LATTE

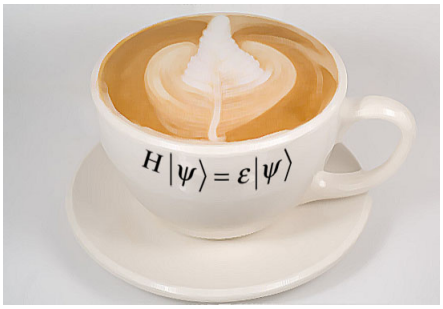
(Los Alamos Transferable Tight-binding for Energetics)
LA-CC 10-004

E. J. Sanville, A. M. N. Niklasson, N. Bock, J. D. Coe, S. P. Rudin, M. J. Cawkwell

Theoretical Division
Los Alamos National Laboratory

User Guide
November 2010

LA-UR 10-01978



I. Introduction

LATTE is a code written in FORTRAN90 for performing self-consistent charge transfer tight-binding (SCC-TB) calculations of total energies and the forces action on atoms in molecules and solids. The development of LATTE is geared principally toward large-scale, long duration, microcanonical molecular dynamics simulations of organic molecular materials.

We identified SCC-TB as the best candidate for the atomistic simulation of organic molecular materials because it is the simplest explicitly quantum mechanical method that describes all the inter- and intramolecular interactions at play in these systems:

i) the making and breaking of intramolecular covalent bonds,
ii) charge transfer between species of different electronegativities and long range electrostatic interactions, and
iii) 'medium'-range van der Waals interactions.

Furthermore, as all *chemists* know, spin-polarization or spin unrestricted calculations are essential if one is going to make or break covalent bonds. Hence, we have the option to run spin-polarized SCC-TB calculations if we want to simulate 'chemistry'.

II. Essential Theory

The most time consuming step in first principles self-consistent field (SCF) or density functional theory (DFT) calculations is the computation of the matrix elements of the Fockian or Hamiltonian. TB-based schemes can be orders of magnitude faster than first principles methods since the terms that enter the TB Hamiltonian are approximated and parameterized rather than computed exactly.[1-5] Nevertheless, methods based on the TB approximation remain explicitly quantum mechanical since the time-independent Schrödinger equation is still constructed and solved.

The SCC-TB formalism is derived from an expansion of the Kohn-Sham equations to second order in charge fluctuations about the self-consistent ground state. This procedure has been described in detail by Elstner,[6] Finnis,[4, 7] and Esfarjani.[8] Our work follows that of Elstner closely, but with the exception that we employ a minimal basis of a real, complete, *orthonormal* set of free-atom-like orbitals. In the following we present a spin-polarized SCC-TB model that we find to be necessary to capture charge redistribution upon the scission of covalent bonds and the formation of radicals.

The spin-independent SCC-TB Hamiltonian, H, is written as a sum of a charge-independent Slater-Koster TB Hamiltonian,[9] $H^{(1)}$, and a charge dependent Hamiltonian, $H^{(2)}$:

$$H_{i\alpha,j\beta} = H^{(1)}_{i\alpha,j\beta} + \sum_{k=1}^{N} \gamma_{ik} q_k \delta_{ij} \delta_{\alpha\beta} \qquad (1)$$

where i and j label atoms, α and β orbitals (s, p_x, p_y, and p_z), γ_{ii} is the Hubbard U for species i, $\gamma_{ik} \to 1/R_{ik}$ at long range, where R_{ik} is the scalar distance between atoms i and k, and at short range is a screened Coulomb potential,[6] q_k is the partial Mulliken charge on atom k, δ_{xy} the Kronecker delta, and N the total number of atoms. The total energy relative to that of isolated atoms is written:

$$E_{\text{total}} = 2\text{tr}[(\rho - \rho_0)H] - 1/2\sum_{i,j}\gamma_{ij}q_iq_j + E_{\text{pair}} \tag{2}$$

where in the first term a factor of two is included to account for spin degeneracy, tr[X] denotes the trace of matrix X, ρ is the density matrix calculated self-consistently from the SCC-TB Hamiltonian and the electronic occupancy, and ρ_0 the density matrix for isolated atoms. The second term is the so-called double counting correction for the electrostatic potential, and E_{pair} is a sum of pair potentials that is strongly repulsive at short range and which at long range provides $-1/R^6$ van der Waals-like interactions.

An explicit dependence of the total energy on electron spin can be built into the SCC-TB formalism through the introduction of two spin populations and the addition of a spin dependent Hamiltonian, $H^{(3)}$.[10] The Hamiltonians for the up and down spins populations are

$$H^{\uparrow} = H^{(1)} + H^{(2)} + H^{(3)} \tag{3}$$

and

$$H^{\downarrow} = H^{(1)} + H^{(2)} - H^{(3)}, \tag{4}$$

respectively. The spin-dependent Hamiltonian is given by:

$$H^{(3)}_{i\alpha,j\beta} = 1/2\sum_{l\in i}\sum_{l'\in i}\left(I_{l,l''} + I_{l',l''}\right)m_{i;l''}\delta_{ij}\delta_{\alpha\beta} \tag{5}$$

where l is the azimuthal quantum number, $I_{l,l'}$ an energy parameter related to the splitting between molecular orbitals as a result of spin polarization, and $m_{i;l}$ the difference between the number of electrons of up and down spin on the l orbitals of atom i. The total energy relative to isolated atoms within the spin-polarized formalism is written:

$$E_{\text{total}} = \text{tr}\left[\left(\rho - \rho_0\right)\left(H^{(1)} + H^{(2)}\right)\right] - 1/2\sum_{i,j}\gamma_{ij}q_iq_j + 1/2\sum_{i}\sum_{l,l'}\left(m_{i;l}m_{i;l'} - m^0_{i;l}m^0_{i;l'}\right)I_{l,l'} + E_{\text{pair}} \tag{6}$$

where $m^0_{i;l}$ is the difference between the number of electrons of up and down spin on the l orbitals of isolated atoms, and $\rho = \rho^{\uparrow} + \rho^{\downarrow}$ where the density matrices for spin up and

down electrons are calculated from the Hamiltonians in Eqns (3) and (4), respectively, subject to the condition that the value of $\mathrm{tr}[\rho^{\uparrow} + \rho^{\downarrow}]$ is equal to the total number of electrons.

