SPILADY

A Spin-Lattice Dynamics Simulation Program

Version 1.0

User Manual

info@spilady.ccfe.ac.uk
http://spilady.ccfe.ac.uk

Pui-Wai (Leo) Ma[1], S. L. Dudarev[1], C. H. Woo[2]

[1]Culham Centre for Fusion Energy,
United Kingdom Atomic Energy Authority,
Abingdon, Oxfordshire OX14 3DB, United Kingdom

[2]Department of Physics and Materials Science,
The City University of Hong Kong,
Hong Kong SAR, China

October 16, 2015

Contents

Acknowledgements

SPILADY (pronounced as [spileidi]) is a computer program written at the Culham Centre for Fusion Energy, United Kingdom Atomic Energy Authority, Oxfordshire OX14 3DB, United Kingdom, from March 2014 to July 2015. It is a spin-lattice dynamics code intended to serve as an introductory computer simulation tool for undergraduate students, scientists, researchers, and others familiar with molecular dynamics. The approach that we followed during the development of SPILADY involved introducing a number of new software architecture and data structure concepts, that differ from those used in other implementations of spin-lattice dynamics, including our own earlier numerical realizations of the method.

An early version of a spin-lattice dynamics simulation program was written by Pui-Wai (Leo) Ma, in collaboration with C. H. Woo and S. L. Dudarev, as a part of his PhD project at the Hong Kong Polytechnic University. The program was made available to students and staff at the Hong Kong Polytechnic University and the City University of Hong Kong.

The development and release of this SPILADY program has been made possible through the support provided by a EUROfusion Enabling Research grant WP14-ER-01/CCFE-02 "Dynamic Evolution of Non-Equilibrium High-Temperature Thermodynamic Properties of Magnetic Metals". The work was 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 number EP/I501045]. The views and opinions expressed herein do not necessarily reflect those of the European Commission.

The authors would like to thank Mark Gilbert and Daniel Mason for stimulating discussions. The authors are grateful to Wayne Arter and Martin O'Brien for reading the manual and making many helpful comments, and to Rob Akers and Jonathan Hollocombe for their interest in the SPILADY project and expert advice on various points.

Leo Ma would like to thank his wife, Ritty Tsui, for all her support in tears and smiles.

License

SPILADY – A Spin-Lattice Dynamics Simulation Program

Version 1.0

Apache License

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice

that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim

in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

You must give any other recipients of the Work or Derivative Works a copy of this License; and You must cause any modified files to carry prominent notices stating that You changed the files; and You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except

as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification

within third-party archives.

Technical notes

Date: 16 Oct 2015
Responsible person: Pui-Wai (Leo) Ma
1. Bug fixed in quantum_noise_CPU.cpp and quantum_noise_GPU.cu
2. Some functions in potential_CPU.cpp and potential_GPU.cu are rewritten for clarity.

1

Introduction

"We know what we are, but know not what we may be."
– William Shakespeare (1564 - 1616)

1.1 Magnetism and Spin-lattice dynamics

Magnetic materials are used in a broad variety of applications. These applications range from the use of magnetic degrees of freedom in storage, recovery, transmission, manipulation, and processing of information, including quantum computing, to technological issues related to materials for fusion plasma confinement arising from the magnetism of ferritic-martensitic steels.

Magnetic excitations influence the stability of crystal phases of magnetic materials, where particularly striking examples are iron, iron-based alloys and steels. The body-centered cubic crystal structure of pure iron is stable at room temperature because iron is magnetic [1, 2]. Other elements in the same group of the Periodic Table, non-magnetic ruthenium (Ru) and osmium (Os), both have hexagonal crystal structure [3].

At high temperatures close to 900°C, magnetic fluctuations make the bcc phase of iron unstable against its transformation into the fcc phase, and this gives rise to softening of elastic constant $c' = (c_{11} - c_{12})/2$ [4, 5]. This elastic constant describes the volume-conserving linear response of the bcc phase of Fe to an infinitesimal deformation that transforms a cube into a parallelepiped on a square base. If the height to side aspect ratio of the parallelepiped equals $\sqrt{2}$, the deformed bcc structure is identical to fcc (with a different lattice parameter), this is known as the Bain transformation [6]. This bcc-fcc transition in pure iron occurs at 1043K. At even higher temperatures the fcc phase transforms back into the bcc phase [7, 8], and at an even higher temperature iron melts. It turns out that the lower-temperature bcc-fcc transition at 1043K is of greater significance to applications than

melting, as the bcc-fcc softening, stimulated by magnetic fluctuations [8], is a microscopic phenomenon that provides the physical foundation for the steel manufacturing industry.

A self-interstitial atom defect in ferromagnetic bcc iron adopts the ⟨110⟩ dumbbell configuration, whereas in all the non-magnetic bcc transition metals like tungsten, molybdenum, or vanadium, a self-interstitial defect adopts a geometrically different ⟨111⟩ crowdion configuration [9, 10]. These examples show that there are remarkable cases where the phase stability [11] and even the mechanical properties of transition metals and alloys are strongly influenced by magnetism [12]. It is therefore desirable to develop an atomistic simulation technique that treats *both* the kinematic motion of atoms *and* precession of magnetic moments resulting from thermal excitation. Conventional molecular dynamics only treats the kinematic motion of atoms and is unable to provide information about the time evolution of magnetic configurations in a material.

The Landau-Lifshitz and Gilbert (LLG) equations [13] describe the classical dynamics of spins. Both equations are purely dissipative. If a spin system evolves according to the LLG equations, it always relaxes into the lowest energy ground-state configuration. This is not consistent with the observed behaviour of thermal spin excitations, where at a finite temperature the spins continuously fluctuate and do not settle into a static lowest energy ground state. To account for this effect, Brown [14] proposed a stochastic integration method involving the use of the fluctuation-dissipation theorem (FDT) [15, 16]. Stochastic evolution equations including dissipation *and* fluctuation terms make it possible to describe finite-temperature fluctuating spin dynamics in thermal equilibrium.

There have been several attempts to incorporate magnetic effects in the embedded atom method (EAM) many-body potential formalism for molecular dynamics (MD) simulations. Dudarev and Derlet [17, 18] and Ackland [19] proposed ways of including magnetism in the semi-empirical potential framework, to enable carrying out MD simulations taking magnetism into account. However, neither approach addressed the directional aspect of magnetic/spin configurations, and the treatments were focused primarily on predicting the magnitudes of atomic magnetic moments [17, 18, 19].

Developing a dynamic simulation model for spin waves, lattice vibrations, and their interaction at a finite temperature proved to be a challenge. Omelyan *et al.* [20] and Tsai *et al.* [21] discovered a way of treating spin and lattice dynamics within a unified approach, still involving highly simplified model assumptions. They noted the advantage of using the Suzuki-Trotter decomposition (STD) algorithm [22] for the treatment of non-commuting coordinate and spin evolution operators. The spin degrees of freedom in this approach were treated purely classically and the question about the role of quantum effects in spin dynamics was not addressed. Also, simulations were limited to relatively small systems since no parallel algorithm for integrating

the spin dynamics equations was available.

We first performed spin-lattice dynamics simulations in 2007 [23, 24], having formulated a model with parameters matching *ab initio* data for bcc ferromagnetic iron. The problem of parallelizing spin-lattice dynamics simulations was addressed and solved, through the use of the Suzuki-Trotter decomposition, in 2009 [25]. We then posed and addressed questions about how to compute the temperature of a dynamically evolving spin ensemble [26] and how to efficiently integrate the stochastic spin equations of motion [27]. We have now included the electronic degrees of freedom in spin-lattice dynamics [28, 29], and developed a model for longitudinal spin fluctuations [30].

Spin-lattice dynamics simulations have been applied to a variety of systems, such as iron thin films [31], the treatment of self-diffusion in iron [32, 33], and dynamic magneto-caloric effects [34]. Significant advances have been made by our colleagues at CEA, France [35, 36] and at Oak Ridge National Laboratory, USA [37].

To broaden the scope of applications of spin-lattice dynamics simulations, we have now fully revised and rewritten the simulation code, producing a new, hopefully more user-friendly, spin-lattice dynamics simulation program. We hope that this will help advance this challenging field of direct time-dependent simulation of magnetic materials, and will enable scientists not yet familiar with the technique, to learn and apply it.

1.2 Papers

Papers, giving a detailed description of the spin-lattice dynamics simulation methodology and explaining the fundamentals of the method, are listed below. A SPILADY user is advised to refer to the material given in the papers when performing simulations, as the papers provide answers to questions that we are not able to cover in detail in this manual.

1. The basics of the spin-lattice dynamics algorithm:

   - "Spin-Lattice Dynamics Simulations of Ferromagnetic Iron", Pui-Wai Ma, C. H. Woo and S. L. Dudarev, AIP Conf. Proc. **999**, 134 (2008)
   - "Large-scale simulation of the spin-lattice dynamics in ferromagnetic iron", Pui-Wai Ma, C. H. Woo, and S. L. Dudarev, Phys. Rev. B **78**, 024434 (2008)

2. Spin-lattice-electron dynamics / molecular-electron dynamics:

   - "Spin-lattice dynamics model for magnon-phonon-electron heat transfer on a million atom scale" Pui-Wai Ma, S. L. Dudarev and C. H. Woo, J. Appl. Phys. **111**, 07D114 (2012)

- "Spin-lattice-electron dynamics simulations of magnetic materials", Pui-Wai Ma, S. L. Dudarev, and C. H. Woo, Phys. Rev. B, **85**, 184301 (2012)

3. Local collective motion of atoms and electrons:

- "Spin-lattice-electron dynamics simulations of magnetic materials", Pui-Wai Ma, S. L. Dudarev, and C. H. Woo, Phys. Rev. B, **85**, 184301 (2012)

4. Fluctuation-dissipation Langevin spin dynamics:

- "Langevin spin dynamics", Pui-Wai Ma and S. L. Dudarev, Phys. Rev. B **83**, 134418 (2011)

5. Longitudinal magnetic fluctuations:

- "Longitudinal magnetic fluctuations in Langevin spin dynamics", Pui-Wai Ma and S. L. Dudarev, Phys. Rev B **86**, 054416 (2012)

6. How to compute the kinematic temperature of atoms and temperature of fluctuating magnetic moments:

- "Temperature for a dynamic spin ensemble", Pui-Wai Ma, S. L. Dudarev, A. A. Semenov, and C. H. Woo, Phys. Rev. E **82**, 031111 (2010)

7. A parallel algorithm for spin-lattice dynamics simulations:

- "Parallel algorithm for spin and spin-lattice dynamics simulations", Pui-Wai Ma and C. H. Woo, Phys. Rev. E **79**, 046703 (2009)

8. A forthcoming paper describing, probably in somewhat lesser detail than the manual, this computer code (SPILADY):

- "SPILADY: A Parallel GPU and CPU Code for Spin-Lattice Dynamics", Pui-Wai Ma and S. L. Dudarev, (in preparation)

2

Theory

"We cannot solve our problems with the same thinking we used when we created them."

– Albert Einstein (1879 - 1955)

We start with a brief summary of equations of motion used in molecular dynamics (MD), spin dynamics (SD) and spin-lattice dynamics (SLD). Then we show how to treat longitudinal fluctuations of magnetic moments, link atomistic simulations to the heat transfer equation for the free electrons, and introduce the notion of local collective motion of atoms and electrons, which is required to make sure that on average electrons move together with the atoms. These methods are described, in greater detail, in the papers mentioned in section 1.2 above.

2.1 Molecular Dynamics

We start from a conventional MD simulation. A classical Hamiltonian describing moving interacting atoms has the form

$$\mathcal{H} = \sum_{i=1}^{N} \frac{\mathbf{p}_i^2}{2m_i} + U(\mathbf{R}_1, \mathbf{R}_2, ..., \mathbf{R}_N), \tag{2.1}$$

where $\mathbf{p}_i$ is the momentum of atom i, m_i is its mass, $\mathbf{R}_i = (x_i, y_i, z_i)$ is the position vector of the atom, and U is the potential energy of interaction between the atoms, which varies depending on where atoms are in real space. The classical mechanics Hamilton equations of motion, which describe how the positions and momenta of each of the N atoms evolve as functions of time, are

$$\frac{d\mathbf{R}_i}{dt} = \frac{\partial \mathcal{H}}{\partial \mathbf{p}_i} = \frac{\mathbf{p}_i}{m}, \tag{2.2}$$

$$\frac{d\mathbf{p}_i}{dt} = -\frac{\partial \mathcal{H}}{\partial \mathbf{R}_i} = -\frac{\partial U}{\partial \mathbf{R}_i}. \tag{2.3}$$

The velocity of an atom $\mathbf{v}_i = \mathbf{p}_i/m_i$ and the force acting on the atom $\mathbf{F}_i = -\partial U/\partial \mathbf{R}_i$ satisfy the Newton equations of motion, which follow from the above Hamilton equations, namely

$$\frac{d\mathbf{R}_i}{dt} = \mathbf{v}_i, \tag{2.4}$$

$$\frac{d\mathbf{v}_i}{dt} = \frac{1}{m_i}\mathbf{F}_i. \tag{2.5}$$

A molecular dynamics simulation algorithm integrates these Newton equations. How this numerical integration is performed in practice is explained on page 78 of the treatise "Computer Simulation of Liquids" by M. P. Allen and D. J. Tildesley [38].

While molecular dynamics equations are entirely deterministic, the motion of atoms is also affected by random forces associated, for example, with electronic excitations. Such stochastic effects can be modelled assuming that in addition to the regular deterministic part, the force acting on an atom also has a stochastic component. As as result, the deterministic Newtonian molecular dynamics is replaced by stochastic Langevin molecular dynamics, where the equations of motion have the form [15, 16]:

$$\frac{d\mathbf{R}_i}{dt} = \frac{\mathbf{p}_i}{m}, \tag{2.6}$$

$$\frac{d\mathbf{p}_i}{dt} = -\frac{\partial U}{\partial \mathbf{R}_i} - \gamma_l \frac{\mathbf{p}_i}{m} + \mathbf{f}_i. \tag{2.7}$$

Here 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 $\langle f_{i\alpha}(t) f_{j\beta}(t') \rangle = \mu_l \delta_{ij} \delta_{\alpha\beta} \delta(t - t')$. Using the fluctuation-dissipation theorem, it is possible to prove that in thermal equilibrium the magnitude of fluctuations of random forces and the damping coefficient are related to each other through the equation $\mu_l = 2\gamma_l k_B T$. If in Langevin dynamics we take T as the temperature of an external thermostat, we can use Langevin equations to thermalize the system through stochastic forces acting on atoms.

