EUROFUSION WP14ER-PR(15) 14790

PW Ma et al.

SPILADY: A Parallel CPU and GPU Code for Spin-Lattice Magnetic Molecular Dynamics Simulations

Preprint of Paper to be submitted for publication in
Computer Physics Communications

This work has been carried out within the framework of the EUROfusion Consortium and has received funding from the Euratom research and training programme 2014-2018 under grant agreement No 633053. The views and opinions expressed herein do not necessarily reflect those of the European Commission.

SPILADY: A Parallel CPU and GPU Code for Spin-Lattice Magnetic Molecular Dynamics Simulations

Pui-Wai Ma

Culham Centre for Fusion Energy, UK Atomic Energy Authority, Culham Science Centre, Abingdon, Oxfordshire, OX14 3DB, United Kingdom

S. L. Dudarev

Culham Centre for Fusion Energy, UK Atomic Energy Authority, Culham Science Centre, Abingdon, Oxfordshire, OX14 3DB, United Kingdom

C. H. Woo

Department of Physics and Materials Sciences, The City University of Hong Kong, Hong Kong SAR, China

Abstract

Spin-lattice dynamics generalizes molecular dynamics to magnetic materials, where the set of dynamic variables describing an evolving atomic system includes not only coordinates and velocities of atoms but also directions and magnitudes of atomic magnetic moments (spins). Spin-lattice dynamics simulates the collective time evolution of spins and atoms, taking into account the effect of non-collinear magnetism on interatomic forces. Applications of the method include atomistic models for defects, dislocations and surfaces in magnetic materials, thermally activated diffusion of defects, magnetic phase transitions, and various magnetic and lattice relaxation phenomena. Spin-lattice dynamics retains all the capabilities of molecular dynamics, adding to them the treatment of non-collinear magnetic degrees of freedom. The spin-lattice dynamics time integration algorithm uses symplectic Suzuki-Trotter decomposition of atomic coordinate, velocity and spin evolution operators, and delivers highly accurate numerical solutions of the evolution equations

Email addresses: leo.ma@ccfe.ac.uk (Pui-Wai Ma), sergei.dudarev@ccfe.ac.uk (S. L. Dudarev)

over extended intervals of time. The code is parallelized in the coordinate and spin spaces, and is written in OpenMP C/C++ for CPU and in CUDA C/C++ for Nvidia GPU implementations. Temperatures of atoms and spins are controlled by Langevin thermostats. Conduction electrons are treated by coupling the discrete spin-lattice dynamics equations for atoms and spins to the heat transfer equation for the electrons. Worked examples include simulations of thermalization of ferromagnetic bcc iron and the dynamics of laser pulse demagnetization.

Keywords: spin-lattice dynamics, molecular dynamics, spin dynamics

---

PROGRAM SUMMARY

Manuscript Title: SPILADY: A Parallel CPU and GPU Code for Spin-Lattice Magnetic Molecular Dynamics Simulations
Authors: Pui-Wai Ma, S. L. Dudarev, C. H. Woo
Program Title: SPILADY, version 1.0
Journal Reference:
Catalogue identifier:
Licensing provisions: Apache License, Version 2.0
Programming language: OpenMP C/C++, CUDA C/C++
Computer: Any computer with an OpenMP capable C/C++ compiler or computer with CUDA capable GPU card and an nvcc compiler.
Operating system: Linux, Unix, Windows
RAM: at least 500MB, depending on the number of atoms or atomic spins, and on the simulation type.
Number of processors used: At least 1
Supplementary material:
available on website: *http://spilady.ccfe.ac.uk*
Keywords: Spin-lattice dynamics, molecular dynamics, spin dynamics
Classification: 7.7
External routines/libraries:
Subprograms used:
Nature of problem: Excitation of magnetic degrees of freedom affects a broad range of properties of magnetic materials, including their equilibrium crystal structure and response to mechanical deformation. Existing atomistic simulation methods, for example molecular dynamics, do not treat magnetic degrees of freedom and do not describe the effect of magnetism on interatomic forces. This is addressed by the spin-lattice dynamics approach. The integration algorithm satisfies the requirement of phase volume conservation in the multi-dimensional space of atomic coordinates, velocities, and atomic spins, which is achieved through the use of the

symplectic Suzuki-Trotter decomposition. Numerical solutions of spin-lattice dynamics equations retain high accuracy over extended intervals of time.

Solution method: An atomic scale simulation technique for modelling the coupled dynamics of atomic coordinates and spins. The method generalizes molecular dynamics to the case of magnetic materials, and uses a parallel Suzuki-Trotter decomposition-based time integration algorithm.

Restrictions: The current version assumes that the material is composed of a single chemical element. Evolution equations assume the validity of localized magnetic moment approximation.

Unusual features: An open source spin-lattice dynamics code. The time integration algorithm uses the Suzuki-Trotter decomposition, which is a symplectic integration method retaining high accuracy over extended intervals of time. The code runs in parallel on multiple CPUs using OpenMP directives, or on an Nvidia GPU card.

Running time: Similar to molecular dynamics, from several minutes to several weeks or months, depending on the number of atoms or spins involved in a simulation, and on the type of the simulation.

1. Introduction

Magnetic materials play pivotal roles in a diverse range of engineering applications, involving storage, recovery, transportation, manipulation and processing of information, and quantum computing. At the same time, magnetic excitations affect the stability of crystal structures of many magnetic alloys, including iron alloys and steels. For example, the body-centred cubic crystal structure of iron appears anomalous in the context of the Periodic Table, since it owes its stability to the fact that the ground state of iron is ferromagnetic [1, 2]. Other elements in the same group of the Periodic Table have hexagonal crystal structure [3]. The high temperature softening of elastic constant C' of iron [4, 5] is fundamentally related to the $\alpha - \gamma$ phase transition, which in turn is associated with the loss of magnetic order at the Curie temperature $T_C = 1043$K. The $\alpha - \gamma$ and $\gamma - \delta$ phase transitions in iron stem from competing effects of phonon and magnon excitations,

where magnetic excitations stabilize the bcc phase whereas thermal atomic vibrations favour fcc crystal structure [6, 7]. Another example is that at low temperatures a self-interstitial defect in iron adopts the $\langle 110 \rangle$ dumbbell configuration, whereas in non-magnetic bcc metals a self-interstitial defect is a $\langle 111 \rangle$ crowdion [8, 9]. These examples show that both the phase stability [10] and dislocation-mediated mechanical properties [11] of materials are strongly influenced by magnetism.

Landau-Lifshits and Gilbert (LLG) equations [12] describe the dissipative dynamics of spins (Spin Dynamics, SD) on a rigid lattice. Solutions of the LLG equations asymptotically converge to the ground state magnetic configuration of the material. Brown [13] extended the LLG treatment and included thermal excitation of spins through the use of a stochastic method based on the fluctuation-dissipation theorem (FDT) [14, 15]. The method makes it possible to drive a spin system to thermal equilibrium at a predefined temperature. Parameters of the model can be defined by assuming that stochastic exchange fields acting on spins describe coupling to an external thermostat.

Kinematic motion of atoms in non-magnetic materials can be modelled by molecular dynamics (MD). MD treats atoms as point objects interacting through forces that depend on atomic positions. MD does not treat the magnetic degrees of freedom of atoms. Models that incorporate magnetic effects in the many-body interatomic potential formalism were developed by Dudarev and Derlet [16, 17] and Ackland [18]. These models predict the magnitudes of atomic magnetic moments for a given atomic configuration [19] but do not take into account the directional aspect of magnetic degrees of freedom necessary for the treatment of non-collinear magnetism, including for example the order-disorder magnetic transitions on a vibrating atomic lattice.

A dynamic model for spin waves, lattice vibrations, and their interaction at a finite temperature requires including magnetic degrees of freedom in molecular dynamics. Omelyan *et al.* [20] and Tsai *et al.* [21] explored models that unify spin and lattice dynamics. They noted the advantages associated with the application of the Suzuki-Trotter decomposition (STD) [22] algorithm to the treatment of non-commuting evolution operators that describe the lattice and spin degrees of freedom. The points that did not receive attention included the development of a realistic parameterization for the interatomic interaction law, the definition of the range of validity of the classical treatment of spin degrees of freedom, and the parallel computer

implementation of spin-lattice dynamics.

In Refs. [23, 24] we formulated a spin-lattice dynamics model with parameters derived from the *ab initio* data on bcc ferromagnetic iron. An algorithm for parallel spin-lattice dynamics simulations was developed in Ref. [25]. Further analysis showed how to evaluate temperature of a dynamic spin ensemble [26]. Analytical formulae that help accelerate spin dynamics simulations were derived in Ref. [27]. Spin-lattice-electron dynamics, also incorporating the electronic degrees of freedom, was developed in Refs. [28, 29] together with the treatment of longitudinal magnetic fluctuations [30].