The SCC-TB equations must be solved self-consistently since the SCC-TB Hamiltonian depends on partial charges and spin-difference densities that are themselves extracted from the density matrix. In the case of spin-polarized calculations these are,

$$q_i = \sum_{\alpha \in i} \left[\rho_{i\alpha,i\alpha} - \rho^0_{i\alpha,i\alpha} \right] \tag{7}$$

and

$$m_{i;l} = \sum_{\alpha \in l} \left[\rho^{\uparrow}_{i\alpha,i\alpha} - \rho^{\downarrow}_{i\alpha,i\alpha} \right]. \tag{8}$$

However, once the self-consistent density matrix has been evaluated, the force acting upon atoms within the system can be computed using the Hellmann-Feynman theorem,[4, 11] i.e.,

$$\mathbf{F}_k = -\mathrm{tr}\left[\rho \frac{\partial H^{(1)}}{\partial \mathbf{R}_k} \right] - \frac{1}{2} \sum_i \sum_{j \neq i} q_i q_j \frac{\partial \gamma_{ij}}{\partial \mathbf{R}_k} - \frac{\partial E_{\mathrm{pair}}}{\partial \mathbf{R}_k}. \tag{9}$$

III. Implementation of SCC-TB into LATTE

III.A *Slater-Koster Hamiltonian*

The matrix elements on the leading diagonal of the Slater-Koster[9] TB Hamiltonian, $H^{(1)}$, are the energies of the valence orbitals on free atoms. The off-diagonal elements of $H^{(1)}$, commonly referred to as hopping integrals, depend on interatomic distance and the angular character of the overlapping valence orbitals. The hopping integrals take the form of angular dependent combinations of a small number fundamental bond integrals. The radial dependence of the hopping integrals is contained within the bond integrals, $h_{ll'\tau}(R)$, where $\tau = \sigma, \pi, \delta$ etc. The angular dependencies of the hopping integrals for *sp*-valent elements have been tabulated,[9] thus one must only represent and parameterize the fundamental bond integrals. We have chosen to represent the bond integrals by Goodwin-Skinner-Pettifor functions,[12, 13] $S(R)$, such that $h(R) = h(R_0)S(R)$, where

$$S(R) = \left(\frac{R_0}{R} \right)^n \exp\left[n \left\{ \left(\frac{R_0}{R_c} \right)^{n_c} - \left(\frac{R}{R_c} \right)^{n_c} \right\} \right] \tag{10}$$

where R_0 is typically an equilibrium bond length, and n, n_c, and R_c are fitting parameters. A polynomial of the form

$$t(R) = \sum_{m=0}^{5} B_m (R - R_s)^m, \tag{11}$$

where B_0 to B_5 are fitting parameters, is added to the bond integrals at $R = R_s$ to ensure a smooth cut-off at a specified distance.

A total of 45 parameters must be defined in order to construct $H^{(1)}$ for hydrocarbons. We find that each of these can be fitted by matching in a least squares sense the energies of occupied *and* unoccupied molecular orbitals calculated using SCC-TB and DFT. The radial dependence of the bond integrals can be extracted by repeating this procedure for a series of homogeneously dilated molecules.

III.B *Electrostatic Hamiltonian*

Only one fitting parameter per element is required in the construction of $H^{(2)}$, namely the Hubbard *U*. The Hubbard *U* can be estimated using experimental values for the ionization potential, *I*, and the electron affinity, *A*, of each element, $U \approx I - A,$ which we apply without modification. The Coulomb potential is screened at short range using the formalism of Elstner[6] and we use the Ewald method[14] to compute the long-range contribution to the electrostatic potential.

III.C *Pairwise Interactions*

We construct a sum of pair potentials for each element pair to provide strong repulsion at short range and weak attraction that mimics van der Waals bonding at long range. The pairwise term in the expression for the total energy takes the form

$$E_{\text{pair}} = \frac{1}{2} \sum_i \sum_{j \neq i} \Phi(R_{ij}) \tag{12}$$

where

$$\Phi = \begin{cases} \displaystyle\sum_{k=1}^{4} A_k \left(R_k - R_{ij} \right)^3 \theta\left[R_k - R_{ij} \right]; & R_{ij} \le R_A \\ t_{\text{join}}(R_{ij}); & R_A < R_{ij} \le R_B \\ -C / R_{ij}^6; & R_B < R_{ij} \le R_s \\ t_{\text{cut}}(R_{ij}); & R_s < R_{ij} \le R_{\text{cut}}. \end{cases} \tag{13}$$

The first term is the short-range part of the pair potential where the node points, R_k, and pre-factors, A_k, are fitting parameters, and $\theta[x]$ is the Heaviside step function.[13] The

adjustable parameters R_k and A_k are fitted such that the overall model reproduces cohesive energies, and the length and force constants of covalent bonds calculated using first principles methods. The third term provides van der Waals-like interactions and the adjustable parameter, C, is taken from the literature for each element pair.[15] The second term, t_{join}, takes the same form as Eqn. (11) and is used to smoothly join the short range pair potential to the long range van der Waals-like pair potential over the range $R_A < R \leq R_B$. The final term applies a smooth cut-off to the pair potential.

III.D *Spin-dependent Hamiltonian*

The adjustable parameters that enter $H^{(3)}$ affect the magnitude of the relative shifts in energy of the molecular orbitals for the two spin channels. The values of $I_{l,l'}$ for carbon and hydrogen have been estimated using spin-polarized DFT calculations of the energies of the molecular orbitals of free atoms and molecules that possess at least one unpaired spin.

IV. Compiling LATTE

A serial version of LATTE can be compiled on a laptop or desktop equipped with a FORTRAN90 compiler and pre-compiled LAPACK/BLAS libraries. The GNU gfortran compiler works very well and we have also tested the more heavily optimized PGI (pgf90), Intel (ifort), and Pathscale (pathf90) compilers. Furthermore, the LAPACK/BLAS libraries can either be downloaded free of charge (http://www.netlib.org/lapack) and compiled locally, or one can try the heavily optimized versions such as ACML from PGI, MKL from Intel, the auto-tuned ATLAS, or GOTO BLAS. It is well worthwhile testing a number of linear algebra library and compiler combinations since it is possible to obtain very significant speed-ups on fixed hardware. Users should note that commercial compilers and libraries may improve performance but publically available software (gfortran: http://gcc.gnu.org/fortran/, LAPACK/BLAS: http://www.netlib.org/lapack/LICENSE) on a regular desktop running LINUX provide identical accuracy and comparable performance.

If you are going to use the CUDA code on a NVIDIA graphics card, you must download the SDK from NVIDIA to get the nvcc compiler, CUBLAS library, and other essential libraries.

