Contents lists available at ScienceDirect

SoftwareX

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

Original software publication

JDFTx: Software for joint density-functional theory

Ravishankar Sundararaman [a,*], Kendra Letchworth-Weaver [b], Kathleen A. Schwarz [c], Deniz Gunceler [d], Yalcin Ozhabes [d], T.A. Arias [d]

[a] Department of Materials Science and Engineering, Rensselaer Polytechnic Institute, Troy, NY, 12180, United States
[b] Center for Nanoscale Materials, Argonne National Laboratory, Lemont, IL 60439, United States
[c] National Institute of Standards and Technology, Material Measurement Laboratory, Gaithersburg, MD, 20899, United States
[d] Department of Physics, Cornell University, Ithaca, NY 14853, United States

A R T I C L E   I N F O

Article history:
Received 13 August 2017
Received in revised form 17 October 2017
Accepted 18 October 2017

Keywords:
Density functional theory
Electronic structure
Solvation
Electrochemistry
Light-matter interactions

A B S T R A C T

Density-functional theory (DFT) has revolutionized computational prediction of atomic-scale properties from first principles in physics, chemistry and materials science. Continuing development of new methods is necessary for accurate predictions of new classes of materials and properties, and for connecting to nano- and mesoscale properties using coarse-grained theories. JDFTx is a fully-featured open-source electronic DFT software designed specifically to facilitate rapid development of new theories, models and algorithms. Using an algebraic formulation as an abstraction layer, compact C++11 code automatically performs well on diverse hardware including GPUs (Graphics Processing Units). This code hosts the development of joint density-functional theory (JDFT) that combines electronic DFT with classical DFT and continuum models of liquids for first-principles calculations of solvated and electrochemical systems. In addition, the modular nature of the code makes it easy to extend and interface with, facilitating the development of multi-scale toolkits that connect to *ab initio* calculations, e.g. photo-excited carrier dynamics combining electron and phonon calculations with electromagnetic simulations.

Code metadata

Current code version	1.3.1
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX_SOFTX-D-17-00063
Legal Code License	GPLv3
Code versioning system used	git
Software code languages, tools, and services used	C++11, MPI, CUDA
Compilation requirements, operating environments & dependencies	GSL, BLAS, LAPACK and FFTW libraries on a POSIX-compliant platform
Link to developer documentation/manual	http://jdftx.org
Support email for questions	sundar@rpi.edu

Software metadata

Current software version	1.3.1
Permanent link to executables of this version	https://github.com/ElsevierSoftwareX/SOFTX_SOFTX-D-17-00063
Legal Software License	GPLv3
Computing platforms/Operating Systems	POSIX-compliant platform (Linux, Unix, OS X, Windows/Cygwin etc.)
Installation requirements & dependencies	MPI C++11 compiler; GSL, BLAS, LAPACK and FFTW libraries
Link to user manual	http://jdftx.org
Support email for questions	sundar@rpi.edu

* Corresponding author.
E-mail address: sundar@rpi.edu (R. Sundararaman).

https://doi.org/10.1016/j.softx.2017.10.006

1. Motivation and significance

Density functional theory (DFT) enables computational prediction of material properties and chemical reactions starting from a quantum mechanical description of the electrons. DFT codes are now widely used to understand and design new materials from first principles through the prediction of electronic properties, structures and dynamics of molecules, solids and surfaces. Such studies commonly employ proprietary software such as GAUSSIAN [1] and VASP [2] as well as open-source software such as Quantum Espresso [3], ABINIT [4] and Qbox [5], to name just a few.[1]

However, DFT offers limited accuracy for certain classes of materials and properties [6], and is extremely computationally expensive for amorphous materials, liquids and nanostructures [7]. The study of new systems, as soon as they become technologically and scientifically relevant, requires continual development of new methods to improve the accuracy of DFT and incorporate it into multi-scale theories to access higher length scales. Developing and testing such methods within production codes is extremely challenging and time consuming.

Systems involving liquids, such as electrochemical interfaces or solvated biomolecules, are particularly challenging for DFT calculations, requiring thermodynamic sampling of several thousands of atomic configurations in *ab initio* molecular dynamics (AIMD) simulations [8]. Joint density-functional theory (JDFT) was proposed as a theoretical framework to address this issue by combining electronic DFT with classical DFT of liquids [9] to directly compute equilibrium properties of quantum-mechanically described solutes in diverse solvent environments [10]. Bringing this method to fruition required the simultaneous development of physical models (free energy functionals) of liquids and their interaction with electrons, algorithms to perform variational free-energy minimization and code that tightly and efficiently coupled these new models with electronic DFT. We began the open-source software project JDFTx in 2012 to facilitate this combined model, algorithm and code development effort.

