HF Final Project

Austin M. Wallace

2023-04-24

Hartree-Fock Overview

The objective of Hartree-Fock (HF) is to solve the Schrodinger Equation through the usage of optimizing coefficients for a linear combination of atomic orbitals (LCAO) to create molecular orbitals that minimize the electronic energy. Because this procedure depends on the electronic energy, the nuclear energy is not added until the very end of the self-consistent field procedure is complete. Within this procedure, orbitals can be optimized by minimizing the electronic energy due to the varitaional theorem.

Effectively, this process relies on nuclear coordinates, charge, multiplicity, and a basis set – the present work uses atom-centered Gaussian functions to describe atomic orbitals. From here, the overalp of atomic orbitals is computed to forumlate S along with compiting kinetic (T) and potential (V) energy intregals at the beginning. Next, the Core Hamiltonian is formed $H_{\mu\nu} = T_{\mu\nu} + V_{\mu\nu}$. Before constructing the core Fock matrix (Equation 1) for a naive initial density matrix guess (Equation 3), one must orthogonalize $S^{-1/2}$ through diagonalizing S and transforming to $S^{-1/2}$.

$$F_0' = (S^{-1/2})^{\dagger} H S^{-1/2} \tag{1}$$

Through diagonlizing the core Fock matrix with the orthogonalized basis, one can construct C_0.

$$C_0 = S^{-1/2} C_0' \tag{2}$$

Using these sorted eigenvectors, we compute our density guess.

$$D_{\mu\nu} = \sum_{i}^{N/2} C_{\mu i} C_{\nu i} \tag{3}$$

With these inital values, the SCF procedure may begin, where we iteratively compute a new Fock matrix (Equation 4), compute electronic energy (Equation 5), transform Fock matrix to the orthonormal basis (Equation 6), diagonlaize F, (Equation 7) update C (Equation 8), and update the density (Equation 9) until converging the energy.

$$F_{\mu\nu} = H_{\mu\nu} + \sum_{\rho\sigma}^{AO} D_{\rho\sigma} 2[\mu\nu|\rho\sigma] - [\mu\rho|\nu\sigma] \tag{4}$$

$$E = \sum_{\mu\nu}^{AO} D_{\mu\nu}(H_{\mu\nu} + F_{\mu\nu}) \tag{5}$$

$$F' = (S^{-1/2})^{\dagger} F S^{-1/2} \tag{6}$$

$$C'^{\dagger} F' C' = \epsilon \tag{7}$$

$$C = S^{-1/2} C' \tag{8}$$

$$D_{\mu\nu} = \sum_{i}^{N/2} C_{\mu i} C_{\nu i} \tag{9}$$

At the end of each iteration, convergence is checked by computing the energy and comparing with the previous iteration and comparing with a user defined threshold to break the iterative process. Once below the threshold, the electronic energy is added to the nuclear energy to get the HF energy.

Specific Implementation Details

I have implemented the abstract described HF procedure above in C++ through the usage of Eigen3 for performing linear algebra operations with Lapack under the hood. C++ was selected due to having strong support with OpenMP and MPI for parallelization and to improve fundamental knowledege for aspring to become a Psi4 developer.

The project allowed me to learn how to create my own cmake files for building the library across OSX and Ubuntu with ease using the pitchfork convention. Using CMakeLists.txt took some initial learning but provides a seamless path for finding libraries and repeatedly building the project during development. Additionally, it makes it easier for others to know what is needed for getting the project to run on their own machine.

The initial test case for getting the library working follows the "Coding Strategy #1" from canvas; however, the T, V, S, and ERI sections were broken into separate files for a simplier parsing function to generate Eigen::MatrixXd objects. The Eigen library provided an easy option for solving for eigenvalues and eigenvectors through passing your matrix to the SelfAdjointEigenSolver object initialization. The eigenvectors and eigenvalues already returned sorted and have methods to reverse the sorting if needed.

Code

The project is available on GitHub, where I plan to continue to implement parallelization towards a distributed HF code.

The most important files are `hf.cpp`, `input.cpp`, and `helper.cpp` which are displayed below.

hf.cpp

```cpp
#include "helper.hpp"
#include "input.hpp"
#include "omp.h"
#include "stdio.h"
#include <Eigen/Dense>
#include <ctime>
#include <fstream>
#include <iostream>
#include <string>
#include <vector>

using namespace std;

void HF() {
  // Specify Data Path
  /* std::string dataPath = "data/t1"; */
  std::string dataPath = "data/t0";
  double t1 = 1e-8, t2 = 1e-8;

  // Make pointers to store input data
  int num_atoms;
  double E = 0, e_nuc = 0;
  std::vector<int> *elements = nullptr;
  std::vector<double> *eri = nullptr;