Note added 11/9/2010: We have noticed absolutely awful performance from the BLAS level 3 SGEMM routine when gfortran is the F90 compiler. Please beware – bad performance may not always be our fault.

IV.A *makefile.h*

To compile LATTE, simply edit `makefile.h` in the src directory to specify your compilers and the location of your libraries. In this file we also specify pre-processor options that select whether single or double precision code will be compiled and whether to link to the CUDA code in MATRIX.

```
#
# Compilation and link flags for LATTE
#

PRECISION = SINGLE
GPUOPT = OFF
CPP = cpp -P -C -D$(PRECISION)PREC -DGPU$(GPUOPT) -traditional < $*.F90 > $*.f90
FC = ifort
FCL = $(FC) -openmp
FFLAGS =  -O3 -openmp
LINKFLAG =
GPU_LINKFLAG = -L../MATRIX -lgfortran -lstdc++
LIB = -L/opt/ACML/acml-4.3.0/ifort64_mp/lib -lacml_mp
GPU_LIB = -L../MATRIX -lmatrix_cuda -L/usr/local/cuda/lib64 -lcublas -lcudart
```

`PRECISION` = `DOUBLE`: run in double precision arithmetic.
= `SINGLE`: run in single precision arithmetic.

`GPUOPT` = `ON`: compile in the CUDA code for SP2 density matrix purification in MATRIX. The library `libmatrix_cuda.a` must be compiled for LATTE is compiled.
= `OFF`: compile only CPU code.

`CPP` Selects the c preprocessor and passes the preprocessor options.

`FC` Your preferred FORTRAN90 compiler.

`FCL` Linker flags for the F90 compiler. Pass the flag to compile the OpenMP parts of the code here.

`FFLAGS` Optimization flags for the F90 compiler. Our testing to date shows that it is safe to 'max-out' these since LATTE is far from exotic.

`GPU_LINKFLAG` These flags are required to for the F90 code to talk to the C++ code in MATRIX when compiling the hybrid CPU/GPU code.

`LIB` Specify your preferred LAPACK and BLAS implementations and the path to them.

`GPU_LIB` Specify the path to `libmatrix_cuda.a` and the CUDA libraries.

Then, simply run '`make`' in the main directory to compile. If the compilation is successful, you will obtain an executable in the main directory whose name reflects the preprocessor options defined in makefile.h.

IV.B *Parallel processing*

We are working toward meshing LATTE with MPI distributed memory, parallelized linear algebra libraries (ScaLAPACK and PBLAS). Makefiles and instructions will be added once these features are merged into the publically available LATTE distribution. We also intend to release parallelized versions of the Ewald sums for the long-range electrostatics.

While a distributed memory version LATTE remains under development, you can run the code on a multi-core shared memory desktop or compute node. Most high performance math libraries (ACML, MKL, and GOTO) are threaded and by setting OMP_NUM_THREADS accordingly, LAPACK and BLAS calls can be run in parallel. Moreover, we have threaded using OpenMP all of the compute-intensive triple loops in the code. These capabilities can be activated by passing to your F90 compiler the appropriate flag (-fopenmp for gfortran, -openmp for ifort, and –mp for pgf90, etc). We have seen a good, but not perfect, scaling of the threaded code up to about 8 cores, with deteriorating performance gains at 16 cores. Nevertheless, this approach will reduce significantly the wall time for your runs with no loss of accuracy.

IV.C *GPU/CUDA acceleration*

LATTE *is* GPU-enabled! Before the user gets carried away, there are a few provisions. First, you must have a NVIDIA graphics card with a compute capability of at least 1.3 if you plan to run double and single precision arithmetic on it. If you don't own suitable hardware, Fermi-based GPUs from NVIDIA such as a GTX480 are very affordable. Slightly more thought must go into powering them – ensure your power supply has a sufficiently high rating (something of the order of 1kW) and that you have sufficient connectors (Fermi's have 4 power inputs, I think). You may, like us, have to build a dedicated machine to ensure the power supply and motherboard can handle the GPU. Once your hardware is up to 'spec', download the NVIDIA's CUDA SDK. This package contains, among other things, the CUDA compiler, nvcc, various CUDA libraries, and the CUBLAS math libraries.

As of November 2010, only the SP2 density matrix purification algorithm has been ported to the GPU. We will release other ports as they become available. The user should note that we do not just shift the level 3 BLAS operations, such as SGEMM or DGEMM, to the GPU, but the entire algorithm. This approach, while more complicated, minimizes the communication between the CPU and GPU, noting that sending large matrices back and forth across a slow communication bus can degrade performance significantly. Our approach, developed by Ed Sanville, yields superior performance on a single GPU than the threaded CPU code running on 16 Xeon cores.

If you wish to use the GPU-enabled code, LATTE must be compiled with the GPUOPT preprocessor flag set to ON.

V. Running LATTE

Running the serial version of LATTE requires a number of files to be at specific locations with respect to the executable. Once these files are in place, LATTE can be run simply by invoking from the command the executable:

```
cawkwell@stelvio:~/LATTE$ ./LATTE
```

where a redirect can be used to log the output, i.e.,

```
cawkwell@stelvio:~/LATTE$ ./LATTE > myoutput.dat
```

LATTE requires that the following files be in the following locations relative to the executable:

`./bl/inputblock.dat` – contains the coordinates of atoms, their chemical symbols, and information on the periodic box.

`./TBparam/control.in` – specifies what type of calculation you want to do and how you want to do it.

`./MDcontroller` – defines how an MD simulation is going to be run.

`./TBparam/bondints.dat` – contains the parameters for the GSP parameters for the bond integrals

`./TBparam/electrons.dat` – defines the basis for each atom and gives the parameters for the on-site energies, Hubbard *U*, atomic mass, and spin parameters.

`./TBparam/ppots.dat` – contains the parameters for the pair potentials.

By default LATTE writes dump files to the directory `./animate` and restart files to the directory `./Restarts`. MD simulations will crash if these directories are not available.

V.A *Required files, their contents, and formats*

In addition to requiring specific names and be placed in specific locations, the files required to run LATTE also have fairly rigid formats. The user is strongly recommended to modify only the arguments of the parameters provided with the distribution of LATTE to their own purpose.

V.A.1 `./bl/inputblock.dat`

The following format is used to input atomic coordinate and the dimensions of the periodic box, etc:

```
Nats= 5
1.0
0.0 19.0 0.0 19.0 0.0 19.0
4.42000485      4.52000922      4.57997533 C
3.94629128      3.97643147      3.73672264 H
5.08066716      5.31744034      4.18177348 H
5.02016006      3.81042226      5.18608617 H
3.63287667      4.97569671      5.21544236 H
```

where

Line 1: `Nats=` the total number of atoms, *N*.