This article introduces JDFTx as a general-purpose user-friendly DFT software that offers a full feature set, yet is simultaneously developer-friendly to enable rapid prototyping of new electronic-structure and related methods. Section 2 presents the overall design of JDFTx using the algebraic formulation of DFT [11,12] to separate implementation into physics, algorithm and hardware layers, enabling rapid development of high-performance code that is easy to use. It also outlines commonly used features of the code, some of which are illustrated in more detail with examples in Section 3. Finally, Section 4 highlights new methods that have already been developed using JDFTx, including a hierarchy of JDFT models for the electronic structure of solvated systems and a toolkit for photo-excited carrier dynamics with *ab initio* electron and phonon properties, as well as key applications of these methods.

2. Software description

The core functionality of any electronic DFT software includes the calculation of ground-state electron densities, energies and forces within the Kohn–Sham DFT formalism [13], given a list of atoms and their positions. This facilitates prediction of structure and dynamics of materials, evaluation of reaction pathways and chemical kinetics, as well as determination of phase equilibria and stability.

[1] Certain commercial software codes are identified in this paper to foster understanding. Such identification does not imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that the materials or equipment identified are necessarily the best available for the purpose.

Due to the nature of quantum-mechanical simulations of matter, DFT calculations become increasingly expensive with the number of atoms and electrons involved; computational complexity ranges from $O(N^3)$ (with a smaller prefactor) to $O(N)$ (with a much larger prefactor), depending on the implementation. A brute force approach to nano and mesoscale systems with several thousands to millions of atoms is therefore not practical; it is instead logical to develop multi-scale theories for the properties of interest while still incorporating DFT electronic structure where appropriate.

JDFTx is an open-source DFT software designed specifically with the goals of coupling electronic DFT with coarse-grained theories to bridge atomic and system length scales, and of facilitating the rapid development of new classes of such combined theories. It implements electronic DFT in the plane-wave basis, which is best suited for periodic systems such as solids and solid surfaces, but is also applicable to molecular systems. A key functionality of JDFTx beyond standard electronic DFT codes is the modeling of liquids using classical DFT [12], and JDFT calculations of electronic structure in liquid environments by combining electronic DFT with classical DFT or simpler solvation models. Section 2.1 presents a birds-eye view of the code architecture along with a code example to illustrate the ease of developing new features, after which Section 2.2 outlines the key features of the code.

2.1. Software architecture

JDFTx achieves its goal of code simplicity and extensibility by using the 'DFT++' algebraic formulation of electronic DFT [11] and its generalization to classical DFT and JDFT [12]. This algebraic formulation cleanly separates the code into physics, algorithm and computational layers. Theories and algorithms are expressed concisely at a high-level of abstraction in the top layers of the code, whereas performance optimizations and support for specialized hardware such as GPUs are handled in the lower layers.

We illustrate this clean separation with an example of a simplified solvation model defined by

$$-4\pi\rho(\vec{r}) = \nabla\cdot(\epsilon(\vec{r})\nabla\phi(\vec{r})), \quad \text{and} \tag{1}$$

$$A_{\text{diel}} = \frac{1}{2}\int \mathrm{d}r \left[\phi(\vec{r}) - \hat{K}\rho(\vec{r})\right]\rho(\vec{r}). \tag{2}$$

Here, the liquid is treated as an inhomogeneous dielectric $\epsilon(\vec{r})$ which interacts with the charge density of the electronic system, $\rho(\vec{r})$. The net electrostatic potential $\phi(\vec{r})$ satisfies the modified Poisson equation (1), and the electrostatic solvation energy A_{diel} is the difference between the dielectric-screened and unscreened electrostatic self energies of $\rho(\vec{r})$ (2), where $\hat{K}$ is the unscreened Coulomb operator. This is the essence of most solvation models used with DFT [14–17]; we have only skipped the determination of $\epsilon(\vec{r})$ from atom positions or electron densities, and additional non-electrostatic correction terms in A_{diel} for brevity. Regardless, solving the Poisson equation above is the most complex and time-consuming operation in these solvation models.

Listing 1 shows the implementation of this model in JDFTx. Class *LinearPCM* derives from a templated abstract base class *LinearSolvable*, which implements the Conjugate Gradients (CG) algorithm [18] on arbitrary vector spaces, instantiated in this case for scalar fields in reciprocal space, *ScalarFieldTilde*. The equation to be solved is defined by the virtual function *hessian*, whose one-line implementation can be recognized as (1) at a moment's glance. Note that the *gradient* (∇) and *divergence* ($\nabla\cdot$) operators apply in reciprocal space (where they are diagonal), while the operators I and J Fourier transform from reciprocal to real space and vice versa [11] (using Fast Fourier Transforms). The function *getAdiel* first calls function *solve* from base class LinearSolvable to solve (1) for $\phi(\vec{r})$ stored in a member variable *state* of the base class, and the second line evaluates A_{diel} using (2). The integral is evaluated as a