2022 Retrospective: Managing Complexity in Psi

Jonathon P. Misiewicz
Emory University

PsiCon
December 9, 2022

Talk Goals

- What in the codebase did I change?

Talk Goals

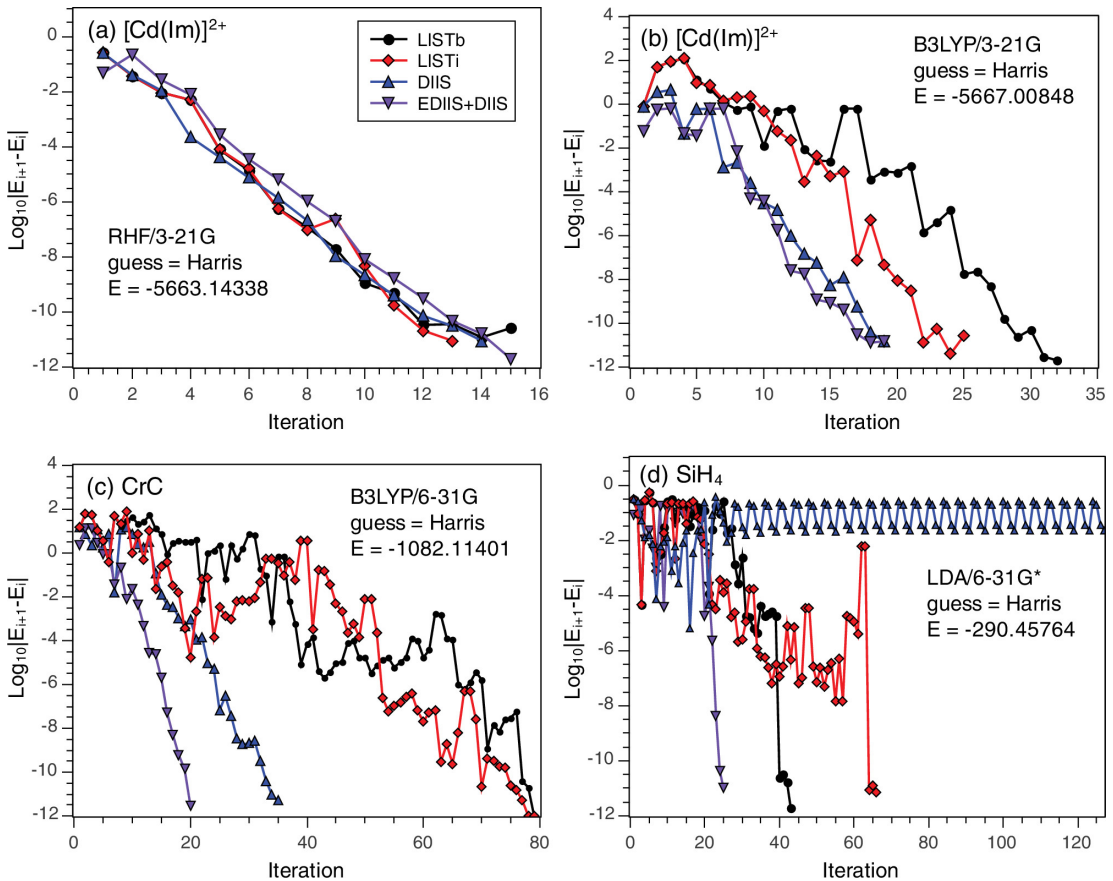
- What in the codebase did I change?
- What in the codebase do I want to change?

Talk Goals

- What in the codebase did I change?
- What in the codebase do I want to change?
- **What in the workflow do I want to change?**

Story 1: ADIIS/EDIIS

ADIIS/EDIIS accelerate SCF convergence when far from converged



DOI: 10.1063/1.4740249; J. Chem. Phys. 137, 054110 (2012); Garza and Scuseria

Story 1: ADIIS/EDIIS

- Technical Problem: Algorithm requires constrained minimization

Story 1: ADIIS/EDIIS

- Technical Problem: Algorithm requires constrained minimization
- Ver. 1: Follow original papers. Transform to unconstrained problem, feed to SciPy BFGS minimizer.

