SoftwareX 7 (2018) 107–111

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

Ergo: An open-source program for linear-scaling electronic structure calculations

Elias Rudberg [a,*], Emanuel H. Rubensson [a], Paweł Sałek [b], Anastasia Kruchinina [a]

[a] Division of Scientific Computing, Department of Information Technology, Uppsala University, 751 05 Uppsala, Sweden
[b] PS Consulting, ul. Zaporoska 8/4, 30-389 Kraków, Poland

ARTICLE INFO

Article history:
Received 29 December 2017
Received in revised form 17 March 2018
Accepted 19 March 2018

Keywords:
Electronic structure
Quantum chemistry
Hartree–Fock
Kohn–Sham density functional theory

ABSTRACT

Ergo is a C++ program for all-electron Hartree–Fock and Kohn–Sham density functional theory electronic structure calculations using Gaussian basis sets. The program uses algorithms for which the computational cost increases linearly with system size for all parts of the calculation, including computation of the Fock/Kohn–Sham matrix and density matrix construction. Both spin-restricted and unrestricted calculations are supported, and both pure and hybrid density functionals. The program also supports linear-scaling computation of highest occupied molecular orbital (HOMO) and lowest unoccupied molecular orbital (LUMO) vectors. This paper briefly describes how the code is organized and provides examples of how it can be used.

Code metadata

Current code version	3.6
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-18-00001
Legal Code License	GPL v3
Code versioning system used	git
Software code languages, tools, and services used	C++, C, BLAS, LAPACK, autoconf, automake
Compilation requirements, operating environments & dependencies	Linux/MacOS/other Unix-like systems
If available Link to developer documentation/manual	http://ergoscf.org/
Support email for questions	info@ergoscf.org

1. Motivation and significance

Electronic structure calculations are of fundamental importance in a wide range of scientific disciplines, including physics, chemistry, and materials science. In this article we describe Ergo, an open-source program that can perform electronic structure calculations using the Hartree–Fock (HF) or Kohn–Sham density functional theory (KS-DFT) models.

Straightforward implementation of the HF and KS-DFT models leads to cubic scaling of the computational cost with system size, $\mathcal{O}(N^3)$, or worse, where N is the number of atoms in the studied system. Much research has been devoted to the development of various linear scaling, $\mathcal{O}(N)$, approaches making the methods applicable also for larger molecular systems [1]. The Ergo program makes use of such linear-scaling algorithms for all parts in a HF or KS-DFT calculation, using Gaussian basis sets.

Ergo implements a number of linear-scaling algorithms for different parts of the calculation, including multipole methods for the Coulomb interaction [2], linear-scaling construction of the HF exchange and KS-DFT exchange correlation matrices directly in sparse form [3,4], and density matrix purification with rigorous error control [5] and automatic stopping criterion [6]. Sparse matrices are stored and operated on using a hierarchical block-sparse matrix data structure [7].

Apart from being a platform for method development research related to electronic structure calculations, previous versions of the Ergo code have been used for example for studying properties of graphene nanoribbons [8], for computation of solvent–solute interaction energies [9], in studies of matrix sparsity [10], and in investigations of computational challenges related to different kinds of density functionals when applied to large protein-like systems [11].

* Corresponding author.
E-mail addresses: elias.rudberg@it.uu.se (E. Rudberg), emanuel.rubensson@it.uu.se (E.H. Rubensson), pawsa0@gmail.com (P. Sałek), anastasia.kruchinina@it.uu.se (A. Kruchinina).

https://doi.org/10.1016/j.softx.2018.03.005

2. Software description

Ergo is written mainly in C++, with parts of the code for different KS-DFT functionals written in C. The program is normally run from the command line in a Linux/Unix-like environment, giving as input a file with xyz coordinates for the molecule and another file with parameters such as what basis set to use and various other settings. The execution generates a text file with results such as computed total energy and dipole moment, as well as a binary file containing the computed electron density that can be used as input for other calculations later on.