In practical simulations, fluctuating forces are modelled using a random number generator (RNG). In SPILADY, we use a thread-safe RNG written following George Marsaglia's MWC (multiply with carry) algorithm and Box-Muller transformation for simulations running on CPU. In GPU simulations, a CUDA built-in RNG following the XORWOW algorithm and Box-Muller transformation is used.

2.2 Spin Dynamics

For an arbitrary spin Hamiltonian $\mathcal{H}$, the equation governing the evolution of a classical spin vector has the form [30]:

$$\frac{d\mathbf{S}_i}{dt} = \frac{1}{\hbar} \left[\mathbf{S}_i \times \mathbf{H}_i \right], \tag{2.8}$$

where the effective magnetic field acting on the spin is $\mathbf{H}_i = -\partial\mathcal{H}/\partial\mathbf{S}_i$. This equation retains its form even if we treat $\mathbf{S}_i$ as an operator, i.e. if we replace vector $\mathbf{S}_i$ in Eq. (2.8) by $\hat{\mathbf{S}}_i$. This quantum-mechanical generalization is valid if the effective field $\mathbf{H}_i$ is a vector. The approximation applies for example if the dynamics of each individual spin in the material is treated in the mean-field approximation [30].

If the magnitudes of all the magnetic moments remain constant in the course of their dynamic evolution, i.e. if each magnetic moment vector remains on the surface of a sphere of constant radius, the Langevin equations of motion can be written as [14, 27]:

$$\frac{d\mathbf{S}_i}{dt} = \frac{1}{\hbar}\left[\mathbf{S}_i \times (\mathbf{H}_i + \mathbf{h}_i) - \gamma_s \mathbf{S}_i \times (\mathbf{S}_i \times \mathbf{H}_i)\right], \tag{2.9}$$

where γ_s is a damping parameter and $\mathbf{h}_i(t)$ is a δ-correlated fluctuating field, satisfying conditions $\langle \mathbf{h}_i(t)\rangle = 0$ 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 for the moments reads $\mu_s = 2\hbar\gamma_s k_B T$.

While the equations of motion for the spins (2.8) are entirely general, in practical applications the Hamiltonian describing interaction between the spins is often taken in the Heisenberg form,

$$\mathcal{H} = -\frac{1}{2}\sum_{i,j} J_{ij}\mathbf{S}_i \cdot \mathbf{S}_j, \tag{2.10}$$

where J_{ij} is the so-called exchange coupling parameter. The magnitude of J_{ij} defines the strength of interactions between neighbouring spins. The magnitude of this interaction determines the Curie or the Neél temperatures T_C or T_N. Values of exchange parameters J_{ij} can be deduced from *ab initio* calculations [41, 42].

In thermodynamic equilibrium, the temperature of a dynamically evolving spin system can be computed using the following equation [26]:

$$T = \frac{1}{2k_B}\frac{\sum_i |\mathbf{S}_i \times \mathbf{H}_i|^2}{\sum_i \mathbf{S}_i \cdot \mathbf{H}_i}. \tag{2.11}$$

This equation is valid only if the spin system is in thermal equilibrium. During transients, the time-dependent temperature values calculated using this equation should only be treated as estimates. A remarkable property of the above formula for the spin temperature is that the temperature computed using this formula is not positive definite. Negative spin temperatures can be realized for configurations where moments $\mathbf{S}_i$ and effective fields $\mathbf{H}_i$ are anti-parallel [26].

2.3 Longitudinal spin fluctuations

In an itinerant magnetic system, both directions *and* magnitudes of magnetic moments fluctuate. To describe directional and longitudinal fluctuations of

magnetic moments on an equal footing, we introduce fully three-dimensional spin Langevin equations of motion [30]

$$\frac{d\mathbf{S}_i}{dt} = \frac{1}{\hbar}\left[\mathbf{S}_i \times \mathbf{H}_i\right] - \gamma_s \frac{\partial \mathcal{H}}{\partial \mathbf{S}_i} + \boldsymbol{\xi}_i \tag{2.12}$$

$$= \frac{1}{\hbar}\left[\mathbf{S}_i \times \mathbf{H}_i\right] + \gamma_s \mathbf{H}_i + \boldsymbol{\xi}_i. \tag{2.13}$$

In the above equations γ_s is a damping parameter and $\boldsymbol{\xi}_i(t)$ is a δ-correlated random noise. The noise function $\boldsymbol{\xi}_i(t)$ satisfies 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 equations (2.13) reads $\mu_s = 2\gamma_s k_B T$. The three-dimensional Langevin equations (2.13) can be mapped onto the two-dimensional Langevin equations (2.9) if the movement of each spin vector is constrained to the surface of a sphere [30].

The spin temperature at equilibrium can now be calculated using the following formula [30]:

$$T = \frac{\sum_{i,\alpha} \left(\partial \mathcal{H}/\partial S_{i\alpha}\right)^2}{k_B \sum_{i,\alpha} \partial^2 \mathcal{H}/\partial S_{i\alpha}^2}. \tag{2.14}$$

For transient non-equilibrium configurations the above formula only provides an estimate for the actual temperature of the system. The formula was derived using a Hamiltonian where spins were treated as 3-dimensional vectors. A representative example is given by the Heisenberg-Landau Hamiltonian [8, 30, 34]:

$$\mathcal{H} = \mathcal{H}_H + \mathcal{H}_L, \tag{2.15}$$

where

$$\mathcal{H}_H = -\frac{1}{2}\sum_{i,j} J_{ij}\mathbf{S}_i \cdot \mathbf{S}_j \tag{2.16}$$

$$\mathcal{H}_L = \sum_i A_i S_i^2 + B_i S_i^4 + C_i S_i^6 + D_i S_i^8 \tag{2.17}$$

The Landau part $\mathcal{H}_L$ of the Heisenberg-Landau Hamiltonian is a polynomial in $S_i^2 = \mathbf{S}_i^2$. The Landau coefficients A_i, B_i, C_i and D_i can be derived from *ab initio* calculations [30, 34].

2.4 Spin-Lattice Dynamics

To simulate the motion of atoms and dynamic evolution of spins, we need a Hamiltonian that treats both the spins and the atoms, as well as interaction between them - in other words, the interaction that couples atomic coordinates and directions of atomic spins. Hence the Hamiltonian is expected to

have a general form like $\mathcal{H} = \mathcal{H}(\{\mathbf{R}\}, \{\mathbf{p}\}, \{\mathbf{S}\})$. In the context of the SPILADY program, we use two different Hamiltonians, depending on whether or not we intend to treat longitudinal magnetic fluctuations.

A Hamiltonian that does *not* include longitudinal fluctuations of magnetic moments has the form

$$\mathcal{H} = \mathcal{H}_{latt} + \mathcal{H}_{spin} + \mathcal{H}_{corr}, \tag{2.18}$$

where

$$\mathcal{H}_{latt} = \sum_i \frac{\mathbf{p_i}^2}{2m} + U(\mathbf{R}), \tag{2.19}$$

$$\mathcal{H}_{spin} = -\frac{1}{2}\sum_{i,j} J_{ij}(\mathbf{R})\mathbf{S}_i \cdot \mathbf{S}_j, \tag{2.20}$$

$$\mathcal{H}_{corr} = \frac{1}{2}\sum_{i,j} J_{ij}(\mathbf{R})|\mathbf{S}_i||\mathbf{S}_j|. \tag{2.21}$$

Here the coordinate-dependent exchange coupling function is taken as a pairwise function of interatomic distances, i.e. $J_{ij}(\mathbf{R}) = J_{ij}(R_{ij})$. Equation (2.21) is a correction term for the Heisenberg spin Hamiltonian. Since the magnitude of each spin vector $|\mathbf{S}_i|$ is a constant, the correction term is a function of atomic coordinates only. We introduce such a correction term only for computational convenience, as this makes it possible to use the existing parameterizations of many-body potentials, for example the DD05 iron potential [17], and modify them by simply adding a Heisenberg-like coordinate-dependent part. The correction that we adopt assumes that the ground state of the system is ferromagnetic.

The spin-lattice dynamics equations of motion are

$$\frac{d\mathbf{R}_i}{dt} = \frac{\mathbf{p}_i}{m}, \tag{2.22}$$

$$\frac{d\mathbf{p}_i}{dt} = -\frac{\partial \mathcal{H}}{\partial \mathbf{R}_i} = -\frac{\partial U}{\partial \mathbf{R}_i} - \sum_j \frac{\partial J_{ij}}{\partial \mathbf{R}_i}(\mathbf{S}_i \cdot \mathbf{S}_j - |\mathbf{S}_i||\mathbf{S}_j|), \tag{2.23}$$

$$\frac{d\mathbf{S}_i}{dt} = \frac{1}{\hbar}\left[\mathbf{S}_i \times \mathbf{H}_i\right], \tag{2.24}$$

where the effective field $\mathbf{H}_i$ is $\mathbf{H}_i = -\partial\mathcal{H}/\partial\mathbf{S}_i = \sum_i J_{ij}\mathbf{S}_j$. From the above equations, we see that the lattice and spin degrees of freedom are coupled via a coordinate-dependent exchange function $J_{ij}(\mathbf{R})$. Langevin thermostat can be added to the spin-lattice dynamics equations by including appropriate fluctuation and dissipation terms in the right-hand side of the above equations.

To treat longitudinal magnetic fluctuations, we use the Heisenberg-Landau Hamiltonian

$$\mathcal{H}_{spin} = \mathcal{H}_H + \mathcal{H}_L, \tag{2.25}$$

where

$$\mathcal{H}_H = -\frac{1}{2}\sum_{i,j} J_{ij}(\mathbf{R})\mathbf{S}_i \cdot \mathbf{S}_j \tag{2.26}$$

$$\mathcal{H}_L = \sum_i A_i(\mathbf{R})S_i^2 + B_i(\mathbf{R})S_i^4 + C_i(\mathbf{R})S_i^6 + D_i(\mathbf{R})S_i^8 \tag{2.27}$$

The Landau Hamiltonian is now also coordinate-dependent. In SPILADY we assume that $A_i(\mathbf{R}) = A_i(\rho_i)$, $B_i(\mathbf{R}) = B_i(\rho_i)$ etc., where ρ_i is the local effective electron density defined as the argument of the embedding function of the embedded atom method (EAM) potential. Equations of motion for the kinematic momenta of atoms now have the form

$$\begin{aligned}\frac{d\mathbf{p}_i}{dt} &= -\frac{\partial U}{\partial \mathbf{R}_i} - \frac{\partial J_{ij}}{\partial \mathbf{R}_i}(\mathbf{S}_i \cdot \mathbf{S}_j - |\mathbf{S}_i||\mathbf{S}_j|), \\ &+ \frac{\partial A_i}{\partial \mathbf{R}_i}S_i^2 + \frac{\partial B_i}{\partial \mathbf{R}_i}S_i^4 + \frac{\partial C_i}{\partial \mathbf{R}_i}S_i^6 + \frac{\partial D_i}{\partial \mathbf{R}_i}S_i^8.\end{aligned} \tag{2.28}$$

If we adopt the Langevin thermostat as mentioned in connection with equation (2.13), we need to exercise caution when defining the effective field. Since the magnitudes of magnetic moments are no longer constant, the correction term, i.e. equation (2.21), needs to be included in the derivation of the effective field, namely

$$\mathbf{H}_i = \sum_j J_{ij}\left(\mathbf{S}_j - \frac{\mathbf{S}_i S_j}{S_i}\right) - (2A_i + 4B_i S_i^2 + 6C_i S_i^4 + 8D_i S_i^6)\mathbf{S}_i. \tag{2.29}$$

Concluding this section, we note that the mathematical expression for spin temperature (2.14) depends sensitively on the form of the spin Hamiltonian. If longitudinal magnetic fluctuations are included in the dynamics, SPILADY computes spin temperature using the Heisenberg-Landau Hamiltonian (2.25).

2.5 Langevin Model and Heat Transfer by Electrons

In a metal, conduction electrons provide the dominant contribution to thermal conductivity. Heat lost to electrons has a significant effect on the motion of atoms, particularly in the treatment of high-energy events like collision cascades [43]. If we treat the Langevin thermostat as a physical representation of electrons, we can relate the Langevin force model to the heat transfer equation [44, 29]. In this way we generalize MD, spin dynamics and spin-lattice dynamics, and include electronic degrees of freedom in a unified discrete atomistic model. Full analysis of such a model and a derivation of the relevant equations are given in [29].

Within the molecular dynamics (MD) framework, a self-consistent "atomic Langevin - electronic heat transfer" model is described by the following system of coupled equations

$$\frac{d\mathbf{R}_i}{dt} = \frac{\mathbf{p}_i}{m}, \tag{2.30}$$

$$\frac{d\mathbf{p}_i}{dt} = -\frac{\partial \mathcal{H}}{\partial \mathbf{R}_i} - \gamma_l \frac{\mathbf{p}_i}{m} + \mathbf{f}_i, \tag{2.31}$$

$$C_e \frac{dT_e}{dt} = \nabla \cdot (\kappa_e \nabla T_e) + G_{el}(T_l - T_e), \tag{2.32}$$

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

$$G_{el} = \frac{3k_B\gamma_l}{m\Omega}, \tag{2.33}$$

where Ω is the volume of an atom. T_l is the lattice temperature evaluated from the kinetic energy of atoms within a given linked-cell. For a definition of what a linked-cell is, see pages 149–152 of Ref. [38].

A model for spins on a rigid lattice, where spins interact with conduction electrons, is defined by the following system of equations

$$\frac{d\mathbf{S}_i}{dt} = \frac{1}{\hbar}\left[\mathbf{S}_i \times (\mathbf{H}_i + \mathbf{h}_i) - \gamma_s \mathbf{S}_i \times (\mathbf{S}_i \times \mathbf{H}_i)\right], \tag{2.34}$$

$$C_e \frac{dT_e}{dt} = \nabla \cdot (\kappa_e \nabla T_e) + G_{es}(T_s - T_e), \tag{2.35}$$

where the coefficient of heat transfer between spins and conduction electrons is given by

$$G_{es} = \frac{2k_B\gamma_s}{\hbar\Omega}\langle \mathbf{S}_i \cdot \mathbf{H}_i \rangle. \tag{2.36}$$

The above model neglects longitudinal magnetic fluctuations. The ensemble average value $\langle \mathbf{S}_i \cdot \mathbf{H}_i \rangle$ and the spin temperature T_s are taken as instantaneous values calculated using time-dependent vectors $\{\mathbf{S}_i(t)\}$ within each linked cell.