Line 2: A scaling parameter, α, for all subsequent parameters with units of length such that, position, $x = \alpha x'$. Thus, one could set α equal to the lattice parameter of a cubic metal and then input the dimensions of the box and atomic coordinates and multiples or fractions of that α. If $\alpha = 1.0$, then the remaining entries are positions in units of Å.

Line 3: Defines the periodic orthorhombic simulation box where the six real numbers correspond to X_{min}, X_{max}, Y_{min}, Y_{max}, Z_{min}, and Z_{max}, respectively. Periodic boundary conditions are applied such that all atoms remain with the specified limits

Line 4 to 4 + *N*: The scaled Cartesian coordinates, $\mathbf{r}' = \mathbf{r}/\alpha$, of the *N* atoms followed by their chemical symbol.

V.A.2 `./TBparam/control.in`

This file tells LATTE what to do and how to do it. An example of the format is provided below, followed by a detailed description of each of the entries.

```
CONTROL= 2
FERMIM= 6
CGORLIB= 1
KBT= 0.1
NORECS= 18
ENTROPYKIND= 1
SPINON= 1 SPINTOL= 1.0e-4
ELECTRO= 1 ELECMETH= 0 ELEC_ETOL= 0.001 ELEC_QTOL= 1.0e-4
COULACC= 1.0e-4 COULCUT= -7.9 COULR1= 40.0
MAXSCF= 50
BREAKTOL= 1.0E-6 MINSP2ITER= 26
FULLQCONV= 0 QITER= 1
QMIX= 0.25 SPINMIX= 0.25
ORDERNMOL= 0
SPARSEON= 0
LCNON= 0 LCNITER= 4 CHTOL= 0.01
SKIN= 1.0
RELAX= 0 MAXITER= 100 RLXFTOL= 0.01
MDON= 1
BOXON= 0
RESTART= 0
XBO= 1
XBODISON= 1
XBODISORDER= 5
```

`CONTROL` Selects the method for computing the density matrix.
`= 1`: Matrix diagonalization with a user-defined electronic temperature.
`= 2`: Niklasson's second order spectral projection (SP2) algorithm (GPU-friendly).[16]
`= 3`: Niklasson's recursive expansion of the Fermi operator (GPU-friendly).[17]
`= 4`: Truncated SP2 algorithm for an approximate Fermi-Dirac distribution (GPU-friendly).
`= 5`: 'SP2Fermi' algorithm for that yields an approximate Fermi-Dirac distribution using a truncated purification with a chemical potential for correcting the occupancy (GPU-friendly).

`FERMIM` Selects the level of the recursion applied during the recursive expansion of the Fermi operator (`CONTROL = 3`).

`CGORLIB` Selects whether a conjugate gradient method or a LAPACK routine is employed to solve $\mathbf{AX} = \mathbf{B}$ when computing the density matrix using the recursive expansion of the Fermi operator (`CONTROL = 3`).
`= 0`: Call the $O(N^3)$ LAPACK routines DGESV or SGESV.
`= 1`: Use the conjugate gradient solver (GPU-friendly and potentially linear scaling).[17]

`KBT` Thermal energy of the electronic subsystem in units of eV. The thermal energy of the electrons comes into play when diagonalization (`CONTROL =`

1) or the recursive expansion of the Fermi operator (CONTROL = 3) is used to compute the density matrix.

NORECS — Selects the number of purification steps for algorithms based on the truncated SP2 algorithm (CONTROL = 4 and CONTROL = 5).

ENTROPYKIND — Selects how the entropy of the electronic subsystem is going to be computed when a finite electronic temperature is employed.
= 0: S = 0 (for testing purposes)
= 1: Logarithmic expression for the entropy that corresponds exactly to Fermi-Dirac statistics, i.e., $S = \rho \ln \rho + (I - \rho)\ln(I - \rho)$.
= 2: A very accurate approximation to the 'exact' entropy at almost the same computational cost.
= 3: An approximate entropy that requires only one matrix-matrix multiplication rather than a diagonalization of the finite temperature density matrix (GPU-friendly).

$S = \mathrm{Tr}[Y(C_1 + C_2 Y)]$, where $Y = \rho(\rho - I)$, $C_1 = 8\ln(2) - 2$, and $C_2 = 16\ln(2) - 8$.

= 4: A higher order, GPU-friendly approximation than that implemented in ENTROPYKIND = 3 that requires two matrix-matrix multiplications.

$S = \mathrm{Tr}[C_1 Y + Y^2(C_2 I + C_3 Y + C_4 Y^2)]$, where $Y = \rho(\rho - I)$, $C_1 = 16\ln(2) - (34/5)$, $C_2 = 96\ln(2) - (844/15)$, $C_3 = 256\ln(2) - (2336/15)$, and $C_4 = 256\ln(2) - (2368/15)$.

SPINON — Selects between spin- and non-spin polarized calculations.
= 0: Spin-polarization off.
= 1: Spin-polarization enabled.

SPINTOL — User-defined tolerance for self-consistency when computing spin-difference densities. This parameter applies mainly to static calculations since when we run MD calculations we typically specify some fixed number of SCF cycles rather than run to full self-consistency at each MD time step.

ELECTRO — Selects whether self-consistent charge transfer or local charge neutrality calculations are to be performed. In the latter the on-site energies are adjusted iteratively at each time step to ensure each atom has a specified amount of charge.
= 0: Local charge neutrality imposed
= 1: Self-consistent charge transfer TB

ELECMETH — Selects a method for computing the electrostatic potential
= 0: Ewald summation

= 1: Real space electrostatics with no long-range component. This option comes in handy for debugging the energies and forces from the Ewald method and allows for very accurate 'gas-phase' calculations on small molecules.

ELEC_ETOL — User-defined tolerance for self-consistency when computing partial charges based on differences between Coulombic energy from one SCF cycle to the next. This flag is currently not used.

ELEC_QTOL — User-defined tolerance for self-consistency when computing partial charges (requires ELECTRO = 1). In static calculations and the first time step of an MD run we perform SCF cycles until the value of the partial charge on each atom at a given SCF cycles differs from the previous cycle by no more than ELEC_QTOL electrons.

COUL_ACC — Relative accuracy of the electrostatic potential computed using the Ewald method (ELECTRO = 1 and ELECMETH = 0).

