

The Quantum Package: developer friendly electronic structure

Emmanuel Giner

Laboratoire de Chimie Théorique, CNRS

Tuesday, 20 February 2024



Context for developers in QC: the code dilemma

- The paradox of Quantum chemistry codes:

Context for developers in QC: the code dilemma

- The paradox of Quantum chemistry codes:
 - HPC software: Large data/flops required !

Context for developers in QC: the code dilemma

- The paradox of Quantum chemistry codes:
 - HPC software: Large data/flops required !
 - Hard problems imply complicated algorithms: flexible coding

Context for developers in QC: the code dilemma

- The paradox of Quantum chemistry codes:
 - ▶ HPC software: Large data/flops required !
 - ▶ Hard problems imply complicated algorithms: flexible coding
 - ▶ HPC and flexible codes don't go along very well ...

Context for developers in QC: the code dilemma

- The paradox of Quantum chemistry codes:
 - ▶ HPC software: Large data/flops required !
 - ▶ Hard problems imply complicated algorithms: flexible coding
 - ▶ HPC and flexible codes don't go along very well ...
- Typical developer's need

Context for developers in QC: the code dilemma

- The paradox of Quantum chemistry codes:
 - HPC software: Large data/flops required !
 - Hard problems imply complicated algorithms: flexible coding
 - HPC and flexible codes don't go along very well ...
- Typical developer's need
 - New idea to tackle an unresolved problem !

Context for developers in QC: the code dilemma

- The paradox of Quantum chemistry codes:
 - ▶ HPC software: Large data/flops required !
 - ▶ Hard problems imply complicated algorithms: flexible coding
 - ▶ HPC and flexible codes don't go along very well ...
- Typical developer's need
 - ▶ New idea to tackle an unresolved problem !
 - ▶ Mathematical equations ← physical quantities

Context for developers in QC: the code dilemma

- The paradox of Quantum chemistry codes:
 - ▶ HPC software: Large data/flops required !
 - ▶ Hard problems imply complicated algorithms: flexible coding
 - ▶ HPC and flexible codes don't go along very well ...
- Typical developer's need
 - ▶ New idea to tackle an unresolved problem !
 - ▶ Mathematical equations ← physical quantities
 - ▶ Easy access/combination of key quantities

Context for developers in QC: the code dilemma

- The paradox of Quantum chemistry codes:
 - ▶ HPC software: Large data/flops required !
 - ▶ Hard problems imply complicated algorithms: flexible coding
 - ▶ HPC and flexible codes don't go along very well ...
- Typical developer's need
 - ▶ New idea to tackle an unresolved problem !
 - ▶ Mathematical equations ← physical quantities
 - ▶ Easy access/combination of key quantities
 - ★ matrix elements

Context for developers in QC: the code dilemma

- The paradox of Quantum chemistry codes:
 - ▶ HPC software: Large data/flops required !
 - ▶ Hard problems imply complicated algorithms: flexible coding
 - ▶ HPC and flexible codes don't go along very well ...
- Typical developer's need
 - ▶ New idea to tackle an unresolved problem !
 - ▶ Mathematical equations ← physical quantities
 - ▶ Easy access/combination of key quantities
 - ★ matrix elements
 - ★ linear algebra routines

Context for developers in QC: the code dilemma

- The paradox of Quantum chemistry codes:
 - ▶ HPC software: Large data/flops required !
 - ▶ Hard problems imply complicated algorithms: flexible coding
 - ▶ HPC and flexible codes don't go along very well ...
- Typical developer's need
 - ▶ New idea to tackle an unresolved problem !
 - ▶ Mathematical equations ← physical quantities
 - ▶ Easy access/combination of key quantities
 - ★ matrix elements
 - ★ linear algebra routines
 - ★ complicated data storage

Context for developers in QC: the code dilemma

- The paradox of Quantum chemistry codes:
 - ▶ HPC software: Large data/flops required !
 - ▶ Hard problems imply complicated algorithms: flexible coding
 - ▶ HPC and flexible codes don't go along very well ...
- Typical developer's need
 - ▶ New idea to tackle an unresolved problem !
 - ▶ Mathematical equations ← physical quantities
 - ▶ Easy access/combination of key quantities
 - ★ matrix elements
 - ★ linear algebra routines
 - ★ complicated data storage
 - ▶ Efficient enough to be tested on "real" systems