```

```
    Eigen::MatrixXd *coords = nullptr;
    Eigen::MatrixXd *T = nullptr;
    Eigen::MatrixXd *V = nullptr;
    Eigen::MatrixXd *e1 = nullptr;
    Eigen::MatrixXd *S = nullptr;

    // Read Data
    input::gatherData(dataPath, num_atoms, &elements, &eri, &coords, &T, &V, &e1,
                      &S, &e_nuc);
    cout << "e_nuc: " << e_nuc << endl;

    // Starting HF Code

    // Allocate Memory for Matrices
    Eigen::MatrixXd *H = nullptr;
    Eigen::MatrixXd *S_12 = nullptr;
    Eigen::MatrixXd *F = nullptr;
    Eigen::MatrixXd *C = nullptr;
    Eigen::MatrixXd *C_0_prime = nullptr;
    Eigen::MatrixXd *D = nullptr;

    // Allocate memory for energy and electron count
    int num_electrons;
    /* int eriSize; */

    // Set Number of Electrons for a Neutral Molecule
    helper::getNumberOfElectrons(num_atoms, elements, &num_electrons);

    H = new Eigen::MatrixXd(T->rows(), T->cols());
    *H = *T + *V;
    cout << endl << "H Matrix: " << endl << endl << *H << endl;

    // Orthogonalization of S
    cout << endl << "S Matrix: " << endl << endl << *S << endl;
    S_12 = new Eigen::MatrixXd(S->rows(), S->cols());
    helper::orthoS(S, S_12);
    cout << endl << "S_12 Matrix: " << endl << endl << *S_12 << endl;

    // Build initial Fock Matrix
    F = new Eigen::MatrixXd(H->rows(), H->cols());
    *F = (*S_12).transpose() * *H * (*S_12);
    cout << endl << "F Matrix: " << endl << endl << *F << endl;

    C_0_prime = new Eigen::MatrixXd(H->rows(), H->cols());
    helper::getC_0_prime(F, C_0_prime);
    cout << endl << "C_0_prime Matrix: " << endl << endl << *C_0_prime << endl;

    C = new Eigen::MatrixXd(H->rows(), H->cols());
    *C = (*S_12) * (*C_0_prime);
    cout << endl << "C Matrix: " << endl << endl << *C << endl;

    D = new Eigen::MatrixXd(H->rows(), H->cols());
    helper::updateDensityMatrix(C, D, num_electrons);
    cout << endl << "D Matrix: " << endl << endl << *D << endl;

    helper::SCF(eri, S_12, H, F, C, D, C_0_prime, num_electrons, &E, e_nuc, t1,
                t2);
    E += e_nuc;
    cout.precision(15);
```

```
  cout << endl << "Final HF Energy: " << E << endl;

  // Final HF Energy:    -74.9659010585405
  // Total Energy =      -74.9659900701199433

  // Free Allocations
  free(elements);
  free(coords);
  free(T);
  free(V);
  free(e1);
  free(S);
  free(eri);
  printf("\nFreed Memory\n");
}

int main() {
  omp_set_num_threads(1);
  Eigen::setNbThreads(1);
  time_t start, end;
  double serial_t;
  start = clock();
  HF();
  end = clock();
  serial_t = (double)(end - start);
  cout << "Serial Time: " << (serial_t / CLOCKS_PER_SEC) << endl;
  double omp_t;
  int num_threads = 2;
  omp_set_num_threads(num_threads);
  Eigen::setNbThreads(num_threads);
  start = clock();
  HF();
  end = clock();
  omp_t = (double)(end - start);
  cout << "Omp Time: " << (double) (omp_t / CLOCKS_PER_SEC) << endl;
  cout << "Omp Speedup: " << (double)(serial_t / omp_t)
       << endl;
  return 0;
}
```

