Reusable libraries for quantum chemistry

Susi Lehtola

University of Helsinki, Finland

12 September 2024

Lab in Helsinki

I've been back in Helsinki since 2020

- located within Dage Sundholm's lab
- recently: part of Finnish Quantum Flagship

Current group:

- Hugo Åström, PhD student (magnetic fields, confinement)
- Tirthonkor Saikia, PhD student
- Milan Kunnatholy Babu, PhD student
- Luukas Nikkanen, MSc student (nuclear-electronic orbitals; collaboration with AlgorithmiQ Oy)
- Joonatan Huhtasalo, MSc student (adaptive basis sets)

We accept visitors; possible funding for postdoc positions

A Statement of Facts

All software is aging software, and everything becomes a legacy code at some point.

How to stay up to date?

A Statement of Facts

All software is aging software, and everything becomes a legacy code at some point.

1. Better mathematical tools are continuously developed

How to stay up to date?

A Statement of Facts

All software is aging software, and everything becomes a legacy code at some point.

1. Better mathematical tools are continuously developed
2. Computing architectures are changing

How to stay up to date?

A Statement of Facts

All software is aging software, and everything becomes a legacy code at some point.

1. Better mathematical tools are continuously developed
2. Computing architectures are changing
3. Programming languages and programming models are evolving

How to stay up to date?

A Statement of Facts

All software is aging software, and everything becomes a legacy code at some point.

1. Better mathematical tools are continuously developed
2. Computing architectures are changing
3. Programming languages and programming models are evolving

How to stay up to date?

- This talk: the solution to all of these issues is reusable open source libraries $\implies$ "standard libraries"

A Statement of Facts

All software is aging software, and everything becomes a legacy code at some point.

1. Better mathematical tools are continuously developed
2. Computing architectures are changing
3. Programming languages and programming models are evolving

How to stay up to date?

- This talk: the solution to all of these issues is reusable open source libraries $\implies$ "standard libraries"
- few reusable libraries exist at the moment, need many more!

A Statement of Facts

All software is aging software, and everything becomes a legacy code at some point.

1. Better mathematical tools are continuously developed
2. Computing architectures are changing
3. Programming languages and programming models are evolving

How to stay up to date?

- This talk: the solution to all of these issues is reusable open source libraries $\implies$ "standard libraries"
- few reusable libraries exist at the moment, need many more!
- but first... what is open source?

What Is Open Source

Common misunderstandings in computational chemisty community about nature of open source!

`https://opensource.org/osd` general definition of open source can be summarized following Lehtola and Karttunen, WIREs Comput Mol Sci. 12, e1610 (2022) as

1. Can be used by **anyone** for **any purpose**
2. Operation of the software can be **freely studied** and **modified at will**
3. Software can be **freely redistributed** in **source** and **binary form**, both **in original** and **modified forms**

- If I have to pay a license fee and/or sign a non-disclosure agremeent to get your code, it is proprietary, **not** open source
- Open source implementations can be used by anyone at no cost, and can thereby replace duplicate efforts across open source and proprietary codes

1. Better Mathematical Tools

A better mathematical algorithm affords better quality results at the same cost (or same quality for cheaper)

- keeping code up to date with the best algorithms requires constant improvements
- improving on something that already works is often not motivating

  $\Longrightarrow$ a typical program has layers upon layers of old (obsolete?) code, which has not been touched in decades

Solution: rely on reusable libraries for mathematical algorithms

2. Ever Changing Compute Architectures

New supercomputers already not limited to CPUs, but feature GPUs and other accelerators

- employing these hardware requires dedicated code
  - and often a different type of algorithm as well!
- how to ensure code will run efficiently on new supercomputers in the future?
  - using all the power of the non-GPU computers on the current top-10 list requires similar programming techniques to GPUs

Solution: abstract implementation from problem, allow use of various backends for solution

- easy to do in a reusable library

3. New Programming Languages

- most programs have been written in Fortran'77
  - Fortran'90: modular paradigm, cleaner code; rolling adoption of newer standards
- new projects: C++ is industry standard for high-performance computing
  - almost all GPU stuff is in C++ (CUDA, SYCL, OpenCL, etc)
- new kids on the block: Python, Julia, Rust

Optimal solution: offer interfaces to all languages.

