linalgwrap: The lazy linear algebra library

...only store your matrix when you have to

Michael F. Herbst*[1], James E. Avery[2], Guido Kanschat[1] and Andreas Dreuw[1]

[1]Interdisciplinary Center for Scientific Computing, Ruprecht-Karls Universität Heidelberg, Germany
[2]Department of Computer Science, Københavns Universitet, Denmark

*EMail: michael.herbst@iwr.uni-heidelberg.de — Website: http://blog.mfhs.eu

Universität Heidelberg
Zukunft. Seit 1386

IWR Interdisciplinary Center for Scientific Computing

Introduction

- **stored matrix**: Rank-2 table into a contiguous chunk of memory
- **lazy matrix**: Object which offers a matrix-like interface

Some properties of lazy matrices:

- Matrix values may be computed **on the fly** from internal state
- State and hence matrix is **updateable**
- Sparsity structure can be built into lazy matrix object

Advantages:

⇒ Reduced access into main memory
⇒ Better scalability
⇒ Especially good for problems with large, sparse matrices
⇒ Sometimes: Reduction of computational complexity (see Application)

Design goals of linalgwrap [1]

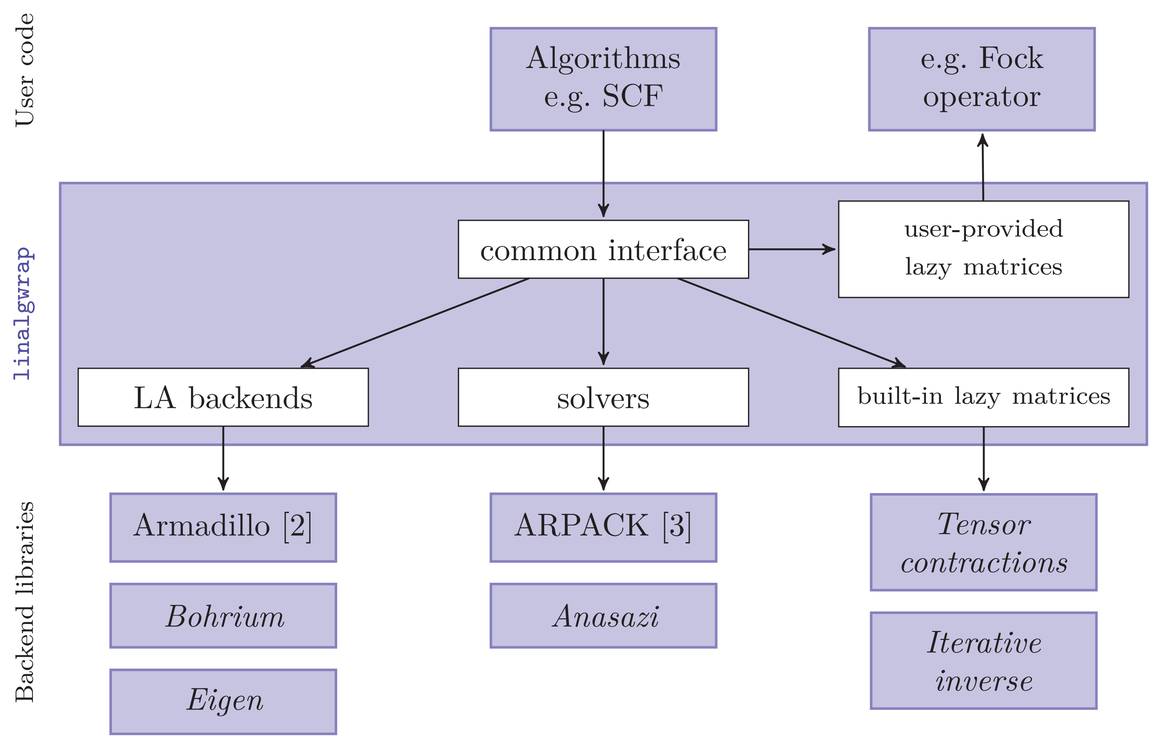

Fig 1. Overview of the planned structure of `linalgwrap`. Italic text indicates that an interface or feature is not yet available.

- Common interface for stored and lazy matrices or linear algebra (LA) backends
  ⇒ **Algorithms independent** of LA implementation details
  ⇒ Easily try new LA libraries or solvers
- Exploit optimisations provided by LA backend as much as possible
  ⇒ **Avoid reinventing** the wheel
  ⇒ Still many potential improvements (e.g. support expression templates)
- Each lazy matrix may influence how LA is performed on it
  ⇒ Knowledge about **internal details** may be exploited
- Modern software design using `C++11` and above
- Property-based **unit tests**

Example code: A very simple SCF

```
int main() {
  typedef SmallVector<double> vector_type;
  typedef MultiVector<vector_type> mulvec_type;

  // Setup problem object somehow ...
  auto problem = setup_problem();

  // Restricted closed shell Fock operator:
  // Sum of lazy matrices -> can be updated
  auto Fock = problem.T() + problem.V0() + 2.0 * problem.J() - problem.K();

  // Initialise by updating with guess of 10 zero vectors:
  mulvec_type zero_guess(10, problem.number_basisfctns());
  Fock.update({{"evec_coefficients", zero_guess}});

  // Perform 10 SCF cycles:
  for (size_t i = 0;; ++i) {
    Eigensolution<double,vector_type> epairs
          = eigensystem_hermitian(Fock, problem.S(), 10);

    // epairs now contains the eigenvalues and eigenvectors
    Fock.update({{"evec_coefficients", epairs.evecs()}});

    if (i >= 9) {
      // Print final eigenvalues and exit.
      print_vector(epairs.evals());
      return 0;
    }
  }
}
```

