NanoPSE: Nanoscience Problem Solving Environment for atomistic electronic structure of semiconductor nanostructures

2005 J. Phys.: Conf. Ser. 16 277

(http://iopscience.iop.org/1742-6596/16/1/038)

Institute of Physics Publishing Journal of Physics: Conference Series **16** (2005) 277–282
doi:10.1088/1742-6596/16/1/038 SciDAC 2005

NanoPSE: Nanoscience Problem Solving Environment for atomistic electronic structure of semiconductor nanostructures

Wesley B Jones[1], Gabriel Bester[1], Andrew Canning[2], Alberto Franceschetti[1], Peter A Graf[1], Kwiseon Kim[1], Julien Langou[3], Lin-Wang Wang[2], Jack Dongarra[3] and Alex Zunger[1]

[1] National Renewable Energy Laboratory, Golden, Colorado 80401, U.S.A.
[2] Lawrence Berkeley National Laboratory, Berkeley, California 94720, U.S.A.
[3] Innovative Computing Laboratory, Department of Computer Science, The University of Tennessee, Knoxville, Tennessee 37996-3450, U.S.A.

E-mail: wesley_jones@nrel.gov

Abstract. Researchers at the National Renewable Energy Laboratory and their collaborators have developed over the past ∼10 years a set of algorithms for an atomistic description of the electronic structure of nanostructures, based on plane-wave pseudopotentials and configuration-interaction. The present contribution describes the first step in assembling these various codes into a single, portable, integrated set of software packages. This package is part of an ongoing research project in the development stage. Components of NanoPSE include codes for atomistic nanostructure generation and passivation, valence force field model for atomic relaxation, code for potential field generation, empirical pseudopotential method solver, strained linear combination of bulk bands method solver, configuration interaction solver for excited states, selection of linear algebra methods, and several inverse band structure solvers. Although not available for general distribution at this time as it is being developed and tested, the design goal of the NanoPSE software is to provide a software context for collaboration. The software package is enabled by fcdev, an integrated collection of best practice GNU software for open source development and distribution augmented to better support FORTRAN.

1. Introduction

Through the Nanoscience Theory grant solicitation, LAB-03-17, our collaboration of physicists, computational scientists and mathematicians is funded under the grant entitled, "Predicting the Electronic Properties of 3D, Million-Atom Semiconductor Nanostructure Architectures." This includes the investigation of the electronic structure of quantum dots [1]-[18] and solving the inverse band structure problem [19, 20]. To solve this problem we cast the inverse problem as an optimization problem and run the forward solvers (i.e., "direct" nanostructure calculations) many times in an automated fashion controlled by an optimization algorithm. NanoPSE was born out of the need to automatically and robustly call the forward solvers along with the requirement to have a standard set of codes. Its design is expected to lead to a software package architecture that will enable collaboration among a diverse group of researchers. NanoPSE is not available for general distribution at this time as it is part of a research project in the development

stage. A description of the NanoPSE software, software architecture and supporting software is the subject of this paper.

In Section 2 we briefly summarize the original codes that are included in the NanoPSE software package. Section 3 describes the design of the NanoPSE software package and Section 4 describes the supporting software for the package, fcdev, based on GNU autotools [21].

2. NanoPSE Components

The most important pieces of the NanoPSE software distribution are the physics codes and mathematical algorithms contained in the package. In the present context we cannot give due justice to the physics problems that may be addressed or the complexity and range of applicability of the underlying algorithms. Here, we simply give references, a very brief description of the components contained in the NanoPSE software package and name the supporting libraries required.

2.1. Generation of atomic coordinates, shapes and its surface passivation (NanoStruct)

NanoPSE currently contains two codes for generating atomic coordinates and shapes of complex nanostructures and an algorithm for placing surface passivant atoms on these structures. These replace the previous passivation codes of [1].

2.2. Relaxation of atomic coordinates via Valence Force Field (VFF)

Keating's VFF was applied to atomic relaxation in alloys in 1984 by Martins and Zunger [2]. This two-parameter potential was generalized to more parameters first by Silverman et al. [3] and by Fu et al. [4]. The algorithm of calculating the relaxation was developed by J. Kim et al. [5] and by K. Kim et al. [6]. To calculate the relaxed atomic positions the strain energy in the system is minimized and the positions are obtained starting from their ideal tetrahedral-bonded configuration. Examples of applications are given in [6, 7].