Context for developers in QC: the code dilemma

- The paradox of Quantum chemistry codes:
 - ▶ HPC software: Large data/flops required !
 - ▶ Hard problems imply complicated algorithms: flexible coding
 - ▶ HPC and flexible codes don't go along very well ...
- Typical developer's need
 - ▶ New idea to tackle an unresolved problem !
 - ▶ Mathematical equations ← physical quantities
 - ▶ Easy access/combination of key quantities
 - ★ matrix elements
 - ★ linear algebra routines
 - ★ complicated data storage
 - ▶ Efficient enough to be tested on "real" systems
 - ★ Easy: implementation for small model systems

Context for developers in QC: the code dilemma

- The paradox of Quantum chemistry codes:
 - ▶ HPC software: Large data/flops required !
 - ▶ Hard problems imply complicated algorithms: flexible coding
 - ▶ HPC and flexible codes don't go along very well ...
- Typical developer's need
 - ▶ New idea to tackle an unresolved problem !
 - ▶ Mathematical equations ← physical quantities
 - ▶ Easy access/combination of key quantities
 - ★ matrix elements
 - ★ linear algebra routines
 - ★ complicated data storage
 - ▶ Efficient enough to be tested on "real" systems
 - ★ Easy: implementation for small model systems
 - ★ Hard: applications on larger systems

Context for developers in QC: the code dilemma

- The paradox of Quantum chemistry codes:
 - ▶ HPC software: Large data/flops required !
 - ▶ Hard problems imply complicated algorithms: flexible coding
 - ▶ HPC and flexible codes don't go along very well ...
- Typical developer's need
 - ▶ New idea to tackle an unresolved problem !
 - ▶ Mathematical equations ← physical quantities
 - ▶ Easy access/combination of key quantities
 - ★ matrix elements
 - ★ linear algebra routines
 - ★ complicated data storage
 - ▶ Efficient enough to be tested on "real" systems
 - ★ Easy: implementation for small model systems
 - ★ Hard: applications on larger systems
 - ▶ Easy to "enter in the community"

Context for developers in QC: the code dilemma

- The paradox of Quantum chemistry codes:
 - ▶ HPC software: Large data/flops required !
 - ▶ Hard problems imply complicated algorithms: flexible coding
 - ▶ HPC and flexible codes don't go along very well ...
- Typical developer's need
 - ▶ New idea to tackle an unresolved problem !
 - ▶ Mathematical equations ← physical quantities
 - ▶ Easy access/combination of key quantities
 - ★ matrix elements
 - ★ linear algebra routines
 - ★ complicated data storage
 - ▶ Efficient enough to be tested on "real" systems
 - ★ Easy: implementation for small model systems
 - ★ Hard: applications on larger systems
 - ▶ Easy to "enter in the community"
 - ★ Diffusing the code to maximize impact/bug reports

Context for developers in QC: the code dilemma

- The paradox of Quantum chemistry codes:
 - HPC software: Large data/flops required !
 - Hard problems imply complicated algorithms: flexible coding
 - HPC and flexible codes don't go along very well ...
- Typical developer's need
 - New idea to tackle an unresolved problem !
 - Mathematical equations ← physical quantities
 - Easy access/combination of key quantities
 - ★ matrix elements
 - ★ linear algebra routines
 - ★ complicated data storage
 - Efficient enough to be tested on "real" systems
 - ★ Easy: implementation for small model systems
 - ★ Hard: applications on larger systems
 - Easy to "enter in the community"
 - ★ Diffusing the code to maximize impact/bug reports
 - ★ Mutual benefits of collaboration

How the Quantum Package was thought

- Typical available codes: "Old but efficient"
 - ▶ Rather ancient but efficient languages (C, Fortran 77/90)
 - ▶ Not necessarily open-source ...
 - ▶ Outsider's point of view: a nightmare !
- How we designed the QP: (short version)
 - ▶ Highly optimized fundamental building blocks
 - ▶ Modular structure: Just pick what you aim for !
 - ▶ Do not need to understand "all the code"
 - ▶ Plugin system: easy to create your own code
- Documentation material
 - ▶ Github webpage (<https://quantumpackage.github.io/qp2/>)
 - ▶ Read-the-Doc documentation
 - ▶ Tutorials (available on Youtube)
 - ▶ Try in your browser interface !