input.cpp

```
#include "input.hpp"
#include "helper.hpp"
#include <Eigen/Dense>
#include <algorithm>
#include <fstream>
#include <iostream>
#include <sstream>
#include <string>
#include <vector>

using namespace std;

void input::readVector(std::string fn, std::vector<std::vector<double>> **arr) {
  std::ifstream file(fn);
  if (!file) {
    std::cout << "Could not open file: " << fn << std::endl;
    return;
  }
  std::string line;
  int count = 0;

  while (getline(file, line)) {
    count++;
  }
  file.clear();
  file.seekg(0);

  *arr = new std::vector<std::vector<double>>(count);
  for (int i = 0; i < count; ++i) {
    (*arr)->at(i) = std::vector<double>(count);
  }

  int i = 0, j;
  while (getline(file, line)) {
    j = 0;
    std::stringstream ss(line);
    std::vector<double> values;
    double value;
    while (ss >> value) {
      (*arr)->at(i).at(j) = value;
      /* std::cout << value << " "; */
      j++;
    }
    /* std::cout << std::endl; */
    i++;
  }
  file.close();
  return;
}

void input::readVector(std::string fn, Eigen::MatrixXd **arr) {
  std::ifstream file(fn);
  if (!file) {
    std::cout << "Could not open file: " << fn << std::endl;
    return;
  }
  std::string line;
  int count = 0;

  while (getline(file, line)) {
    count++;
  }
  file.clear();
  file.seekg(0);

  *arr = new Eigen::MatrixXd(count, count);

  int i = 0, j;
  while (getline(file, line)) {
    j = 0;
    std::stringstream ss(line);
    std::vector<double> values;
    double value;
    while (ss >> value) {
      /* std::cout << value << " " << i << " " << j << std::endl; */
      (**arr)(i, j) = value;
      j++;
    }
    /* std::cout << std::endl; */
    i++;
  }
  file.close();
  return;
}
void input::readVector(std::string fn, std::vector<double> **arr) {
  std::ifstream file(fn);
  if (!file) {
    std::cout << "Could not open file: " << fn << std::endl;
    return;
  }
  std::string line;
  int count = 0;

  while (getline(file, line)) {
    count++;
  }
  file.clear();
  file.seekg(0);

  *arr = new std::vector<double>(count);

  int i = 0;
  while (getline(file, line)) {
    std::stringstream ss(line);
    ss >> (*arr)->at(i);
    i++;
  }
  file.close();
  return;
}

void input::readERI(std::string fn, std::vector<double> **arr, int n_basis) {
  std::ifstream file(fn);
  if (!file) {
    std::cout << "Could not open file: " << fn << std::endl;
    return;
  }
  std::string line;
  int count = 0;

  while (getline(file, line)) {
    count++;
  }
  file.clear();
  file.seekg(0);
  int arrSize =
      /* n_basis * (n_basis + 1) / 2  * (n_basis + 2) / 2 * (n_basis  + 3) / 2 ;
       */
      /* n_basis * n_basis * n_basis * n_basis / 8; */
      /* helper::indexIJKL(n_basis, n_basis, n_basis, n_basis); */
      helper::indexIJKL(n_basis - 1, n_basis - 1, n_basis - 1, n_basis - 1) + 1;

  cout << "arrSize: " << arrSize << endl;
  cout << "count: " << count << endl;

  *arr = new std::vector<double>(count);

  int i, j, k, l;
  double value;
  int ijkl;
  while (getline(file, line)) {
    std::stringstream ss(line);
    ss >> i >> j >> k >> l >> value;
    ijkl = helper::indexIJKL(i, j, k, l);
    /* std::cout << ijkl << " " << i << " " << j << " " << k << " " << l << " "
     */
    /*             << value << std::endl; */
    (*arr)->at(ijkl) = value;
  }
  file.close();
  return;
}

void input::gatherData(std::string dataPath, int &num_atoms,
                       std::vector<int> **elements, std::vector<double> **eri,
                       std::vector<std::vector<double>> **coords,
                       std::vector<std::vector<double>> **T,
                       std::vector<std::vector<double>> **V,
                       std::vector<std::vector<double>> **e1,
                       std::vector<std::vector<double>> **overlap

) {
  std::string geom = dataPath + "/geom.xyz";
  std::string eriFN = dataPath + "/eri.dat";
  std::string TFN = dataPath + "/T.dat";
  std::string VFN = dataPath + "/V.dat";
  std::string e1FN = dataPath + "/e1.dat";
  std::string overlapFN = dataPath + "/overlap.dat";
  // Gathering Geometry
  input::readGeometry(geom, num_atoms, elements, coords);
  std::cout << "Number of atoms: " << num_atoms << std::endl;
  input::printElements(*elements);
  input::printVector(*coords);

  // Gathering T, V, e1, overlap
  input::readVector(TFN, T);