COUL_CUT — Specifies the range in Å for the real-space summation of Coulombic interactions. If the Ewald method employed (ELECMETH = 0) then COUL_CUT is equal to the cut-off for the error-function summation part. During Ewald summation, if COUL_CUT < 0 then the code determines automatically the optimal value for the real-space cut-off. It is advised that the user checks the value since it is likely to be machine and compiler dependent. If the electrostatic potential is being computed entirely in real-space (ELECMETH = 1), then COUL_CUT is equal to the radial distance beyond which partial charges are not included in the sum.

COULR1 — Specifies the start of the 5^{th}-order polynomial employed to smoothly truncate the real-space computation of the electrostatic potential if ELECMETH = 1. Of course, COULR1 < COUL_CUT when ELECMETH = 1.

MAXSCF — The maximum number of SCF cycles to be used computing self-consistent partial charges and/or spin-difference densities during static calculations or the first time step in an MD run. LATTE will stop if the number of SCF cycles is exceeded since it means something is probably is not good with the atomic geometry and/or the model parameterization and/or the mixing parameters.

BREAKTOL — Specifies the tolerance on the error of trace of the density matrix when computing the density matrix (or density matrices in the case of a spin-polarized calculation) using diagonalization (CONTROL = 1) the SP2 purification algorithm (CONTROL = 2). The iterative application of the purification steps will cease once this tolerance has been reached *and* we've performed at least MINSP2ITER purification steps if CONTROL = 2.

MINSP2ITER Specifies the minimum number of iterations within the SP2 density matrix purification algorithm before any other convergence criteria are checked. The use of a minimum number of purification steps is important to prevent the other convergence criteria being satisfied fortuitously and a poor density matrix being produced. These criteria should be studied closely, especially when running LATTE in single precision arithmetic.

FULLQCONV Selects whether we will run to full self-consistency (as specified by the values SPINTOL and ELEC_QTOL) at each time step in an MD run or run some user-defined number of SCF cycles only.
= 0: Do not run to full self-consistency at each time self: run QITER SCF cycles instead
= 1: Run to full self-consistency at each time step

QITER During MD calculations run QITER SCF cycles at each time step. This only applies if FULLQCONV = 0.

QMIX Mixing parameter when self-consistently updating partial charges, i.e., $\{q(\text{SCF+1})\} = \text{QMIX} \times \{q(\text{SCF})\} + (1 - \text{QMIX}) \times \{q(\text{SCF-1})\}$.

SPINMIX Mixing parameter when self-consistently updating the spin-difference densities, i.e.,
$\{m(\text{SCF+1})\} = \text{QMIX} \times \{m(\text{SCF})\} + (1 - \text{QMIX}) \times \{m(\text{SCF-1})\}$.

ORDERNMOL Currently not implemented

SPARSEON Selects between dense (DGEMM or SGEMM) matrix-matrix multiplication or M.J.C.'s primitive (as in primordial) sparse matrix-matrix multiplication scheme during SP2 purification (CONTROL = 2) or the recursive expansion of the Fermi operator (CONTROL = 3). Not implemented in SP2 for spin-polarized calculations yet.
= 0: All dense matrix matrix multiplication
= 1: The sparse matrix method.

LCNON Selects whether local charge neutrality is applied to within a user-defined tolerance (CHTOL) or whether a user-defined number of iterations (LCNITER) of the local charge neutrality procedure are performed. This option applies only when ELECTRO = 0 and when an MD simulation is being performed.
= 0: Run a specified number of iterations of the adjustments for local charge neutrality.
= 1: Run the LCN calculations until we reach the user-defined tolerance of the amount of charge per atom

`LCNITER` Number of iterations of the adjustments in on-site energies when imposing local charge neutrality. Invoked during an MD simulation only when `ELECTRO = 0` and `LCNON = 0`.

`CHTOL` User-defined tolerance in units of electrons when imposing local charge neutrality. This tolerance is invoked during static calculations, on the first time step of an MD simulation when `LCNON = 0`, and during all time steps of a MD simulation when `LCNON = 1`.

`SKIN` This is the value in units of Å of the 'skin' used when constructing the neighbor lists such that we store atoms in the neighbor lists that are outside the cut-offs but which may enter the cut-offs between neighbor list updates.

`RELAX` Selects whether a steepest descent molecular statics relaxation of an assembly of atoms is to be performed.
`= 0`: Relaxation not performed.
`= 1`: Perform the relaxation.

`MAXITER` Maximum number of atom-moving steps during a molecular statics relaxation.

`RLXFTOL` The tolerance in units of eV $Å^{-1}$ on the magnitude of the force acting on any atom for the termination of a molecular statics relaxation.

`MDON` Selects whether a molecular dynamics simulation is to be performed.
`= 0`: A MD simulation is not performed.
`= 1`: Perform a MD simulation.

`BOXON` Selects whether reflecting boundaries are to be used in an MD simulation. This is an old feature when local charge neutrality was typically invoked during a simulation. It doesn't make much sense to put atoms in a reflecting box when long-range electrostatic interactions based on three-dimensional periodic boundary conditions are employed.
`= 0`: Use non-reflecting periodic boundary conditions.
`= 1`: Use reflecting walls during a MD simulation.

`RESTART` Specifies whether a MD simulation is to start from scratch or resume from a restart file. In the former, coordinates are read from `./bl/inputblock.dat` and velocities are initialized in the code. In the latter, the number of the last time step from the previous run, coordinates, and velocities are read from `./bl/restart.dat`. This file must be put in place by hand by copying `./restartMD.dat` to `./bl/restart.dat`. Note that 'perfect' restarts in terms of reading in partial charges, spin densities, and their histories are not yet possible but this feature will be implemented in the near future. The accuracy of restarts could also be improved

(potentially at the expense of portability) by using binary restart files as in LAMMPS.
= 0: Start an MD simulation from scratch.
= 1: Restart an MD simulation from a restart file.

XBO Selects whether during a MD simulation self-consistently calculated quantities (partial charge, spin-difference densities, the chemical potential) are to be propagated using Niklasson's extended Lagrangian Born-Oppenheimer MD formalism.[18, 19] It is strongly recommended that XLBOMD are switched on whenever MD is performed.
= 0: No not use XLBOMD propagation.
= 1: Propagate quantities in a time reversible manner using XLBOMD.

XBODISON Selects whether dissipation is to be employed during a XLBOMD run (XBO = 1) to counteract the accumulation of numerical noise.[19]
= 0: No dissipation included.
= 1: Employ a dissipation scheme.

