constants
 */
template <TLAPACK_REAL real_t>
constexpr real_t uroundoff() noexcept
{
    return ulp<real_t>() * std::numeric_limits<real_t>::round_error();
}

/** Digits
 *
 * Number of digits p in the mantissa.
 * @see std::numeric_limits<real_t>::digits.
 *
 * @ingroup constants
 */
template <typename real_t>
int digits() noexcept
{
    return std::numeric_limits<real_t>::digits;
}
```

Version Control System

Tracks all file changes of a project over time.

Some concepts:

- Repository: stores all data and metadata of the project.
- Commit: Change in one or multiple files.
- Branch: Linear sequence of commits.
- Merge: Special commit that combines the commits of two branches.

Why Git?

- Distributed, Fast, Robust.

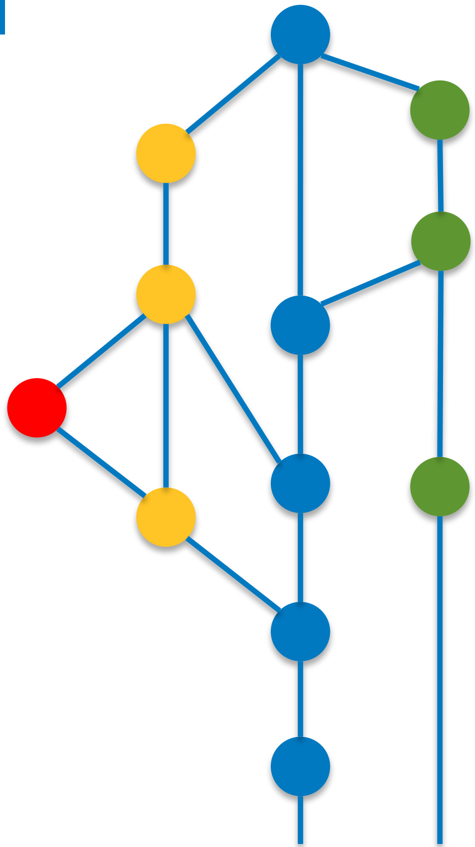
Version Control System… why?

For groups:

- Alice and Bob read file C from a filesystem.
- Alice and Bob do some modifications in file C.
- Alice writes their file in the filesystem.
- Bob writes their file in the filesystem.
- What to expect? Race condition!

In general:

- Well… the code was working yesterday but not today, I wonder why?



Issue Tracking System

“Software is never finished, only abandoned.” – John Saddington.

Issues usually fit into two categories:

- Feature request
- Bug report

You can find such systems inside:

- Github, Gitlab, Bitbucket.

Continuous Integration (CI) and Continuous Development (CD)

CI: Pipeline to automatically build/install code and run tests.

- Runs after every event (commit, merge, week, etc.)
- Test different software configurations, debug/release, Windows/Linux/MacOS, dependency package v1/v2/v3, etc.

CD: Pipeline for automatically deploying the software.

- Update website, documentation, software in a server.

Examples:

- Github Actions, Jenkins, Travis CI, GitLab CI/CD.

Let’s look at https://github.com/tlapack/tlapack.

Software metrics

Quantitative:

- Rate of resolution of issues. (active development)
- Rate of merge requests reviewed. (active development)
- # of citations. (usability)
- # of first-time contributors per release. (community engagement)
- Coverage of tests. (robustness)
- Coverage of documentation. (maintainability)
- OpenSSF scorecard grade. (security)

Software metrics

Qualitative:

- Feedback/quotes.
- Undergrads can use it and contribute to it.
- OS dependent. Dependent on stale software.
- Integrated in other software.
- Impacts other software.
- Impacts research.

Some LAPACK metrics

- Maintainers in close contact with partners (e.g., Intel MKL and Matlab).
- Still being updated, e.g., EAD QZ in v3.10.0.
- LAPACK Users’ Guide 3rd Edition (1999) has 300-500 citations per year.
- # first-time contributors:
  7 out of 15 contributors in LAPACK 3.10.1 (2022).
  10 out of 19 contributors in LAPACK 3.11.0 (2022).
  23 out of 42 contributors in LAPACK 3.12.0 (2023).

Some LAPACK metrics

- 55 PRs accepted 25 issues closed for LAPACK 3.10.
- 70 PRs accepted 61 issues closed for LAPACK 3.11.
- 100 issues open, 20 PRs ready for review as of 07/14/2024.
- 1.5k stars and 429 forks on Github.
- All tests in the CI are passing.
- OpenSSF scorecard: 6.8.

Impact of <T>LAPACK on other software

- Eigen: bug report, block() called recursively.
- GCC: added feature, make std::complex compatible with Eigen::half.
- StarPU: bug reports and fixes.
- BLAS: test suite for Inf and NaN propagation and corner cases.
- SLATE: bug reports and fixes.