  /* input::printVector(*T); */
  input::readVector(VFN, V);
  /* input::printVector(*V); */
  input::readVector(e1FN, e1);
  /* input::printVector(*e1); */
  input::readVector(overlapFN, overlap);
  input::readVector(eriFN, eri);
  /* printVector(*eri); */
}

void input::gatherData(std::string dataPath, int &num_atoms,
                       std::vector<int> **elements, std::vector<double> **eri,
                       Eigen::MatrixXd **coords, Eigen::MatrixXd **T,
                       Eigen::MatrixXd **V, Eigen::MatrixXd **e1,
                       Eigen::MatrixXd **overlap, double *enuc) {
  std::string geom = dataPath + "/geom.xyz";
  std::string eriFN = dataPath + "/eri.dat";
  std::string TFN = dataPath + "/T.dat";
  std::string VFN = dataPath + "/V.dat";
  std::string e1FN = dataPath + "/e1.dat";
  std::string overlapFN = dataPath + "/overlap.dat";
  std::string enucFN = dataPath + "/enuc.dat";
  // Gathering Geometry
  input::readGeometry(geom, num_atoms, elements, coords);
  /* std::cout << "Number of atoms: " << num_atoms << std::endl; */
  /* input::printElements(*elements); */
  /* input::printVector(*coords); */

  // Gathering T, V, e1, overlap
  input::readVector(TFN, T);
  input::readVector(VFN, V);
  input::readVector(e1FN, e1);
  input::readVector(overlapFN, overlap);

  // Gathering eri
  /* input::readVector(eriFN, eri); */
  int n_basis = (*T)->rows();
  cout << "n_basis: " << n_basis << endl;
  input::readERI(eriFN, eri, n_basis);
  input::readNumber(enucFN, *enuc);
}

void input::numAtoms(std::string filename, int &num_atoms) {
  std::ifstream file(filename);
  if (!file) {
    std::cout << "Could not open file " << filename << std::endl;
    return;
  }
  file >> num_atoms;
}

void input::readGeometry(std::string filename, int &num_atoms,
                         std::vector<int> **elements,
                         Eigen::MatrixXd **coords) {
  std::ifstream file(filename);
  if (!file) {
    std::cout << "Could not open file " << filename << std::endl;
    return;
  }

  file >> num_atoms;
  *elements = new std::vector<int>(num_atoms);
  *coords = new Eigen::MatrixXd(num_atoms, 3);

  std::string line;
  std::getline(file, line); // read in the comment line

  for (int i = 0; i < num_atoms; ++i) {
    int el = -1;
    double x, y, z;
    file >> el >> x >> y >> z;
    (*elements)->at(i) = el;
    (**coords)(i, 0) = x;
    (**coords)(i, 1) = y;
    (**coords)(i, 2) = z;
  }
```

```cpp
  file.close();
  return;
}

void input::readGeometry(std::string filename, int &num_atoms,
                         std::vector<int> **elements,
                         std::vector<std::vector<double>> **coords) {
  std::ifstream file(filename);
  if (!file) {
    std::cout << "Could not open file " << filename << std::endl;
    return;
  }

  file >> num_atoms;
  *elements = new std::vector<int>(num_atoms);
  *coords = new std::vector<std::vector<double>>(num_atoms);
  for (int i = 0; i < num_atoms; ++i) {
    (*coords)->at(i) = std::vector<double>(3);
  }

  std::string line;
  std::getline(file, line); // read in the comment line

  for (int i = 0; i < num_atoms; ++i) {
    int el = -1;
    double x, y, z;
    file >> el >> x >> y >> z;
    (*elements)->at(i) = el;
    (*coords)->at(i).at(0) = x;
    (*coords)->at(i).at(1) = y;
    (*coords)->at(i).at(2) = z;
  }
  file.close();
  return;
}

void input::printVector(std::vector<std::vector<double>> *matrix) {
  std::cout << std::endl;
  for (int i = 0; u_int64_t(i) < matrix->size(); ++i) {
    for (int j = 0; u_int64_t(j) < matrix->at(i).size(); ++j) {
      std::cout.precision(12);
      std::cout << matrix->at(i).at(j) << " ";
    }
    std::cout << std::endl;
  }
  std::cout << std::endl;
}

void input::printVector(std::vector<double> *matrix) {
  std::cout << std::endl;
  for (int i = 0; u_int64_t(i) < matrix->size(); ++i) {
    std::cout.precision(12);
    std::cout << matrix->at(i) << " ";
  }
  std::cout << std::endl;
}

void input::printElements(std::vector<int> *matrix) {
  for (int i = 0; u_int64_t(i) < matrix->size(); ++i) {
    std::cout << matrix->at(i) << " ";
    std::cout << std::endl;
  }
}

void input::readNumber(std::string filename, double &number) {
  std::ifstream file(filename);
  if (!file) {
    std::cout << "Could not open file " << filename << std::endl;
    return;
  }
  file >> number;
  file.close();
}
```