If longitudinal magnetic fluctuations are taken into account, the model acquires the form [30],

$$\frac{d\mathbf{S}_i}{dt} = \frac{1}{\hbar}\left[\mathbf{S}_i \times \mathbf{H}_i\right] + \gamma_s \mathbf{H}_i + \boldsymbol{\xi}_i, \tag{2.37}$$

$$C_e \frac{dT_e}{dt} = \nabla \cdot (\kappa_e \nabla T_e) + G_{es}(T_s - T_e). \tag{2.38}$$

The coefficient of heat transfer between spins and electrons is

$$G_{es} = \frac{k_B\gamma_s}{\Omega}\left\langle \sum_\alpha \frac{\partial^2 \mathcal{H}}{\partial S_{i\alpha}^2} \right\rangle, \tag{2.39}$$

where index α refers to the three degrees of freedom of a spin vector.

In the full spin-lattice dynamics (SLD) model, the heat transfer equation has the form

$$C_e \frac{dT_e}{dt} = \nabla \cdot (\kappa_e \nabla T_e) + G_{el}(T_l - T_e) + G_{es}(T_s - T_e). \tag{2.40}$$

Equations of motion for the spins and the atoms, and definitions of various constants and variables, remain unchanged.

2.6 Local Collective Motion of Atoms

If there is a group of atoms moving uniformly in the same direction, the kinetic energy and temperature of atoms *in the moving centre of mass frame* can be evaluated as

$$T_l = \frac{2}{3Nk_B} \sum_i \frac{(\mathbf{p}_i - \mathbf{P})^2}{2m}, \tag{2.41}$$

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

The heat transfer equation assumes that the local temperature of the lattice is defined in the centre of mass frame, and that the kinetic energy associated with the uniform motion of atoms is not included in the definition of temperature. In practical terms, this implies that in the treatment of heat transfer from atoms to electrons and *vice versa*, only the motion of atoms *relative* to conduction electrons should be taken into account when evaluating the rate of heat transfer.

The way to address the issue is to treat lattice-electron energy transfer using locally moving frames. If we take lattice-electron interactions as local properties of the material, we write [29]:

$$\frac{d\mathbf{R}_i}{dt} = \frac{\mathbf{p}_i}{m}, \tag{2.42}$$

$$\frac{d\mathbf{p}_i}{dt} = -\frac{\partial \mathcal{H}}{\partial \mathbf{R}_i} - \frac{\gamma_l}{m}(\mathbf{p}_i - \mathbf{p}_A) + (\mathbf{f}_i - \mathbf{f}_A), \tag{2.43}$$

$$C_e \frac{dT_e}{dt} = \nabla \cdot (\kappa_e \nabla T_e) + G_{el}(T_l - T_e), \tag{2.44}$$

where

$$\mathbf{p}_A = 1/N_A \sum_{i \in A} \mathbf{p}_i, \tag{2.45}$$

$$\mathbf{f}_A = 1/N_A \sum_{i \in A} \mathbf{f}_i, \tag{2.46}$$

$$G_{el} = \frac{3k_B(N_A - 1)\gamma_l}{mV_A}, \tag{2.47}$$

where A refers to a local domain of the system, N_A is the number of atoms in this domain, and V_A is the volume of A. In practice, we identify domains with linked cells.

2.7 Suzuki-Trotter decomposition

This section is intended for those who are interested in the computer algorithm underlying spin-lattice dynamics simulations. If one has no intention of changing the integration algorithm, one may skip reading this section.

The Suzuki-Trotter decomposition (STD) is a way of performing practical calculations with non-commuting evolution operators. It is a symplectic integration method that conserves the volume of phase space during dynamic evolution of the system. As a result, numerical errors are unbiased, and this results in better conservation of quantities that are expected to be conserved under the relevant dynamics, for example the total energy. Hatano *et al.* [22] wrote a comprehensive and easy-to-read paper on this topic. We briefly describe the method here and discuss its applications.

Suppose that there is a partial differential equation describing the time evolution of variable x:

$$\frac{dx}{dt} = (A + B)x, \tag{2.48}$$

where A and B are arbitrary operators. The solution of the above equation is

$$x(t + \Delta t) = e^{(A+B)\Delta t}x(t) \tag{2.49}$$

where Δt is the time step. In general, operators A and B do not commute. The STD is an approximate method, where up to the second order in the time step the evolution operator can be written as:

$$e^{(A+B)\Delta t} = e^{A(\Delta t/2)}e^{B\Delta t}e^{A(\Delta t/2)} + O\left(\Delta t^3\right). \tag{2.50}$$

The error of the method is of the order of Δt^3, and the advantage offered by the method is that it allows integration of non-commuting evolution operators separately, one by one. The order in which the operators are taken has no effect on the magnitude of the error term. Higher order STD forms can also be derived, they are given in Ref. [22]. In our program, we only use the second order STD formula (2.50).

If we take MD as an example, we write

$$\frac{d}{dt}\begin{pmatrix} \mathbf{R} \\ \mathbf{p} \end{pmatrix} = \begin{pmatrix} 0 & \delta\mathcal{R} \\ \delta\mathcal{P} & 0 \end{pmatrix}\begin{pmatrix} \mathbf{R} \\ \mathbf{p} \end{pmatrix} \tag{2.51}$$

where $\mathbf{R} = \{\mathbf{R}_i\}$ and $\mathbf{p} = \{\mathbf{p}_i\}$. Operators $\delta\mathcal{R}$ and $\delta\mathcal{P}$ refer to the following operations:

$$\frac{d\mathbf{R}}{dt} = \delta\mathcal{R}\mathbf{p} = \frac{\mathbf{p}}{m}, \tag{2.52}$$

$$\frac{d\mathbf{p}}{dt} = \delta\mathcal{P}\mathbf{R} = -\frac{\partial U}{\partial \mathbf{R}}. \tag{2.53}$$

Using the second order STD, we write

$$e^{(\delta\mathcal{P}+\delta\mathcal{R})\Delta t} \approx e^{\delta\mathcal{P}(\Delta t/2)} e^{\delta\mathcal{R}\Delta t} e^{\delta\mathcal{P}(\Delta t/2)}, \tag{2.54}$$

where

$$e^{\delta\mathcal{P}\Delta t} \equiv I + \delta\mathcal{P}\Delta t \tag{2.55}$$

$$e^{\delta\mathcal{R}\Delta t} \equiv I + \delta\mathcal{R}\Delta t. \tag{2.56}$$

It can be shown [22] that this is exactly equivalent to the well-known velocity Verlet algorithm of molecular dynamics simulations [38].

A sum of several operators can also be decomposed in a similar way. For example,

$$e^{(A+B+C+D)\Delta t} \approx e^{A(\Delta t/2)} e^{B(\Delta t/2)} e^{C(\Delta t/2)} e^{D\Delta t} e^{C(\Delta t/2)} e^{B(\Delta t/2)} e^{A(\Delta t/2)} \tag{2.57}$$

These operators may, for example, refer to the dissipation and fluctuation terms in the Langevin equations, or to operations on spins $\delta\mathcal{S}$, or to electron temperature $\delta\mathcal{T}_e$.

Fig. 2.1 shows the flow chart of the SPILADY program, using spin-lattice-electron dynamics as an example. The core part of the integrator uses the second order STD formula, where $A = \delta\mathcal{T}_e$, $B = \delta\mathcal{P}$, $C = \delta\mathcal{S}$ and $D = \delta\mathcal{R}$ in the equation above. The sequence in which the operators are taken is immaterial. Our aim here is to minimize the number of times forces are evaluated in a single cycle.

Parallel programming in the case of MD is straightforward. Since the time derivatives of variables $\mathbf{p}$ and $\mathbf{R}$ depend on their conjugate variables, if we apply STD, an operation on a given atom does not affect other atoms over a time step. Therefore, a parallel calculation can be performed by parallelizing individual calculations for different atoms.

In contrast, in the case of spin dynamics, the change of value of a spin variable $\mathbf{S_i}$ affects the effective fields acting on other spins. In principle, one is therefore forced to decompose an operation on the entire spin system into operators acting on spins one by one, such that

$$e^{\delta\mathcal{S}\Delta t} = e^{\delta\mathcal{S}_1(\Delta t/2)} e^{\delta\mathcal{S}_2(\Delta t/2)} ... e^{\delta\mathcal{S}_N\Delta t} ... e^{\delta\mathcal{S}_2(\Delta t/2)} e^{\delta\mathcal{S}_1(\Delta t/2)} \tag{2.58}$$

where $\delta\mathcal{S}_i$ refers to an operation performed on spin i. It appears as if it is impossible to perform calculations in parallel in this situation.

Fortunately, any conventional interatomic potential has a cut-off distance. This makes it possible to parallelize calculations at the level of linked-cells. Details of the method can be found in Ref. [25]. The idea is to segregate the linked-cells, which are defined in almost any MD program, into non-interacting groups. Each linked-cell within a group must have no interaction with other linked-cells that are members of the same

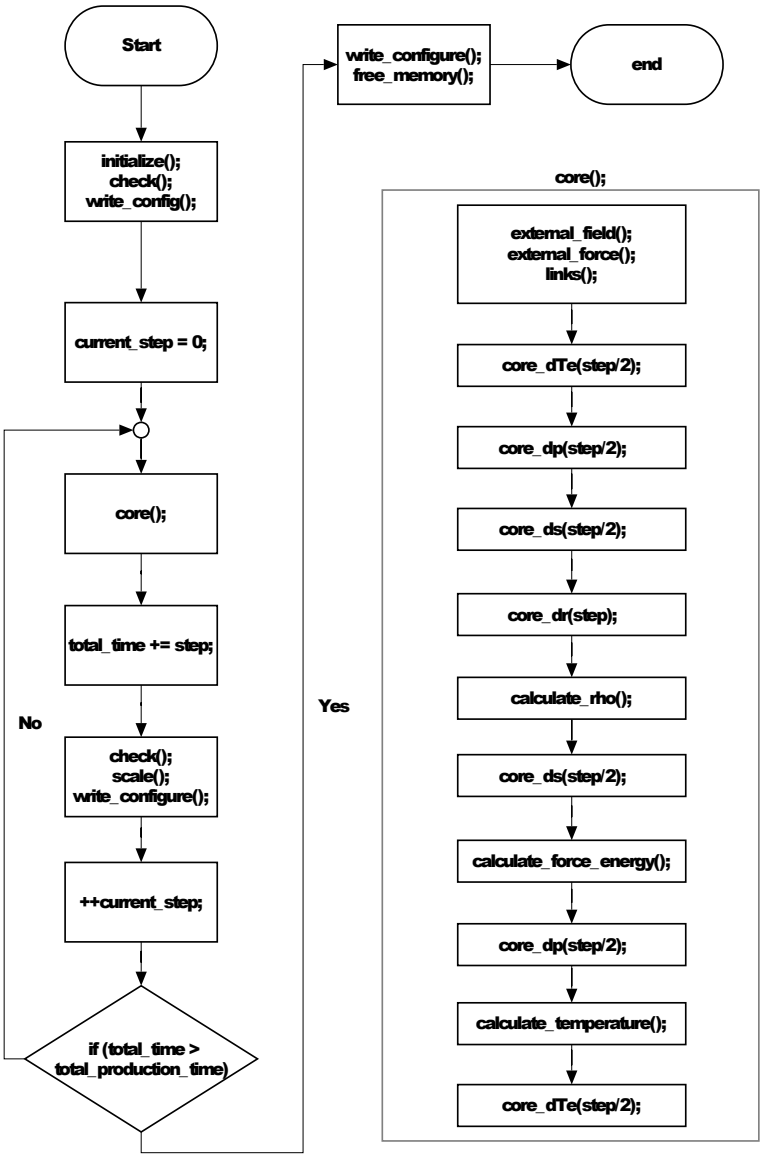

Figure 2.1: SPILADY flow chart

group. Parallel computation can then be performed by parallelizing with respect to members of the same group. The evolution of spins within a linked-cell still needs to be treated using the STD. Calculations involving different groups should also obey the STD formula. A suitable choice of STD at the level of spins can reduce the time required for calculating the effective fields by a factor of two without introducing any error greater than the third power of the time step.

3

Controls

"Do not dwell in the past, do not dream of the future, concentrate the mind on the present moment."

– Buddha (563 BC - 483 BC)

The section explains how the program works. Although the mathematical aspects of spin dynamics and spin-lattice dynamics are more involved than those of a conventional molecular dynamics (MD) simulations, a common rationale underlies the algorithms implemented in SPILADY. Hence running SPILADY successfully is of comparable difficulty to making MD simulations.

SPILADY is written using the OpenMP C/C++ and CUDA C/C++ computer languages. The program and all the examples were tested 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 the following compilers: gcc version 4.4.7, icc version 12.0.0, and nvcc version 5.5. SPILADY was also tested on computers with Nvidia GeForce GTX 480, GTX 680, GTX Titan, Tesla K40c GPU cards and Tesla M2090 GPU module.

SPILADY was written using standard syntax and not optimized for a particular type of hardware architecture or operating system. In principle, it should work on most common system. No external libraries are required.

3.1 Files

One downloads the program as a compressed file *spilady1.0.tar.gz*, and decompresses it using one of the standard commands, for example

```
$ tar -zxvf spilady1.0.tar.gz
```

All the files and working examples should decompress into a folder *spilady1.0*. It is advisable to work from inside the folder by typing

```
$ cd spilady1.0
```

To run the program, one needs to have several files available. They are *make.sh*, *control.h* and *variables.in*. If one intends to perform an MD or an SLD simulation, i.e. simulations involving atomic coordinates, a file defining an interatomic potential is also required, for example *DD05.cpp*. If one intends to perform a spin dynamics (SD) simulation on a rigid lattice or a spin-lattice dynamics (SLD) simulation, a Heisenberg-Landau function file *JijFe.cpp* is also needed. If one uses the Langevin electron thermostat, the electron heat capacity file *heatcapacity_CPU.cpp* (and the file *heatcapacity_GPU.cu* if GPU is used) is also required.

3.2 Compiling and running the program

First, the program needs to be compiled. In principle, SPILADY can be compiled using only a single command line. To simplify it further, a script file is provided. The file that contains all the commands needed to compile the program is called *make.sh*. Now one needs to convert it into an executable file. In Unix or Linux environments, this can be accomplished by typing the following command (which changes the nature of the file from a text file into an executable file):

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

One may need to alter the content of the script file depending on the *in situ* requirements or computer settings. If one is using CPUs, it is mandatory to use an OpenMP capable compiler. One can use, for example, *g++* or *icc* to compile the program. For Nvidia GPU, the default compiler is *nvcc*. Also, one needs to use appropriate options to compile *all* the files in the current folder, for example

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

Option *-fopenmp* allows the compiler to recognize OpenMP directives in the code. If one chooses the *icc* compiler, the corresponding option is *-openmp*. Second, *-o spilady* gives the resulting executable file a name *spilady*. Third, *-DCPU* defines the word *CPU* in the computer code, similarly *-DOMP* and *-DMD* define words OMP and MD. Option *-DXXX* has the same effect as adding directive *#define XXX* to the content of all the files.