Application: Coulomb-Sturmian-based Hartree-Fock

- Alternative basis functions for electronic structure calculations:

$$\chi_\mu(\underline{\boldsymbol{x}}) = \chi_{nlm}(\underline{\boldsymbol{x}}) = \underbrace{R_{nl}(r)}_{\text{Radial part}} \cdot \underbrace{Y_{lm}(\theta,\varphi)}_{\text{Spherical harmonics}}$$

- $R_{nl}(r) = \frac{u_{nl}(r)}{r}$ satisfies [4]

$$\left(\frac{\mathrm{d}^2}{\mathrm{d}r^2}\frac{l(l+1)}{r^2} + \frac{k^2}{2} + \frac{kn}{r}\right) u_{nl}(r) = 0.$$

- **Exponential-type** orbitals: $u_{nl}(r) = \mathcal{O}\big(\exp(-kr)\big)$ for large r
- Correctly reproduce nuclear cusp
- One-electron integrals are sparse

Overlap $\quad S_{\mu',\mu} = \delta_{m',m}\delta_{l',l} \begin{cases} n = n' & 1 \\ |n-n'| = 1 & a_{n,n'} \in \mathbb{R} \\ \text{else} & 0 \end{cases}$

Nuclear attraction $\quad (V_0)_{\mu',\mu} = \delta_{\mu',\mu}\dfrac{kZ}{n}$

Kinetic energy $\quad T_{\mu',\mu} = k^2\left(\delta_{\mu',\mu} - S_{\mu',\mu}\right)$

- Two-electron integrals formed by sequence of tensor contractions

Coulomb $$J_{\mu_3,\mu_4} = \sum_{\mu_1\mu_2}\sum_n c_{\mu_1}^{(n)} c_{\mu_2}^{(n)} \sum_{\mu\mu'} \mathcal{C}^{\mu'}_{\mu_1,\mu_2} I_{\mu',\mu} \mathcal{C}^{\mu}_{\mu_3,\mu_4} \tag{1}$$

Exchange $$K_{\mu_3,\mu_2} = \sum_{\mu_1\mu_4}\sum_n c_{\mu_1}^{(n)} c_{\mu_4}^{(n)} \sum_{\mu\mu'} \mathcal{C}^{\mu'}_{\mu_1,\mu_2} I_{\mu',\mu} \mathcal{C}^{\mu}_{\mu_3,\mu_4} \tag{2}$$

where

$\mathcal{C}^{\mu'}_{\mu_1,\mu_2}$ and $I_{\mu',\mu}$		sparse, precomputed tensor
$c_\mu^{(n)}$		SCF coefficients for n-th orbital

Ideal **use case for linalgwrap**:

- **Scaling** of matrix-vector products $\mathbf{K}\underline{\boldsymbol{c}}^{(n)}$ and $\mathbf{J}\underline{\boldsymbol{c}}^{(n)}$ **reduced** by one order if contractions in (1) and (2) are performed **on the fly**.
- $\mathbf{S}$, $\mathbf{T}$ and $\mathbf{V}_0$ **sparse**
- Matrix elements can be computed as **simple analytic expressions**.

The molsturm program

- Extremely modular Hartree-Fock SCF program
- Support for many different integral backends intended

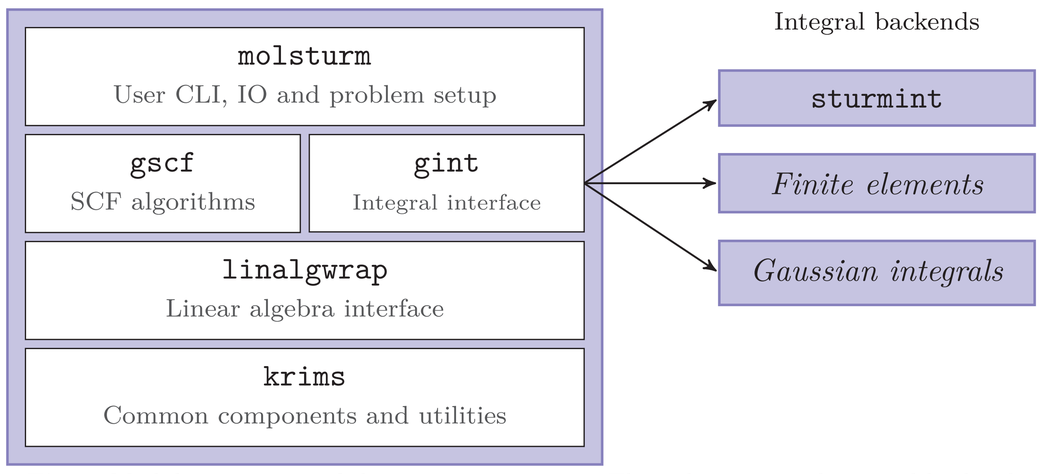

Fig 2. Modular structure of the `molsturm` program. The only supported integral backend so far is `sturmint`, i.e. the Coulomb Sturmian integral library.

Outlook

- Improved support of expression templates within `linalgwrap` [1]
- Interface to Anasazi and Bohrium
- Benchmark `linalgwrap` using standard problems of Quantum Mechanics
- Highly accurate Coulomb Sturmian SCF calculations using `linalgwrap`
- Comparison of Coulomb Sturmian and Gaussian basis for SCF

References

[1] https://linalgwrap.org/, *The linear algebra wrapper library.*

[2] C. Sanderson and R. Curtin, *J. Open Source Software*, **1** (2016), 26.

[3] R. B. Lehoucq, D. C. Sorensen and C. Yang. *ARPACK users' guide: solution of large-scale eigenvalue problems with implicitly restarted Arnoldi methods.* SIAM, 1998.

[4] J. Avery and J. Avery, *Generalized Sturmians and Atomic Spectra.* World Scientific, 2006.