How the Quantum Package was thought

- Modular language

How the Quantum Package was thought

- Modular language
 - Modified Fortran 90 language (Scemama)

How the Quantum Package was thought

- Modular language
 - Modified Fortran 90 language (Scemama)
 - Handles all complex dependencies

How the Quantum Package was thought

- Modular language
 - Modified Fortran 90 language (Scemama)
 - Handles all complex dependencies
 - Just use the quantities you want

How the Quantum Package was thought

- Modular language
 - ▶ Modified Fortran 90 language (Scemama)
 - ▶ Handles all complex dependencies
 - ▶ Just use the quantities you want
 - ▶ Don't need to know how they are built

How the Quantum Package was thought

- Modular language
 - Modified Fortran 90 language (Scemama)
 - Handles all complex dependencies
 - Just use the quantities you want
 - Don't need to know how they are built
- Modular structure

How the Quantum Package was thought

- Modular language
 - ▶ Modified Fortran 90 language (Scemama)
 - ▶ Handles all complex dependencies
 - ▶ Just use the quantities you want
 - ▶ Don't need to know how they are built
- Modular structure
 - ▶ "Core QC objects":

How the Quantum Package was thought

- Modular language
 - ▶ Modified Fortran 90 language (Scemama)
 - ▶ Handles all complex dependencies
 - ▶ Just use the quantities you want
 - ▶ Don't need to know how they are built
- Modular structure
 - ▶ "Core QC objects":
 - ★ One- and two-e integrals (efficient storage)

How the Quantum Package was thought

- Modular language
 - ▶ Modified Fortran 90 language (Scemama)
 - ▶ Handles all complex dependencies
 - ▶ Just use the quantities you want
 - ▶ Don't need to know how they are built
- Modular structure
 - ▶ "Core QC objects":
 - ★ One- and two-e integrals (efficient storage)
 - ★ Slater determinants manipulation

How the Quantum Package was thought

- Modular language
 - ▶ Modified Fortran 90 language (Scemama)
 - ▶ Handles all complex dependencies
 - ▶ Just use the quantities you want
 - ▶ Don't need to know how they are built
- Modular structure
 - ▶ "Core QC objects":
 - ★ One- and two-e integrals (efficient storage)
 - ★ Slater determinants manipulation
 - ★ Hartree-Fock routines

How the Quantum Package was thought

- Modular language
 - ▶ Modified Fortran 90 language (Scemama)
 - ▶ Handles all complex dependencies
 - ▶ Just use the quantities you want
 - ▶ Don't need to know how they are built
- Modular structure
 - ▶ "Core QC objects":
 - ★ One- and two-e integrals (efficient storage)
 - ★ Slater determinants manipulation
 - ★ Hartree-Fock routines
 - ★ DFT quantities (grid, functionals)

How the Quantum Package was thought

- Modular language
 - ▶ Modified Fortran 90 language (Scemama)
 - ▶ Handles all complex dependencies
 - ▶ Just use the quantities you want
 - ▶ Don't need to know how they are built
- Modular structure
 - ▶ "Core QC objects":
 - ★ One- and two-e integrals (efficient storage)
 - ★ Slater determinants manipulation
 - ★ Hartree-Fock routines
 - ★ DFT quantities (grid, functionals)
 - ★ Highly efficient selected CI code
parallel efficiency $\approx 100\%$ up to 6×10^3 CPU