helper.cpp

```
#include "helper.hpp"
#include "stdio.h"
#include <Eigen/Dense>
#include <iostream>
#include <vector>

using namespace Eigen;
using namespace std;

void helper::orthoS(Eigen::MatrixXd *S, Eigen::MatrixXd *S12) {
  // Diagonalize S
  Eigen::SelfAdjointEigenSolver<Eigen::MatrixXd> es(*S);
  Eigen::MatrixXd LAMBDA = es.eigenvalues().asDiagonal();
  Eigen::MatrixXd U = es.eigenvectors();
  // Invert D
  for (int i = 0; i < LAMBDA.rows(); i++) {
    LAMBDA(i, i) = 1 / sqrt(LAMBDA(i, i));
  }
  // Calculate X
  *S12 = U * LAMBDA * U.transpose();
}

void helper::initialFockMatrix(Eigen::MatrixXd *X, Eigen::MatrixXd *H,
                               Eigen::MatrixXd *F) {
  // Calculate F
  *F = (*X).transpose() * *H * (*X);
}

void helper::getC_0_prime(Eigen::MatrixXd *F, Eigen::MatrixXd *C) {
  // Diagonalize F
  SelfAdjointEigenSolver<MatrixXd> eigensolver(*F);
  if (eigensolver.info() != Success)
    abort(); // check for errors
  *C = eigensolver.eigenvectors();
}

void helper::computeEnergy(Eigen::MatrixXd *D, Eigen::MatrixXd *H,
                           Eigen::MatrixXd *F, double *E) {
  // Calculate E
  *E = 0;
  *E = (*D).cwiseProduct((*H) + (*F)).sum();
```

```cpp
}

void helper::getNumberOfElectrons(int num_atoms, std::vector<int> *elements,
                                  int *num_electrons) {
  // Calculate number of electrons
  *num_electrons = 0;
  for (int i = 0; i < num_atoms; i++) {
    *num_electrons += elements->at(i);
  }
}

int helper::indexIJKL(int i, int j, int k, int l) {
  if (j > i){
    std::swap(i, j);
  }
  if (l > k){
    std::swap(k, l);
  }
  int ij = i * (i + 1) / 2 + j;
  int kl = k * (k + 1) / 2 + l;
  if (ij < kl){
    std::swap(ij, kl);
  }
  int ijkl = ij * (ij + 1) / 2 + kl;
  return ijkl;
}

void helper::updateDensityMatrix(Eigen::MatrixXd *C, Eigen::MatrixXd *D,
                                 int num_electrons) {
  // Calculate D
    for (int i = 0; i < C->rows(); i++) {
      for (int j = 0; j < C->rows(); j++) {
        (*D)(i, j) = 0;
        for (int k = 0; k < num_electrons / 2; k++) {
          (*D)(i, j) += (*C)(i, k) * (*C)(j, k);
        }
      }
    }
}


void helper::updateFockMatrix(Eigen::MatrixXd *H, Eigen::MatrixXd *D,
                              Eigen::MatrixXd *F, std::vector<double> *eri) {
  // Update Fock Matrix
  *F = *H;
  for (int mu = 0; mu < H->rows(); mu++) {
    for (int nu = 0; nu < H->cols(); nu++) {
      for (int rho = 0; rho < H->rows(); rho++) {
        for (int sig = 0; sig < H->cols(); sig++) {
          (*F)(mu, nu) += (*D)(rho, sig) *
                          (2 * eri->at(helper::indexIJKL(mu, nu, rho, sig)) -
                           eri->at(helper::indexIJKL(mu, rho, nu, sig)));
        }
      }
    }
  }
}

void helper::SCF(std::vector<double> *eri, Eigen::MatrixXd *S_12,
                 Eigen::MatrixXd *H, Eigen::MatrixXd *F, Eigen::MatrixXd *C,
                 Eigen::MatrixXd *D, Eigen::MatrixXd *C_0_prime,
                 int num_electrons, double *E, double e_nuc, double t1,
                 double t2) {
  // Calculate SCF

  bool converged = false;
  double E2 = 0;
  int iter = 0, max_iter = 100;
  while (!converged) {
    // Update Fock Matrix
    helper::updateFockMatrix(H, D, F, eri);
    /* cout << endl <<"F Matrix: " << endl << endl <<*F << endl; */
    /* cout << endl <<"E: " << endl << endl <<*E << endl; */
    helper::computeEnergy(D, H, F, E);
    *F = (*S_12).transpose() * *F * (*S_12);

    helper::getC_0_prime(F, C_0_prime);
    /* cout << endl <<"C_0_prime Matrix: " << endl <<endl << *C_0_prime << endl;
     */

    *C = (*S_12) * (*C_0_prime);
    /* cout << endl <<"C Matrix: " << endl <<endl << *C << endl; */
    helper::updateDensityMatrix(C, D, num_electrons);
    /* cout << endl <<"D Matrix: " << endl << endl <<*D << endl; */

    cout << "iter: " << iter << " Energy: " << *E << " Delta E: " << (*E - E2) << endl
        ↪ ;
    if (abs(*E - E2) < t1) {
      converged = true;
    } else if (iter > max_iter) {
      cout << "Max iterations reached" << endl;
      converged = true;
    } else {
      E2 = *E;
    }
    iter++;
  }
}
```