Depending on whether one intends to run an MD, or an SD, or an SLD simulation, the last option in the compilation command should be either

-DMD (for molecular dynamics), or *-DSDH* for spin dynamics in the Heisenberg Hamiltonian approximation, and so on, as described in the table 3.1 below. The table gives all the currently available compilation options, apart from the *-DSLDNC* option that is still under development.

Table 3.1: Compilation 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
-DSLDNC	SLD	Noncollinear Many-body

If the program is compiled with the intention of using a Nvidia GPU, the compilation procedure is broadly similar. We advise use of a CUDA version no earlier than 5.5, for which the program has been tested. To compile the program, execute the command

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

In the above example, option *-arch=sm_35* instructs the compiler to produce an executable file for devices with hardware architecture of version 3.5. The program has been successfully tested with option *-arch=sm_20*, i.e. version 2.0, too. There is no guarantee that compilation for hardware version earlier than 2.0 is going to be successful. If one intends to compile the program in a way that the executable can be run in various hardware architectures, please refer to the CUDA program manual.

Option *-rdc=true* is important, where *-rdc* is the short form of *--relocatable-device-code*. It allows device codes to be placed in different files to be recognized by other codes in other files. Although this is a default feature of CPU compilers, this option must be used for the GPU compiler *nvcc*. Other compilation options are similar to those used for CPU compilations.

One should note that all the files in the current folder are going to be compiled. If there are files that are not required for a given round of compilation, it is the best to move them elsewhere out of the current folder before executing the compilation command. In particular, this has implications for the choice of interatomic potential. Only one single file describing a potential, say *DD05.cpp*, can be retained in the folder during compilation, and it is this specific potential, described in the potential file, that is going to be used by the compiled SPILADY program for carrying out simulations.

SPILADY can be executed by typing

```
$ ./spilady
```

Running the program requires no extra argument.

3.3 Potential and the specific heat files

Up to two files are required to describe interactions between atoms and/or the spins. To perform an MD or SLD simulation, a file describing interaction between the atoms, in the form of an EAM potential - for example *DD05.cpp* - is required. If one would like to perform an SD or SLD simulation where directions (and magnitudes) of magnetic moments are expected to evolve, one requires a file defining the part of the potential that depends on the directions of magnetic moments, in the form of a Heisenberg(-Landau) function file, for example *JijFe.cpp*.

In an EAM potential file, the parametrization assumes that the potential energy of interaction between the atoms has a standard EAM functional form

$$U(\mathbf{R}) = \sum_i F_i(\rho_i) + \frac{1}{2}\sum_{i,j} V_{ij}(R_{ij}), \tag{3.1}$$

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

One needs to input the exact functional forms of F_i, f_{ij} and V_{ij}, and their parameters into the potential file. One should create a potential file with functions **bigf_gen**, **smallf_gen** and **pair_gen**. The Dudarev-Derlet 2005 [17] potential is coded in file *DD05.cpp* as an example. The program converts these functions into tables at the beginning of a simulation run for the actual calculation of energies and forces.

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

$$\begin{aligned} \mathcal{H}_{spin} &= -\frac{1}{2}\sum_{i,j} J_{ij}(R_{ij})\mathbf{S}_i \cdot \mathbf{S}_j \\ &+ \sum_i \left(A_i(\rho_i)S_i^2 + B_i(\rho_i)S_i^4 + C_i(\rho_i)S_i^6 + D_i(\rho_i)S_i^8 \right). \end{aligned} \tag{3.2}$$

One needs to input exact functional forms of J_{ij}, A_i, B_i, C_i and D_i and numerical values of parameters describing these variables into the potential file. One needs to create a potential file with functions **Jij_gen**, **LandauA_gen**, **LandauB_gen**, **LandauC_gen** and **LandauD_gen**. Parameters for iron are given in *JijFe.cpp* as an example. The program converts these functions into tables at the beginning of a simulation run for the actual calculations of energies and forces. If one works with the Heisenberg Hamiltonian only, the Landau coefficients can be ignored.

One needs to be careful about one point, if option #**define magmom** in file *control.h* is switched on, the Heisenberg-Landau Hamiltonian is assumed

to be defined in terms of magnetic moments, not spins, namely

$$\begin{aligned}\mathcal{H}_{spin} &= -\frac{1}{2}\sum_{i,j} J_{ij}(R_{ij})\mathbf{M}_i \cdot \mathbf{M}_j \\ &+ \sum_i \left(A_i(\rho_i)M_i^2 + B_i(\rho_i)M_i^4 + C_i(\rho_i)M_i^6 + D_i(\rho_i)M_i^8\right). \quad (3.3)\end{aligned}$$

All the parameters for the Hamiltonian in this case need to be re-fitted bearing in mind that it is now the magnetic moment of an atom rather than atomic spin that represents the evolving dynamic variable.

If one intends to use the Langevin electron thermostat, one needs to edit files *heatcapacity_CPU.cpp* (and the file *heatcapacity_GPU.cu* if GPU is used). One needs to edit functions **Ce**, **Te_to_Ee** and **Ee_to_Te** for the CPU case. In the GPU case, one also needs to edit functions **Ce_d**, **Te_to_Ee_d** and **Ee_to_Te_d**. SPILADY assumes that the heat capacity (specific heat) per atom has the following functional form $C_e = a\tanh(bT_e)$ [44], where T_e is the temperature of the electrons. One needs to establish a relationship between the electron temperature T_e and electron energy per atom E_e, and input analytical expressions for both quantities into the file.

3.4 Triclinic simulation box

One can create a triclinic simulation box according to one's needs. Even if a simulation box initially is a rectangle, its shape can change according internal stresses developing in the system. If one would like to start with a triclinic box, a configuration input file either in the SPILADY configuration output format or in a V_sim compatible format should be supplied. Their formats are going to be discussed below.

The box vectors are shown in Fig. 3.1, and are defined as

$$\mathbf{a} = \{d_{xx}, 0, 0\} \quad (3.4)$$
$$\mathbf{b} = \{d_{yx}, d_{yy}, 0\} \quad (3.5)$$
$$\mathbf{c} = \{d_{zx}, d_{zy}, d_{zz}\} \quad (3.6)$$

where **a** is in red, **b** is in blue and **c** is in green. Their components are given in Å units. For all the cases, the simulation box is set up assuming periodic boundary conditions. The program chooses the number of linked cells using the *initial* box vectors. Therefore, one needs to be careful if the box is going to be deformed severely during a simulation. In the case of compression, one may need to increase the input variable *min_length_link_cell* in the file *variables.in*. This will be discussed later.

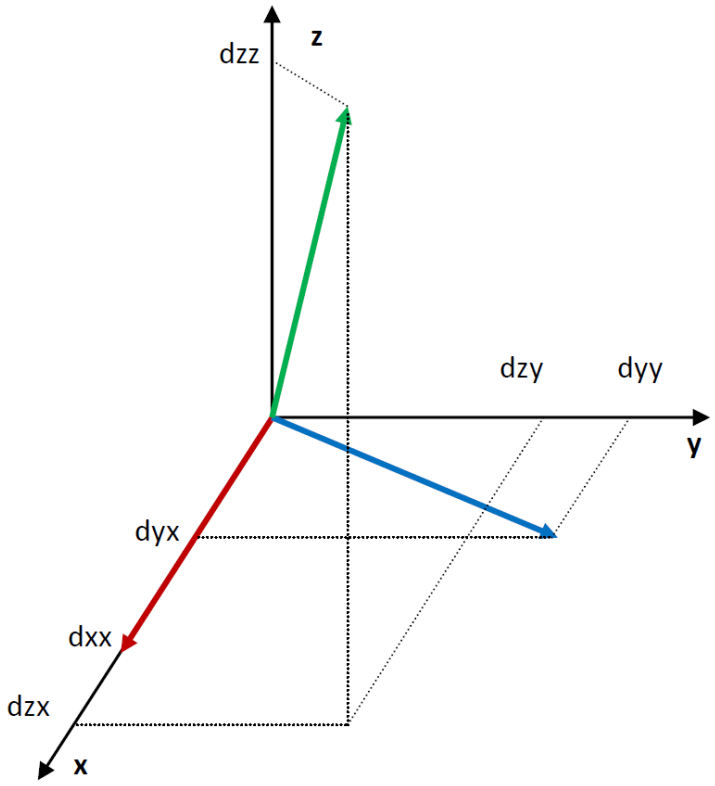

Figure 3.1: Triclinic simulation box vectors

3.5 Options

One can switch additional options on or off in the file *control.h* by uncommenting or commenting the lines containing them. If one changes any option in *control.h*, the program must be recompiled.

When one starts a new simulation, the simulation box vectors, atomic positions, momenta and spin vectors, need to be supplied or initialized. Therefore, one must switch on one of the options *readconf*, *readvsim*, *bcc100*, *bcc111*, *fcc100* or *hcp0001*. They allow the program to read in, or to create, atomic configurations. If one is creating a new simulation box, the momenta and spin vectors should also be initialized.

Words in **bold** refer to variables in file *variables.in*.

3.5.1 #define bcc100

By switching on this option one is constructing a body-centred cubic structure. The simulation box vectors are in the [100], [010] and [001] directions. A unit cell has 2 atoms. The size of a unit cell is $a \times a \times a$, where a is the lattice constant.

3.5.2 #define bcc111

By switching on this option one is constructing a body-centred cubic structure. The simulation box vectors are in the [111], [$\bar{2}$11] and [0$\bar{1}$1] directions. A unit cell has 12 atoms. The size of the unit cell is $\sqrt{3}a \times \sqrt{6}a \times \sqrt{2}a$, where a is the lattice constant.

3.5.3 #define changestep

If this option is switched on, the size of the time step changes depending on the maximum displacement of atoms, i.e. **displace_limit**, and/or the maximum change of the angle of rotation of magnetic moments/spins, i.e. **phi_limit**, over a single time step. This procedure is known as the adaptive time step integration.

If this option is switched off, the program uses a fixed time step value **step**.

3.5.4 #define eltemp

If this option is switched on, one must also switch on the Langevin thermostats for the lattice and/or the spins. Heat transfer in the electron subsystem is also taken into account. Electron temperature is defined for each linked-cell. Electronic temperature in each linked-cell changes as a result of energy transfer. Electron temperature can be spatially inhomogeneous. The strength of fluctuating forces in the Langevin equation depends on the

electron temperature in each linked-cell.

If this option is switched off, Langevin forces are defined by the input temperature, i.e. **temperature**, which is a constant.

3.5.5 #define extfield

If this option is switched on, it imposes constant external magnetic fields on every atomic spin or magnetic moments throughout the simulation. One needs to supply an extra input file *extfield.in* to specify the magnitude of fields on each atomic site. The format of the file should look like

N			
1	H_{1x}^{ext}	H_{1y}^{ext}	H_{1z}^{ext}
2	H_{2x}^{ext}	H_{2y}^{ext}	H_{2z}^{ext}
3	H_{3x}^{ext}	H_{3y}^{ext}	H_{3z}^{ext}
$\vdots$			
N	H_{Nx}^{ext}	H_{Ny}^{ext}	H_{Nz}^{ext}

where N is the number of atoms in the entire simulation cell. External field vectors are given in Tesla units. Sample files can be found in the examples given below.

If this option is switched off, there is no external magnetic field.

3.5.6 #define extforce

If this option is switched on, it imposes constant external forces on every atom in the simulation box. One needs to supply an extra input file *extforce.in* to specify the force on each atom. The format of the file should look like

N			
1	F_{1x}^{ext}	F_{1y}^{ext}	F_{1z}^{ext}
2	F_{2x}^{ext}	F_{2y}^{ext}	F_{2z}^{ext}
3	F_{3x}^{ext}	F_{3y}^{ext}	F_{3z}^{ext}
$\vdots$			
N	F_{Nx}^{ext}	F_{Ny}^{ext}	F_{Nz}^{ext}

where N is the number of atoms. External force vectors are given in eV/Å units.

If this option is switched off, there are no external forces.

3.5.7 #define fcc100

By switching on this option one is constructing a face-centred cubic structure. The simulation box vectors are in [100], [010] and [001] directions. A unit cell has 4 atoms. The size of a unit cell is $a \times a \times a$, where a is the lattice constant.

3.5.8 #define hcp0001

By switching on this option one is creating a hexagonal close-packed structure. The simulation box vectors are in $[\bar{1}2\bar{1}0]$, $[\bar{1}010]$ and $[0001]$ directions. A unit cell has 2 atoms. The size of a unit cell is $a \times \sqrt{3/2}a \times c$, where a and c are the a and c hcp lattice constants.

3.5.9 #define initmomentum

If this option is switched on, it generates two different results depending on option *lattlang*. If *lattlang* is switched on, all the momenta are initialized to zero. If *lattlang* is switch off, SPILADY generates momenta randomly and rescales the kinetic temperature to **initTl**. One should note that temperature defined using the kinetic energy criterion can then vary as a function of time because the kinetic energy of atoms is converted into potential energy and *vice versa*.

If this option is switched off, either *readconf* or *readvsim* should be switched on. One should also supply input files containing information about the momenta of atoms.

3.5.10 #define initspin

If this option is switched on, spins will be initialized as a collinear ferromagnetic configuration. One should also supply the value of magnetic moments i.e. **mag_mom**, in the Bohr magneton μ_B units.

If this option is switched off, either *readconf* or *readvsim* should be switched on. One should also supply input files containing information about the spins or magnetic moments.

3.5.11 #define inittime

If this option is switched on, the internal variable that represents the total simulation time (not the total CPU time) will be initialized and set equal to **start_time**. It can have any value. If *readconf* and *readvsim* are switched off, this option needs to be switch on, too. There is no reverse requirement.

If this option is switched off, either *readconf* or *readvsim* should be switched on. One should also supply input files containing information about the initial simulation time.

3.5.12 #define lattlang

If this option is switched on, the equations of motion for atoms with Langevin thermostat are going to be used.

If this option is switched off, the equations of motion for atoms with no thermostat are going to be used.

3.5.13 #define localcolmot

This is a sub-option to *eltemp*. It only applies if *eltemp* is switched on.

If this option is switched on, the local collective motion of atoms is treated using locally moving frames. Theoretical details can be found in Ref. [29].

If this option is switched off, all the atoms are treated in a single static frame.

3.5.14 #define localvol

If this option is switched on, the volume of each atom will be calculated using the formula

$$\Omega_i = \frac{4}{3}\pi r_\Omega^3, \tag{3.7}$$

where

$$r_\Omega = \frac{\sum_j 1/r_{ij}}{2\sum_j 1/r_{ij}^2}. \tag{3.8}$$

Then, the value of each Ω_i is re-scaled to match the total volume of the system. One needs to supply the cut-off radius **rcut_vol** to determine over what distance the neighbourhood should be included in the summation. We suggest choosing the cut-off radius just larger than the 2nd nearest neighbour distance in the bcc case, and just larger than the 1st nearest neighbour distance in the fcc case.