XBODISORDER Selects the order (integers in the range 3 through 9) of the dissipation algorithm employed in an XLBOMD trajectory (XBO = 1) with dissipation (XBODISON = 1).[19]

V.A.3 ./MDcontroller

This file provides the basic parameters (temperature, time step, frequency of writing outputs etc.) for the control of an MD simulation. Parameters related to the computation of interatomic forces are defined in ./TBparam/control.in leaving ./MDcontroller dedicated to generic parameters related to MD simulations.

```
MAXITER= 250000
UDNEIGH= 25
DT= 0.25
TEMPERATURE= 300.0
DUMPFREQ= 250
RSFREQ= 250
WRTFREQ= 50
TOINITTEMP= 1
THERMPER= 500
THERMRUN= 20000
NVTON= 0 AVEPER= 400
SHOCKON= 0
SHOCKSTART= 25000
SHOCKDIR= 1
UPARTICLE= 2000.0 USHOCK= 4590.0
```

`MAXITER` Specifies the number of time steps over which a MD simulation will be run. If a simulation is resumed from a restart file (`RESTART = 1` in `./TBparam/control.in`) from a simulation of duration M_A steps, then `MAXITER` should be set equal to $M_A + M_B$ where M_B is the total number of MD time steps that the new simulation will span.

`UDNEIGH` The number of MD time steps between updates of the neighbor lists. This parameter will depend on the thickness of the 'skin' region (defined in `./TBparam/control.in)`, the temperature, and the size of the time step.

`DT` The size of the time step for the integration of the equations of motion for the nuclei in units of femtoseconds.

`TEMPERATURE` Equal to the initialized or thermostated temperature of a system in units of K. LATTE will try to adjust the kinetic energy of a system toward a value consistent with the user-specified `TEMPERATURE` in two ways. First, upon the start of a new MD simulation, the velocities of all atoms will be initialized to values consistent with the value of `TEMPERATURE`. Second, the velocities of all atoms will be periodically rescaled during a *NVT* MD simulation such that system temperature is equal to the value of `TEMPERATURE`. If a simulation is resumed from a restart file, the velocities are not reinitialized but a thermostated simulation will still push the velocities of atoms to those consistent with the value of `TEMPERATURE`.

`DUMPFREQ` The number of MD time steps between writing a snap shot of the simulation to file. Dump files are written to the directory `./animate` and are index by the time step (the total time step if one or more restarts have been applied). The files are currently written in the .cfg format as we find this provides efficient, dense storage (cf. PDB), can be extendable to include a number of auxiliary properties (partial charges, spin-difference densities, etc.), and they can be visualized directly, very well, and very easily using Ju Li's Atomeye code.[20] See: http://mt.seas.upenn.edu/Archive/Graphics/A/.

`RSFREQ` The number of MD time steps between writing a restart file. These files are written to `./restartMD.dat` and are not indexed by the time step. This file must be copied to `./bl/restart.dat` with `RESTART = 1` in `./TBparam/control.in` to restart a MD run. Note that in their current guise a 'perfect' restart is not possible since the values of self-consistently calculated quantities and their time histories are not stored. Nevertheless, the current capabilities of LATTE appear to be reasonably robust.

`WRTFREQ` The number of MD time steps between computing energies, the virial pressure, and writing them to the standard output. There is no need to

write these every time step as the energies will be strongly correlated, and there is a computational cost to computing $\mathrm{tr}[\rho H]$, and especially the entropic contribution to the free energy when $k_B T_e \neq 0$. The format of this output is:

```
#   Time (ps)          Free energy (eV)  T (K)    Pressure (GPa)
     0.00250         -75.6319688288638 3210.2     3.842633
```

This data can be written to a user-specified file during a MD simulation via a redirect.

`TOINITTEMP` `= 1`: Initialize the velocities at the first time step of a new MD run based on the value of `TEMPERATURE`.
`= 0`: Do not initialize velocities.

`THERMPER` The number of MD time step between velocity rescalings during *NVT* molecular dynamics (`NVTON = 1`).

`THERMRUN` The number of MD time steps over which *NVT* MD is performed (`NVTON = 1`). LATTE has been constructed such that a microcanonical (*NVE*) MD simulation will continue directly from a *NVT* simulation if `MAXITER > THERMRUN`. For example, `MAXITER = 100000` and `THERMRUN = 50000`, then the simulation will start with 50000 time steps of *NVT* MD followed by 50000 time steps of *NVE* MD without interruption. Alternatively, if `MAXITER = THERMRUN` then *NVT* MD will be performed for the entire duration of the simulation.

`NVTON` Specifies whether a *NVE* or *NVT* simulation is to be performed. The latter will usually be required to thermalize a system toward `TEMPERATURE` before switching to a microcanonical XLBOMD simulation. We take full advantage of the uncorrupted, energy conserving dynamics afforded by the XLBOMD formalism during *NVT* simulations by computing an average temperature, $\langle T \rangle$, over `AVEPER` time steps of an XLBOMD simulation (`XBO = 1` in `./TBparam/control.in`) and then rescaling the velocities of all atoms by a factor of $\sqrt{T'/\langle T \rangle}$, where T' = `TEMPERATURE`, once every `THERMRUN` time steps.
`= 0`: Perform *NVE* MD and do not apply a themostat.
`= 1`: Perform *NVT* MD by rescaling velocities as specified by the values of `AVEPER`, `THERMPER`, and `TEMPERATURE`.

`AVEPER` Compute the average temperature over this many MD time steps in order to rescale the velocities of atoms during *NVT* MD.

SHOCKON We have developed and implemented a simple 'Hugoniostat' to mimic shock compression in a MD simulation. The passage of a shock wave causes a change of density via uniaxial compression that is specified by the Hugoniot relation $\rho = \rho_0 / \left(1 - U_p / U_s\right)$, where ρ_0 is the initial density, and U_p and U_s are the particle and shock wave velocities, respectively. The dynamic compression imparted by a shock wave can be captured in a small, periodic unit cell by noting that the shock wave produces the change in density given by the Hugoniot relation in the period of time required for the shock wave to traverse the simulation cell, l_0 / U_s. Thus, during the passage of the shock wave, the length of the simulation cell parallel to the direction of the propagation of the shock wave will be $l(t) = l_0 - U_p t$, where l_0 is the length of the simulation cell before the shock arrives, and t is time, where $0 \le t < l_0 / U_s$.

= 1: Use the Hugoniostat during the MD run

= 0: Do not use the Hugoniostat.

SHOCKSTART Specifies the time step at which the Hugoniostat will be applied. The Hugoniostat will stop distorting the simulation cell $l_0 / (U_s \Delta t)$ time steps later.