2.3. Constructing a screened pseudopotential

The original "Empirical Pseudopotential Method" (EPM) is modified by deriving it from LDA, following empirical adjustments to overcome the "LDA errors" [8]. The codes performing this are not incorporated yet into NanoPSE. A library of potentials is being constructed.

2.4. Solving the plane-wave pseudopotential Schrödinger equations via the Folded Spectrum Method (FSM)

To solve the Schrödinger equation, for the single particle eigenvalues using a fixed empirical pseudopotential we use the folded spectrum method [9] parallelized in [10]. The combination of the use of the empirical pseudopotential method with the folded spectrum method to solve for electronic states near conduction and valence band edges of the nanostructure allows million atom systems to be addressed. Examples of applications are given in [11, 12].

2.5. Solving the plane-wave pseudopotential Schrödinger equation via the Strained Linear Combination of Bulk Bands (SLCBB) Method

In the SLCBB method [13] the Schrödinger equation is solved non-self-consistently like, in the previous case, but the wave functions are expanded using a basis of strained Bloch functions calculated from Bulk materials. The method has been developed further by S. Nair et al. [14]. The advantage of this method is a strongly reduced size of the basis and a sub-linear scaling. The computational cost of the method mainly scales with the complexity of the system, not its size. Examples of applications are given in [15, 16].

2.6. Coulomb and exchange matrix elements and the configuration interaction calculation for excited states (MX)
Once the single-particle Schrödinger equation has been solved using methods 2.4 and 2.5, the ensuing single particle energies and wave functions are used to construct the basis set for the configuration interaction (CI) Hamiltonian. This requires the calculation of four-point Coulomb and exchange integrals between single-particle wave functions. The CI Hamiltonian is then iteratively diagonalized to obtain the electronic excited states (e.g. excitons, multi excitons, etc.) of the nanostructure [17]. Examples of applications are given in [18].

2.7. Inverse band structure solver
The inverse band structure problem [19] (i.e. for a given band structure, find the corresponding atomic configuration) is cast as an optimization problem and solved via a hierarchical parallel genetic algorithm [20]. This code uses a number of components recast as subroutines solving the forward problem (i.e. for a given atomic configuration, find the corresponding band structure) many times to find the optimal solution. Applications are given in [20].

2.8. Selection of iterative methods for eigenvalue computation
Interior eigenvalue problems are one of the common difficult mathematical problems that SLCBB and PESCAN need to solve [22]. Solvers to address these problems will need to feature: speed, robustness, accuracy, parallel distributed memory, a common API based on reverse communication, spectral transformation for interior eigenvalues, block algorithm, use of preconditioner for acceleration of the convergence, focus on the Hermitian case, and modified structure of the matrix when the k-point is 0. The most promising candidate to improve upon currently implemented FSM solvers is the LOBPCG method.

2.9. Supporting software and libraries
The following supporting libraries are called by NanoPSE: FFTW, BLAS, LAPACK, ARPACK, HDF5, PGAPACK and MPI.

3. NanoPSE Software Package Design

The NanoPSE software package was implemented to enable the optimization solvers to more easily interact with the forward solvers and allow for more rapid implementation of different types of optimization problems. The design has proven useful for the latter case and in addition contains the current standard version of codes and helps enable collaboration among the current researchers and code developers. The properties that have made this possible are as follows:

- A comprehensive software context;
- Integration with a revision control system, the concurrent version system (CVS);
- Division of the software and responsibility into software components enforced by CVS;
- Portability to many architectures, including the development environment;
- Ease of integration and maintenance of new software components via the fcdev framework;
- Integration into a software management system (i.e. modules) for ease of developer and end user interaction with multiple revisions.

A comprehensive software context is a collection of all of the software needed to solve a problem or a set of problems with clearly defined interfaces to software not contained in the collection. In our case an interface is simply defined by the application-programming interface (API) of the supporting libraries. Having a clear comprehensive software context, at least as a base, distinctly defines the details of investigations that are immediately possible and separable from larger, more interesting questions concerning physics and math algorithms.

NanoPSE is stored in CVS. The revision control system starts by helping code developers on a day-to-day basis more rapidly develop codes, enables the tracking of regressions, and allows the NanoPSE software manager to more easily ensure the integrity of a release.