If this option is switched off, the volume of each atom equals the total volume of the simulation box divided by the number of atoms.

3.5.15 #define magmom

If this option is switched on, atomic magnetic moments are used as input and output variables. The magnetic part of the Hamiltonian has the form

$$\mathcal{H}_{spin} = -\frac{1}{2}\sum_{i,j} J_{ij}\mathbf{M}_i \cdot \mathbf{M}_j + \sum_i A_i M_i^2 + B_i M_i^4 + \cdots . \tag{3.9}$$

If this option is switched off, atomic spins are used as input and output variables. The spin Hamiltonian is then assumed to have the form

$$\mathcal{H}_{spin} = -\frac{1}{2}\sum_{i,j} J_{ij}\mathbf{S}_i \cdot \mathbf{S}_j + \sum_i A_i S_i^2 + B_i S_i^4 + \cdots . \tag{3.10}$$

One needs to pay attention to the definitions here, and be consistent about whether the magnetic moments or spins are used as variables. The values of J_{ij}, A_i, B_i, C_i and D_i in file e.g. *JijFe.cpp* should be chosen according to this definition. A remark for an interested developer is that all the internal calculations in SPILADY are performed and coded in terms of atomic spins, not magnetic moments. Magnetic moments and atomic spins are related through the equation $\mathbf{M}_i = -g\mu_B\mathbf{S}_i$.

3.5.16 #define PRESSURE

If this option is switched on, the size of the simulation box changes isotropically according to the difference between internal pressure and the desired **pressure** through a Berendsen type barostat. One needs to exercise care in the case where simulation box undergoes contraction. Since mapping of linked-cells is performed for the initial atomic configuration, if the linear size of a linked-cell is smaller than the cut-off radius of the potential, some atoms may be missed in a calculation of energies and forces. The value of **min_length_link_cell** should be increased to ensure the correct choice of the size of linked-cells. If *PRESSURE* is switched on, *STRESS* should be switched off.

If this option is switched off, the size of the simulation box remains constant, unless *STRESS* is switched on.

3.5.17 #define quantumnoise

This is a sub-option to *lattlang*. It only works if *lattlang* is switched on.

If this option is switched on, the classical Langevin thermostat for the lattice part will be replaced by the Quantum Langevin thermostat developed by J.-L. Barret and D. Rodney [39].

If this option is switched off, the classical Langevin thermostat is used.

This option has not been tested with *eltemp*. A proper theoretical framework incorporating the quantum thermostat into a two or three temperature model has not yet been established. One should also note that if this option is switched on, the program will consume a few times (about 4 to 5 times) more memory than a conventional case. Extra care should be taken when using a GPU, where memory limitations may present a significant issue.

3.5.18 #define readconf

If this option is switched on, the program will read in configuration file **in_config** containing information about the number of atoms, initial time, simulation box vectors, atomic positions, momenta, forces, magnetic moments/spins etc. A description of the format of the file is given in section *In/Output files*.

3.5.19 #define readTe

This is a sub-option to *eltemp*. It only works if *eltemp* is switched on.

If this option is switched on, the program reads the linked cell temperatures file, which contains values of electronic temperature in each linked-cell. Details of the file format are given in section *In/Output files*.

If this option is switched off, the initial electronic temperature in each linked-cell equals the input value of **temperature**.

3.5.20 #define readvsim

By switching on this option, the program will be prompted to read in the V_sim compatible file **in_vsim_atom** containing information about the number of atoms, the initial time, simulation box vectors, atomic positions and momenta, and file **in_vsim_spin** that contains information about spin or magnetic moment vectors. A description of the format of the files is given in section *In/Output files*. V_sim is free software developed by CEA, France. It can visualize atoms and spins, and generate multiple figures for producing animated movie files. It can be obtained from http://inac.cea.fr/L_Sim/V_Sim/ .

3.5.21 #define renormalizeEnergy

This is a sub-option to *eltemp*. It only works if *eltemp* is switched on.

If this option is switched on, after every time step, the total energy of the

whole system is renormalized numerically to the initial total energy. This is to conserve the total energy despite the effect of numerical errors caused by the differential equation describing the evolution of the specific heat of electrons.

If this option is switched off, there is no renormalization of energy.

3.5.22 #define runstep

If this option is switched on, the program runs until the total number of steps reaches **no_of_production_steps**.

If this option is switched off, the program runs until the total simulation time reaches **total_production_time**.

3.5.23 #define spinlang

If this option is switched on, equations of motion for spins with the Langevin thermostat applied are going to be used.

If this option is switched off, equations of motion for spins with no thermostat will be used.

3.5.24 #define STRESS

If this option is switched on, the simulation box vectors will change according to the difference between internal stresses and the desired stresses **stress_xx**,**stress_yx**, **stress_yy**, **stress_zx**, **stress_zy** and **stress_zz** using a Berendsen type barostat. One needs to exercise care if the simulation box undergoes contraction. Since the mapping of linked-cells is performed for the initial atomic configuration, if the size of a linked-cell is smaller than the cut off radius of the potential, some atoms may be missed in the calculation of energies and forces. The value of **min_length_link_cell** should be increased to ensure the correct choice of the size of linked-cells. If *STRESS* is switched on, *PRESSURE* should be switched off.

If this option is switched off, the size of the simulation box remains constant, unless *PRESSURE* is switched on.

3.5.25 #define writevsim

If this option is switched on, the program will output V_sim compatible files for visualization purposes. V_sim is free software developed by CEA, France. It can visualize atoms and spins, and generate multiple figures for producing animated movie files. It can be obtained from http://inac.cea.fr

/L_Sim/V_Sim/.

If this option is switched off, there will be no output files for visualization.

3.6 Variables

Apart from the control options, one should also supply values of various variables as input data in file *variables.in*. Some of the variables have default values. If there is no input for a particular variable, either a default value will be used, or the program will not run properly, potentially generating meaningless numerical values. If file *variables.in* is amended but no other files are changed, there is no need to recompile the program. Once the program starts running, file *variables.all* will be generated, showing the values of all the variables, both the input and default values, used in the current run. It has the same format as *variables.in*. One could and should check the contents of the *variables.all* file to ensure that input for SPILADY program is provided properly. File *variables.in* should be formatted as

a_lattice 2.87
atmass 55.847
element Fe
mag_mom 2.2
out_body test
total_production_time 1e-9
...

where a_lattice is a variable representing the value of the lattice constant in Å, and atmass is the atomic mass in atomic mass units etc. It is not important in which order they are listed in the file.

3.6.1 a_lattice

This is the lattice constant of bcc or fcc lattices. It is also the a lattice constant of the hcp lattice, given in Å units. Its default value is 2.83 for bcc lattice, 3.5 for fcc lattice, and 3.629 for the hcp lattice case. It has no default value if #**define readconf** or #**define readvsim** are switched on.

3.6.2 atmass

This is the atomic mass in the units of u (atomic mass unit). Its default value is 55.847, which is the average, over isotopes, atomic mass of naturally occurring iron.

3.6.3 baro_damping_time

If one of the options #**define STRESS** or #**define PRESSURE** is switched on, this variable gives the relaxation time for stress or pressure. The value is given in the units od seconds. Its default value is 1×10^{-13}.

3.6.4 c_lattice

This is the c lattice constant of hcp lattice, given in Å units. Its default value is 5.796.

3.6.5 current_device

This variable is used if simulations are performed using Nvidia GPU, where the device index of a particular GPU card should be supplied. Its default value is 0.

3.6.6 displace_limit

If #**define changestep** is switched on, this is the maximum amplitude of atomic displacement over one time step. If there is even a single atom in the entire cell that is displaced by more than this maximum displacement limiting value, the size of the time step **step** is decreased by the factor of 0.8. Otherwise, **step** increases by a factor of 1.05, provided that a similar criterion is satisfied for the spin subsystem. The value of the maximum displacement limit is given in Å units. Its default value is 0.01.

3.6.7 element

If a new atomic configuration is initialized, one should supply the element symbol of the material for which atomistic simulations are performed, such as Fe, W, C, Au. The element name can contain one or two characters. It has no default value.

3.6.8 gamma_L_over_mass

If #**define lattlang** is switched on, it defines the damping constant divided by the atomic mass, i.e. γ_l/m, for the lattice Langevin thermostat. It has the dimensionality of seconds^{-1}. Its default value is $6 \times 10^{11}\mathrm{s}^{-1}$. Some of the values for various materials can be obtained experimentally assuming that the damping forces in the Langevin equation are equivalent to the effect of electronic stopping power. For example, such experimental data can be found in Ref. [40]. Electron stopping can also be calculated using SRIM, but the values derived from SRIM only describe the high atomic velocity limit.

3.6.9 gamma_S_H

If #**define spinlang** is switched on, it defines the damping constant γ_s for the spin Langevin thermostat associated with the Heisenberg Hamiltonian. This quantity is dimensionless. Its default value is 8×10^{-3}.

3.6.10 gamma_S_HL

If #**define spinlang** is switched on, it defines the damping constant γ_s for the spin Langevin thermostat associated with the Heisenberg-Landau Hamiltonian. The units are eV^{-1} s^{-1}. Its default value is 5.88×10^{13}.

3.6.11 in_config

If #**define readconf** is switched on, this is the name of the input file required for supplying information about the number of atoms, initial simulation time, simulation box vectors, atomic positions, momenta, forces, spin or magnetic moment vectors etc. The format of the file is described in section *In/Output files*. It has no default value.

3.6.12 in_eltemp

If #**define readTe** and #**define eltemp** are switched on, this is the name of the input file supplying information about the electronic temperatures in each linked-cell. The format of the file is described in section *In/Output files*. It has no default value.

3.6.13 in_vsim_atom

If #**define readvsim** is switched on, this is the name of the input file supplying information about the number of atoms, initial simulation time, simulation box vectors, atomic positions and momenta. The format of the file is discussed in section *In/Output files*. It has no default value.

3.6.14 in_vsim_spin

If #**define readvsim** is switched on, this is the name of the input file supplying information about spin or magnetic moment vectors. The format of the file is discussed in section *In/Output files*. It has no default value.

3.6.15 initTl

If #**define initmomentum** is switched on and #**define lattlang** is switched off, the program generates atomic momenta randomly and rescales kinetic temperature to the pre-defined temperature value. Its default value is **temperature**.

3.6.16 interval_of_config_out

This is the interval between time steps when all the information about the configuration of the system at a particular instant, such as atomic positions, atomic kinetic momenta, forces, magnetic moments, is printed into an output file. The format of the configuration file is discussed in section *In/Output files*. Its default value is 1000.

3.6.17 interval_of_print_out

This is the interval between time steps when some general information, such as the average energies, temperatures, stresses, pressure, magnetic moment, is sent to an output file. The format of file is described in section *In/Output files*. Its default value is 1.

3.6.18 interval_of_scale_pressure

If #**define PRESSURE** is switched on, this is the interval between time steps when the size of the simulation box is re-scaled isotropically according to pre-defined pressure using a Berendsen type barostat. Its default value is 1.

3.6.19 interval_of_scale_stress

If #**define STRESS** is switched on, this is the interval between time steps when rescaling of the simulation box vectors is performed according to pre-defined stresses using a Berendsen type barostat. Its default value is 1.

3.6.20 interval_of_vsim

If #**define writevsim** is switched on, this is the interval between time steps when information about positions, momenta and spins is sent to generate V_sim compatible output files for visualization purposes. The format of the files is described in section *In/Output files*. Its default value is 1000.

3.6.21 mag_mom

If #**define initspin** is switched on, this is the initial value of the magnitude of magnetic moment of all the atoms, given in μ_B units. Its default value is 2.2.

3.6.22 no_of_production_steps

If #**define runstep** is switched on, the program runs until the total number of time steps reaches this value. The default value is 1000.

3.6.23 no_of_threads

If simulations are performed using Nvidia GPU, this is the number of threads per block for a particular GPU card, which should be supplied. The value depends on the hardware architecture of the card. It is advisable to use 32 threads for hardware versions lower than 3.5. For hardware versions higher than or equal to 3.5, it is advisable to use 64 or 192 threads. Tests can be performed to find the best value to achieve maximum efficiency. The default value is 32.

3.6.24 no_of_unit_cell_x

The number of unit cells in the x direction.

3.6.25 no_of_unit_cell_y

The number of unit cells in the y direction.

3.6.26 no_of_unit_cell_z

The number of unit cells in the z direction.

The default values of no_of_unit_cell_x, no_of_unit_cell_y and no_of_unit_cell_z are as follows:

Option	x	y	z
bcc100	10	10	10
bcc111	6	4	7
fcc100	10	10	10
hcp0001	10	10	10
readconf	N/A	N/A	N/A
readvsim	N/A	N/A	N/A

3.6.27 OMP_threads

If simulations are performed using CPUs, the number of threads for running the program in parallel using OpenMP should be supplied. If a serial run is intended, this value should still be supplied and should be defined as 1. Note that its default value is 2.

3.6.28 out_body

This is the body of all the output files' names. For example, if its value is *text*, the output file for temperatures is going to be *tmp-text.dat* etc. It has no default value.

3.6.29 phi_limit

If #**define changestep** is switched on, this is the maximum value of spin rotation angle over one time step **step**. If there is even a single spin in the entire simulation cell that rotates by more than this limiting value over one simulation time step, the size of the time step will be reduced by multiplying it by 0.8. Otherwise, **step** will increase by 1.05, provided that a similar separate criterion is satisfied for the lattice subsystem. The value is given in the units of Radian. Its default value is $2\pi/10$.

3.6.30 pressure

If #**define PRESSURE** is switched on, this is the desired pressure, given in GPa units. Its default value is 0.

3.6.31 random_seed

This is the seed for the random number generator. Its default value is 1234.

3.6.32 rcut_mag

If a simulation involves spin-spin interactions, this is the cut-off radius of the exchange coupling function. The parameter is given in Å units. Its default value is 3.75.

3.6.33 rcut_max

This is the maximum value of **rcut_pot**, **rcut_mag**, and **rcut_vol**. It needs to be input manually. The parameter is given in Å units. Its default value is 4.1.

3.6.34 rcut_pot

If the simulation involves computing interatomic forces, this parameter defines the cut-off radius for the EAM potential. It is given in Å units. Its default value is 4.1.

3.6.35 rcut_vol

If #**define localvol** is switched on, this is the cut-off radius used when calculating the local atomic volume using an approximate method described above. The parameter is given in Å units. Its default value is $1.2a$, where a is the lattice constant.