SHOCKDIR Specifies the Cartesian direction along with the simulation cell will be distorted (1 = *x*, 2 = *y*, 3 = *z*).

UPARTICLE The particle velocity in units of m/s.

USHOCK The shock wave velocity in units of m/s.

V.A.4 ./TBparam/bondints.dat

This file contains the parameters for the GSP functions for each of the Slater-Koster bond integrals. LATTE gives the user the option to define a radial scaling individually for each bond integral. The number of bond integrals depends on basis of each element as well as the number of elements.

Recall that,

$$h_{ll'\tau}(R) = \begin{cases} h_{ll'\tau}(R_0) S(R) & R \le R_1 \\ \sum_{k=0}^{5} B_k \left(R - R_1\right)^k & R_1 < R \le R_{\text{cut}} \end{cases}$$

where

$$S(R) = \left(\frac{R_0}{R}\right)^n \exp\left[n\left\{\left(\frac{R_0}{R_c}\right)^{n_c} - \left(\frac{R}{R_c}\right)^{n_c}\right\}\right].$$

Thus, the radial dependence of each bond integral is described completely by the seven adjustable parameters $h(R_0)$, R_0, R_1, $R_{\rm cut}$, n, n_c, and R_c. The six parameters $\{B_k\}$ are not adjustable – they are determined uniquely by the conditions that at $R = R_1$ the cut-off tail should match the corresponding GSP function in value and first and second derivative, and at $R = R_{\rm cut}$ the value and first and second derivative of the cut-off tail should equal zero. The parameters $\{B_k\}$ are calculated in LATTE when the GSP parameters are read from file.

An example of the format of `./TBparam/bondints.dat` is given below:

```
Noints= 7
Element1 Element2 Kind  h(R_0)      R_0   R_c      n           n_c      R_1 R_cut
C        C        sss  -4.986742022 1.531 2.242063 2.56158     5.982062 2.0 3.0
C        C        sps   4.661822786 1.531 0.524225 0.597312    0.883757 2.0 3.0
C        C        pps   5.428514026 1.531 0.318689 6.291311e-4 4.167031 2.0 3.0
C        C        ppp  -1.954959    1.531 1.0075   1.645437    0.292716 2.0 3.0
H        H        sss  -7.521920    0.743 0.128033 0.090587    1.256029 1.5 2.5
H        C        sss  -5.928782    1.093 0.858752 0.565763    1.272500 2.0 3.0
H        C        sps   7.279468    1.093 1.540020 0.811385    2.511200 2.0 3.0
```

where the parameter `Noints=` specifies the number of GSP functions to be read into LATTE. The only entries in the remainder of the file that may not be self-explanatory given the preamble in this subsection are the `Kind` identifiers for the $ll'\tau$ bond integrals. These are:

`sss` $\equiv ss\sigma$
`sps` $\equiv sp\sigma$
`pps` $\equiv pp\sigma$
`ppp` $\equiv pp\pi$

and LATTE is written such that $h_{sp\sigma} = -h_{ps\sigma}$. However, LATTE can handle the situation where we have two sp-valent elements and *p*(element 1)*s*(element 2) is not the same as *p*(element 2)*s*(element 1). Nevertheless, we never define $h_{ps\sigma}$, in the input file, only $h_{sp\sigma}$.

V.A.5 `./TBparam/electrons.dat`

Parameters relating mainly to free atoms are defined for each element in `./TBparam/electrons.dat`. An example of the format and detailed descriptions of each entry are provided below.

```
Noelem= 2
Element Basis  Numel   Es          Ep         Mass   HubbardU   Iss  Isp  Ipp
C       sp     4.0    -10.966922 -2.767504  12.01  9.9986    -0.5 -0.7 -0.7
H       ss     1.0    -5.365000   0.0       1.0079 12.8437   -2.5  0.0  0.0
```

The first entry, `Noelem=`, specifies the number of sets of atomic parameters to be read from the file. The header entries on the next line are fairly self-explanatory, nevertheless:

`Element` The chemical symbol of the element. This must be consistent across all of the input files.

`Basis` Specifies the valence orbitals possessed by each element where `ss` corresponds to one valence *s* orbital per atoms, and `sp` to one valence *s* orbital and three *p* orbitals (p_x, p_y, and p_z) per atom.

`Numel` Specifies the number of valence electrons on an isolated atom of each element, N_e. The total number of electrons in the system will be equal to $N_e^{\text{total}} = \sum_{i=1}^{N_{\text{atoms}}} N_e^i$, but of course when the self-consistent transfer of charge between atoms is enabled, we determine partial charges by calculating the Mulliken charges relative to the value of N_e.

`Es` The energy of the valence *s* orbital on a free atom in units of eV.

`Ep` The energy of the valence *p* orbitals on a free atom in units of eV.

`Mass` The atomic mass of the element in units of a.m.u.

`HubbardU` The value of the Hubbard *U* for each element in units of eV.

`Iss` The value of the Stoner-like parameter, in units of eV, that describes the splitting of the molecular orbital energies resulting from the spin polarization of valence *s* orbitals.

`Isp` The value of the Stoner-like parameter, in units of eV, that couples the spin-polarization of the valence *p*(*s*) orbitals to the valence *s*(*p*) orbitals.

`Ipp` The value of the Stoner-like parameter, in units of eV, that describes the splitting of the molecular orbital energies resulting from the spin polarization of valence *p* orbitals.

V.A.6 `./TBparam/ppots.dat`

The parameters for the short-range and long-range pair potentials along with the positions of the start and end of the joining and cut-off functions are provided in `./TBparam/ppots.dat`. An example of the format of this file and a detailed description of each entry are provided below.

```
Nopps= 3 Max_k= 4
ELE1: C ELE2: C k: 4
Rk      Ak
1.5307  129.788382
1.9000 -3.614438
2.4000  7.169733
3.0000  1.816112
Join_R1: 2.0 Join_Rcut: 2.6
vdW_C: 27.787715795
vdW_R1: 8.5 vdW_Rcut: 10.0
Done
ELE1: H ELE2: H k: 4
Rk      Ak
0.7428  35.32452626
1.0     4.40710494383
1.6     9.30030199743
2.7     0.124942629563
Join_R1: 2.3 Join_Rcut: 2.6
vdW_C: 1.1274666209
vdW_R1: 8.5 vdW_Rcut: 10.0
Done
ELE1: C ELE2: H k: 4
Rk      Ak
1.0933  17.56543067
1.3     27.2682130272
1.85    4.01861782146
2.4     3.09839775397
Join_R1: 2.0 Join_Rcut: 2.3
vdW_C: 5.99291873107
vdW_R1: 8.5 vdW_Rcut: 10.0
Done
```

`Nopps` Specifies the total number of pair potentials to be read from file. Each pair potential is described in a block of parameters whose last line is `Done`.