Applications of spin-lattice dynamics (SLD) include simulations of magnetism in iron thin films [31], self-diffusion in iron [32, 33], and a dynamic model for the magnetocaloric effect in iron and gadolinium [34]. SLD simulations have been recently performed by several research groups worldwide [35, 36, 37, 38, 39]. Since a practical implementation of SLD is somewhat more intricate than that of MD, it appears timely that a working version of a spin-lattice dynamics computer code (SPILADY), suitable for carrying out a diverse range of atomistic simulations involving also the magnetic degrees of freedom of atoms, should be made available as an open source computer program.

In what follows we briefly describe the underlying theory, algorithms and parameters that control the execution of a spin-lattice dynamics simulation, and give two worked examples of application of spin-lattice dynamics. For further information we refer an interested reader to the user manual of SPILADY, version 1.0, which describes in detail the control options and variables of spin-lattice dynamics. The manual also gives extra examples showing how to use SPILADY to perform MD, SD and SLD simulations.

2. Theory

In this section we briefly describe the Hamiltonian and equations of motion for MD, SD and SLD, the incorporation of longitudinal magnetic fluctuations, the coupled heat transfer equation for the conduction electronic subsystem, and a method for treating local collective motion of atoms. Although these theories were described in detail elsewhere, we give a brief summary of the relevant concepts here.

2.1. Molecular Dynamics

We start from an MD simulation. An atomic scale classical Hamiltonian can be written as

$$\mathcal{H} = \sum_i \frac{\mathbf{p}_i^2}{2m} + U(\mathbf{R}), \tag{1}$$

where $\mathbf{p}_i$ is the momentum of atom i and U is the potential energy, which is a function of atomic coordinates $\mathbf{R} = \{\mathbf{R}_i\}$. The Hamilton equations of motion have the form

$$\frac{d\mathbf{R}_i}{dt} = \frac{\partial \mathcal{H}}{\partial \mathbf{p}_i} = \frac{\mathbf{p}_i}{m}, \tag{2}$$

$$\frac{d\mathbf{p}_i}{dt} = -\frac{\partial \mathcal{H}}{\partial \mathbf{R}_i} = -\frac{\partial U}{\partial \mathbf{R}_i}. \tag{3}$$

The above equations can be generalized to the case of Langevin dynamics as [14, 15]:

$$\frac{d\mathbf{R}_i}{dt} = \frac{\mathbf{p}_i}{m}, \tag{4}$$

$$\frac{d\mathbf{p}_i}{dt} = -\frac{\partial U}{\partial \mathbf{R}_i} - \gamma_l \frac{\mathbf{p}_i}{m} + \mathbf{f}_i, \tag{5}$$

where γ_l is a damping parameter and $\mathbf{f}_i$ is a δ-correlated fluctuating force satisfying conditions $\langle \mathbf{f}_i \rangle = 0$ and $\langle f_{i\alpha}(t) f_{j\beta}(t') \rangle = \mu_l \delta_{ij} \delta_{\alpha\beta} \delta(t - t')$. The fluctuation-dissipation theorem states that parameters μ_l and γ_l are related through $\mu_l = 2\gamma_l k_B T$. If we take T to be the temperature of an external thermostat, then Langevin equations provide a practical way of thermalizing atomic lattice to a given temperature T.

2.2. Spin Dynamics

For an arbitrary spin Hamiltonian $\mathcal{H}$, in which the spin operator variables are treated in the mean-field approximation, the equations of motion for each individual spin vector have the form [30]:

$$\frac{d\mathbf{S}_i}{dt} = \frac{1}{\hbar} \left[\mathbf{S}_i \times \mathbf{H}_i \right]. \tag{6}$$

The spin of atom i is related to its atomic magnetic moment *via* $\mathbf{S}_i = -\mathbf{M}_i/(g\mu_B)$, and the effective exchange field acting on spin $\mathbf{S}_i$ is $\mathbf{H}_i =$

$-\partial\mathcal{H}/\partial\mathbf{S}_i$. Here $\mathbf{M}_i$ is the magnetic moment of atom i, $g \approx 2.0023$ is the electron g-factor, and μ_B is the Bohr magneton.

Assuming that the magnitudes of all the atomic magnetic moments are constant, we write the Langevin equations of motion for the spins as [13, 27]:

$$\frac{d\mathbf{S}_i}{dt} = \frac{1}{\hbar}\left[\mathbf{S}_i \times (\mathbf{H}_i + \mathbf{h}_i) - \gamma_s \mathbf{S}_i \times (\mathbf{S}_i \times \mathbf{H}_i)\right], \tag{7}$$

where γ_s is a damping parameter and $\mathbf{h}_i$ is a δ-correlated fluctuating field, satisfying conditions $\langle \mathbf{h}_i \rangle = 0$ and $\langle h_{i\alpha}(t) h_{j\beta}(t') \rangle = \mu_s \delta_{ij} \delta_{\alpha\beta} \delta(t - t')$. The fluctuation-dissipation relation has the form $\mu_s = 2\hbar\gamma_s k_B T$.

Although the above spin equations of motion are valid for an arbitrary spin-dependent mean-field Hamiltonian, the Heisenberg Hamiltonian is often used in applications. The Heisenberg Hamiltonian has the form

$$\mathcal{H} = -\frac{1}{2}\sum_{i,j} J_{ij} \mathbf{S}_i \cdot \mathbf{S}_j, \tag{8}$$

where J_{ij} are the exchange coupling parameters [40]. These parameters define the strength of interaction between the spins, and determine the Curie or Neél temperatures. The exchange coupling parameters can be derived from *ab initio* calculations [40, 41].

Temperature of a dynamically evolving spin system can be evaluated using the following equation [26]:

$$T = \frac{\sum_i |\mathbf{S}_i \times \mathbf{H}_i|^2}{2k_B \sum_i \mathbf{S}_i \cdot \mathbf{H}_i}. \tag{9}$$

This formula assumes that the spin system is in thermodynamic equilibrium. Formula (9) also shows that spin temperature may take positive or *negative* values.

The above equation, derived in Ref. [26] for the Heisenberg Hamiltonian, is in fact valid for an arbitrary spin Hamiltonian, treated in the mean-field approximation. An expression for spin temperature, the derivation of which assumes that spins evolve according to equation (7), has the form

$$T = \frac{\sum_i |\mathbf{S}_i \times (-\partial\mathcal{H}/\partial\mathbf{S}_i)|^2}{2k_B \sum_i \mathbf{S}_i \cdot (-\partial\mathcal{H}/\partial\mathbf{S}_i)}. \tag{10}$$

2.3. Longitudinal spin fluctuations

If spins exhibit longitudinal fluctuations, i.e. fluctuations of magnitudes of atomic spins (or magnetic moments), they are described by another type of Langevin equations of motion [30]:

$$\begin{aligned} \frac{d\mathbf{S}_i}{dt} &= \frac{1}{\hbar}\left[\mathbf{S}_i \times \mathbf{H}_i\right] - \gamma_s' \frac{\partial \mathcal{H}}{\partial \mathbf{S}_i} + \boldsymbol{\xi}_i(t), \\ &= \frac{1}{\hbar}\left[\mathbf{S}_i \times \mathbf{H}_i\right] + \gamma_s' \mathbf{H}_i + \boldsymbol{\xi}_i(t). \end{aligned} \tag{11}$$

Here γ_s' is a damping parameter and $\boldsymbol{\xi}_i(t)$ is a δ-correlated random process satisfying conditions $\langle \boldsymbol{\xi}_i \rangle = 0$ and $\langle \xi_{i\alpha}(t)\xi_{j\beta}(t')\rangle = \mu_s' \delta_{ij}\delta_{\alpha\beta}\delta(t-t')$. The fluctuation-dissipation relation for the longitudinal spin noise has the form $\mu_s' = 2\gamma_s' k_B T$. Equation (11) is equivalent to equation (7) if the spin vector is constrained to the surface of a sphere [30].

Equilibrium spin temperature can now be calculated using the following formula [30]:

$$T = \frac{\sum_{i,\alpha} \left(\partial \mathcal{H}/\partial S_{i\alpha}\right)^2}{k_B \sum_{i,\alpha} \partial^2 \mathcal{H}/\partial S_{i\alpha}^2}, \tag{12}$$

where α denotes the Cartesian components.