3.6.36 min_length_link_cell

This is the minimum length of any edge of a linked-cell. The number of linked cells in a simulation box is determined by this value. It must always be larger than **rcut_max**. Its default value is **rcut_max** + 0.1, in Å units.

3.6.37 start_time

If #**define inittime** is switched on, this is the initial value of the total simulation time. Its default value is 0.

3.6.38 step

This is the simulation time step, expressed in seconds. If #**define changestep** is switched on, this is the initial value of the time step. Its default value is 1×10^{-15} s.

3.6.39 stress_xx

If #**define STRESS** is switched on, this is the desired stress σ_{xx}, in GPa units. Its default value is 0.

3.6.40 stress_yx

If #**define STRESS** is switched on, this is the desired stress σ_{yx}, in GPa units. Its default value is 0.

3.6.41 stress_yy

If #**define STRESS** is switched on, this is the desired stress σ_{yy}, in GPa units. Its default value is 0.

3.6.42 stress_zx

If #**define STRESS** is switched on, this is the desired stress σ_{zx}, in GPa units. Its default value is 0.

3.6.43 stress_zy

If #**define STRESS** is switched on, this is the desired stress σ_{zy}, in GPa units. Its default value is 0.

3.6.44 stress_zz

If #**define STRESS** is switched on, this is the desired stress σ_{zz}, in GPa units. Its default value is 0.

3.6.45 temperature

This is the thermostat temperature for both spins and the lattice atoms, given in Kelvin units. Its default value is 300.

3.6.46 total_production_time

If #**define runstep** is switched off, the program runs until the total simulation time reaches this value. This parameter is given in seconds. Its default value is 1×10^{-13}.

3.6.47 vsim_prec

If #**define writevsim** is switched on, this is the precision of each output value in V_sim compatible output files. Since these files are created for visualization purposes, the output values do not need to have high precision, as this saves output time and disk space. Its default value is 4.

3.7 In/Output files

In/Output files and their formats are described in this section. All the output files have the same common name as the value of **out_body**. For the convenience of our discussion here, we set **out_body** = *xxx*. There is a limit of 256 characters on the length of this variable.

3.7.1 enr-xxx.dat

This is a file that accumulates data on energies per atom for every **interval_of_print_out** step. The format is as follows:

Column 1	2	3	4	5	6	7
current step	total time	P.E.	K.E.	E_{spin}	E_{corr}	$E_{electron}$

where E_{spin} and E_{corr} are the energies corresponding to Hamiltonians $\mathcal{H}_{spin}$ and $\mathcal{H}_{corr}$. The respective columns only appear if one considers the corresponding degrees of freedom in simulations. For example, if one performs a lattice-electron simulation, the format of the file is:

Column 1	2	3	4	5
current step	total time	P.E.	K.E.	$E_{electron}$

All the energies are given in the eV per atom units.

3.7.2 prs-xxx.dat

This is a file that accumulates data on pressure per atom for every **interval_of_print_out** step. The format of the file is as follows:

Column 1	2	3	4	5	6	7	8
current step	total time	d_{xx}	d_{yx}	d_{yy}	d_{zx}	d_{zy}	d_{zz}

9	10
density	pressure

The simulation box vectors are in Å units, and pressure is in GPa units.

3.7.3 str-xxx.dat

This is a file that accumulates data on stresses per atom for every **interval_of_print_out** step. The format is as follows:

Column 1	2	3	4	5	6	7	8
current step	total time	d_{xx}	d_{yx}	d_{yy}	d_{zx}	d_{zy}	d_{zz}

9	10	11	12	13	14	15	16
density	$\sum_\alpha \sigma_{\alpha\alpha}/3$	σ_{xx}	σ_{yy}	σ_{zz}	σ_{xy}	σ_{zy}	σ_{zx}

The simulation box vectors are in Å units, and stresses are in GPa units.

3.7.4 tmp-xxx.dat

This is a file that accumulates data on temperatures of various subsystems in the simulation box for every **interval_of_print_out** step. The format is as follows:

Column 1	2	3	4	5	6
current step	total time	T_l	T_s	T_s'	T_e

where T_l is the lattice temperature, T_s is the spin temperature calculated using equation (2.11), T_s' is the spin temperature calculated using equation (2.14), and T_e is the electronic temperature. If longitudinal spin fluctuations are not treated, only T_s is going to be calculated. Otherwise, both T_s and T_s' are going to be calculated. The respective columns only appear if one considers the corresponding degrees of freedom in a simulation. All the temperatures are given in Kelvin units.

3.7.5 spn-xxx.dat

This file accumulates data on average values of spins and magnetic moments for every **interval_of_print_out** step. The format is as follows:

Column 1	2	3	4	5	6
current step	total time	$\langle S_{ix}\rangle$	$\langle S_{iy}\rangle$	$\langle S_{iz}\rangle$	$\lvert\langle \mathbf{S}_i\rangle\rvert$

6	7	8	9
$\langle M_{ix}\rangle$	$\langle M_{iy}\rangle$	$\langle M_{iz}\rangle$	$\lvert\langle \mathbf{M}_i\rangle\rvert$

Note that $\mathbf{M}_i = -g\mu_B\mathbf{S}_i$, where spin is dimensionless. Magnetic moments have the dimensionality of Bohr magneton μ_B.

3.7.6 cel-xxx_nnnn.dat

This file contains data on temperatures in each linked cell at a particular moment of time. This file will be generated only if option **#define eltemp** is switched on. The SPILADY program then produces an output file for every **interval_of_config_out** step. An extra part of the file name is then going to be added. *nnnn* spans values from 0000 to 9999. It represents the sequence of output files. After initialization, before any real calculation, a file *cel-xxx_0000.dat* is produced. Then, the index increases by 1 for each new output file. When the program reaches the desired total number of production steps or production time, a file *cel-xxx_9999.dat* is generated, regardless of the previous output file numbering. The format is as follows

	1	2	3	4	5	6	7
1	no. of cells	total time					
2	d_{xx}	d_{yx}	d_{yy}				
3	d_{zx}	d_{zy}	d_{zz}				
4	cell 1 x	cell 1 y	cell 1 z	T_{1e}	T_{1l}	T_{1s}	T'_{1s}
5	cell 2 x	cell 2 y	cell 2 z	T_{2e}	T_{2l}	T_{2s}	T'_{2s}
⋮							

The first line is the total number of linked cells and the total time. The second and third lines are the simulation box vectors. Starting from the fourth line, it is the x, y and z coordinates of the centre of linked-cell i, followed by the local electron temperature T_{ie}, local lattice temperature T_{il} and the local spin temperatures T_{is} and T'_{is} calculated using equations (2.11) and (2.14), respectively, for each linked-cell. These columns only appear when one considers the relevant degrees of freedom in a simulation.

If **#define readTe** and **#define eltemp** are switched on, this file can be used as input for a new simulation, which supplies information about local electronic temperature. One should define the file name using variable

in_eltemp.

3.7.7 con-xxx_nnnn.dat

This file contains information about the atomic configuration at a given moment of time. The program produces an output file for every **interval _of_config_out** step. An extra part of the file name is added. The value of *nnnn* spans the interval from 0000 to 9999. It represents the sequence of output files. After initialization, before any real calculation, a file named *con-xxx_0000.dat* is generated. Then, the index is increased by 1 for each output file. When the program attains the decided total production steps or production time, a file *con-xxx_9999.dat* is generated, regardless of the previous output file numbering. The format is as follows

	1	2	3	4	5	6	7	8
1	no. of atoms	total time	type					
2	d_{xx}	d_{yx}	d_{yy}					
3	d_{zx}	d_{zy}	d_{zz}					
4	1	element$_1$	R_{1x}	R_{1y}	R_{1z}	p_{1x}	p_{1y}	p_{1z}
5	2	element$_2$	R_{2x}	R_{2y}	R_{2z}	p_{2x}	p_{2y}	p_{2z}
⋮								

	9	10	11	12	13	14	15	16	17
1									
2									
3									
4	F_{1x}	F_{1y}	F_{1z}	$\sigma_{1,xx}$	$\sigma_{1,yy}$	$\sigma_{1,zz}$	$\sigma_{1,xy}$	$\sigma_{1,zy}$	$\sigma_{1,zx}$
5	F_{2x}	F_{2y}	F_{2z}	$\sigma_{2,xx}$	$\sigma_{2,yy}$	$\sigma_{2,zz}$	$\sigma_{2,xy}$	$\sigma_{2,zy}$	$\sigma_{2,zx}$
⋮									

	18	19	20	21	22	23	24	25	26	27
1										
2										
3										
4	ρ_1	(K.E.)$_1$	(P.E.)$_1$	S_{1x}	S_{1y}	S_{1z}	$\lvert\mathbf{S}_1\rvert$	H_{H1x}	H_{H1y}	H_{H1z}
5	ρ_2	(K.E.)$_2$	(P.E.)$_2$	S_{2x}	S_{2y}	S_{2z}	$\lvert\mathbf{S}_2\rvert$	H_{H2x}	H_{H2y}	H_{H2z}
⋮										

	28	29	30	31	32	33
1						
2						
3						
4	H_{L1x}	H_{L1y}	H_{L1z}	$E_{spin,1}$	$E_{corr,1}$	Ω_1
5	H_{L2x}	H_{L2y}	H_{L2z}	$E_{spin,2}$	$E_{corr,2}$	Ω_2
$\vdots$						

The first line is the total number of atoms, the total time and the type of simulation, i.e. MD, SDH, SDHL, SLDH or SLDHL. The second and third lines are the simulation box vectors. Starting from the fourth line, it gives information about atom i, where i is the index of an atom, element symbol, atomic position, momentum, force, stresses, effective electron density, kinetic energy, potential energy, spin vector, magnitude of the spin vector, the Heisenberg part of the effective field, the Landau part of the effective field (including the correction term), the energy corresponding to $\mathcal{H}_{spin}$, the energy corresponding to $\mathcal{H}_{corr}$ and the local atomic volume. If #**define magmom** is switched on, spin vectors are replaced by magnetic moments. The relevant columns only appear if the corresponding degrees of freedom are included in a simulation.

If #**define readconf** is switched on, this file can be used as an input file for a new simulation, which supplies information about atoms/spins. One should define the file name using variable **in_config**. The **total time** in the first line of the file will be used as the initial simulation time, if #**define inittime** is switched off. One should also note that file output from an MD simulation can only be used as input for another MD simulation (not for an SD or SLD simulation). The same applies to SD and SLD simulations.

3.7.8 vsm-xxx_nnnn.ascii/.spin/.dat

If #**define writevsim** is switched on, the program generates files containing information about atoms at a given moment of time. They are produced in a V_sim readable format for visualization purposes for every **interval_of_vsim** step. An extra part of the file name is added. The value of *nnnn* spans interval from 0000 to 9999. It represents a sequence of output files. After initialization, before any real calculation, files *vsm-xxx_0000.ascii/.spin/.dat* are generated. Then, the file index increases by 1 for each output file. When the program attains a pre-defined number of total production steps or production time, files *vsm-xxx_9999.ascii/.spin/.dat* are generated, regardless of the previous output file numbering.

vsm-xxx_nnnn.ascii

This file allows V_sim to visualize atoms. The format is as follows

	1	2	3	4	5	6	7
1	no. of atoms	total time					
2	d_{xx}	d_{yx}	d_{yy}				
3	d_{zx}	d_{zy}	d_{zz}				
4	R_{1x}	R_{1y}	R_{1z}	element$_1$	p_{1x}	p_{1y}	p_{1z}
5	R_{2x}	R_{2y}	R_{2z}	element$_2$	p_{2x}	p_{2y}	p_{2z}
$\vdots$							

The first line is the total number of atoms and the total time. The second and the third lines are the simulation box vectors. Starting from the fourth line, it gives information about atom i, specifying its position, element symbol, and kinematic momentum.

vsm-xxx_nnnn.spin

This file allows V_sim to visualize spins. It is not mandatory to supply this file. The file *vsm-xxx_nnnn.ascii* should also be supplied in the case if spin visualization is intended. The format is as follows

	1	2	3
1	no. of atoms		
2	$\|\mathbf{S}_1\|$	θ_1	ϕ_1
3	$\|\mathbf{S}_2\|$	θ_2	ϕ_2
$\vdots$			

The first line is the total number of atoms. Starting from the second line, the file contains information about spin i in spherical coordinates, which is the magnitude of the spin vector, and its polar and azimuthal angles θ and ϕ. Angles θ and ϕ are given in degrees. If **#define magmom** is switched on, the magnetic moment of an atom is used as an output variable instead of an atomic spin.

vsm-xxx_nnnn.dat

This file allows use of a user-defined color scheme in V_sim. It is not mandatory to supply this file. The file *vsm-xxx_nnnn.ascii* should also be supplied. The format is as follows

	1
1	$(\text{P.E.})_1$
2	$(\text{P.E.})_2$
⋮	

Starting from the first line, it gives information about the potential energy of an atom. In the case of spin dynamics with/without longitudinal fluctuations, the format is

	1
1	$\lvert\mathbf{S}_1\rvert$
2	$\lvert\mathbf{S}_2\rvert$
⋮	

It gives the magnitudes of the spin vectors (or magnetic moments if #**define magmom** is switched on). One could change the output into K.E. (kinetic energy), stresses, etc. by editing the source code.

If #**define readvsim** is switched on, these files can be used as input files for a new simulation, which supplies information about atoms/spins. One could assign values to variables **in_vsim_atom** = *vsm-xxx_nnnn.ascii* and **in_vsim_spin** = *vsm-xxx_nnnn.spin*. The **total time** in the first line of *vsm-xxx_nnnn.ascii* will be used as the initial simulation time, if #**define inittime** is switched off.

It should be noted that these files are only in V_sim *readable* format. It does not mean that SPILADY can read all the files in conventional V_sim format. Besides, since these files are produced as output for visualization purposes, the default value of **vsim_prec** = 4. If one intends to use them as input files for a new simulation, one should increase the value of **vsim_prec** to 16.

4

Examples

"Failures, repeated failures, are finger posts on the road to achievement. One fails forward toward success."

– C. S. Lewis (1898 - 1963)

In this section we are going to describe applications of SPILADY. They use parameters and the functional form of the EAM potential for pure iron. The many-body potential [17], the exchange coupling function [24], the Landau coefficients [34], and electron heat capacity [44] are already defined in files *DD05.cpp*, *JijFe.cpp* and *heatcapacity_CPU.cpp* or *heatcapacity_GPU.cu*. There is a small adjustment in the value of J_{ij} in comparison to Ref. [24]. This is to make the results agree better with the experimental Curie temperature T_C. There are some other potential files in folder *potential*.