- easy for simple APIs, e.g. coordinates and forces

Yet, cross-language interface standard is C

- can be hard to write interfaces for object-oriented APIs for abstract problems between high-level languages

$\Longrightarrow$ second-best solution: specific libraries for every language

- number of duplicate implementations still greatly reduced

Status Quo: All Programs are Modular, but Not Reusable

Comput. Chem. Vol. 12, No. 2, pp. 85–89, 1988
Printed in Great Britain

0097-8485/88 $3.00 + 0.00
Pergamon Press plc

MODULAR LIBRARIES AND LITERATE PROGRAMMING IN SOFTWARE FOR *AB INITIO* ATOMIC AND MOLECULAR ELECTRONIC STRUCTURE CALCULATIONS

CARLOS F. BUNGE[1] and GERARDO CISNEROS[2]

[1]Instituto de Física, Universidad Nacional Autónoma de México, Apartado Postal 20-364, 01000 México D. F. and [2]Departamento de Aplicación de Microcomputadoras, Instituto de Ciencias, Universidad Autónoma de Puebla, Apartado Postal 461, 72000 Puebla, Puebla, México

(*Received* 20 *April* 1987)

Abstract—Modular libraries and literate programming in software for *ab initio* atomic and molecular electronic structure calculations are proposed as a short way out of a software crisis which is hindering innovation in the field.

Modularity has been an essential part of all electronic structure packages for a long time

- that problem has been solved a long time ago with e.g. the Fortran 90 standard, allowing splitting off large programs into compilation units

Status Quo: All Programs are Modular... but Not Reusable

The real problem is duplicated code and duplicated effort **across programs**

- core focus (e.g. recent JCP special issue, the CECAM meetings) traditionally on *modularity* in program packages, not increased *reusability* of software libraries **across** packages
- solution: explicitly focus making **reusable** software

A call to arms: Making the case for more reusable libraries

Cite as: J. Chem. Phys. 159, 180901 (2023); doi: 10.1063/5.0175165
Submitted: 5 September 2023 • Accepted: 23 October 2023 •
Published Online: 10 November 2023

Susi Lehtola[a)]

Reusable Software is an Old Idea, as Well

ACM SIGSOFT SOFTWARE ENGINEERING NOTES vol 13 no 1 Jan 1988 Page 17

Software Reuse Myths

Will Tracz

Program Analysis and Verification Group[1]

Computer Systems Laboratory - ERL 402

Stanford, California 94305

TRACZ@SIERRA.STANFORD.EDU

Abstract

Reusing software is a simple, straightforward concept that has appealed to programmers since the first stored-program computer was created. Unfortunately, software reuse has not evolved beyond its most primitive forms of subroutine libraries and brute force program modification. This paper analyzes nine commonly believed software reuse myths. These myths reveal certain technical, organizational, and psychological software engineering research issues and trends.

- initial start-up costs important: software designed for reuse costs 20%–25% more to develop; break-even after 2nd or 3rd use
- big payoff is decreased maintenance costs; reductions of up to 90% have been reported in development of new systems

Reused Software is not Reusable Software

2.8. Myth #8: Reused Software is the Same as Reusable Software.
A corollary to this myth is that "A good way to develop reusable software is to take an existing program and add parameters." Both these myths fail to emphasize the need to design for and document for reuse. Unplanned reuse of software (also called software salvaging) occurs frequently in the software community. Programmers often extract modules or code segments, and then modify them to meet their needs. This is an error-prone and time-consuming process, which could be avoided if the software were designed initially with reuse in mind. As the corollary implies, reuse should be considered at design time, not after the implementation has been completed. The emphasis should be on *planned* reuse. Special attention needs to be placed on interface design and modularization (e.g., low coupling and high cohesion).

