Elevia

Elevia is an electronic structure program owned by **uovie**, whose name stems from the combination of "electron" and "uovie". It is a homework of a quantum chemistry course.

- Author: Haoyu Lin
- E-mail: vileoy@pku.edu.cn
- Repo: uovie/Elevia

1 Releases

The version 1.0.0 of Elevia has been released, which can be obtained from here. **Elevia-1.0.0** can only carry out restricted Hartree-Fock calculations with some Pople basis sets. And this release has already included an executable file `Elevia-1.0.0`, which is compiled in the Fedora 29 system. So, you can use it directly.

- `Elevia-1.0.0` usage tips

  Elevia only accepts `.vie` type input file. Type the following command to run a calculation:

  ```
  ./Elevia-1.0.0 test.vie
  ```

- compile tips

  If you want to compile it by yourself, just invoke `make` under `src` directory . Note that Eigen C++ library and libint-2.5.0 are needed. Please correctly install them before compile.

Elevia-1.0.1 is a new version, and the the following test results are produced based on it.

2 Input file format

Elevia assumes a new input format:

```
[method] [basis] [total number of atoms] [charge] [spin multiplicity]

[atom symbol]    [ Rx ]    [ Ry ]    [ Rz ]
[atom symbol]    [ Rx ]    [ Ry ]    [ Rz ]
[atom symbol]    [ Rx ]    [ Ry ]    [ Rz ]
... ...
```

For instance, an input file for test is

```
RHF STO-3G 3 0 1

O   -0.53792437   -0.43835419    0.00000000
H    0.42207560   -0.43811309    0.00000000
H   -0.85815167    0.46666209    0.00000000
```

3 Supports

1. Method

   Only support RHF.

2. Basis Set

   Currently supported basis sets:

   - STO-3G
   - 3-21G
   - 6-31G
   - 6-31G(d)
   - 6-31G(d,p)
   - 6-311G(d,p)

   Basis must strictly be typed in these displayed forms, only for that could Elevia recognize. The source code about reading basis data (`dat/basis/*.g94`) into the program is included in `intro_basis.cpp`.

3. Others

   Charge and spin multiplicity are not considered currently.

4 Development details

4.1 Files

- headers
  - Global.h

    contains global class definitions: `atom`, `system`, `fio`.
  - umath.h

    aims to establish a versatile uovie math library. But now it only contains factorial functions.
  - Mol_Int.h

    invokes libint2 library to calculate molecular integrals.
- .cpp
  - main.cpp

    includes basic flow of this program for the beauty of simplicity :)
  - open.cpp

    contains the definition of member function `open(*,*)` in the class `fio`.
  - intro_basis.cpp

    includes a `intro_basis(*)` function, which can introduce a basis set according to a given system configuration.
  - core.cpp

    is the key part of Elevia.
  - close.cpp

    closes those opened files before terminating program.

4.2 Procedures [1,2,3]

1. read system information into program
2. count the number of electrons and calculate the number of doubly occupied orbitals
3. calculate nuclear repulsion energy V_{nn}
4. introduce a basis set
5. compute single-electron integrals
    - overlap integrals $\mathbf{S}$
    - kinetic energy integrals $\mathbf{T}_{\mathrm{e}}$
    - nuclear attraction integrals $\mathbf{V}_{\mathrm{ne}}$
    - core Hamiltonian $\mathbf{H}_{\mathrm{core}} = \mathbf{T}_{\mathrm{e}} + \mathbf{V}_{\mathrm{ne}}$
6. assume an initial electron-charge bond-order matrix $\mathbf{P}$
    - use Superposition-Of-Atomic-Densities (SAD) guess for minimal basis sets
    - use core Hamiltonian guess for other basis sets
7. main iterative loop (SCF)
    - record initial time
        - code

        ```
        const auto tstart = std::chrono::high_resolution_clock::now();
        ```

    - Save a copy of the energy and the density for comparison
    - build a new Fock matrix $\mathbf{F} = \mathbf{H}_{\mathrm{core}} + \mathbf{PQ}$, $\mathbf{Q}$ is a two-electron repulsion integral matrix
        - implementation