We advise the users to try a CPU version of SPILADY before progressing to a GPU implementation. This is because the source code is more readable in the CPU version. We also advise use of CPUs, instead of GPUs, for spin dynamics and spin-lattice dynamics cases, if a system is smaller than 54,000 spins (atoms). This is because the parallel algorithm based on the Suzuki-Trotter decomposition only performs parallel calculations at the level of linked-cells, not atoms. Due to the limited number of linked-cells within a non-interacting group, the powerful massive parallel environment of a GPU cannot be fully utilized if the total number of atoms is relatively small. On the other hand, if one is interested in a million atoms scale simulation, GPU is more efficient. One should do the testing on one's local computer.

We also advise creation of a folder to keep all the original files, and copy them to a working examples folder, so that input and output files and potential files for a particular example are not mixed up with other examples. We further recommend reading all the files in the example folders, especially the files *variables.in* and *control.h*, before starting a simulation, to become familiar with the meaning of input parameters.

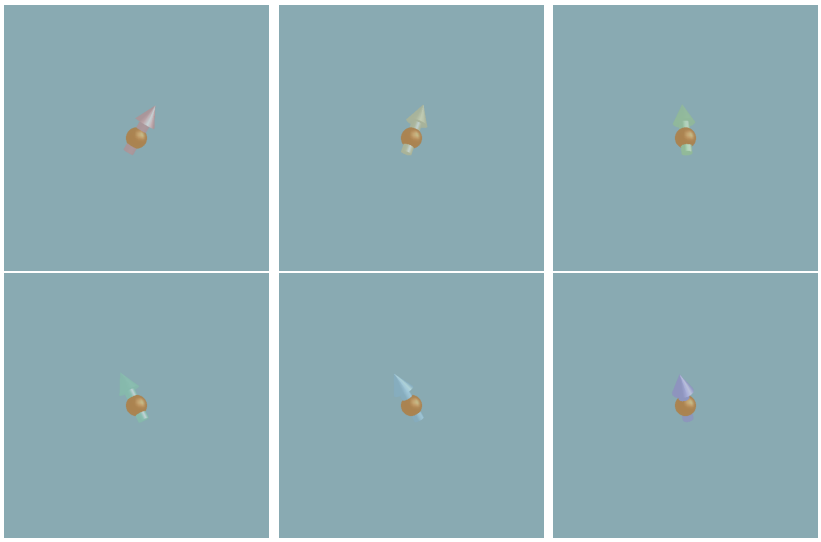

Figure 4.1: A single magnetic moment with magnitude of $2.2\mu_B$ interacting with external magnetic field of 100 Tesla. From left to right, and top to bottom, they are snapshots corresponding to t =0fs, 50fs, 100fs, 150fs, 200fs and 250fs.

When one runs SPILADY, one can plot the data accumulated in the output files. Temperatures, magnetization, pressure, stresses and energies can be plotted as functions of time using any plotting software, for example *gnuplot* in Linux or *Origin* in Windows. For visualizing atoms and spin, we recommend using V_sim, which has been developed at CEA, France. It is free software, which can be downloaded from http://inac.cea.fr/L_Sim/V_Sim/ .

4.1 Non-interacting magnetic moment

The first example illustrates rotation of a single magnetic moment interacting with external magnetic field. All the necessary files are in folder *example1*. One needs to copy them into the working folder, and compile and run the program as described in previous chapters.

A single magnetic moment with magnitude of $2.2\mu_B$ is placed in a simulation box. It interacts with external magnetic field of 100 Tesla pointing in the z direction. The angle between the external field and magnetic moment is 30 degrees. The initial configuration can be observed in *vsim-example1.ascii* and *vsim-example1.spin*. The external field is controlled by file *extfield.in*. Although there is no spin-spin interaction, one still needs to supply the spin-spin interaction file *Jij0.cpp* for a spin dynamics simulation, where function *Jij_gen* only returns zero values.

Fig. 4.1 shows rotation of magnetic moment in the anti-clockwise direction. We should note that the rotation direction of magnetic moments is opposite to that of atomic spins, since they are related through $\mathbf{M}_i = -g\mu_B\mathbf{S}_i$. Option #**defined magmom** is switched on here. A movie of this is in folder *example1*.

4.2 Interacting spins - Heisenberg Hamiltonian

The second example shows a spin system interacting through the Heisenberg Hamiltonian. All the necessary files are in *example2*. We are now working with atomic spins, not atomic magnetic moments, so that option #**defined magmom** is switched off. Function *Jij_gen* in *JijFe_fix_lattice.cpp* is the exchange coupling J_{ij} of the Heisenberg Hamiltonian for atomic spins $\mathbf{S}_i$. Parameters of the simulation correspond to those of bcc iron.

A bcc lattice with 2000 spins is set up. We have input files *vsim-example2.ascii* and *vsim-example2.spin*. The directions of spin vectors are assigned at random. The simulation does not impose any thermostat on the spin system. We work with the microcanonical spin ensemble. The magnitudes of atomic spins are fixed at $2.2/g$, where $g = 2.0023$ is the electronic g-factor. From the output file *tmp-2000_interact_spins.dat*, one can see that the spin temperature calculated according to Eq. (2.14) is approximately 233K. Its value is fluctuating, since only either temperature or energy can be kept constant, but not both.

Fig. 4.2 shows that spins rotate in the clockwise direction. There is no external magnetic field. Spins rotate because there are effective Weiss fields acting on them. The effective fields result from the interaction of spins with other, neighbouring, spins. One can observe that the speed of rotation of atomic spins in iron is about 10 to 15fs per cycle. A movie of it is in folder *example2*.

4.3 Thermalization - Heisenberg-Landau Hamiltonian

The third example illustrates the dynamics of thermalization of an interacting spin system described by the Heisenberg-Landau Hamiltonian. All the necessary files are in *example3*. A bcc lattice with 16000 spins is set up. We do not have input configuration files in the example. We initialize the lattice and spin configuration via *variables.in* and *control.h*. All the spins are initially collinear. The Langevin thermostat is applied to the spin system.

Fig. 4.3 illustrates the dynamics of thermalization of the spin system interacting through the Heisenberg-Landau Hamiltonian. The figure was plotted using *gnuplot*. One can type the following commands in the *gnuplot* environment after completing the simulation.

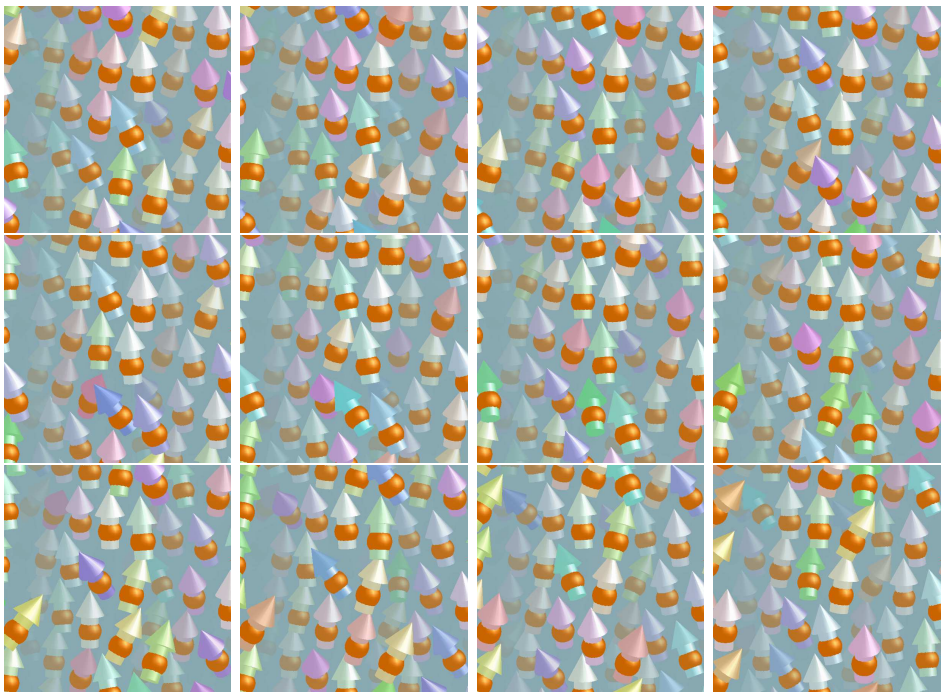

Figure 4.2: A bcc lattice with 2000 interacting spins. From left to right, and top to bottom, the snapshots correspond to 0fs, 2fs, 4fs, 6fs, 8fs, 10fs, 12fs, 14fs, 16fs, 18fs, 20fs and 22fs.

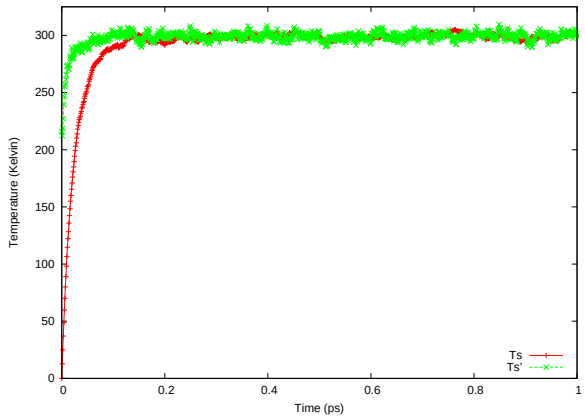

Figure 4.3: A fluctuating spin system with longitudinal degrees of freedom is thermalized from a collinear ferromagnetic perfect lattice configuration to 300K. The red line shows spin temperature calculated using equation (2.11). The green line is the spin temperature calculated using equation (2.14).

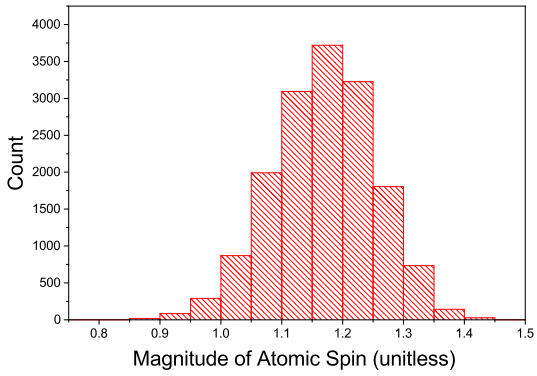

Figure 4.4: A histogram of magnitudes of atomic spins of a system with 16000 spins at 300K.

gnuplot> plot "tmp-16000_interacting_spins_HL_0K_to_300K.dat" using 2:3, "" using 2:4

The command means that the second column of the data file is used as x-coordinates and the third or the fourth columns are treated as y-coordinates.

Fig. 4.3 shows how spin temperatures vary as functions of time. Spin temperatures are calculated using equation (2.11) and (2.14). We note that both equations are only valid if the system is in thermal equilibrium. In the case of non-equilibrium, the values derived from equations for the temperature can only be treated as estimates. The figure shows that the system is heated up to 300K as prescribed. Atomic spins are non-collinear at a finite temperature.

Fig. 4.4 shows a histogram of magnitudes of atomic spins. The histogram shows that the magnitudes of spins are no longer constant, and they exhibit a considerable spread of their values. All the atomic spins are initialized with the magnitude of $2.2/g$, where $g = 2.0023$. When the system is in thermal equilibrium, magnitudes of spins fluctuate, this is a direct consequence of dynamics associated with the longitudinal (Landau) part of the spin Hamiltonian.

4.4 MD - Stresses control

The fourth example is a molecular dynamics simulation. It illustrates how to control internal stress components by changing the simulation box vectors. A bcc lattice with 2000 atoms is set up in a cubic box. We put $\sigma_{xx} = \sigma_{yy} = \sigma_z = 0$GPa, $\sigma_{zx} = \sigma_{zy} = 0.1$GPa and $\sigma_{xy} = -0.2$GPa in *variables.in*,

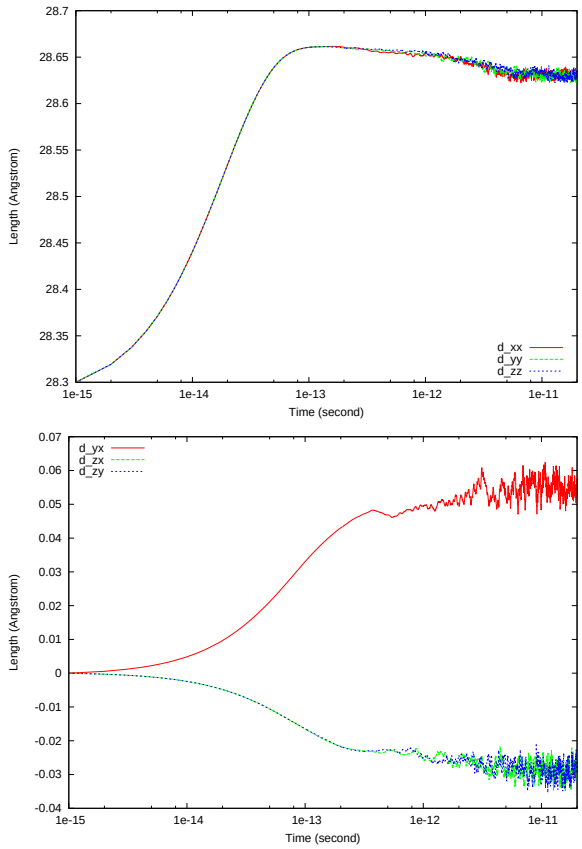

Figure 4.5: Components of the simulation box vectors plotted as functions of time.

and #**defined STRESS** is switched on in *control.h* The simulation box is thermalized from 0K to 300K using the Langevin thermostat.

Fig. 4.5 shows how the components of the simulation box vectors vary as functions of time. Fig. 4.6 illustrates how internal stresses reach pre-defined values at about 1ps. Box vectors components d_{yx}, d_{zx} and d_{zy} deviate from zero, in order to impose the desired stresses onto the box. The simulation box is no longer cubic, it now has a triclinic shape.

4.5 Thermalizing a spin-lattice system

The fifth example shows how to thermalize a spin-lattice system with longitudinal fluctuations from a collinear ferromagnetic perfect lattice configuration to 300K. The size of the simulation box varies according to its internal pressure.

A bcc iron configuration with 16,000 atoms has been set up according to input files. There are $20 \times 20 \times 20$ unit cells, where each unit cell contains

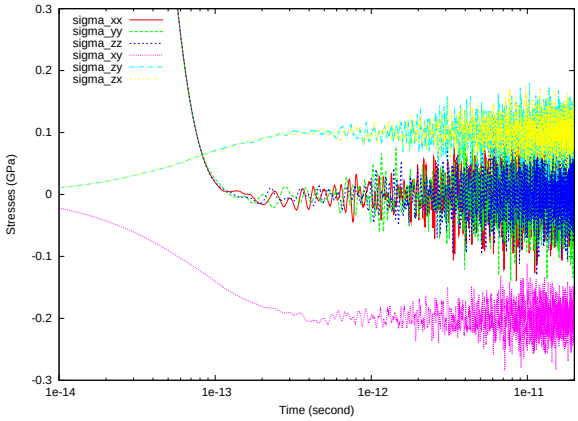

Figure 4.6: Average stresses plotted as functions of time.