Story 1: ADIIS/EDIIS

- Technical Problem: Algorithm requires constrained minimization
- Ver. 1: Follow original papers. Transform to unconstrained problem, feed to SciPy BFGS minimizer.
- Problem: Users (and test suite) keep reporting the solver fails to work.

Story 1: ADIIS/EDIIS

- Technical Problem: Algorithm requires constrained minimization
- Ver. 1: Follow original papers. Transform to unconstrained problem, feed to SciPy BFGS minimizer.
- Problem: Users (and test suite) keep reporting the solver fails to work.
- Ver. 2: Use SciPy SLSQP solver on constrained minimization.

Story 1: ADIIS/EDIIS

- Technical Problem: Algorithm requires constrained minimization
- Ver. 1: Follow original papers. Transform to unconstrained problem, feed to SciPy BFGS minimizer.
- Problem: Users (and test suite) keep reporting the solver fails to work.
- Ver. 2: Use SciPy SLSQP solver on constrained minimization.
- Works for physical geometries!

Everything Else

- That was my sexy contribution to Psi this year. I specialize in "special forces debugging and refactoring."

Everything Else

- That was my sexy contribution to Psi this year. I specialize in "special forces debugging and refactoring."
- What does Psi need special forces for?

Everything Else

- That was my sexy contribution to Psi this year. I specialize in "special forces debugging and refactoring."
- What does Psi need special forces for?
- Dealing with the consequences of our past decisions

Story 2: Pythonizing UHF Instability Following

- Psi wants to determine if an HF solution is at a minimum.
  Had custom matrix-vector product.

Story 2: Pythonizing UHF Instability Following

- Psi wants to determine if an HF solution is at a minimum.
  Had custom matrix-vector product.
- Psi wants to solve TDDFT.

Story 2: Pythonizing UHF Instability Following

- Psi wants to determine if an HF solution is at a minimum. **Had custom matrix-vector product.**
- Psi wants to solve TDDFT.
- Problem: Code duplication! Both algorithms star the same step.

Story 2: Pythonizing UHF Instability Following

- Psi wants to determine if an HF solution is at a minimum. **Had custom matrix-vector product.**
- Psi wants to solve TDDFT.
- Problem: Code duplication! Both algorithms star the same step.
- By unifying, **we can determine if a KS solution is at a minimum**.

Story 2: Pythonizing UHF Instability Following

- Psi wants to determine if an HF solution is at a minimum. **Had custom matrix-vector product.**
- Psi wants to solve TDDFT.
- Problem: Code duplication! Both algorithms star the same step.
- By unifying, **we can determine if a KS solution is at a minimum**.
- (UKS-LDA currently supported. RKS and UKS supported for LDA and GGA is planned for 1.8. We can already handle exact exchange and long-range.)

Story 3: IrrepedVector

- Old Psi had **completely separate classes for vector of double and vector of int**.

Story 3: IrrepedVector

- Old Psi had **completely separate classes for vector of double and vector of int**.
- When refactoring MOM, I **kept needing to change both classes**

Story 3: IrrepedVector

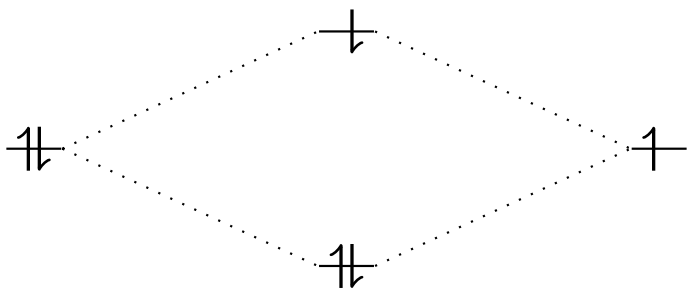
- Old Psi had **completely separate classes for vector of double and vector of int**.
- When refactoring MOM, I **kept needing to change both classes**
- Unified them into template classes, so I can change both at once