NanoPSE is separated into components. Each component is a software package unto itself so that it may be licensed, tested and released individually. With each component is associated a primary contributing researcher who is either responsible for the code or contributed to the code. The separation of components is enforced via UNIX permissions within CVS so that only a subset of the researchers may make changes to a component. Each component, however, is integrated into the NanoPSE hierarchy so that an end user may configure, make, and install all or a subset of the NanoPSE components easily.

NanoPSE is ported to a number of architectures and, due to the inherent portability enabled by fcdev, should be easily ported to other platforms when the need arises. The uniform build environment allows us to more easily adapt to new architectures by identifying and addressing issues once and having the solution easily propagate to all of the application codes. Currently the NanoPSE software distribution is known to run on IA32-Linux, AMD64-Linux, IA64-Linux, MAC OS X, IBM AIX, and SGI Irix. The entire software package may be built, a subset of the software package may be built or the entire software package minus a subset may be built. This enables porting specific components individually to new architectures and allows developers to concentrate on their own software.

The ease with which new software components are integrated into and tested for the overall architecture allows for a software distribution manager to manage specific components that cannot be managed directly by the developers who have contributed or are responsible for a component. This is enabled by fcdev.

The ease with which NanoPSE and fcdev integrate into a software management system such as modules [23], enhanced with pkg-config, allows for the support and testing of new versions of NanoPSE as well as new revisions of libraries. With automatic query and configuration via pkg-config, we are able to load one set of modules, build and test the software, and compare against the same software built with a different set of loaded modules. In addition, end users are more easily able to load a module with one version of NanoPSE and test against a different version of NanoPSE.

The properties of the NanoPSE software distribution described above have helped us in a number of ways and at the same time have not increased the net burden on the researchers. In some cases the system has helped collaborators with an incomplete knowledge of the whole system, enhance the system and have their enhancements tested. This includes the work to the individual codes to make them more portable and ported to various architectures. Without the software context and components provided by NanoPSE, linking the forward solvers together for optimization — especially with support for hierarchical parallelism — would be virtually impossible. Thus, the NanoPSE software distribution design has enabled a broader set of problems to be investigated using the inverse band structure method than might otherwise have been the case.

4. Fcdev

The GNU Autotools — autoconf, automake, libtool — provide a foundational mechanism for developers to construct source code software distributions. These tools are widely used in the open source community and define the de-facto standard software distribution development environment in the Linux open source community.

Once constructed, software distributions based on GNU autotools have a number of characteristics that make them very desirable. First, the near universal use of open source software distributions based on these tools makes knowledge about these tools easy to obtain, either via in house staff expertise or external web based knowledge. Second, the software

distributions can be made such that installation across the platform spectrum is possible, including Linux, Unix, MAC and Windows. Third, the installation process can be easy or at a minimum diagnosable via the first characteristic.

The problems with these tools for scientific software development are twofold. The first is that the support of the tools for FORTRAN is minimal, archaic, and/or not integrated across the toolset. Autoconf-2.58 and later is the partial exception. The second is a lack of ease of use and access to implementation specifics, especially when it comes to using FORTRAN. We have addressed the first by creating fcdev and the second by creating fcdeveg, examples of fcdev for the beginner.

Fcdev is an integrated collection of best practice GNU software for open source development and distribution including autoconf, automake, libtool and pkg-config. This software is augmented to better support FORTRAN by enhancing the underlying development software packages and includes extra tools, e.g. fcdep, and fcdeveg. fcdeveg is designed such that it integrates well with a revision control system such as CVS and with a software management system such as modules [23].

Fcdev has the following software components: 1) Autoconf-2.58, 2) Automake-1.7.6 : modified, 3) Libtool-1.4.3, 4) Pkgconfig-0.15.2: modified, 5) Fcdep-0.1.1, 6) Fcdeveg-0.1.0, 7) M4-1.4 (or equivalent) : usually already installed. We worked with the autoconf developers to include desired features for support of FORTRAN and follow their conventions, such that autoconf-2.58 does not need to be modified for our use.