2.1. Software architecture

The source code is divided into a set of modules, each in a separate sub-directory. The modules that provide the main functionality are the following:

- `basisset` : code for managing Gaussian-type orbital basis sets; text files with specifications for a set of common basis sets are available in the `basis` directory.
- `densfromf` : code for construction of the density matrix from the Fock matrix using linear-scaling methods based on block-sparse matrix–matrix multiplication.
- `dft` : implementation of KS-DFT functionals, with construction of the KS-DFT exchange–correlation potential matrix and exchange–correlation contributions to linear response calculations directly in sparse form.
- `integrals` : code related to electron repulsion integrals, including linear-scaling Coulomb (J) and Hartree–Fock exchange (K) matrix construction.
- `matrix` : hierarchical matrix library (HML) [7] for block-sparse matrix operations.
- `scf` : code managing self-consistent field (SCF) iterations. A virtual base class `SCF_general` is used for functionality that is needed for both spin-restricted and unrestricted calculations, with functionality specific for those cases in classes `SCF_restricted` and `SCF_unrestricted`, respectively.
- `test` : unit tests testing the functionality in the other modules. Each file in the `source/test` directory gives a separate executable program that tests some specific part of the functionality in the other modules.
- `utilities` : utilities used by other modules, including conversions to/from HML format.
- `utilities_basic` : basic utilities used by other modules, not dependent on matrix library.

There are also the following modules, less important for the main linear-scaling functionality:

- `ci` : experimental configuration interaction (CI) implementation, that can be used after a HF calculation.
- `electron_dynamics` : simple electron dynamics implementation, so far without truncation so not linear scaling.

The main program in `ergo_scripted.cc` uses yacc/bison and flex for input parsing.

Shared-memory parallelization is handled using Pthreads in some parts and OpenMP in some other parts of the code. The default number of threads used is determined by checking the `OMP_NUM_THREADS` environment variable. The number of threads can also be specified for individual parts of the code using parameters in the input file, e.g. `J_K.threads_J = 4`.

Source code comments are written in a format compatible with Doxygen; running the `doxygen` command generates HTML-formatted documentation in the `documentation` directory.

2.2. Configuration, compilation, and test suite

Ergo is configured by running the `configure` script. Running "`./configure -h`" gives a list of available configuration options. After running the `configure` script, the program is built using `make` with the resulting main executable `ergo` ending up in the `source` directory. Then the test suite can be run using "`make check`". Each test runs in a separate directory so that parallel make using "`make check -j`" also works.

The configure script and makefiles are generated using the autoconf and automake tools and controlled by the `configure.ac` and `Makefile.am` files. The `bootstrap.sh` script contains the relevant autoconf and automake commands. Examples of how Ergo can be configured with different compiler optimization flags etc. for different systems are available in the `config_examples.txt` file.

Code coverage checking can be turned on using the `--enable-coverage` configure option. Then, after running the test suite a tool such as `gcov` or `lcov` can be used to check the test coverage.

2.3. Software functionalities

The main functionality of the Ergo program is the ability to perform large-scale HF and KS-DFT calculations using linear-scaling techniques. Both spin-restricted and unrestricted calculations are supported. The program performs all-electron calculations; effective core potentials are not used. Sparse matrices and other sparse data structures are used in all parts of the calculation, leading to linear scaling of both computational time and memory usage with system size. The program uses Gaussian-type orbital (GTO) basis sets that are common in quantum chemistry programs, making it possible to directly compare results to other quantum chemistry programs using such basis sets. Spin–orbit coupling effects are not taken into account.

In each SCF iteration, the current density matrix is written to a file called `density.bin` that also includes information about the basis set used. Such a saved density file can then be used as a starting guess for another calculation. If the new calculation uses a different basis set, a basis set projection is performed taking into account the basis set information for the saved density.

Construction of the Fock/Kohn–Sham matrix is performed using a multipole method with dynamically truncated multipole expansions [2] and linear-scaling construction of the HF exchange matrix directly in sparse form [3]. In KS-DFT calculations, the exchange–correlation matrix is also computed directly in sparse form [4].

Linear-scaling density matrix construction is achieved using density matrix purification with rigorous error control [5] using a recently developed automatic stopping criterion [6]. The implemented purification methods include also nonmonotonic expansions [12].

A number of common KS-DFT functionals are implemented, including both local-density approximation (LDA) and generalized gradient approximation (GGA) functionals such as BLYP and PBE. Hybrid functionals like B3LYP and PBE0 that include a fraction of HF exchange are also available, as well as range-separated hybrid functionals such as CAM-B3LYP. See [4] for details about the exchange–correlation matrix computation. Note that hybrid functionals can help alleviate the problems with too small HOMO–LUMO gaps encountered for pure functionals, see e.g. [11].

The direct inversion in the iterative subspace (DIIS) [13] scheme is used for SCF convergence acceleration. In cases where DIIS does not work the code resorts to damping with reduced step length if the energy increases. The SCF procedure is considered converged when the largest magnitude element of the matrix $FDS - SDF$ is smaller than the chosen SCF convergence threshold value, see [4].

Linear-scaling computation of highest occupied molecular orbital (HOMO) and lowest unoccupied molecular orbital (LUMO)