A Hamiltonian involving both the Heisenberg and longitudinal Landau terms can have the form [30, 34]:

$$\mathcal{H} = \mathcal{H}_H + \mathcal{H}_L, \tag{13}$$

where

$$\mathcal{H}_H = -\frac{1}{2}\sum_{i,j} J_{ij}\mathbf{S}_i \cdot \mathbf{S}_j, \tag{14}$$

$$\mathcal{H}_L = \sum_i A_i S_i^2 + B_i S_i^4 + C_i S_i^6 + D_i S_i^8. \tag{15}$$

The Landau Hamiltonian $\mathcal{H}_L$ is a polynomial in $S_i^2 = \mathbf{S}_i^2$. Numerical coefficients in the Landau Hamiltonian can also be determined from *ab initio* calculations [30, 34].

2.4. Spin-Lattice Dynamics

One of our goals is to develop a self-consistent treatment of coupled magnetic excitations and atomic vibrations. We start from a Hamiltonian that treats the atomic and spin subsystems, and interaction between them. Hamiltonian now has the form $\mathcal{H} = \mathcal{H}(\mathbf{R}, \mathbf{p}, \mathbf{S})$. In SPILADY we use two different Hamiltonians, depending on whether or not the longitudinal fluctuations of magnetic moments are taken into account.

If there are no longitudinal fluctuations, we use a Hamiltonian that has the form

$$\mathcal{H} = \mathcal{H}_{latt} + \mathcal{H}_{spin} + \mathcal{H}_{corr}, \tag{16}$$

where

$$\mathcal{H}_{latt} = \sum_i \frac{\mathbf{p_i}^2}{2m} + U(\mathbf{R}), \tag{17}$$

$$\mathcal{H}_{spin} = -\frac{1}{2}\sum_{i,j} J_{ij}(\mathbf{R})\mathbf{S}_i \cdot \mathbf{S}_j, \tag{18}$$

$$\mathcal{H}_{corr} = \frac{1}{2}\sum_{i,j} J_{ij}(\mathbf{R})|\mathbf{S}_i||\mathbf{S}_j|. \tag{19}$$

Here the coordinate-dependent exchange coupling function is assumed to be a pairwise function of atomic coordinates $J_{ij}(\mathbf{R}) = J_{ij}(R_{ij})$. Equation (19) is a correction term for the spin Hamiltonian. Since the magnitude of spin $|\mathbf{S}_i|$ is constant, this term is a function of atomic coordinates only. The introduction of this correction term stems from simple convenience, as it makes it possible to use the existing parametrizations of many-body potentials, for example the Dudarev-Derlet 2005 (DD05) iron potential [16]. This also implies that at $T = 0$K the system is ferromagnetically ordered. Parameterizations required for modelling materials characterized by antiferromagnetic or non-collinear magnetic order in their magnetic ground state, have not yet been included in SPILADY. This does not of course affect the form of the spin or spin-lattice dynamics equations.

Equations of motion for the atomic coordinates, momenta, and spins can now be written as:

$$\frac{d\mathbf{R}_i}{dt} = \frac{\mathbf{p}_i}{m}, \tag{20}$$

$$\frac{d\mathbf{p}_i}{dt} = -\frac{\partial \mathcal{H}}{\partial \mathbf{R}_i} = -\frac{\partial U}{\partial \mathbf{R}_i} - \frac{\partial J_{ij}}{\partial \mathbf{R}_i}(\mathbf{S}_i \cdot \mathbf{S}_j - |\mathbf{S}_i||\mathbf{S}_j|), \tag{21}$$

$$\frac{d\mathbf{S}_i}{dt} = \frac{1}{\hbar}\left[\mathbf{S}_i \times \mathbf{H}_i\right], \tag{22}$$

where the local effective field $\mathbf{H}_i = -\partial\mathcal{H}/\partial\mathbf{S}_i = \sum_i J_{ij}\mathbf{S}_j$. From the above equations, we see that the lattice and spin subsystems are coupled via a coordinate dependent exchange function $J_{ij}(\mathbf{R})$. A Langevin thermostat can now be added using the same fluctuation and dissipation terms as those discussed above.

If longitudinal magnetic fluctuations are included, we rewrite the spin part of the Hamiltonian (18) as

$$\mathcal{H}_{spin} = \mathcal{H}_H + \mathcal{H}_L, \tag{23}$$

where

$$\mathcal{H}_H = -\frac{1}{2}\sum_{i,j} J_{ij}(\mathbf{R})\mathbf{S}_i \cdot \mathbf{S}_j \tag{24}$$

$$\mathcal{H}_L = \sum_i A_i(\mathbf{R})S_i^2 + B_i(\mathbf{R})S_i^4 + C_i(\mathbf{R})S_i^6 + D_i(\mathbf{R})S_i^8. \tag{25}$$

The Landau Hamiltonian is now a function of atomic coordinates. In SPILADY we assume that $A_i(\mathbf{R}) = A_i(\rho_i)$, where ρ_i is the local effective electron density entering the EAM potential. Other Landau coefficients are treated in the same way. We now need to modify equations of motion for the kinematic atomic momenta as

$$\begin{aligned}\frac{d\mathbf{p}_i}{dt} &= -\frac{\partial U}{\partial \mathbf{R}_i} - \frac{\partial J_{ij}}{\partial \mathbf{R}_i}(\mathbf{S}_i \cdot \mathbf{S}_j - |\mathbf{S}_i||\mathbf{S}_j|), \\ &+ \frac{\partial A_i}{\partial \mathbf{R}_i}S_i^2 + \frac{\partial B_i}{\partial \mathbf{R}_i}S_i^4 + \frac{\partial C_i}{\partial \mathbf{R}_i}S_i^6 + \frac{\partial D_i}{\partial \mathbf{R}_i}S_i^8.\end{aligned} \tag{26}$$

If we adopt the Langevin thermostat method mentioned in connection with Eq. (11), then there is a subtlety associated with the definition of the effective field. Since the magnitudes of magnetic moments are no longer constant, the correction term given by Eq. (19) needs to be included in the calculation of the effective field, namely

$$\mathbf{H}_i = \sum_j J_{ij}(\mathbf{S}_j - \frac{S_j}{S_i}\mathbf{S}_i) - (2A_i + 4B_iS_i^2 + 6C_iS_i^4 + 8D_iS_i^6)\mathbf{S}_i. \tag{27}$$

2.5. Langevin Treatment of the Electron Subsystem

Thermal conductivity of metals is primarily associated with the transport of conduction electrons. Heat dissipation through electron subsystem has a significant effect on the motion of atoms, especially in high energy events like collision cascades [42, 43]. The stochastic Langevin equation treatment of atomic degrees of freedom can be linked to the heat transfer equation describing conduction electrons [42, 29]. This makes it possible to incorporate electronic degrees of freedom in MD, SD or SLD [29].

MD equations of motion, describing atoms coupled to the electrons, are

$$\frac{d\mathbf{R}_i}{dt} = \frac{\mathbf{p}_i}{m}, \tag{28}$$

$$\frac{d\mathbf{p}_i}{dt} = -\frac{\partial \mathcal{H}}{\partial \mathbf{R}_i} - \gamma_l \frac{\mathbf{p}_i}{m} + \mathbf{f}_i, \tag{29}$$

$$C_e \frac{dT_e}{dt} = \nabla(\kappa_e \nabla T_e) + G_{el}(T_l - T_e), \tag{30}$$

where C_e is the electronic specific heat and κ_e is the thermal conductivity of the electrons. Heat transfer between the lattice and the electrons is described by the coefficient

$$G_{el} = \frac{3k_B\gamma_l}{m\Omega}, \tag{31}$$

where Ω is the volume per atom. In equations (30) T_l is the temperature of the lattice evaluated from the *local* kinetic energy of the atoms.

A spin dynamics simulation, where spin excitations are treated as pure rotations coupled to electrons, is described by the following set of equations

$$\frac{d\mathbf{S}_i}{dt} = \frac{1}{\hbar}\left[\mathbf{S}_i \times (\mathbf{H}_i + \mathbf{h}_i) - \gamma_s \mathbf{S}_i \times (\mathbf{S}_i \times \mathbf{H}_i)\right], \tag{32}$$

$$C_e \frac{dT_e}{dt} = \nabla(\kappa_e \nabla T_e) + G_{es}(T_s - T_e). \tag{33}$$

The coefficient of heat transfer between the spins and electrons is

$$G_{es} = \frac{2k_B\gamma_s}{\hbar\Omega}\langle \mathbf{S}_i \cdot \mathbf{H}_i \rangle. \tag{34}$$

In practical simulations, the ensemble average values of $\langle \mathbf{S}_i \cdot \mathbf{H}_i \rangle$ and T_s are calculated using vectors $\{\mathbf{S}_i(t)\}$ within a particular linked cell.