Output

The initial output is from using the “Coding Strategy #1” water geometry wiht the STO-3G basis set. The results with the serial version and OpenMP 4 threaded version yeilded the following ouptuts. The speedup with Eigen detecting OpenMP threads and using them is very poor for this small system at about 1.15, meaning that the most naive parallelization is hardly an improvement from the serial version.

Results

The final energy produced from the present work yielded -74.965901 Ha while Psi4 on the same geometry and basis set with default options yields -74.965990 Ha. The difference between these two energies is likely due to me not implementing a cutoff threshold for the commutator of the density and Fock matrix for checking convergence, along with me not using density fitting like Psi4. The speedup from adding 4 Omp threads is quite underwhelming at 1.15; however, it is likely due to the size of the system not being large enough and only calling OMP threads for a few of the steps while mostly remaining serial besides the eigensolvers.

```
-- The CXX compiler identification is GNU 12.2.0
-- Checking whether CXX compiler has -isysroot
-- Checking whether CXX compiler has -isysroot - yes
-- Checking whether CXX compiler supports OSX deployment target flag
-- Checking whether CXX compiler supports OSX deployment target flag - yes
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Check for working CXX compiler: /usr/local/bin/g++-12 - skipped
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Found OpenMP_CXX: -fopenmp (found version "4.5")
-- Found OpenMP: TRUE (found version "4.5")
-- Looking for sgemm_
-- Looking for sgemm_ - not found
-- Performing Test CMAKE_HAVE_LIBC_PTHREAD
-- Performing Test CMAKE_HAVE_LIBC_PTHREAD - Success
-- Found Threads: TRUE
-- Looking for dgemm_
-- Looking for dgemm_ - found
-- Found BLAS: /Applications/Xcode.app/Contents/Developer/Platforms/MacOSX.platform/Developer/SDKs/MacOSX13.3.sdk/System/Library/Frameworks/Accelerate.framework
-- Looking for cheev_
-- Looking for cheev_ - found
-- Found LAPACK: /Applications/Xcode.app/Contents/Developer/Platforms/MacOSX.platform/Developer/SDKs/MacOSX13.3.sdk/System/Library/Frameworks/Accelerate.framework;-lm;-ldl
-- Found MPI_CXX: /Users/austinwallace/miniconda3/envs/qcn/lib/libmpicxx.dylib (found version "3.1")
-- Found MPI: TRUE (found version "3.1")
-- Found OpenMP_CXX: -fopenmp (found version "4.5")
-- Configuring done
-- Generating done
-- Build files have been written to: /Users/austinwallace/gits/HF/cpp/build
[ 25%] Building CXX object src/CMakeFiles/hf.dir/hf.cpp.o
[ 50%] Building CXX object src/CMakeFiles/hf.dir/input.cpp.o
[ 75%] Building CXX object src/CMakeFiles/hf.dir/helper.cpp.o
[100%] Linking CXX executable hf
ld: warning: directory not found for option '-L/usr/include/eigen3'
[100%] Built target hf
n_basis: 7
arrSize: 406
count: 406
e_nuc: 8.90771

H Matrix:

  -32.6851   -7.60432          0          0 -0.0186797    -1.6196    -1.6196
  -7.60432   -9.30206          0          0   -0.22216   -3.54321   -3.54321
         0          0   -7.43084          0          0          0          0
         0          0          0   -7.56702          0   -1.89086    1.89086
-0.0186797   -0.22216          0          0   -7.52666   -1.65879   -1.65879
   -1.6196   -3.54321          0   -1.89086   -1.65879   -4.95649   -1.56026
   -1.6196   -3.54321          0    1.89086   -1.65879   -1.56026   -4.95649

S Matrix:

        1  0.236704         0         0        -0 0.0500137 0.0500137
 0.236704         1         0         0        -0  0.453995  0.453995
```

```
        0          0          1          0          0          0          0
        0          0          0          1          0   0.292739  -0.292739
        0         -0          0          0          1   0.245551   0.245551
0.0500137   0.453995          0   0.292739   0.245551          1   0.251002
0.0500137   0.453995          0  -0.292739   0.245551   0.251002          1

S_12 Matrix:

     1.02406    -0.141659 -3.22048e-18            0   -0.0100026    0.0212116
       ↪ 0.0212116
   -0.141659      1.22192 -2.96899e-17  2.47692e-18     0.105912    -0.275658
     ↪ -0.275658
-3.22048e-18 -2.96899e-17            1  4.23252e-17  1.97909e-17  1.30612e-16   8.2861
  ↪ e-17
           0  2.47692e-18  4.23252e-17      1.09816 -1.62843e-16    -0.213038
          ↪ 0.213038
  -0.0100026     0.105912  1.97909e-17 -1.62843e-16      1.05609    -0.146486
    ↪ -0.146486
   0.0212116    -0.275658  1.30612e-16    -0.213038    -0.146486      1.19048
     ↪ -0.0903463
   0.0212116    -0.275658   8.2861e-17     0.213038    -0.146486   -0.0903463
     ↪ 1.19048

F Matrix:

    -32.3609     -2.78101  5.24172e-17  1.11022e-16    0.0165515    -0.273188
      ↪ -0.273188
    -2.78101     -8.32926  -5.1327e-17  4.44089e-16    -0.281188    -0.481149
      ↪ -0.481149
 5.24172e-17  -5.1327e-17     -7.43084 -6.96731e-16 -5.38842e-16 -1.57009e-15 -9.77471
   ↪ e-16
  1.8735e-16  1.66533e-16 -6.96731e-16     -7.66432  2.35922e-15    -0.134212
    ↪ 0.134212
   0.0165515    -0.281188 -5.38842e-16  2.44249e-15     -7.57829    -0.146808
     ↪ -0.146808
   -0.273188    -0.481149 -1.57009e-15    -0.134212    -0.146808     -4.24477
     ↪ -0.0501421