How the Quantum Package was thought

- Modular language
 - ▶ Modified Fortran 90 language (Scemama)
 - ▶ Handles all complex dependencies
 - ▶ Just use the quantities you want
 - ▶ Don't need to know how they are built
- Modular structure
 - ▶ "Core QC objects":
 - ★ One- and two-e integrals (efficient storage)
 - ★ Slater determinants manipulation
 - ★ Hartree-Fock routines
 - ★ DFT quantities (grid, functionals)
 - ★ Highly efficient selected CI code
parallel efficiency $\approx 100\%$ up to 6×10^3 CPU
 - ★ Highly optimized Coupled Cluster code

How the Quantum Package was thought

- Modular language
 - ▶ Modified Fortran 90 language (Scemama)
 - ▶ Handles all complex dependencies
 - ▶ Just use the quantities you want
 - ▶ Don't need to know how they are built
- Modular structure
 - ▶ "Core QC objects":
 - ★ One- and two-e integrals (efficient storage)
 - ★ Slater determinants manipulation
 - ★ Hartree-Fock routines
 - ★ DFT quantities (grid, functionals)
 - ★ Highly efficient selected CI code
parallel efficiency $\approx 100\%$ up to 6×10^3 CPU
 - ★ Highly optimized Coupled Cluster code
 - ▶ Plugin system: developer's playground !

How the Quantum Package was thought

- Modular language
 - Modified Fortran 90 language (Scemama)
 - Handles all complex dependencies
 - Just use the quantities you want
 - Don't need to know how they are built
- Modular structure
 - "Core QC objects":
 - ★ One- and two-e integrals (efficient storage)
 - ★ Slater determinants manipulation
 - ★ Hartree-Fock routines
 - ★ DFT quantities (grid, functionals)
 - ★ Highly efficient selected CI code
parallel efficiency $\approx 100\%$ up to 6×10^3 CPU
 - ★ Highly optimized Coupled Cluster code
 - Plugin system: developer's playground !
 - ★ External codes/routines

How the Quantum Package was thought

- Modular language
 - Modified Fortran 90 language (Scemama)
 - Handles all complex dependencies
 - Just use the quantities you want
 - Don't need to know how they are built
- Modular structure
 - "Core QC objects":
 - ★ One- and two-e integrals (efficient storage)
 - ★ Slater determinants manipulation
 - ★ Hartree-Fock routines
 - ★ DFT quantities (grid, functionals)
 - ★ Highly efficient selected CI code
parallel efficiency $\approx 100\%$ up to 6×10^3 CPU
 - ★ Highly optimized Coupled Cluster code
 - Plugin system: developer's playground !
 - ★ External codes/routines
 - ★ Easy to install and connect with the core

How the Quantum Package was thought

- Modular language
 - Modified Fortran 90 language (Scemama)
 - Handles all complex dependencies
 - Just use the quantities you want
 - Don't need to know how they are built
- Modular structure
 - "Core QC objects":
 - ★ One- and two-e integrals (efficient storage)
 - ★ Slater determinants manipulation
 - ★ Hartree-Fock routines
 - ★ DFT quantities (grid, functionals)
 - ★ Highly efficient selected CI code
parallel efficiency $\approx 100\%$ up to 6×10^3 CPU
 - ★ Highly optimized Coupled Cluster code
 - Plugin system: developer's playground !
 - ★ External codes/routines
 - ★ Easy to install and connect with the core
 - ★ CASSCF, DFT, Transcorrelation