Story 4: Compute DOCC/SOCC

Old Psi **stored the numbers of doubly and singly occupied alpha orbitals**

Story 4: Compute DOCC/SOCC

Old Psi **stored the numbers of doubly and singly occupied alpha orbitals**



Story 4: Compute DOCC/SOCC

- Singly occupied beta orbitals **assumed impossible**

Story 4: Compute DOCC/SOCC

- Singly occupied beta orbitals **assumed impossible**
- Led to **crashes and tortured MOM-SCF**

Story 4: Compute DOCC/SOCC

- Singly occupied beta orbitals **assumed impossible**
- Led to **crashes and tortured MOM-SCF**
- Now we store the alpha and beta occupations and compute the occupations

The Problem

- Those three examples were “big” but straightforward and unavoidable

The Problem

- Those three examples were "big" but straightforward and unavoidable
- Remaining examples are "big" but highly technical, suggest (to me) Psi workflow should change

Story 5: DCT Convergence

- In 2006, Kutzelnigg proposed DCT, with orbitals be chosen to satisfy $[\gamma_{\text{ref}}, h + \bar{g}\gamma] = 0$

Story 5: DCT Convergence

- In 2006, Kutzelnigg proposed DCT, with orbitals be chosen to satisfy $[\gamma_{\text{ref}}, h + \bar{g}\gamma] = 0$
- When $\gamma = \gamma_{\text{ref}}$, get Hartree–Fock

Story 5: DCT Convergence

- In 2006, Kutzelnigg proposed DCT, with orbitals be chosen to satisfy $[\gamma_{\text{ref}}, h + \bar{g}\gamma] = 0$
- When $\gamma = \gamma_{\text{ref}}$, get Hartree–Fock
- In 2012, it was proposed that orbitals instead be chosen to minimize the energy

Story 5: DCT Convergence

- DCT was failing to converge, for no apparent reason

Story 5: DCT Convergence

- DCT was failing to converge, for no apparent reason
- Tracked convergence failures in DCT down to requirement that γ_{ref} be converged in SO space

Story 5: DCT Convergence

- DCT was failing to converge, for no apparent reason
- Tracked convergence failures in DCT down to requirement that γ_{ref} be converged in SO space
- In linearly dependent case, this requirement not met

Story 5: DCT Convergence

- DCT was failing to converge, for no apparent reason
- Tracked convergence failures in DCT down to requirement that γ_{ref} be converged in SO space
- In linearly dependent case, this requirement not met
- **Solution: Remove the check. It's redundant.**

Story 5: DCT Convergence

- DCT was failing to converge, for no apparent reason
- Tracked convergence failures in DCT down to requirement that γ_{ref} be converged in SO space
- In linearly dependent case, this requirement not met
- **Solution: Remove the check. It's redundant.**
- γ_{ref} converged when orbitals converged, and we already check that (with lin. dep. accounted for).

Story 5: DCT Convergence

- DCT was failing to converge, for no apparent reason
- Tracked convergence failures in DCT down to requirement that γ_{ref} be converged in SO space
- In linearly dependent case, this requirement not met
- **Solution: Remove the check. It's redundant.**
- γ_{ref} converged when orbitals converged, and we already check that (with lin. dep. accounted for).
- 2006 Orbitals: We already check that the commutator is small
- 2012 Orbitals: It's better to check the orbital gradient

The Point

- Purpose of check not written down. Obnoxiously technical theory expertise required to find it.

The Point

- Purpose of check not written down. Obnoxiously technical theory expertise required to find it.
- **This is not the open-source way.**

The Point

- Purpose of check not written down. Obnoxiously technical theory expertise required to find it.
- **This is not the open-source way.**
- Usual solution of institutional knowledge worked here. What about when I'm gone?

The Point

- Purpose of check not written down. Obnoxiously technical theory expertise required to find it.
- **This is not the open-source way.**
- Usual solution of institutional knowledge worked here. What about when I'm gone?
- **How can we reasonably lower the expertise threshhold?**

Story 6: CCSD Properties

- Stumbled onto CCSD dipoles being qualitatively wrong

Story 6: CCSD Properties

- Stumbled onto CCSD dipoles being qualitatively wrong
- 0.495 vs. -0.134 a.u.