   -0.273188    -0.481149 -9.77471e-16     0.134212    -0.146808   -0.0501421
     ↪ -4.24477

C_0_prime Matrix:

   -0.993361    -0.104697 -6.45607e-16    0.0476199 -7.98146e-15   4.0881e-16
     ↪ 0.00222819
   -0.113883     0.887058    5.797e-15    -0.417863  6.99798e-14 -2.99761e-14
     ↪ -0.159843
 3.15273e-19  1.64226e-15  4.80473e-15  1.71258e-13            1 -1.14056e-16 -7.14186
   ↪ e-16
  9.7715e-18 -2.46936e-15     0.998516  8.28671e-15 -4.55392e-15    0.0544598 -9.44874
    ↪ e-15
-0.000754957     0.418666 -6.32026e-15     0.906926 -1.55863e-13 -8.64117e-15
  ↪ -0.0469432
  -0.0114923     0.115943    0.0385089   -0.0174439  3.15953e-15    -0.706057
    ↪ 0.697224
  -0.0114923     0.115943   -0.0385089   -0.0174439  3.58246e-15     0.706057
    ↪ 0.697224

C Matrix:

   -1.00161    -0.232145 -1.42291e-15    0.0981483  -1.6388e-14  1.02331e-14
       ↪ 0.054973
 0.00781923      1.07916  6.54511e-15    -0.411669  6.82443e-14 -1.08663e-13
     ↪ -0.584993
 4.4273e-18  1.6493e-15  4.84883e-15  1.71285e-13            1 -1.45466e-16 -5.61539
     ↪ e-16
-4.33681e-18 -2.62637e-15      1.08012  8.75992e-15 -4.86851e-15     0.360639 -6.61415
   ↪ e-14
 0.000444284     0.503178 -6.19296e-15     0.918173 -1.58082e-13 -5.01404e-14
    ↪ -0.270795
 -0.00221056    -0.180521    -0.163398   -0.0358456    7.9105e-15    -0.915938
    ↪ 0.818024
 -0.00221056    -0.180521     0.163398   -0.0358456  6.46414e-15     0.915938
    ↪ 0.818024

D Matrix:

     1.06675    -0.298759  3.60303e-17 -6.31019e-17   -0.0271382    0.0406029
        ↪ 0.0406029
   -0.298759      1.33412 -4.88497e-16  6.29026e-16     0.165031    -0.180072
      ↪ -0.180072
 3.60303e-17 -4.88497e-16            1  3.68824e-16  1.73216e-17  6.80666e-16  8.18881
    ↪ e-16
-6.31019e-17  6.29026e-16  3.68824e-16      1.16667  3.24209e-17     -0.17649
   ↪ 0.17649
  -0.0271382     0.165031  1.73216e-17  3.24209e-17      1.09623    -0.123747
     ↪ -0.123747
   0.0406029    -0.180072  6.80666e-16     -0.17649    -0.123747    0.0605765
      ↪ 0.00717853
   0.0406029    -0.180072  8.18881e-16      0.17649    -0.123747   0.00717853
      ↪ 0.0605765
iter: 0 Energy: -82.1486 Delta E: -82.1486
iter: 1 Energy: -83.8388 Delta E: -1.69013
iter: 2 Energy: -83.8722 Delta E: -0.0333948
iter: 3 Energy: -83.8734 Delta E: -0.00128959
iter: 4 Energy: -83.8736 Delta E: -0.000137273
iter: 5 Energy: -83.8736 Delta E: -2.36545e-05
iter: 6 Energy: -83.8736 Delta E: -4.48631e-06
iter: 7 Energy: -83.8736 Delta E: -8.72434e-07
iter: 8 Energy: -83.8736 Delta E: -1.70848e-07
iter: 9 Energy: -83.8736 Delta E: -3.35253e-08
iter: 10 Energy: -83.8736 Delta E: -6.58248e-09

Final HF Energy: -74.9659010585405

Freed Memory
Serial Time: 0.011638
n_basis: 7
arrSize: 406
count: 406
e_nuc: 8.9077081

H Matrix:

-32.6850823  -7.6043227          0          0  -0.0186797  -1.6196035  -1.6196035
 -7.6043227  -9.3020628          0          0  -0.2221598  -3.5432107  -3.5432107
          0           0 -7.4308356          0           0           0           0
          0           0          0 -7.5670222           0  -1.8908561   1.8908561
 -0.0186797  -0.2221598          0          0  -7.5266557  -1.6587893  -1.6587893
 -1.6196035  -3.5432107          0  -1.8908561  -1.6587893  -4.9564901  -1.5602636
 -1.6196035  -3.5432107          0   1.8908561  -1.6587893  -1.5602636  -4.9564901

S Matrix:

         1  0.2367039          0          0         -0  0.0500137  0.0500137
 0.2367039          1          0          0         -0  0.4539953  0.4539953
         0          0          1          0          0          0          0
         0          0          0          1          0  0.2927386 -0.2927386
         0         -0          0          0          1  0.2455507  0.2455507
 0.0500137  0.4539953          0  0.2927386  0.2455507          1  0.2510021
 0.0500137  0.4539953          0 -0.2927386  0.2455507  0.2510021          1

S_12 Matrix:

     1.02406182657061     -0.141659273415172  -3.22048111124557e-18
         ↪ 0     -0.010002569640787     0.0212115716053913     0.0212115716053913
   -0.141659273415172       1.22191752943491  -2.96898659569188e-17   2.47691638127938e
       ↪ -18      0.105911538776014     -0.275657558591939     -0.275657558591939
-3.22048111124557e-18  -2.96898659569188e-17                      1   4.23252413596324e
    ↪ -17   1.97909022691313e-17   1.30611707135044e-16   8.28610253903473e-17
                    0   2.47691638127938e-18   4.23252413596325e-17
                        ↪ 1.09816107111058  -1.62842650124526e-16     -0.213037590176114
                        ↪       0.213037590176114
   -0.010002569640787      0.105911538776014   1.97909022691313e-17  -1.62842650124526e
       ↪ -16       1.05609001943927     -0.146486280475907     -0.146486280475907
   0.0212115716053913     -0.275657558591939   1.30611707135044e-16
       ↪ -0.213037590176114     -0.146486280475907       1.19047902077767
       ↪ -0.0903463197977045
   0.0212115716053913     -0.275657558591939   8.28610253903472e-17
       ↪ 0.213037590176114     -0.146486280475907    -0.0903463197977044
       ↪ 1.19047902077767

F Matrix:

    -32.3609072054604      -2.78101317027712   5.24172002424587e-17   1.11022302462516e
        ↪ -16     0.0165514684591211     -0.273188318077003     -0.273188318077005