In an SD simulation that includes longitudinal fluctuations [30], we use equations

$$\frac{d\mathbf{S}_i}{dt} = \frac{1}{\hbar}\left[\mathbf{S}_i \times \mathbf{H}_i\right] + \gamma_s' \mathbf{H}_i + \boldsymbol{\xi}_i, \tag{35}$$

$$C_e \frac{dT_e}{dt} = \nabla(\kappa_e \nabla T_e) + G_{es}(T_s - T_e), \tag{36}$$

where the coefficient of heat transfer between the spins and electrons is

$$G_{es} = \frac{k_B \gamma_s'}{\Omega} \left\langle \sum_\alpha \frac{\partial^2 \mathcal{H}}{\partial S_{i\alpha}^2} \right\rangle. \tag{37}$$

Here α is the index of a Cartesian component of a spin vector.

In an SLD simulation, the heat transfer equations have the form [29]

$$C_e \frac{dT_e}{dt} = \nabla(\kappa_e \nabla T_e) + G_{el}(T_l - T_e) + G_{es}(T_s - T_e). \tag{38}$$

Equations of motion for the spins and lattice atoms, and the definitions of constants and variables remain the same as above.

2.6. Local Collective Motion of Atoms and Electrons

For a group of atoms moving uniformly in the same direction, a suitable definition of kinetic energy requires using the moving frame associated with the centre of mass of the entire group, namely

$$T_l = \frac{2}{3Nk_B} \sum_i \frac{(\mathbf{p}_i - \mathbf{P})^2}{2m}, \tag{39}$$

where $\mathbf{P} = 1/N \sum_i \mathbf{p}_i$ is the average kinematic momentum of the atoms.

In the heat transfer equation, the local temperature of the lattice is normally calculated assuming that there is no local collective motions in the system. In the presence of local collective motion, this approximation strongly overestimates the rate of energy dissipation from the lattice subsystem to the electrons. A way to tackle this is to treat the lattice-electron energy transfer in a local moving frame. Assuming that the lattice-electron interactions occur locally, we write [29]

$$\frac{d\mathbf{R}_i}{dt} = \frac{\mathbf{p}_i}{m}, \tag{40}$$

$$\frac{d\mathbf{p}_i}{dt} = -\frac{\partial \mathcal{H}}{\partial \mathbf{R}_i} - \frac{\gamma_l}{m}(\mathbf{p}_i - \mathbf{p}_A) + (\mathbf{f}_i - \mathbf{f}_A), \quad (41)$$

$$C_e \frac{dT_e}{dt} = \nabla(\kappa_e \nabla T_e) + G_{el}(T_l - T_e), \quad (42)$$

where

$$\mathbf{p}_A = 1/N_A \sum_{i \in A} \mathbf{p}_i, \quad (43)$$

$$\mathbf{f}_A = 1/N_A \sum_{i \in A} \mathbf{f}_i, \quad (44)$$

$$G_{el} = \frac{3k_B(N_A - 1)\gamma_l}{mV_A}. \quad (45)$$

Here A refers to a local region in the material, N_A is the number of atoms in region A, and V_A is the volume of A. In practical simulations, we identify A with a linked cell.

3. Algorithm

Fig. 1 shows the flow chart of SPILADY. The basic structure of SPILADY is similar to that of a conventional MD program [44]. It starts with the initialization of the time variable, the lattice structure, momenta, spins, the linked cells structure, the tables describing interatomic potentials, various random numbers, and - if necessary - the parameters controlling the execution of the program on a GPU. It also calculates the initial effective electron densities, energies, forces and temperatures. Then, it performs an initial check on energies, temperatures, pressure, stresses, and magnetic moments, followed by the generation of a full record of the initial configuration. The content of the output files is described in the user manual, which is distributed together with SPILADY.

After the initialization, the program starts integrating the equations of motion, taking a time step of size **step**. It performs a consistency check again after each, or several, time steps. After a certain number of time steps, the box size can be re-scaled to make it consistent with the internal stresses or pressure. The size of the time step can also be adjusted according to the maximum displacement of atoms and/or the maximum precession angle of spins. The program generates intermediate output files after a certain

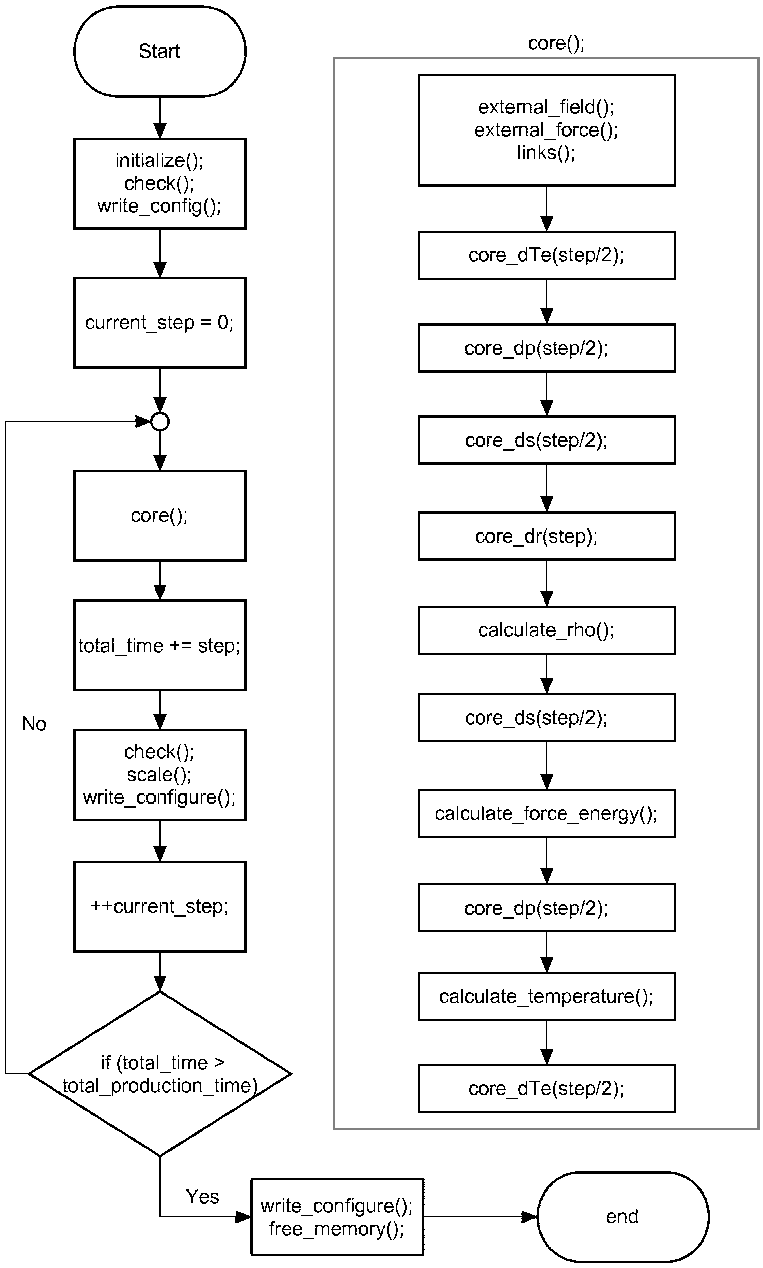

Figure 1: A flow chart for SPILADY

number of time steps. When the total time reaches a limiting value, the program ends and writes the final configuration into a file.

An unusual aspect of the program is the integration algorithm, which is different from that used in the majority of MD programs. We adopted an algorithm based on the Suzuki-Trotter decomposition (STD) [22]. The Suzuki-Trotter algorithm is symplectic, and it conserves volume in phase space, in this way minimizing numerical errors over extended intervals of time. Omelyan *et al.* [20] and Tsai *et al.* [21] investigated applications of STD to spin-lattice dynamics. However, due to the mathematical structure of the STD integration algorithms, it is intrinsically difficult to parallelize STD simulations in the spin subspace. We addressed this problem in Ref. [25] and implemented the related algorithm in SPILADY.