Story 6: CCSD Properties

- Stumbled onto CCSD dipoles being qualitatively wrong
- 0.495 vs. -0.134 a.u.
- In 2016, previous student modernized CC densities, but misunderstood orbital indexing

Story 6: CCSD Properties

- Stumbled onto CCSD dipoles being qualitatively wrong
- 0.495 vs. -0.134 a.u.
- In 2016, previous student modernized CC densities, but misunderstood orbital indexing
- Not noticed for lack of tests

Story 6: CCSD Properties

- Stumbled onto CCSD dipoles being qualitatively wrong
- 0.495 vs. -0.134 a.u.
- In 2016, previous student modernized CC densities, but misunderstood orbital indexing
- Not noticed for lack of tests
- Orbital indexing conventions not (clearly) documented, and the student's PI wrote the original code

Story 6: CCSD Properties

- Stumbled onto CCSD dipoles being qualitatively wrong
- 0.495 vs. -0.134 a.u.
- In 2016, previous student modernized CC densities, but misunderstood orbital indexing
- Not noticed for lack of tests
- Orbital indexing conventions not (clearly) documented, and the student's PI wrote the original code
- Institutional knowledge failed.

Story 7: Fixing DF-wK

- TD-DFT first excitation energy with MemDF is wrong by 3 eV with wK.

Story 7: Fixing DF-wK

- TD-DFT first excitation energy with MemDF is wrong by 3 eV with wK.
- The problem: a 14-argument DGEMM multiplied T2 * T1 rather than T1 * T2. (Credit to Holger for debugging help.)

Story 7: Fixing DF-wK

- TD-DFT first excitation energy with MemDF is wrong by 3 eV with wK.
- The problem: a 14-argument DGEMM multiplied T2 * T1 rather than T1 * T2. (Credit to Holger for debugging help.)
- The PR passed tests; we just didn't think to test TD-DFT

Story 7: Fixing DF-wK

- TD-DFT first excitation energy with MemDF is wrong by 3 eV with wK.
- The problem: a 14-argument DGEMM multiplied T2 * T1 rather than T1 * T2. (Credit to Holger for debugging help.)
- The PR passed tests; we just didn't think to test TD-DFT
- Original coder unavailable

Story 8: TD-GGA

- In 2017, began coding the matrix-vector product terms needed for unstricted TD-DFT with GGAs

Story 8: TD-GGA

- In 2017, began coding the matrix-vector product terms needed for unstricted TD-DFT with GGAs
- I reached out to them for the papers...

Story 8: TD-GGA

- In 2017, began coding the matrix-vector product terms needed for unstricted TD-DFT with GGAs
- I reached out to them for the papers...
- Cross-referenced against Psi4Numpy...

Story 8: TD-GGA

- In 2017, began coding the matrix-vector product terms needed for unstricted TD-DFT with GGAs
- I reached out to them for the papers...
- Cross-referenced against Psi4Numpy...
- Looked at (sparse) code comments...

Story 8: TD-GGA

- In 2017, began coding the matrix-vector product terms needed for unstricted TD-DFT with GGAs
- I reached out to them for the papers...
- Cross-referenced against Psi4Numpy...
- Looked at (sparse) code comments...
- Mostly debugged the function!