[Tracz, ACM SIGSOFT Software Engineering Notes 13, 17 (1988)]

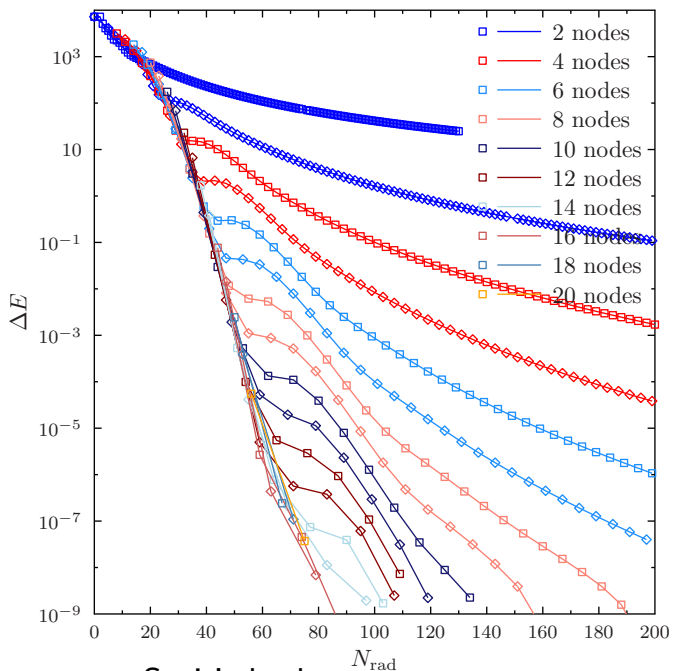
- traditional practice in quantum chemistry: take a program, fork it and hack it to death
- not a maintainable tradition!

Motivation: Numerical Atomic Orbitals for Quantum Chemistry

Main motivation for present project: develop a new open source numerical atomic orbital (NAO) code for quantum chemistry

- NAOs used in solid state, e.g. SIESTA and FHI-aims, but more powerful numerical methods now available
- High-order finite elements allow converging atomic energies to $10^{-9} E_h$ with $\mathcal{O}(100)$ radial basis functions

Error in HF energy of Xe atom [Lehtola, IJQC 119, e25945 (2019)]



Steps to New NAO program

To get the new program up and running, need to at minimum

- implement one and two electron integrals and their derivatives for NAO basis (in practice: use density fitting)
- implement density functional interface and necessary integrals
- implement self-consistent field (SCF) theory

Ideally, one would also have

- post-Hartree-Fock methods (CASSCF, CC, DMRG, etc.)
- geometry optimizer: ground state, transition states, reaction paths)
- response equation solvers to enable computing Hessians
  - various spectroscopies (IR, Raman, NMR, etc)

Steps to New NAO program

To get the new program up and running, need to at minimum

- implement one and two electron integrals and their derivatives for NAO basis (in practice: use density fitting)
- implement density functional interface and necessary integrals
- implement self-consistent field (SCF) theory

Ideally, one would also have

- post-Hartree-Fock methods (CASSCF, CC, DMRG, etc.)
- geometry optimizer: ground state, transition states, reaction paths)
- response equation solvers to enable computing Hessians
  - various spectroscopies (IR, Raman, NMR, etc)

1. problem: this is a lot of work!

Steps to New NAO program

To get the new program up and running, need to at minimum

- implement one and two electron integrals and their derivatives for NAO basis (in practice: use density fitting)
- implement density functional interface and necessary integrals
- implement self-consistent field (SCF) theory

Ideally, one would also have

- post-Hartree-Fock methods (CASSCF, CC, DMRG, etc.)
- geometry optimizer: ground state, transition states, reaction paths)
- response equation solvers to enable computing Hessians
  - various spectroscopies (IR, Raman, NMR, etc)

1. problem: this is a lot of work!
2. problem: a lot of the required algorithms aren't really tied to the use of NAO basis sets!