The basic principle of STD is to break a global evolution operation into several simpler sub-evolution steps [22]. The second order STD reads:

$$e^{(A+B)\Delta t} = e^{A\Delta t/2} e^{B\Delta t} e^{A\Delta t/2} + O(\Delta t^3), \tag{46}$$

where A and B are arbitrary operators. In the treatment of SLD, if we represent all the degrees of freedom by a generalized coordinate $\mathbf{x} = \{\mathbf{R}, \mathbf{p}, \mathbf{S}\}$, and if there is an operation on $\mathbf{x}$ such that:

$$\frac{d\mathbf{x}}{dt} = (\mathcal{R} + \mathcal{P} + \mathcal{S})\mathbf{x}, \tag{47}$$

where $\mathcal{R}$, $\mathcal{P}$ and $\mathcal{S}$ are operators acting on positions, momenta and spins, respectively, a solution corresponding to a time interval Δt can be written as follows:

$$\mathbf{x}(t + \Delta t) = e^{(\mathcal{R}+\mathcal{P}+\mathcal{S})\Delta t}\mathbf{x}(t). \tag{48}$$

Using the second order STD, we write:

$$e^{(\mathcal{R}+\mathcal{P}+\mathcal{S})\Delta t} = e^{\mathcal{P}\Delta t/2} e^{\mathcal{S}\Delta t/2} e^{\mathcal{R}\Delta t} e^{\mathcal{S}\Delta t/2} e^{\mathcal{P}\Delta t/2} + O(\Delta t^3). \tag{49}$$

This is only one of the ways how the global evolution operator can be decomposed. We adopt this specific decomposition because it minimizes the number of times when forces are evaluated. This speeds up the execution of the program.

Examining Fig. 1 we see that in function **core()** the updating of positions, momenta and spins is performed according to Eq. (49). In the Langevin treatment of the electron subsystem, we treat the electronic temperature T_e

as an extra degree of freedom, where now $\mathbf{x} = \{\mathbf{R}, \mathbf{p}, \mathbf{S}, T_e\}$, and T_e is defined individually in each linked cell. We can insert the evolution operator for T_e immediately before and after function **core()** in accord with the second order STD.

There is a subtlety associated with the treatment of the spin evolution operator, since it affects its own evolution. The effective field $\mathbf{H}_i$ acting on a particular spin depends on all its neighbours, and even on itself. It is possible to decompose the evolution operator for a system of spins into a series of single spin operations using the STD, such that:

$$e^{\mathcal{S}\tau} = e^{\mathcal{S}_1\tau/2} e^{\mathcal{S}_2\tau/2} \cdots e^{\mathcal{S}_N\tau} \cdots e^{\mathcal{S}_2\tau/2} e^{\mathcal{S}_1\tau/2} + O(\tau^3), \tag{50}$$

where τ is a time step. However, since a single spin operation depends on a previous operation, it is impossible to perform the calculation simultaneously. Therefore, it appears as if the mathematical structure of the algorithm intrinsically prohibits the parallelization of spin dynamics simulations.

However it is still possible to parallelize computations if the coupling between the spins extends only to a finite number of neighbouring atoms [25]. In an interatomic potential, a cut-off distance is used to limit the number of neighbouring atoms that contribute to the evaluation of force acting on an atom, in this way reducing computation time. This means that the evolution operator for an atom or a spin only depends on a finite number of its neighbours and does not depend on the configuration of the entire system. Parallelization of the evolution operators for the spin subspace involves the manipulation of the linked cell structure. We divide the linked cells into groups, where the atoms/spins belonging to a particular linked cell do not interact with atoms/spins in other linked cells in the same group. Therefore, we can parallelize the evaluation of evolution operators for linked cells in the same group. Notably, one still needs to treat each group in accordance with the STD. For example, if we separate the linked cells into groups A to H, the second order decomposition must be performed as follows:

$$e^{(A+B+\cdots+H)\Delta t} \approx e^{A\Delta t/2} e^{B\Delta t/2} \cdots e^{H\Delta t} \cdots e^{B\Delta t/2} e^{A\Delta t/2}. \tag{51}$$

Evolution operators for atoms/spins within a linked cell still need to be treated sequentially.

4. Controls

There are various control parameters and files that direct the execution of SPILADY, including compilation of the program, and the way how it in-

puts and outputs data. After downloading the program as a compressed file *spilady1.0.tar.gz*, one needs to decompress it. There are several files that require attention here. They are *make.sh*, *control.h*, and *vairables.in*. To perform an MD or an SLD simulation, one needs a many-body potential file, for example *DD05.cpp*. To perform an SD or SLD simulation, one needs a Heisenberg-Landau function file, for example *JijFe.cpp*. To perform a simulation involving the electronic subsystem using Langevin thermostat, the electron heat capacity file *heatcapacity_CPU.cpp* or *heatcapacity_GPU.cu* is required.

We tested our program and examples on a computer with Dual Intel Xeon Processors E5 2680v2 2.8GHz (10 cores) and Nvidia GeForce GTX Titan Black GPU cards, in the Linux environment using gcc verion 4.4.7, icc version 12.0.0 and nvcc version 5.5. SPILADY was also tested on Nvidia GeForce GTX 480, GTX 680, GTX Titan, Tesla K40c GPU cards and Tesla M2090 module. The program is written using standard syntax and is expected to run on any generic computer system. No linking to external libraries is required.

4.1. Compilation of the program

In principle, SPILADY can be compiled using a single command. To simplify it further, a script file called *make.sh* is provided. In Unix or Linux environments, one can generate an executable file by entering the following command in a shell:

```
$ chmod +x make.sh
```

If a CPU computer is used, it is assumed that an OpenMP capable compiler is used, for example *icc* or *g++*. When using an Nvidia GPU, the default option is to use *nvcc*. One needs to compile *all* the files in the working directory with suitable options, for example:

```
$ g++ -fopenmp -o spilady -DCPU -DOMP -DMD *.cpp
```

Option *-fopenmp* allows the compiler to recognize OpenMP directives in the code. If *icc* is used, it should be replaced with *-openmp*. Option *-o spilady* tells the compiler to name the resulting executable file *spilady*. Option *-DCPU* defines word *CPU* in the code, with *-DOMP* and *-DMD* performing similar functions, with the latter specifying that the program is going to be

compiled with the sole purpose of performing only MD simulations. Basically, at the compilation stage there is no need to change anything except the last option (which in the above example is *-DMD*) specifying the type of a simulation that a user intends to perform. The list of options is listed in table 1.

Table 1: Compiling options

Option	Type of calculation	Hamiltonian
-DMD	MD	EAM
-DSDH	SD	Heisenberg
-DSDHL	SD	Heisenberg-Landau
-DSLDH	SLD	EAM + Heisenberg
-DSLDHL	SLD	EAM + Heisenberg-Landau

Compiling the program for the execution on a GPU is similar. Users are advised to use at least CUDA version 5.5, which is the version that was used for testing the program. The program can be compiled using a suitably edited version of the command:

```
$ nvcc -arch=sm_35 -rdc=true -o spilady -DGPU -DMD *.cpp *.cu
```

Option *-arch=sm_35* tells the compiler to produce an executable file for devices with hardware architecture version 3.5 (Fermi). It is safe to use *-arch=sm_20*, however using versions lower than 2.0 is not advisable. Option *-rdc=true* is important. It allows the device codes placed in different files to be recognized by other codes in other files. Although this happens automatically in CPU compilers, this is not the case for the GPU compiler *nvcc*. Other compilation options are similar to those of the CPU case.

All the files in the working directory are going to be compiled. If there are unnecessary files, it is best to move them out of the working directory. For example, if there are multiple interatomic potential files, e.g. *DD05.cpp*, only one of them should remain in the folder.

4.2. Potential and heat capacity files

To preparing SPILADY for performing MD and SLD simulations, one needs to retain one and only one embedded atom method (EAM) many-body potential file in the working directory. Similarly, to perform an SD or

an SLD simulation one should retain one and only one Heisenberg-Landau function file in the working directory.

The potential file must contain parameters describing the potential energy of interaction between the atoms in a standard EAM function format:

$$U(\mathbf{R}) = \sum_i F_i(\rho_i) + \frac{1}{2}\sum_{i,j} V_{ij}(R_{ij}), \tag{52}$$

where $\rho_i = \sum_j f_{ij}(R_{ij})$ is the effective electron density.

It is necessary to provide exact functional forms of F_i, V_{ij} and f_{ij} in the interatomic potential file. This can be accomplished by editing functions **bigf_gen**, **pair_gen** and **smallf_gen** in *DD05.cpp*. At the start of a simulation SPILADY converts these functions into tables for the actual calculation of energies and forces.

In the Heisenberg-Landau function file, one needs to enter parameters for the Heisenberg-Landau Hamiltonian:

$$\begin{aligned} \mathcal{H}_{spin} &= -\frac{1}{2}\sum_{i,j} J_{ij}(R_{ij})\mathbf{S}_i \cdot \mathbf{S}_j \\ &+ \sum_i \left(A_i(\rho_i)S_i^2 + B_i(\rho_i)S_i^4 + C_i(\rho_i)S_i^6 + D_i(\rho_i)S_i^8\right). \end{aligned} \tag{53}$$

It is necessary to input the exact functional forms of J_{ij}, A_i, B_i, C_i and D_i and their parameterizations into the Heisenberg-Landau function file. This can be done by editing functions **Jij_gen**, **LandauA_gen**, **LandauB_gen**, **LandauC_gen** and **LandauD_gen** in *JijFe.cpp*. At the start of a simulation these functions are going to be converted into tables for the actual calculations of energies and forces.