Story 8: TD-GGA

```
for (int P = 0; P < npoints; P++) {
    rho_ak[P] = 0.5 * C_DDOT(nlocal, phi[P], 1, Tap[P], 1);
    rho_bk[P] = 0.5 * C_DDOT(nlocal, phi[P], 1, Tbp[P], 1);
}

// Rho^d_k and gamma_k
if (ansatz >= 1) {
    for (int P = 0; P < npoints; P++) {
        // Alpha
        rho_ak_x[P] = C_DDOT(nlocal, phi_x[P], 1, Tap[P], 1);
        rho_ak_y[P] = C_DDOT(nlocal, phi_y[P], 1, Tap[P], 1);
        rho_ak_z[P] = C_DDOT(nlocal, phi_z[P], 1, Tap[P], 1);
        gamma_aak[P] = rho_ak_x[P] * rho_ax[P];
        gamma_aak[P] += rho_ak_y[P] * rho_ay[P];
        gamma_aak[P] += rho_ak_z[P] * rho_az[P];
        gamma_aak[P] *= 2.0;

        // Beta
        rho_bk_x[P] = C_DDOT(nlocal, phi_x[P], 1, Tbp[P], 1);
        rho_bk_y[P] = C_DDOT(nlocal, phi_y[P], 1, Tbp[P], 1);
        rho_bk_z[P] = C_DDOT(nlocal, phi_z[P], 1, Tbp[P], 1);
        gamma_bbk[P] = rho_bk_x[P] * rho_bx[P];
        gamma_bbk[P] += rho_bk_y[P] * rho_by[P];
        gamma_bbk[P] += rho_ak_z[P] * rho_bz[P];
        gamma_bbk[P] *= 2.0;

        // Alpha-Beta
        gamma_abk[P] = rho_ak_x[P] * rho_bx[P] + rho_bk_x[P] * rho_ax[P];
        gamma_abk[P] += rho_ak_y[P] * rho_by[P] + rho_bk_y[P] * rho_ay[P];
        gamma_abk[P] += rho_ak_z[P] * rho_bz[P] + rho_bk_z[P] * rho_az[P];
    }
}
```

Story 8: TD-GGA

```
// This one is a doozy
for (int P = 0; P < npoints; P++) {
    // V alpha contributions
    if (rho_a[P] > v2_rho_cutoff_) {
        tmp_val = v2_rho_a_gamma_aa[P] * gamma_aak[P];
        tmp_val += v2_rho_a_gamma_ab[P] * gamma_abk[P];
        tmp_val += v2_rho_a_gamma_bb[P] * gamma_bbk[P];
        C_DAXPY(nlocal, (0.5 * w[P] * tmp_val), phi[P], 1, Tap[P], 1);
    }

    // V beta contributions
    if (rho_b[P] > v2_rho_cutoff_) {
        tmp_val = v2_rho_b_gamma_aa[P] * gamma_aak[P];
        tmp_val += v2_rho_b_gamma_ab[P] * gamma_abk[P];
        tmp_val += v2_rho_b_gamma_bb[P] * gamma_bbk[P];
        C_DAXPY(nlocal, (0.5 * w[P] * tmp_val), phi[P], 1, Tbp[P], 1);
    }

    // => Alpha W terms <= //
    if ((rho_a[P] < v2_rho_cutoff_) || (rho_b[P] < v2_rho_cutoff_)) continue;

    // rho_ak
    v2_val_aa = v2_rho_a_gamma_aa[P] * rho_ak[P];
    v2_val_ab = v2_rho_a_gamma_ab[P] * rho_ak[P];

    // rho_bk
    v2_val_aa += v2_rho_b_gamma_aa[P] * rho_bk[P];
    v2_val_ab += v2_rho_b_gamma_ab[P] * rho_bk[P];

    // gamma_aak
    v2_val_aa += v2_gamma_aa_gamma_aa[P] * gamma_aak[P];
    v2_val_ab += v2_gamma_aa_gamma_ab[P] * gamma_aak[P];
```

Moralizing

- Tests work, let's have more of them.
  - Caught numerous issues during my refactoring
  - Would have caught CCSD dipoles, DF-wK

Moralizing

- Reviewers should only approve code they understand.
  - Comments and Psi4NumPy and paper/equation references

Moralizing

- Reviewers should only approve code they understand.
  - Comments and Psi4NumPy and paper/equation references
  - Internal discussion on this point for the best.
  - This is an up-front cost to method addition
  - This is an up-front cost to reviewing
  - This makes debugging so much easier

Moralizing

- Reviewers should only approve code they understand.
  - Comments and Psi4NumPy and paper/equation references
  - Internal discussion on this point for the best.
  - This is an up-front cost to method addition
  - This is an up-front cost to reviewing
  - This makes debugging so much easier
  - Great for new developers

A Word to New Devs

- There's a lot of Psi. Best way to learn is "small projects," e.g., forum topics
- Comments and cleanup welcome as you go

Future Spec. Ops.