Steps to New NAO program

To get the new program up and running, need to at minimum

- implement one and two electron integrals and their derivatives for NAO basis (in practice: use density fitting)
- implement density functional interface and necessary integrals
- implement self-consistent field (SCF) theory

Ideally, one would also have

- post-Hartree-Fock methods (CASSCF, CC, DMRG, etc.)
- geometry optimizer: ground state, transition states, reaction paths)
- response equation solvers to enable computing Hessians
  - various spectroscopies (IR, Raman, NMR, etc)

1. problem: this is a lot of work!
2. problem: a lot of the required algorithms aren't really tied to the use of NAO basis sets!

big problem for the field! Example to follow

SCF Algorithms Still Developed

Although SCF theory has been around for long, improved algorithms are still coming out in 2024!

- comparing methods difficult as implementations tied to single packages, here OpenMolcas and MPQC, respectively
- reusable solution would be useful not just for new codes, but for old ones as well: access to faster and robust algorithms

A Story of Three Levels of Sophistication in SCF/KS-DFT Orbital Optimization Procedures

Published as part of The Journal of Physical Chemistry A virtual special issue "Gustavo Scuseria Festschrift".

Daniel Sethio, Emily Azzopardi, Ignacio Fdez. Galván,* and Roland Lindh*

Cite This: *J. Phys. Chem. A* 2024, 128, 2472–2486 Read Online

PAPER View Article Online View Journal | View Issue

Check for updates

Cite this: *Phys. Chem. Chem. Phys.*, 2024, **26**, 6557

Economical quasi-Newton unitary optimization of electronic orbitals†

Samuel A. Slattery, [a] Kshitijkumar A. Surjuse, [a] Charles C. Peterson, [b] Deborah A. Penchoff [c] and Edward F. Valeev *[a]

Problems Also Affect Existing Codes

I have worked on SCF implementations in many codes

- ERKALE, Gaussian-basis code I wrote during my PhD
- Q-Chem, contributions during my Berkeley postdoc
- Psi4, various contributions
- PySCF, various contributions
- HelFEM, finite element methods for atoms and diatomic molecules I started writing in 2018
  - an assembly of three programs with code copied from ERKALE

and already this set is unmaintainable, without the new NAO program!

- want to have new methods in all programs

Really need a reusable solution!

- SCF is just iteratively solving $\mathbf{FC} = \mathbf{SCE}$ or direct minimization
- same code should work in all programs!

How Does Reusable Software Differ From Modular Software?

Characteristics of easily reusable software

1. separation of concerns / modularity: separate code into independent modules
2. high cohesion: elements in the module strongly belong together
3. loose coupling: only few components interact
4. information hiding: the implementation is hidden so that it can be modified without affecting downstream programs

Modularity is only item 1, but one also needs to consider criteria 2, 3, and 4 for reusability!

- Reusability gives a more concise design model for libraries.
- Any task that satisfies many or all of these issues has high potential for standardization across programs.

Case Study: Libxc

A highly successful example of reusable software is our Libxc library whic specializes in the evaluation of density functional approximations

$$E_{\mathrm{xc}} = E(n_\alpha, n_\beta, \gamma_{\alpha\alpha}, \gamma_{\alpha\beta}, \gamma_{\beta\beta}, \tau_\alpha, \tau_\beta, \nabla^2 n_\alpha, \nabla^2 n_\beta)$$

and their derivatives.