To perform a Langevin dynamics simulation of coupled atomic and electron subsystems, one needs to edit file *heatcapacity_CPU.cpp*, provided that the simulation is going to be performed on a CPU computer. It may also be necessary to amend functions **Ce**, **Te_to_Ee** and **Ee_to_Te**. If the simulation is going to be performed on a GPU, it is also necessary to edit file *heatcapacity_GPU.cu* and functions **Ce_d**, **Te_to_Ee_d** and **Ee_to_Te_d**. The functional form of electron heat capacity per atom $C_e = a\tanh(bT_e)$ assumed in the program follows Ref. [42]. If necessary, this function can be modified by defining a relationship between the electron temperature T_e and electron energy per atom E_e in analytical form.

If any of the potential or the heat capacity files have been amended or modified, the program needs to be recompiled.

4.3. Input, output and control options

Full description of input, output and control directives can be found in the user manual, which is distributed together with the program.

Variables can be initialized in the file *variables.in*. Some variables have default values, while some do not. It is important to input initial numerical values of the relevant variables properly in accord with the user manual. If changes are restricted to file *variables.in* there is no need to recompile the program. For performing a GPU simulation, it is necessary to input the GPU card index using variable **current_device**. For a hardware architecture lower than 3.5, we suggest using 32 threads per block. Otherwise, we suggest using 64 or 192. This can be defined using variable **no_of_threads**.

All the control options can be defined in file *control.h*. These options can be switched on or off by uncommenting or commenting them. For example, one can switch on or off the Langevin thermostats for the spin and lattice subsystems, include the Langevin treatment of the electron subsystem, switch on the option describing the separation of local collective motion etc. SPILADY also includes an implementation of the quantum thermostat for the atomic lattice, following the method developed by Barret and Rodney [45]. Any changes in file *control.h* should be followed by the recompilation of the program to take effect.

In an SD or SLD simulation, one can choose whether to use magnetic moments or atomic spins as input and output variables. This point requires attention, for example if we choose to input and output magnetic moments, it is necessary to uncomment #**define magmom** in *control.h*. The Heisenberg-Landau Hamiltonian then acquires the form:

$$\begin{aligned} \mathcal{H}_{spin} &= -\frac{1}{2}\sum_{i,j} J_{ij}(R_{ij})\mathbf{M}_i \cdot \mathbf{M}_j \\ &+ \sum_i \left(A_i(\rho_i)M_i^2 + B_i(\rho_i)M_i^4 + C_i(\rho_i)M_i^6 + D_i(\rho_i)M_i^8\right). \end{aligned} \quad (54)$$

Parameters of this Hamiltonian would need to be fitted for the magnetic moments, since now $\{\mathbf{M}_i\}$ are used as fundamental magnetic variables. In particular, one needs to pay attention to making the correct choice of J_{ij} and the Landau coefficients in the Heisenberg-Landau function file $JijFe.cpp$. Independently of the choice of the input and output variables, all the internal calculations in SPILADY are performed using atomic spins as variables.

Values of magnetic moment and atomic spin vectors are related simply as $\mathbf{M}_i = -g\mu_B \mathbf{S}_i$.

All the output files have the same body as the content of variable **out_body**. For example, if we include a line "*out_body xxx*" in *variables.in*, SPILADY will generate output files with the following names: (1) *enr-xxx.dat*, (2) *prs-xxx.dat*, (3) *str-xxx.dat*, (4) *tmp-xxx.dat*, (5) *spn-xxx.dat*, (6) *cel-xxx_nnnn.dat*, (7) *con-xxx_nnnn.dat*, and (8) *vsm-xxx_nnnn.ascii/.spin/.dat*. The first five files accumulate the data on the average energy, pressure, stresses, temperatures, and magnetic moments in the simulation cell every **interval_of_print_out** steps.

The sixth file contains data on the temperature of each linked cell in this particular instance. This file is generated only if #**define eltemp** is switched on. The program produces an output file every **interval_of_config_out** number of steps. An extra part is added to the file name to form a sequence of output files. The value of *nnnn* spans the interval from 0000 to 9999. After initialization, before any calculation is performed, a file *cel-xxx_0000.dat* is generated. Then, the file index increases by 1 for each output file. When the program reaches the upper limit for the number of steps or the upper limit for the execution time, a file of the form *cel-xxx_9999.dat* is generated, regardless of the previous output file numbering. Any of the numbered files can be used as input for a new calculation using variable **in_eltemp**.

The seventh file contains information about an atomic configuration at a given moment of time. The program produces an output file every **interval_of_config_out** number of steps. If #**define magmom** is switched on, values of spin vectors are replaced by magnetic moments. Columns of data only appear when specific degrees of freedom are considered. Any of the numbered files can be used as an input file for a new calculation, this is controlled by variable **in_config**.

The eighth set of files contains information about atomic positions, momenta and spin vectors. The program then produces an output file for every **interval_of_vsim** steps. Details can be found in the user manual. These files can be used to visualize atoms and spins using the freely available program V_sim, developed by CEA, France [46], an example of application of which is given below. These files can also be used as input files similarly to *con-xxx_nnnn.dat*. However, since they are generated for the visualization purposes, their default precision includes only 4 meaningful digits. One should change it into 16 digits using variable **vsim_prec** if the data are to be used as input for another simulation.

5. Applications

Two examples illustrating application of SPILADY program are given below, and more examples can be found in the user manual. Interatomic potential for pure iron is used in both cases. The many-body potential [16], the exchange coupling function [24], the Landau coefficients [34], and electron heat capacity [42] are already included in files *DD05.cpp*, *JijFe.cpp* and *heatcapacity_CPU.cpp* or *heatcapacity_GPU.cu*. Function J_{ij} is slightly adjusted in comparison with Ref. [24] to achieve better agreement with the experimentally observed Curie temperature T_C.

It is advisable to use the CPU version of SPILADY before trying the GPU version. The content of the source code is easier to follow in the CPU version. Users are also advised to use the CPU version, instead of the GPU version, for carrying out SD and SLD simulations if the system size is smaller than 54,000 spins/atoms. This is because parallelization of the Suzuki-Trotter decomposition algorithm involves linked cells rather than atoms. If the system size is small, the number of linked cells within a non-interacting group of cells is limited, and the powerful parallel environment of a GPU cannot be fully utilized. On the other hand, if the system size approaches a million atoms, the use of a GPU is justified.

5.1. Thermalization of Magnetic BCC Iron

In this application we are going to illustrate the dynamics of thermalization of a spin-lattice system with longitudinal fluctuations, starting from a collinear ferromagnetic perfect lattice configuration and evolving the system into a finite temperature state. The size of the simulation box is relaxed according to its internal pressure. This worked example can be used as a test to see if the compilation and running of the program were performed correctly. Files *control.h* and *variables.in* are in folder *example5*, these files should be copied to the working directory before the compilation of the program.

An atomic configuration of bcc iron containing 16000 atoms is initialized according to the content of the input files. There are $20 \times 20 \times 20$ unit cells, where each unit cell contains 2 atoms. The thermostat temperature is set at 300K. Fig. 2 shows how temperatures change as functions of time. In the figure, T_l is the kinetic temperature of the lattice, T_s and T_s' are the spin temperatures calculated using Eq. (9) and (12), respectively. The plots show that all the temperatures reach 300K eventually, following their own timescales, which are defined by the respective damping parameters.

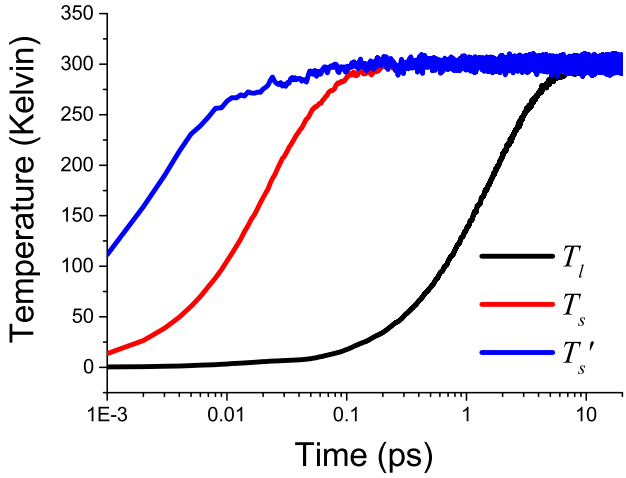

Figure 2: A spin-lattice system with longitudinal fluctuation is thermalized from collinear ferromagnetic and perfect lattice configuration to 300K.