`Max_k` Specifies the maximum number of node points used in the evaluation of the short range pairwise term (see Eqn. (13)) in any of the `Nopps` pair potentials to be read. This parameter is included in the input file to assist with the dimensioning of arrays in LATTE rather than for any physical reason.

`ELE1: ELE2:` Specify the pair of elements that will interact via the corresponding pair potential. The order of the elements is not important but only one of `ELE1: X ELE2: Y` and `ELE1: Y ELE2: X` need to be defined.

`k:` Specifies the actual number of node points in the short-range pair potentials between elements `ELE1:` and `ELE2:`. Note that `k:` $\leq$ `Max_k`.

`Rk Ak` The value of the `k:` node points, R_k, and the corresponding coefficients, A_k, are listed below the corresponding headers in units of Å and eV, respectively.

`Join_R1:` Specifies in units of Å the radial position of the start of the function that joins smoothly the short-range pair potential to the long-range van der Waals pair potential. The parameters $\{B\}$ for the joining function are calculated within LATTE.

`Join_Rcut:` Specifies in units of Å the radial position of the end of the function that joins smoothly the short-range pair potential to the long-range van der Waals pair potential.

`vdW_C:` Specifies in units of eV the value of the parameter C in the van der Waals-like $-C/R^6$ pair potential between elements `ELE1:` and `ELE2:`.

`vdW_R1:` Specifies in units of Å the radial position of the start of the cut-off tail that is added to the long-range pair potential. The parameters $\{B\}$ for the cut-off tail are calculated with LATTE.

`vdW_Rcut:` Specifies in units of Å the radial position the end of the cut-off tail that is added to the long-range pair potential.

V.B *Other files*

V.B.1 *Files for plotting the radial dependences of model parameters*

During the initialization of a static calculation or MD simulation LATTE will write the files `./GSPscaling.dat` and `./ppot_plot_k.*.dat` where the value of the asterisk run from 1 to the total number of pair potentials employed in the simulation, `Nopps`. These files can be used to plot the radial dependence of the bond integrals and pair potentials, respectively. The order of the GSP functions written in `./GSPscaling.dat` is the same as the order employed in `./TBparam/bondints.dat`.

V.B.2 *Relaxed coordinates*

The file `./restartREL.dat` will be written upon the termination of a molecular statics relaxation (`RELAX= 1` in `./TBparam/control.in`) based on the maximum number of iterations or the tolerance on the maximum force. This file contains the coordinates of the atoms at the last iteration of the relaxation procedure.

V.B.3 *Features to come*

Too numerous to mention – stay tuned to our savannah page

VI. Postamble

VI.A *Problems, bugs, etc.*

In the (inevitable) event you identify a bug, mistake, or any 'issue' with LATTE, please contact the one developers. We are happy to help, and only with community involvement will LATTE improve.

VI.B *Feature requests*

If LATTE doesn't do something you think it should, please let us know. If you've developed something for LATTE and would like to see it added to the main distribution, send us the source code *along with documentation for the manual* and we'll do our best.

VII. LATTE Developers

Marc Cawkwell:	cawkwell-at-lanl.gov
Ed Sanville:	edsanville-at-gmail.com
Anders Niklasson:	amn-at-lanl.gov
Nicolas Bock:	nbock-at-lanl.gov
Josh Coe:	jcoe-at-lanl.gov
Sven Rudin:	srudin-at-lanl.gov

VIII. References

[1] A. P. Sutton, M. W. Finnis, D. G. Pettifor, and Y. Ohta, J. Phys. C: Solid State **21**, 35 (1988).

[2] A. P. Sutton, *Electronic Structure of Materials* (Oxford University Press, Oxford, 1993).

[3] D. G. Pettifor, *Bonding and Structure of Molecules and Solids* (Oxford University Press, Oxford, 1995).

[4] M. W. Finnis, *Interatomic Forces in Condensed Matter* (Oxford University Press, Oxford, 2003).

[5] W. A. Harrison, *Electronic Structure and the Properties of Solids* (Dover Publications, Inc., New York, 1989).

[6] M. Elstner, D. Porezag, G. Jungnickel, J. Elsner, M. Haugk, T. Frauenheim, S. Suhai, and G. Seifert, Phys. Rev. B **58**, 7260 (1998).

[7] M. W. Finnis, A. T. Paxton, M. Methfessel, and M. van Schilfgaarde, Phys. Rev. Lett. **81**, 5149 (1998).

[8] K. Esfarjani and Y. Kawazoe, J. Phys.: Condens. Matter **10**, 8257 (1998).

[9] J. C. Slater and G. F. Koster, Phys. Rev. **94**, 1498 (1954).

[10] T. Frauenheim, G. Seifert, M. Elstner, Z. Hajnal, G. Jungnickel, D. Porezag, S. Suhai, and R. Scholz, Phys. Stat. Sol. (b) **217**, 41 (2000).

[11] R. P. Feynman, Phys. Rev. **56**, 340 (1939).

[12] L. Goodwin, A. J. Skinner, and D. G. Pettifor, Europhys. Lett. **9**, 701 (1989).

[13] M. J. Cawkwell, D. Nguyen-Manh, D. G. Pettifor, and V. Vitek, Phys. Rev. B **73**, 064104 (2006).
[14] M. P. Allen and D. J. Tildesley, *Computer Simulation of Liquids* (Oxford University Press, Oxford, 1987).
[15] G. D. Smith, D. Bedrov, O. Byutner, O. Borodin, C. Ayyagari, and T. D. Sewell, J. Phys. Chem. A **107**, 7552 (2003).
[16] A. M. N. Niklasson, Phys. Rev. B **66**, 155115 (2002).
[17] A. M. N. Niklasson, J. Chem. Phys. **129**, 244107 (2008).
[18] A. M. N. Niklasson, Phys. Rev. Lett. **100**, 123004 (2008).
[19] A. M. N. Niklasson, P. Steneteg, A. Odell, N. Bock, M. Challacombe, C. J. Tymczak, E. Holmstrom, G. S. Zheng, and V. Weber, J. Chem. Phys. **130**, 214109 (2009).
[20] J. Li, Modelling Simul. Mater. Sci. Eng. **11**, 173 (2003).