Original software publication

Recent developments in LIBXC — A comprehensive library of functionals for density functional theory

Susi Lehtola [a,1], Conrad Steigemann [b], Micael J.T. Oliveira [c,*], Miguel A.L. Marques [b]

[a] Chemical Sciences Division, Lawrence Berkeley National Laboratory, Berkeley, CA 94720, United States
[b] Institut für Physik, Martin-Luther-Universität Halle-Wittenberg, D-06099 Halle, Germany
[c] Max Planck Institute for the Structure and Dynamics of Matter, Luruper Chaussee 149, 22761 Hamburg, Germany

[Lehtola et al, SoftwareX 7, 1 (2018)]

Programs Utilizing Libxc

Over 40 programs using different numerical approaches, algorithms, and licenses—academic, commercial programs, as well as free and open source—currently rely on Libxc

- **if you use Libxc, please cite it!**

- Abinit
- ACE-Molecule
- ADF
- APE
- AtomPAW
- BAGEL
- BigDFT
- CASTEP
- Chronus Quantum
- CP2K
- deMon2k
- DFT-FE
- DP
- Elk
- entos
- ERKALE
- exciting
- FHI-AIMS
- GAMESS (US)
- GPAW
- HelFEM
- Horton
- INQ
- JDFTx
- MADNESS
- MOLGW
- Molpro
- MRCC
- NWChem
- Octopus
- OpenMolcas
- ORCA
- PROFESS
- Psi4
- PySCF
- QuantumATK
- Quantum Espresso
- Serenity
- Turbomole
- VASP
- VeloxChem
- WIEN2k
- Yambo

Plane wave (9) Real-space basis (8) Gaussian basis (19) LAPW (3) Atomic solver (2) LCAO (2)

Libxc Is Reusable Software

Libxc is so successful, since it exhibits all of the characteristics of easily reusable software

- clear scope: evaluation of density functional approximations and their derivatives, e.g.
  - first $\partial E_{\mathrm{xc}}/\partial n_\sigma$, $\partial E_{\mathrm{xc}}/\partial \gamma_{\sigma\sigma'}$,
  - second $\partial^2 E_{\mathrm{xc}}/\partial n_\sigma \partial n_{\sigma'}$, ...
- no coupling: the code is a standalone C library
- information hiding: given the API, the implementations can be improved

Surely many other problems amenable to a standard implementation also exist!

Benefits of Reusable Software

Reusable software

- is straightforward to develop, since problem is preset and actual implementation can be hidden
- eliminates duplicated effort across program packages, freeing up developer time
- enables keeping algorithms up to date; easily done within a single centralized solution (or within a few competing projects sharing the same API)
- enables best practices: what algorithm is efficient on modern architectures?

Economic Efficiency of Open Source

Open source projects are known in the economic literature

- to promote innovation through competition of free and open source packages
- to promote peer review and free exchange of ideas
- to be more motivating to develop and support than proprietary software, and to provide economic benefits
- to be **a public good**

Received: 26 November 2021 | Revised: 14 February 2022 | Accepted: 22 February 2022

DOI: 10.1002/wcms.1610

OVERVIEW

WIREs COMPUTATIONAL MOLECULAR SCIENCE WILEY

Free and open source software for computational chemistry education

Susi Lehtola[1] | **Antti J. Karttunen**[2]