Damping constants γ_l and γ_s are assigned according to our previous studies [29, 30]. Since coupling between the lattice and the thermostat is weaker than between the spin subsystem and the thermostat, the rate of thermalization of the lattice degrees of freedom is slower.

The two spin temperatures take different values during the thermalization process because the first (T_s) describes collective magnetic excitations, which take longer to reach equilibrium, whereas the second (T_s') includes longitudinal fluctuations, which evolve towards the Gibbs distribution faster. Because of that T_s' reaches the final asymptotic value sooner than T_s. At equilibrium T_s equals T_s'.

If option #**defined PRESSURE** is switched on, the simulation box relaxes in response to its internal pressure. Fig. 3 shows that initially the internal pressure is positive, and forces acting on atoms expand the simulation cell. As a result, the average lattice constant increases and reaches a maximum. At approximately the same moment of time pressure reaches zero, which is the target value of **pressure** variable in *variables.in*.

Using the output files *vsm-test_SLDHL_9999.ascii/.spin* we can visualize atoms and spins using V_sim. Fig. 4 shows that atoms are displaced from the initially perfect lattice positions, and also that spin configurations become non-collinear, due to thermal excitations. Forces and effective exchange fields

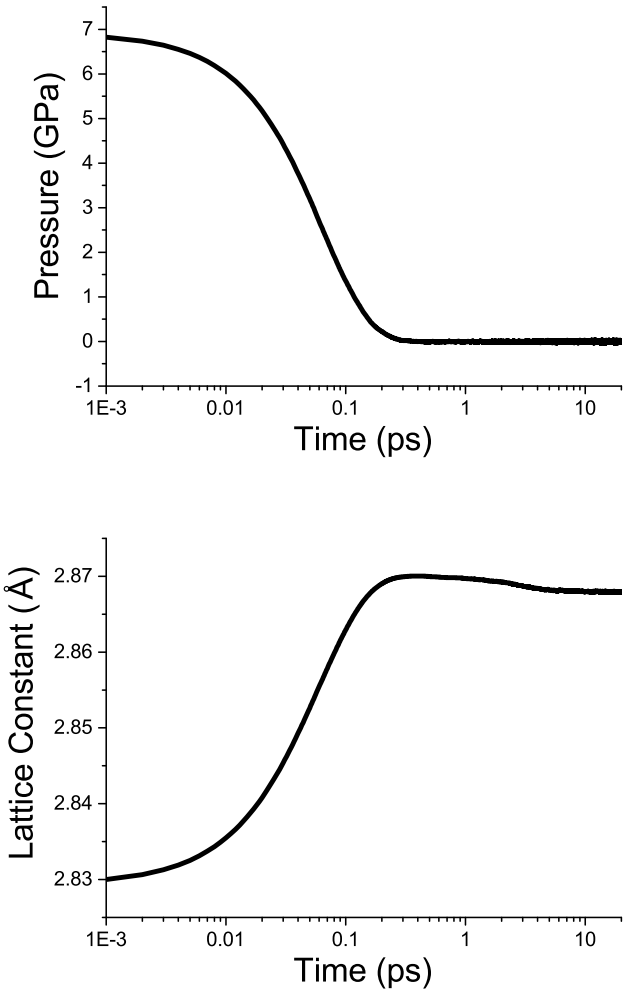

Figure 3: (Top) The change of the internal pressure of a simulation box during a thermalization to 300K. (Bottom) The change of the lattice constant as a function of time.

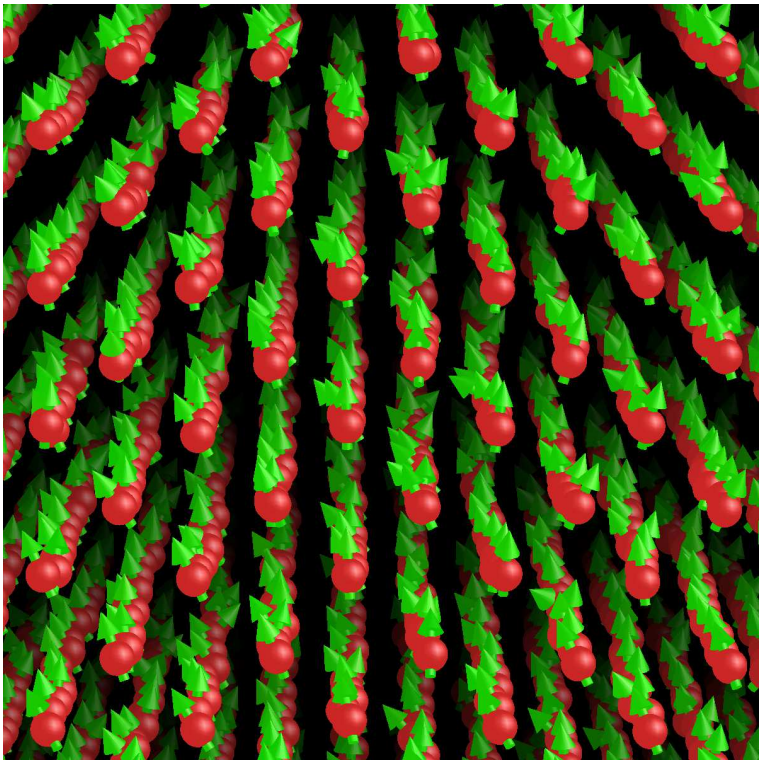

Figure 4: The atomic and spin configuration at 300K visualized by V_sim.

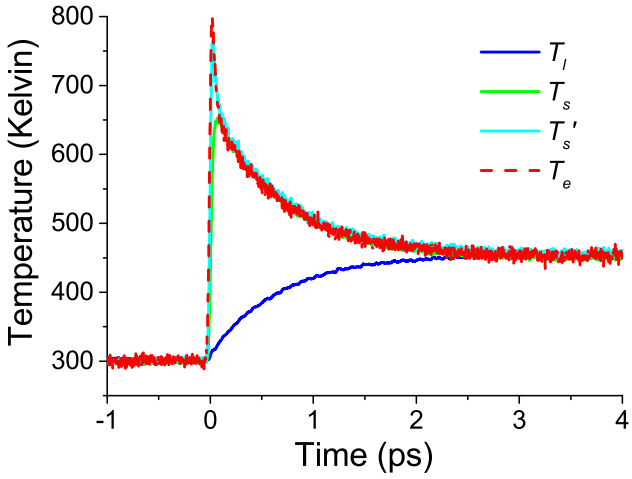

Figure 5: Variation of temperatures as functions of time for a case where a laser pulse occurs at 0ps.

acting on each atom depend on both atomic and spin configurations.

5.2. Laser Pulse Demagnetization

The second example is a little more challenging. Magnetic iron interacts with a laser pulse occurring at $t = 0$. This is mimicked by adding an extra amount of energy to the electron subsystem, effectively assuming that all the energy of the laser pulse is absorbed by the electrons. A Gaussian profile of the energy pulse is assumed in agreement with the experimental realizations of laser pulse demagnetization effects [47]. In our earlier work, we simulated a demagnetization experiment, neglecting longitudinal fluctuations [29]. Here, simulations involve both the longitudinal and transverse spin degrees of freedom.

To perform the simulations we need to edit a small part of the source code. All the edited files can be found in directory *example7*. To perform the simulations described in this worked example one needs to copy all the files into the working directory, and compile and run the program. File *laser_demagnetization_CPU.cpp* contains a function describing a Gaussian energy pulse with the standard deviation of 15fs, corresponding to a 60fs experimental pulse absorbed by the electrons. We remind the reader that when a new function is added to the program, its prototype needs

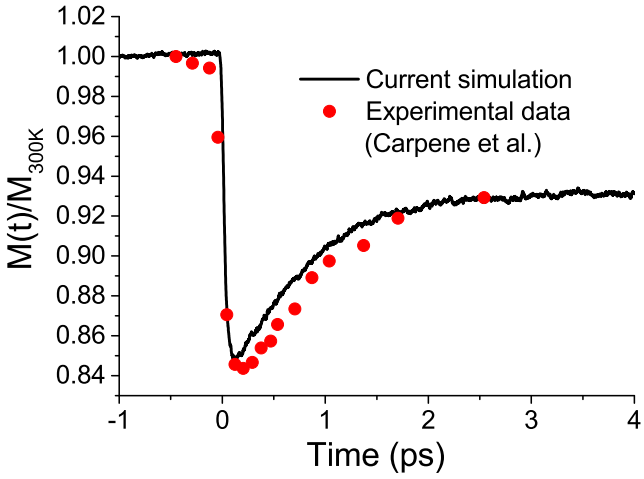

Figure 6: Dynamics of demagnetization and recovery corresponding to the case where a laser pulse occurs at 0ps. Experimental data [47] are shown for comparison.

to be added to file *prototype_CPU.cpp*. In a GPU device code, the prototype needs to be added to file *prototype_GPU.cu*. We have already included function **laser_demagnetization_CPU()** in *spilady.cpp*, and added it to *prototype_CPU.cpp*.