    -2.78101317027712      -8.32925561645821  -5.13270290872514e-17   4.44089209850063e
        ↪ -16     -0.281188185381983     -0.481149427898156     -0.481149427898158
 5.24172002424587e-17  -5.13270290872513e-17      -7.43083559999999  -6.96731231395782e
     ↪ -16  -5.38841970692135e-16  -1.57009226303266e-15   -9.7747116702658e-16
 1.87350135405495e-16   1.66533453693773e-16  -6.96731231395782e-16
     ↪ -7.66432453273666   2.35922392732846e-15     -0.134212006461572
     ↪ 0.134212006461574
   0.0165514684591212     -0.281188185381983  -5.38841970692135e-16   2.44249065417534e
       ↪ -15       -7.5782870430736     -0.146808221404036     -0.146808221404037
   -0.273188318077004     -0.481149427898157  -1.57009226303266e-15
       ↪ -0.134212006461572     -0.146808221404037      -4.24477070218156
       ↪ -0.0501420820421337
   -0.273188318077005     -0.481149427898159   -9.7747116702658e-16
       ↪ 0.134212006461574     -0.146808221404037    -0.0501420820421338
       ↪ -4.24477070218156

C_0_prime Matrix:

   -0.993360946807231     -0.104696768441391  -6.45607357871904e-16
       ↪ 0.047619861479712  -7.98145833312514e-15   4.08810439798154e-16
       ↪ 0.00222818958438748
   -0.113882888380018      0.887057650718452   5.79699625674734e-15
```

```
       ↪ -0.417863112682388  6.99797794137324e-14 -2.99760857482156e-14
       ↪ -0.15984314528772
 3.15272521982205e-19  1.64225978658692e-15  4.80473010276224e-15  1.71258418115006e
     ↪ -13                     1 -1.14056123398505e-16 -7.14186418134342e-16
 9.77150076534427e-18 -2.46935679192742e-15     0.998515963197578  8.28671139854403e
     ↪ -15 -4.55392017498224e-15    0.0544598130699557 -9.44874050494495e-15
-0.000754957031892415     0.418666423379276 -6.32026050815077e-15
    ↪ 0.906925679062221 -1.55863042394564e-13 -8.64117352984491e-15
    ↪ -0.0469432490589991
  -0.0114923264038581     0.115943384706922    0.0385089031239175
      ↪ -0.0174439174151013  3.15953114908042e-15    -0.706057408699895
      ↪ 0.697223613858515
  -0.0114923264038582     0.115943384706923   -0.0385089031239176
      ↪ -0.0174439174151022  3.58245893480923e-15     0.706057408700153
      ↪ 0.697223613858253

C Matrix:

    -1.00161044750755    -0.232144963446622 -1.42290693116998e-15
        ↪ 0.0981482541870465 -1.63879728445281e-14  1.02331337847872e-14
        ↪ 0.0549730380561187
  0.00781922696659781      1.07916282558611  6.54511167486049e-15
      ↪ -0.411669067669377  6.82443017090394e-14 -1.08663078535187e-13
      ↪ -0.584992535025317
 4.42730077727251e-18   1.6492968752263e-15  4.84883135828343e-15  1.71284896132709e
     ↪ -13                     1 -1.45465821282211e-16 -5.61538691641825e-16
-4.33680868994202e-18 -2.62637134262889e-15      1.08012367182238  8.75991987281388e
    ↪ -15  -4.86851309946075e-15     0.360639184404274 -6.61415366920437e-14
  0.00044428381163818      0.50317807835031   -6.1929628092372e-15
      ↪ 0.918172900946269 -1.58081720406738e-13 -5.01404473496336e-14
      ↪ -0.270795205621142
 -0.00221056111666548    -0.180520707470594    -0.163398255593119
     ↪ -0.0358455758061886  7.91050438005288e-15    -0.915938208301685
     ↪ 0.818024274039899
 -0.00221056111666562    -0.180520707470594     0.163398255593117
     ↪ -0.0358455758061863  6.46413796363787e-15     0.915938208301988
     ↪ 0.81802427403956

D Matrix:

     1.06674785240988    -0.298758634414375  3.60302859820339e-17 -6.31019445971245e
         ↪ -17    -0.0271381886434372    0.0406029134607188    0.0406029134607187
   -0.298758634414375      1.33412496571312 -4.88497293589624e-16  6.29025778196539e
       ↪ -16       0.16503116866963    -0.180072006857659    -0.180072006857658
 3.60302859820339e-17 -4.88497293589624e-16                     1  3.68824431297332e
     ↪ -16  1.73215629949631e-17  6.80666040100179e-16  8.18880794903004e-16
-6.31019445971245e-17  6.29025778196539e-16  3.68824431297332e-16
    ↪ 1.16666714643106  3.24209007142285e-17     -0.17649032380061
    ↪ 0.176490323800609
  -0.0271381886434372      0.16503116866963  1.73215629949631e-17  3.24209007142285e
      ↪ -17       1.0962298519525    -0.123747481128067    -0.123747481128067
   0.0406029134607188    -0.180072006857659  6.80666040100179e-16
       ↪ -0.17649032380061    -0.123747481128067    0.0605765076418855
       ↪ 0.00717852778013748
   0.0406029134607187    -0.180072006857658  8.18880794903004e-16
       ↪ 0.176490323800609    -0.123747481128067   0.00717852778013748
       ↪ 0.0605765076418848
iter: 0 Energy: -82.1486234977129 Delta E: -82.1486234977129
iter: 1 Energy: -83.8387582899332 Delta E: -1.69013479222033
```

```
iter: 2 Energy: -83.8721530704074 Delta E: -0.0333947804741541
iter: 3 Energy: -83.8734426611114 Delta E: -0.00128959070397627
iter: 4 Energy: -83.8735799342989 Delta E: -0.000137273187519327
iter: 5 Energy: -83.8736035888434 Delta E: -2.36545445346792e-05
iter: 6 Energy: -83.8736080751508 Delta E: -4.48630736116229e-06
iter: 7 Energy: -83.8736089475846 Delta E: -8.7243387270064e-07
iter: 8 Energy: -83.8736091184327 Delta E: -1.7084808234813e-07
iter: 9 Energy: -83.873609151958 Delta E: -3.35253247385481e-08
iter: 10 Energy: -83.8736091585405 Delta E: -6.58248211493628e-09

Final HF Energy: -74.9659010585405

Freed Memory
Omp Time: 0.010108
Omp Speedup: 1.15136525524337
```