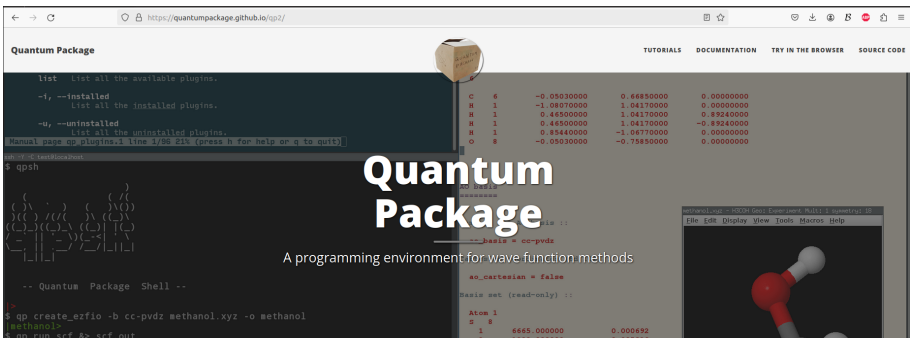
How the Quantum Package was thought

- Modular language
 - Modified Fortran 90 language (Scemama)
 - Handles all complex dependencies
 - Just use the quantities you want
 - Don't need to know how they are built
- Modular structure
 - "Core QC objects":
 - ★ One- and two-e integrals (efficient storage)
 - ★ Slater determinants manipulation
 - ★ Hartree-Fock routines
 - ★ DFT quantities (grid, functionals)
 - ★ Highly efficient selected CI code
parallel efficiency $\approx 100\%$ up to 6×10^3 CPU
 - ★ Highly optimized Coupled Cluster code
 - Plugin system: developer's playground !
 - ★ External codes/routines
 - ★ Easy to install and connect with the core
 - ★ CASSCF, DFT, Transcorrelation
 - ★ Can create yours easily ☺

A few examples of applications with the QP

- Used/developed in
 - Europe: France, Poland
 - North America: U.S.A, Canada
- Ground state and excited states with SCI
 - State-of-the art calculations
 - Benchmark done by Loos *et. al.*
 - Typical system size: Benzene
- Multi-reference range-separated DFT (Julien Toulouse, E. G.)
- Positron-binding calculations (E. G.)
- Transcorrelated/QMC calculations (Ammar, Scemama, E. G.)
- Complex Gaussian basis (Ammar)
- Application to core-excitations/ionizations (Ferté *et. al.*)
- Multi-reference adiabatic connection (Pernal *et. al.*, Poland)
- SCI for QMC (Benali *et. al.*, U.S.A)
- Extension to solid-state of SCI (Caffarel *et. al.*, U.S.A.)
- High-level Coupled Cluster calculations (Piecuch *et. al.*, U.S.A.)
- Pair-natural orbitals functionals (Hollet, Canada)

Visit our website !



The screenshot displays the Quantum Package website interface. At the top, the URL <https://quantumpackage.github.io/qpz/> is visible. The page header includes navigation links for TUTORIALS, DOCUMENTATION, TRY IN THE BROWSER, and SOURCE CODE. The main content area features a dark terminal window on the left with the following text:

```
list List all the available plugins.
-i, --installed List all the installed plugins.
-u, --uninstalled List all the uninstalled plugins.
Manual page on plugins: file 2/23 21% (press h for help or q to quit)
$ qpsh
  ( ) ( )
  ( \ ' ) ( )
  (( )) /(( )) \(( ))
  ((-))((-)_)\((-) |((-)
  \_-|||_-)\_-|_|
  |_|_|_|_|_|_|_|_|_|_|_|

-- Quantum Package Shell --
>
$ qp create_ezfn -b cc-pvdz methanol.xyz -o methanol
[methanol]
$ qp run scf > scf.out
```

In the center, a 3D ball-and-stick molecular model of methanol is shown, with a red oxygen atom and a white hydrogen atom bonded to a grey carbon atom.

On the right side, a table of numerical values is displayed:

C	6	-0.05030000	0.66850000	0.00000000
H	1	-1.08070000	1.04170000	0.00000000
H	1	0.46500000	1.04170000	0.89240000
H	1	0.46500000	1.04170000	-0.89240000
H	1	0.85440000	-1.06770000	0.00000000
O	8	-0.05030000	-0.75850000	0.00000000

Quantum Package

A programming environment for wave function methods

Quantum Package is an open-source programming environment for quantum chemistry specially designed for wave function methods. Its main goal is the development of determinant-driven selected configuration interaction (sCI) methods and multi-reference second-order perturbation theory (PT2).

The determinant-driven framework allows the programmer to include any arbitrary set of determinants in the reference space, hence providing greater methodological freedoms. The sCI method implemented in Quantum Package is based on the CIPSI (Configuration Interaction using a Perturbative Selection made Iteratively) algorithm