We use the final output configuration file from the previous example as an input configuration file for this simulation, and switch on option #**define eltemp**. We now deal with a spin-lattice-electron system where the total energy is conserved apart from the extra amount associated with the laser pulse. The laser pulse is introduced at around 0ps. Fig. 5 shows a sharp peak of T_e at around 0ps, associated with the energy pulse. T_s' responds swiftly to changes in T_e because they are strongly coupled. T_s responds slower because T_s describes the relaxation of collective spin excitations. It takes slightly longer for the transverse fluctuations to absorb energy and achieve maximum entropy. The T_l response is the slowest, since the coupling between the electrons and the lattice is comparatively weak.

Fig. 6 shows magnetization plotted as a function of time, on the same timescale as in Fig. 5. Magnetization is normalized to its value at 300K. It drops significantly when the laser pulse is introduced. Then it gradually recovers. Experimental data points taken from Ref. [47] are also shown for comparison. They match each other fairly well. Magnetization does not

recover back to its original value because the temperature of the system increases from 300K to about 450K.

6. Conclusion

In this paper we describe the main features of a spin-lattice dynamics simulation program, SPILADY verison 1.0. A user of SPILADY can perform molecular dynamics, spin dynamics, spin-lattice dynamics, and spin-lattice-electron dynamics simulations, including where necessary longitudinal spin fluctuations. The program is written in OpenMP C/C++ and CUDA C/C++. It can be run in parallel on CPUs, or on an Nvidia GPU. This paper describes the underlying theoretical concepts, algorithms and applications of the program, as well as its limitations.

Acknowledgement

This work has been carried out within the framework of the EUROFusion Consortium and has received funding from the Euratom research and training programme 2014-2018 under grant agreement No. 633053 and from the RCUK Energy Programme [grant No. EP/I501045]. To obtain further information on the data and models underlying the paper please contact PublicationsManager@ccfe.ac.uk. The views and opinions expressed herein do not necessarily reflect those of the European Commission.

[1] H. C. Herper, E. Hoffmann, and P. Entel, Phys. Rev. B 60 (1999) 3839

[2] M. Friák, M. Sob, and V. Vitek, Phys. Rev. B 63 (2001) 052405

[3] D. G. Pettifor, *Bonding and Structure of Molecules and Solids* (Clarendon Press, Oxford, 1996)

[4] D.J. Dever, J. Appl. Phys. 43 (1972) 3293

[5] H. Hasegawa, M. W. Finnis, and D. G. Pettifor, J. Phys. F: Met. Phys. 17 (1987) 2049

[6] H. Hasegawa and D. G. Pettifor, Phys. Rev. Lett. 50 (1983) 130

[7] M. Y. Lavrentiev, D. Nguyen-Manh, and S. L. Dudarev. Phys. Rev. B 81 (2010) 184202

[8] D. Nguyen-Manh, A. Horsfield, and S. L. Dudarev, Phys. Rev. B 73 (2006) 020101

[9] P. M. Derlet, D. Nguyen-Manh, and S. L. Dudarev, Phys. Rev. B 76 (2007) 054107

[10] F. Körmann, A. A. H. Breidi, S. L. Dudarev, N. Dupin, G. Ghosh, T. Hickel, P. Korzhavyi, J. A. Muñoz, I. Ohnuma, Physica Status Solidi B 251 (2014) 53

[11] S. L. Dudarev, R. Bullough, and P. M. Derlet, Phys. Rev. Lett. 100 (2008) 135503

[12] T. L. Gilbert, IEEE Trans. Magn. 40 (2004) 3443

[13] W. F. Brown Jr., Phys. Rev. 130 (1963) 1677

[14] S. Chandrasekhar, Rev. Mod. Phys. 15 (1943) 1

[15] R. Kubo, Rep. Prog. Phys. 29 (1966) 255

[16] S. L. Dudarev and P. M. Derlet, J. Phys.: Condens. Matter 17 (2005) 7097; P. M. Derlet and S. L. Dudarev, Prog. Mater. Sci. 52 (2007) 299

[17] S. L. Dudarev and P. M. Derlet, J. Computer-Aided Mater. Des. 14 (2007) 129

[18] G. J. Ackland, J. Nucl. Mater. 351 (2006) 20

[19] P.-W. Ma, W.C. Liu, C.H. Woo, S.L. Dudarev, J. Appl. Phys. 101 (2007) 073908

[20] I. P. Omelyan, I. M. Mryglod, and R. Folk, Phys. Rev. Lett. 86 (2001) 898; I. P. Omelyan, I. M. Mryglod, and R. Folk, Phys. Rev. E 64 (2001) 016105; I. P. Omelyan, I. M. Mryglod, and R. Folk, Phys. Rev. E 66 (2002) 026701

[21] S. H. Tsai, M. Krech, and D. P. Landau, Brazil. J. Phys. 34 (2004) 382; S. H. Tsai, H. K. Lee, and D. P. Landau, Am. J. Phys. 73 (2005) 615

[22] N. Hatano and M. Suzuki, Lect. Notes Phys. 679 (2005) 37

[23] Pui-Wai Ma, C. H. Woo, and S. L. Dudarev, AIP Conf. Proc. 999 (2008) 134

[24] Pui-Wai Ma, C. H. Woo, and S. L. Dudarev, Phys. Rev. B 78 (2008) 024434

[25] Pui-Wai Ma and C. H. Woo, Phys. Rev. E 79 (2009) 046703

[26] Pui-Wai Ma, S. L. Dudarev, S. L. Semenov, and C. H. Woo, Phys. Rev. E 82 (2010) 031111

[27] Pui-Wai Ma, and S. L. Dudarev, Phys. Rev. B 83 (2011) 134418

[28] Pui-Wai Ma, S. L. Dudarev, and C. H. Woo, J. Appl. Phys. 111 (2012) 07D114

[29] Pui-Wai Ma, S. L. Dudarev, and C. H. Woo, Phys. Rev. B, 85 (2012) 184301

[30] Pui-Wai Ma and S. L. Dudarev, Phys. Rev. B, 86 (2012) 054416

[31] Pui-Wai Ma, C. H. Woo, and S. L. Dudarev, Phil. Mag. 89 (2009) 2921

[32] Haohua Wen, Pui-Wai Ma, and C. H. Woo 440 (2013) 428

[33] Haohua Wen and C. H. Woo 455 (2014) 31

[34] Pui-Wai Ma and S. L. Dudarev 90 (2014) 024425

[35] D. Beaujouan, P. Thibaudeau, and C. Barreteau, Phys. Rev. B 86 (2012) 174409

[36] P. Thibaudeau and D. Beaujouan, Physica A: Stat. Mech. Appl. 391 (2012) 1963

[37] D. Perera, D. P. Landau, D. M. Nicholson, G. M. Stocks, M. Eisenbach, J. Q. Yin, and G. Brown, J. Appl. Phys. 115 (2014) 17D124

[38] T. Shimada, K. Ouchi, I. Ikeda, Y. Ishii and T. Kitamura, Comp. Mater. Sci. 97 (2015) 216

[39] A. K. Zhuravlev and Yu. N. Gornostyrev, J. Exp. Theo. Phys. 119 (2014) 503

[40] M. van Schilfgaarde and V. P. Antropov, J. Appl. Phys. 85 (1999) 4827

[41] S.-T. Pi, R. Nanguneri, and S. Y. Savrasov, Phys. Rev. Lett. 112 (2014) 077203

[42] D. M. Duffy and A. M. Rutherford, J. Phys.: Condens. Matter 19 (2007) 016207; J. Nucl. Mater. 386 (2009) 19

[43] A. E. Sand, S. L. Dudarev, and K. Nordlund, Euro. Phys. Lett. 103 (2013) 46003

[44] M. P. Allen and D. J. Tildesley, "Computer Simulation of Liquids" (Clarendon Press, Oxford, 1987).

[45] J.-L. Barret and D. Rodney, J. Stat. Phys. 144 (2011) 679

[46] http://inac.cea.fr/L_Sim/V_Sim/

[47] E. Carpene, E. Mancini, C. Dallera, M. Brenna, E. Puppin and S. De Silvestri, Phys. Rev. B 78 (2008) 174422