        ```
        auto F = Hcore;
        F += Elevia::two_body_fock(shells, P);
        ```

    - solve $\mathbf{FC} = \boldsymbol{\epsilon}\mathbf{SC}$, with the aid of an Eigen module `GeneralizedSelfAdjointEigenSolver` defined in the header `Eigen\src\Eigenvalues\GeneralizedSelfAdjointEigenSolver.h`.
        - Account for Eigen eigenvalues_module `GeneralizedSelfAdjointEigenSolver` It Computes eigenvalues and eigenvectors of the generalized self-adjoint eigen problem. This class solves the generalized eigenvalue problem $\mathbf{Av} = \boldsymbol{\lambda}\mathbf{Bv}$. In this case, the matrix $\mathbf{A}$ should be self-adjoint and the matrix $\mathbf{B}$ should be positive definite.
    - compute density, $\mathbf{P} = \mathbf{C}_{\mathrm{occ}}^{\mathrm{T}}\mathbf{C}_{\mathrm{occ}}$, note that $\mathbf{P}$ in this context is not contain an efficient 2.
    - compute HF energy, $E_{\mathrm{elec}} = \mathrm{Tr}[\mathbf{P}(\mathbf{H}_{\mathrm{core}} + \mathbf{F})]$
        - Due the symmetry of $\mathbf{P}$, we can straightforwardly carry out as follows

        ```
        Eelec = 0.0;
            for (auto i = 0; i < nao; i++)
                for (auto j = 0; j < nao; j++)
                    Eelec += P(i, j) * (Hcore(i, j) + F(i, j));
        ```

    - compute difference with last iteration

        - $\Delta E = E_{\text{elec}} - E_{\text{elec\_last}}$
        - $\text{RMSD} = (\mathbf{P} - \mathbf{P}_{\text{last}}).\text{norm}()$, note that `.norm()` calculate Frobenius norm of a matrix
    - calculate time elapsed in the iteration
        - code

```
const auto tstop = std::chrono::high_resolution_clock::now();
const std::chrono::duration<double> time_elapsed = tstop - tstart;
```

    - print the results of this iteration
    - check if it meets the convergence condition or arrives at the max number of Iterations
        - if not, enter into next iteration
        - if so, terminate the loop
8. calculate the total Hartree-Fock energy, $E_{\text{tot}} = E_{\text{elec}} + V_{\text{nn}}$
9. normal termination. Congratulations!
10. close files and close Elevia.

5 Tests

Compare total energy with Gaussian 09W. The test files can be found in `test` directory.

1. STO-3G

Molecules	Elevia / a.u.	Gaussian 09W / a.u.
H_2	-1.11750589	-1.1175059
HF	-98.57284734	-98.5728474
O_2	-147.55157167	-147.5515717
H_2O	-74.96072325	-74.9607233

2. 3-21G

Molecules	Elevia / a.u.	Gaussian 09W / a.u.
H_2	-1.12256431	-1.1225643
HF	-99.45988912	-99.4598891
O_2	-148.68659130	-148.6865913
H_2O	-75.58581250	-75.5858125

3. 6-31G(d,p)

Molecules	Elevia / a.u.	Gaussian 09W / a.u.
H_2	-1.13101745	-1.1310174
HF	-100.00824819	-100.0082482
O_2	-149.52702867	-149.5270287
H_2O	-76.02255415	-76.0225541

The results of Elevia coincides well with the computational outcomes of Gausssian 09W.

References

[1] libint/tests: https://github.com/evaleev/libint/tree/master/tests

[2] L. Piela. *Ideas of Quantum Chemistry*. Elsevier, USA, second edition, 2014.

[3] A. Szabo and N. S. Ostlund. *Modern Quantum Chemistry*. Dover Publications, USA, 1996.

[4] S. Obara and A. Saika. Efficient recursive computation of molecular integrals over cartesian gaussian functions. *J. Chem. Phys.*, 84(7):3963–3974, 1986.