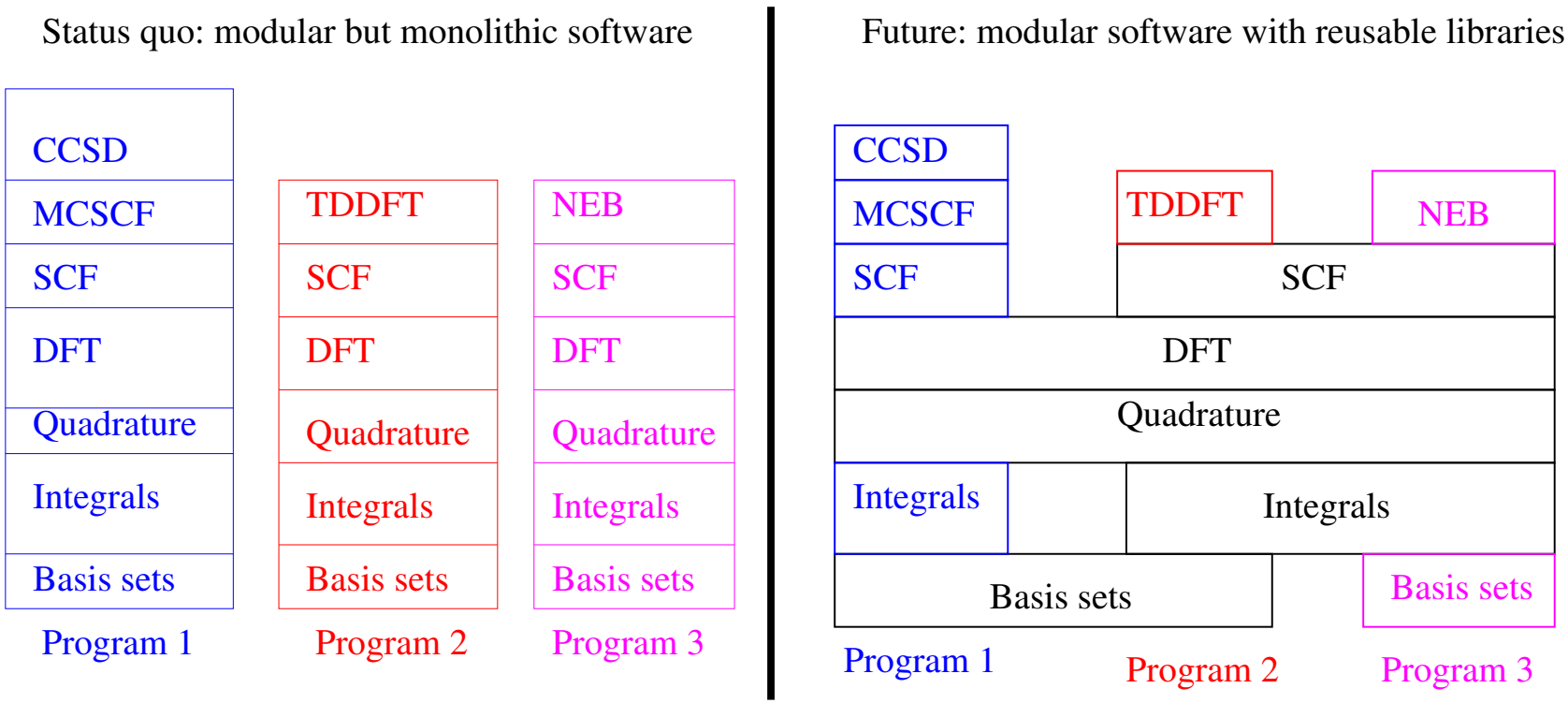
[WIREs Comput Mol Sci. 12, e1610 (2022)]

Summary: Alternative Model

It is clear that an alternative model is possible: need to identify potential common pieces and design them for maximal reuse

- somewhat harder than the traditional modular monolith design

Why develop code for only one program, if same code can work the same way in many programs?



More Work is Needed

We need more reusable libraries!

- discussions between programs can help to find common issues and fashion new solutions
- existing open source projects could be refactored to avail functionalities to other codes
- interoperability less of an issue when functionalities are more accessible

Book Your Calendars

WATOC satellite meeting in Helsinki next summer

Reusable libraries for quantum chemistry

June 29–July 3 2025 (after WATOC)

- purpose: discuss available libraries and the potential for new ones
- room for 60 posters with $12 + 3$ minute flash talks
- lots of time allocated for discussions
- size: ~ 100 participants

Helsinki Winter School 2024

The next **Helsinki Winter School**
held **9–12 December 2024** on the topic
Solvation and Embedding

- registration will open as soon as we have confirmed speakers

Acknowledgments

- Academy of Finland
  - Academy Postdoctoral Fellow grant 2017–2020
  - Academy Research Fellow grant 2022–2027
- Molecular Sciences Software Institute
  - NSF grants CHE-2136142 (current), OAC-1547580 (previous)
- Travel grants
  - Finnish Society of Sciences and Letters
  - Ehrnrooth Foundation
  - French Ministry of Higher Education and Research
  - Embassy of France in Finland
  - French Cultural Institute in Helsinki