- `libpsio` to `hdf5`

Future Spec. Ops.

- `libpsio` to `hdf5`
  - I/O system from the 90s coded together in a weekend to simplify the API of an older I/O system

Future Spec. Ops.

- `libpsio` to `hdf5`
  - I/O system from the 90s coded together in a weekend to simplify the API of an older I/O system
  - Nowadays, difficult to debug and steep learning curve

Future Spec. Ops.

- `libpsio` to `hdf5`
  - I/O system from the 90s coded together in a weekend to simplify the API of an older I/O system
  - Nowadays, difficult to debug and steep learning curve
  - Let's use a standard library!

Future Spec. Ops.

- `libpsio` to `hdf5`
  - I/O system from the 90s coded together in a weekend to simplify the API of an older I/O system
  - Nowadays, difficult to debug and steep learning curve
  - Let's use a standard library!
  - One of our afternoon talks uses `hdf5` I/O

Future Spec. Ops.

- Gradient Refactor

Future Spec. Ops.

- Gradient Refactor
  - Gradient involves complicated input (IWL with special prefactors)

Future Spec. Ops.

- Gradient Refactor
  - Gradient involves complicated input (IWL with special prefactors)
  - High barrier for anybody coding gradient

Future Spec. Ops.

- Gradient Refactor
  - Gradient involves complicated input (IWL with special prefactors)
  - High barrier for anybody coding gradient
  - Blocks us from frozen core gradients

Future Spec. Ops.

- Gradient Refactor
  - Gradient involves complicated input (IWL with special prefactors)
  - High barrier for anybody coding gradient
  - Blocks us from frozen core gradients
  - OPDM and Lagrangian down, TPDM to go

Future Spec. Ops.

- DFT

Future Spec. Ops.

- DFT
  - UKS-GGA stability analysis and TD-DFT
  - RKS stability analysis (LDA, GGA, Exchange, LRC)

Future Spec. Ops.

- DFT
  - UKS-GGA stability analysis and TD-DFT
  - RKS stability analysis (LDA, GGA, Exchange, LRC)
  - Gradient-wise, VV10 is next. Not sure what that needs.

Future Spec. Ops.

- DFT
  - UKS-GGA stability analysis and TD-DFT
  - RKS stability analysis (LDA, GGA, Exchange, LRC)
  - Gradient-wise, VV10 is next. Not sure what that needs.
  - For hessians, RKS next step is GGA and UKS is LDA

Future Spec. Ops.

- DFT

Future Spec. Ops.

- DFT
  - UKS-GGA stability analysis and TD-DFT
  - RKS stability analysis (LDA, GGA, Exchange, LRC)

Future Spec. Ops.

- DFT
  - UKS-GGA stability analysis and TD-DFT
  - RKS stability analysis (LDA, GGA, Exchange, LRC)
  - Gradient-wise, VV10 is next. Not sure what that needs.

Future Spec. Ops.

- DFT
  - UKS-GGA stability analysis and TD-DFT
  - RKS stability analysis (LDA, GGA, Exchange, LRC)
  - Gradient-wise, VV10 is next. Not sure what that needs.
  - For hessians, RKS next step is GGA and UKS is LDA

Summary

- A year of special forces debugging and refactoring

Summary

- A year of special forces debugging and refactoring
- The answer to “can other people understand this code?” is usually no. This is bad. What are we going to do about that?

Summary

- A year of special forces debugging and refactoring
- The answer to "can other people understand this code?" is usually no. This is bad. What are we going to do about that?
- Next targets are refactoring gradients and I/O, and some new DFT tech

Summary

- A year of special forces debugging and refactoring
- The answer to "can other people understand this code?" is usually no. This is bad. What are we going to do about that?
- Next targets are refactoring gradients and I/O, and some new DFT tech
- My experience doing so reinforces "can other people understand this code?"

Fin