Fcdev and fcdeveg have the following features:

- Ease of creation of FORTRAN distributions from many FORTRAN source files
- Support for industry standard preprocessing via the C preprocessor with fixed form extension, .F and .fpp and free form extensions, .F90 and .f90pp
- Support for industry standard extensions of fixed form, .f and free form, .f90
- Automatic dependency resolution of FORTRAN90 modules: fcdep.pl
- Automatic library API compiling and linking resolution: pkg-config −−variable=FCFLAGS −−variable=FCLIBS
- Automatic with environmental variable override for most features such as specification of compilers and library linking
- Examples for the features in full software distribution form
- Macros and environmental variable settings for Linux-IA32, Linux-IA64, Linux-AMD64, IBM AIX, SGI Irix, MAC OS X
- Integrated compatibility with CVS or other revision control system
- Integrated compatibility with modules or other software management system

Although not complete, especially with respect to our prototype support for configuration of C preprocessing of FORTRAN files in automake, fcdev contains all of the pieces for, is a context for, and represents another step toward enabling computational scientists who support FORTRAN applications to more easily use GNU autotools for creating and maintaining software distributions.

This work was supported by U.S. DOE-SC-ASCR-MICS through LAB-03-17 Theory Modeling in Nanoscience Initiative under Contract No. DE-AC36-99GO10337.

References

[1] Wang L-W and Zunger A 1996 *Studies in Surface Science and Catalysis* vol 103 ed P V Kamat and D Meisel (Amsterdam, Elsevier Science) p 161
[2] Martins J L and Zunger A 1984 *Phys. Rev.* B **30** R6217
[3] Silverman A, Zunger A, Kalish R and Adler J 1995 *Phys. Rev.* B **51** 10795

[4] Fu H, Ozolins V and Zunger A 1998 *Phys. Rev.* B **59** 2881
[5] Kim J, Wang L-W and Zunger A 1998 *Phys. Rev.* B **57** R9408; Wang L-W, Kim J and Zunger A 1999 *Phys. Rev.* B **59** 5678
[6] Kim K, Kent P R C, Zunger A and Geller C B 2002 *Phys. Rev.* B **66** 045208
[7] Shumway J, Williamson A J, Zunger A, Passaeso A, DeGiorgi M, Cingolani R, Catalano M and Crozier P 2001 *Phys. Rev.* B **64** 125302
[8] Wang L-W and Zunger A 1995 *Phys. Rev.* B **51** 17398; Fu H and Zunger A 1997 *Phys. Rev.* B **55** 1642
[9] Wang L-W and Zunger A 1994 *J. Chem. Phys.* **100** 2394
[10] Canning A, Wang L-W, Williamson A J and Zunger A 2000 *J. Comput. Phys.* **160** 29
[11] Fu H and Zunger A 1997 *Phys. Rev.* B **56** 1496; Fu H and Zunger A 1998 *Phys. Rev. Lett.* **80** 5397
[12] Franceschetti A, Williamson A J and Zunger A 2000 *J. Phys. Chem.* B **104** 3398
[13] Wang L-W and Zunger A 1999 *Phys. Rev.* B **59** 15806
[14] Bester G, Nair S and Zunger A, submitted to Phys. Rev. B, Rapid Communications.
[15] Shumway J, Franceschetti A and Zunger A 2001 *Phys. Rev* B **63** 155316
[16] Bester G and Zunger A 2003 *Phys. Rev.* B **68** 073309
[17] Franceschetti A, Fu H, Wang L-W and Zunger A 1999 *Phys. Rev.* B **60** 1819
[18] Franceschetti A and Zunger A 2000 *Phys. Rev.* B **62** 2614; Williamson A J, Franceschetti A and Zunger A 2000 *Europhysics Lett.* **53** 59; Reboredo F, Franceschetti A and Zunger A 1999 *App. Phys. Lett.* **75** 2972
[19] Franceschetti A and Zunger A 1999 *Nature* **402** 60
[20] Kim K, Graf P A and Jones W B 2005 *J. Comput. Phys.* **208** 735; Dudiy S and Zunger A (In preparation)
[21] Vaughan G V, Elliston B, Tromey T and Taylor I L 2000 *GNU Autoconf, Automake, and Libtool* (Pearson Education) 1st ed.
[22] Tomov S, Langou J, Canning A, Wang L-W and Dongarra J 2005 *Proc. of 5th Int. Conf. on Computational Sciences (ICCS), Atlanta, GA*, Part III. Springer's Lecture Notes in Computer Science, LNCS-3516, p 317
[23] Furlani J L 1991 *Proc. of the 5th Large Installation Systems Admin. Conf. (LISA V)*, San Diego, CA p 141