- LAPACK:
  - more robust algorithm for reciprocal scaling.
  - more robust algorithm for generation of Givens rotations.

Research software, and research again

Research software

Software made for and mainly used for research.

Common issues on the software side:

- Researcher is not a software engineer.
- Focus on publications.
- No time to plan the software.
- Lack of funding.
- Team of developers change in short time windows.

Why even care?

- Reduce waste of resources. (human time, money, etc.)
- Software may be a tool for reproducibility.

Research Software Engineer

Professional that uses expertise in programming to advance research.

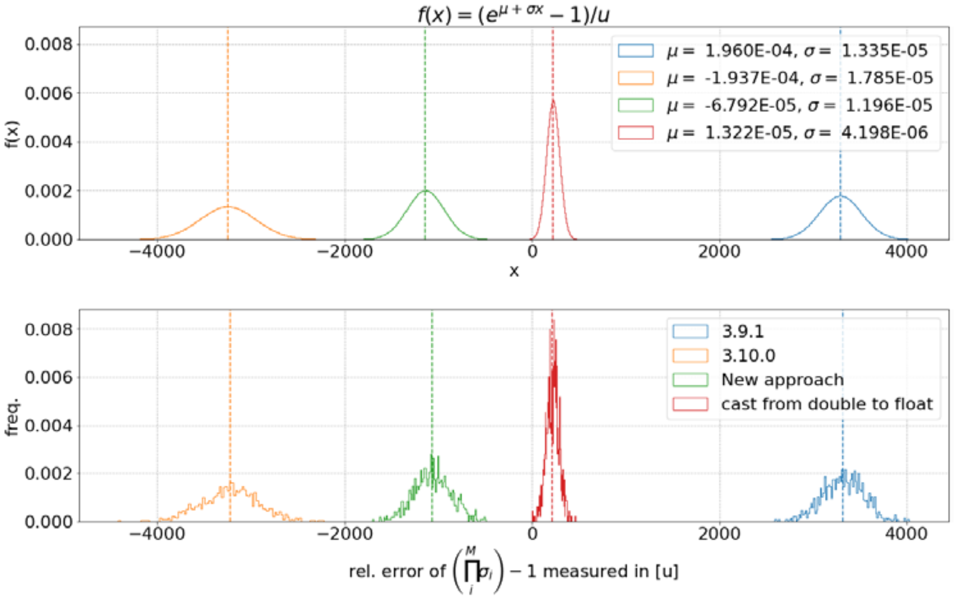
- British Society of RSE, https://society-rse.org/ (since 2019).
- US-RSE, https://us-rse.org (since 2021). Goals:
  - Build a community to share knowledge, connections and resources.
  - Promote RSE's impact on research, highlighting its value.
  - Provide resources for current and future RSEs.

Givens rotation algorithms

Problem: Find (c,s,r) such that $c^2 + |s|^2 = 1$ *and*

$$\begin{bmatrix} c & s \\ -\bar{s} & c \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} r \\ 0 \end{bmatrix}$$

- Algorithm was modified in LAPACK 3.10.0.
- Users reported a lack of accuracy in the new version.
- In the end, a new algorithm was proposed for LAPACK 3.11.0.



Top: Estimated PDF for the errors in the singular values of M = 100,000 composed rotations.
Bottom: Distributions measured through sampling.
Image from Pereira et al. (2022).

Reciprocal scaling of a complex

<u>Problem</u>: Given a vector x, compute x/(a+ib) efficiently with no overflow.

- Algorithm: x * (1/(a+ib)) with proper scaling.
- New routine on LAPACK 3.12.0.
- Change propagated to OpenBLAS. PR #4126.
- Details in (Pereira, 2023).

Takeaways

- A software has more aspects than its core functionality.
- Nonfunctional software requirements can be improved by using development tools.
- Support to research software can lead to better research and reduce waste of resources.

References & Links

Pereira, Lotfi, and Langou. 2022. “Numerical analysis of Givens rotation.” arXiv preprint. arXiv:2211.04010.

Pereira. 2023. “An algorithm for scaling vectors by the reciprocal of a complex number.” arXiv preprint. arXiv:2311.05736

https://github.com/Reference-LAPACK/lapack

https://github.com/tlapack/tlapack

https://github.com/NREL/bbopt

Obrigado!

wdasilv@nrel.gov

www.nrel.gov

NREL/PR-2C00-90579

This work was authored by the National Renewable Energy Laboratory, operated by Alliance for Sustainable Energy, LLC, for the U.S. Department of Energy (DOE) under Contract No. DE-AC36-08GO28308. Funding provided by the National Renewable Energy Laboratory. The views expressed in the slides do not necessarily represent the views of the DOE or the U.S. Government. The U.S. Government retains and the publisher, by accepting the article for publication, acknowledges that the U.S. Government retains a nonexclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this work, or allow others to do so, for U.S. Government purposes.

Photo from iStock-627281636