2 atoms. Values of damping constants γ_l and γ_s are the same as in Refs. [29, 30].

Fig. 4.7 shows temperature changes, as a function of time, for T_l, T_s and T_s'. It shows clearly that all the temperatures reach 300K eventually, but over different timescales, which depend on the values of damping parameters.

If #**defined PRESSURE** is switched on, the size of the simulation box relaxes according to its internal pressure. In Fig. 4.8, internal pressure has a positive value initially. It means that there are forces that try to expand the cell. The lattice constant, which can be obtained by dividing the length of a simulation box edge by the number of unit cells on that edge, increases and reaches a maximum. At around the same time, pressure drops to zero, which is the pre-defined value of **pressure** in *variables.in*. The simulation box is still heating up until it reaches 300K. The length of the box edges reaches equilibrium eventually.

4.6 SLD - Atomic spin chain

The sixth example illustrates the dynamics of an atomic spin chain. We show that SPILADY can deal with an arbitrary crystal structure. A spin chain containing 10 atoms is placed in a box with periodic boundary condition. The input configuration file is in the SPILADY format.

The atomic spin chain is thermalized from 0K to 50K. Both the spins and the atoms are heated up. Fig. 4.9 shows snapshots of the spin chain at 0ps, 1ps and 10ps. The atoms and spin vectors deviate from their initial values due to thermal excitations.

If you can reproduce the results shown in the above examples, you are now able to run SPILADY successfully. You may now try to change the val-

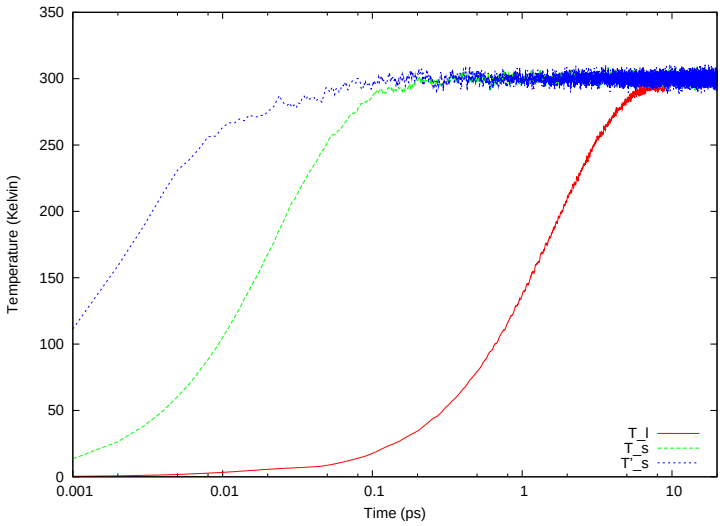

Figure 4.7: A spin-lattice system with longitudinal fluctuations is thermalized from a collinear ferromagnetic perfect lattice configuration to 300K.

ues of variables or try the GPU mode of SPILADY. Below, we give two more examples. They require a more advanced level of understanding of the program. Modification of the source code is required.

4.7 Laser pulse demagnetization

The seventh example is a little more challenging. This simulation is limited to the CPU version only. We are going to heat up the system by a laser pulse. This could be mimicked by adding extra energy to the electronic subsystem, assuming that all the energy from a laser pulse is absorbed by the electrons. A Gaussian form of the energy pulse is assumed. In our previous work, we performed a similar simulation on a spin-lattice system without longitudinal fluctuations. Detailed discussions of the physical content of the method and technique can be found in [29]. We are going to perform a simulation on a spin-lattice system with longitudinal fluctuations.

We need to edit a small part the of source code. All the files that require editing are in folder *example7*. Parts of the source code have been modified. They are *spilady.cpp* and *prototype_CPU.h*, and a new file *laser_demagnetization_CPU.cpp*. One may just copy these files into a working folder, and compile and run the program.

The file *laser_demagnetization_CPU.cpp* contains a function describing how the energy pulse is absorbed by the electronic subsystem. Whenever a new function is added to the program, one needs to put its prototype in *prototype_CPU.cpp*. If it is a GPU device code, its prototype

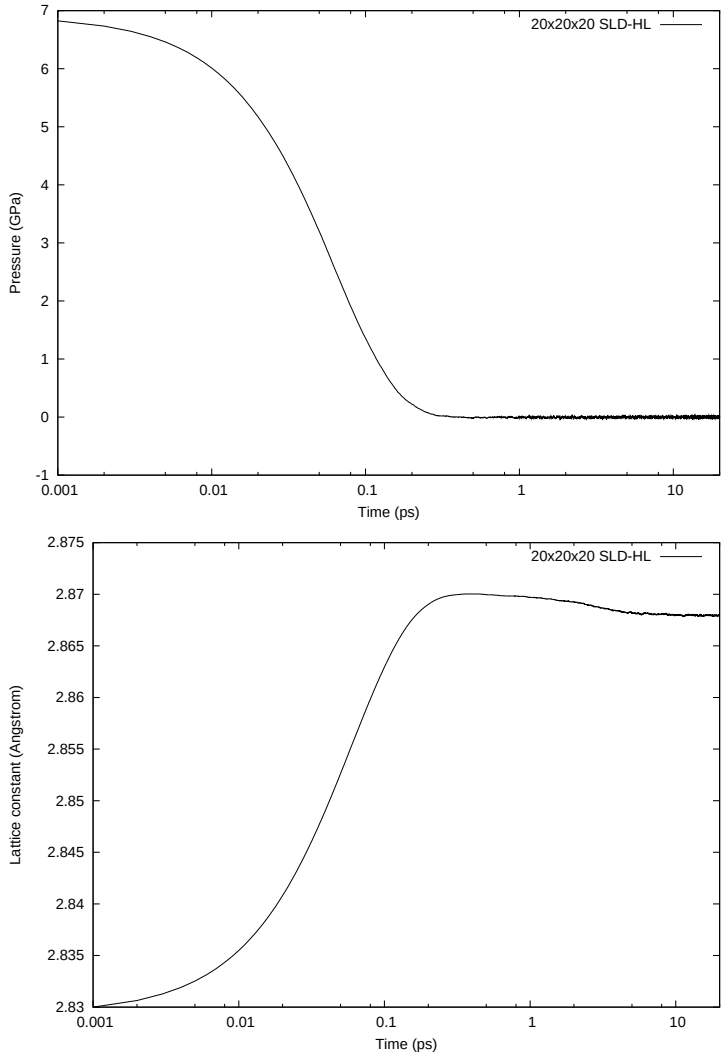

Figure 4.8: (Top) Change of internal pressure in the simulation box during thermalization to 300K. (Bottom) Change of the lattice constant as a function of time.

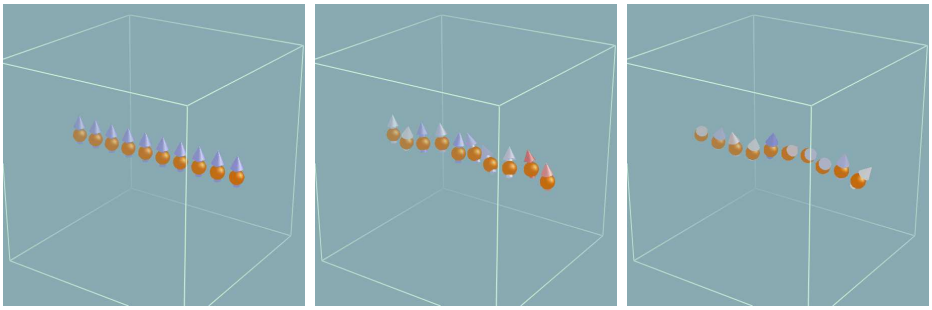

Figure 4.9: A spin chain thermalizing from 0K to 50K. From left to right, snapshots correspond to 0ps, 1 ps and 10 ps.

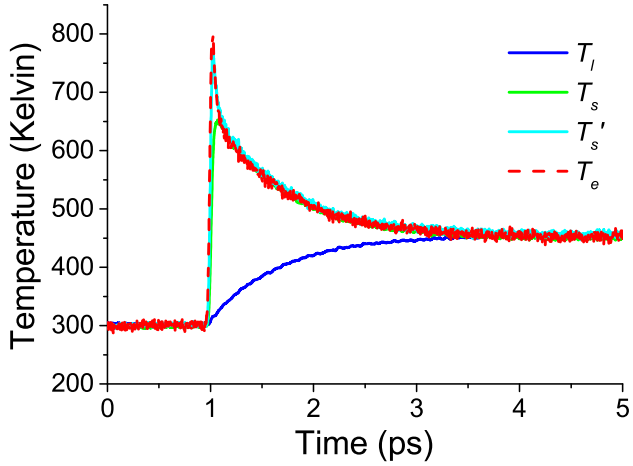

Figure 4.10: Temperatures change as a function of time, assuming that a laser pulse occurs at around 1ps.

needs to be put in *prototype_GPU.cu*. We have already inserted function **laser_demagnetization _CPU()** in *spilady.cpp*, and added it to *prototype_CPU.cpp* as shown in the files in folder *example7*.

We use a final output configuration file of a spin-lattice system at 300K with 16000 atoms/spins as the input configuration file, and switched on #**define eltemp**. We now deal with the spin-lattice-electron system. The total energy is conserved apart from the extra energy delivered by the laser pulse. The laser pulse comes at around 1 ps. In Fig. 4.10, we see a sharp peak of T_e at around 1ps, which is due to the laser pulse. T_s' responds quickly to T_e, because electrons and longitudinal spin degrees of freedom are strongly coupled. T_s responds a little more slowly. This is because T_s primarily describes relaxation of spin-spin directional collective excitations.

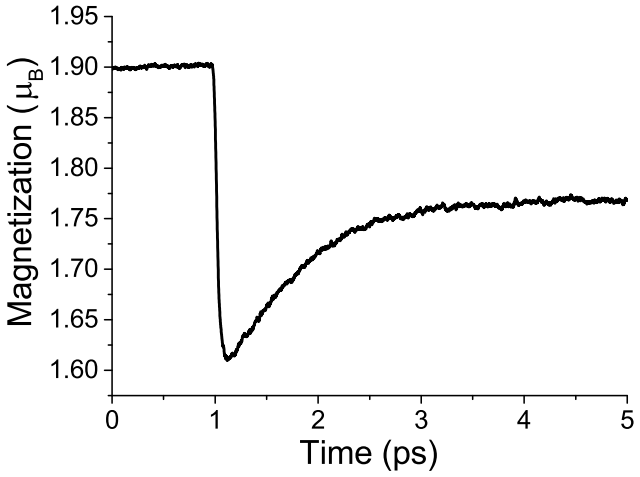

Figure 4.11: Demagnetization and recovery process, where a laser pulse occurs at around 1ps.

They take a slightly longer time than longitudinal fluctuations to absorb energy and achieve maximum entropy. T_l is the slowest evolving component of temperature, since coupling between electrons and the lattice is comparatively weak.

In Fig. 4.11, magnetization is plotted as a function of time on the same scale as in Fig. 4.10. It drops significantly at the point when a laser pulse is introduced. Then, it gradually recovers. However, it does not recover fully back to its original value, because the temperature of the system increases from 300K to about 450K.

4.8 Compressive wave

Similar to the above example, the eighth example requires modification of SPILADY source code. This simulation is limited to the GPU version only. We introduce a compressive wave by reducing the length of the simulation box in the z direction, which essentially has the same effect as adding soft pistons at both sides of the box. Detailed discussions of the physical content of the method and technique can be found in Ref. [29]. Again, we are going to perform a simulation of a spin-lattice system with longitudinal fluctuations.

All the edited files are in folder *example8*. Parts of the source code have been modified. The amended files are *spilady.cpp* and *prototype_CPU.h*, and there is a new file *compressive_wave_GPU.cu*. One may just copy these files into a working folder, and compile and run the program. File

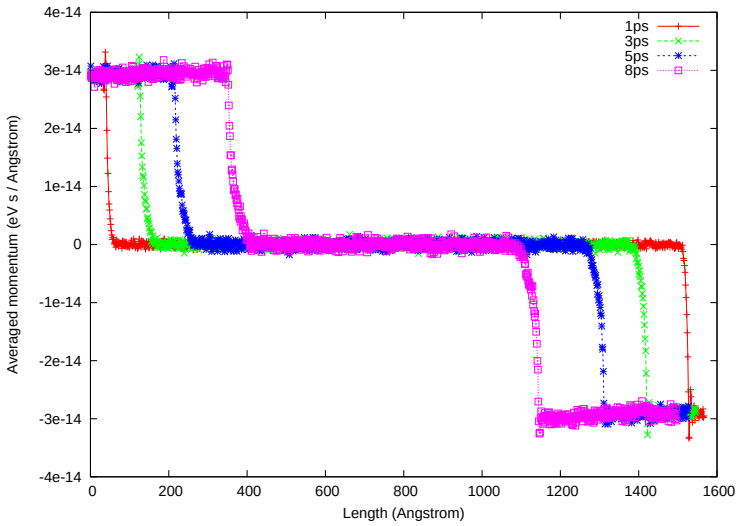

Figure 4.12: Average momentum across the simulation box at 1, 3, 5, and 8 ps.

compressive_wave_GPU.cu contains a function describing how to introduce the compressive wave.

We use the final output configuration file of a spin-lattice system at 300K with $30 \times 30 \times 550$ unit cell, i.e. 990000 atoms/spins, as the input configuration file, and switch on #**define eltemp** and #**define localcolmot**. Because of the large size of the box, it may take up to a week of computer simulation time before the system reaches 10ps. One could run this example in the MD mode, which can substantially reduce the computation time.

Fig. 4.12 shows the average momenta calculated as a function of position across the simulation box in planes normal to the z direction at 1, 3, 5 and 8 ps. An analysis file *cal_momentum.f* is included in the example folder for producing a similar figure. We can see a compressive wave propagating along the box at a constant velocity. It results from the initial compression occurring at the edges of the box.

If you have been able to reproduce all of the above examples, congratulations, you should now be able to perform simulations using this SPIn-LAttice-DYnamics program, SPILADY.

Bibliography

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

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

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

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

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

[6] L. G. Wang and M. Sob, Phys. Rev. B **60**, 844 (1999)

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

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

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

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

[11] F. Körmann *et al.*, Physica Status Solidi B **251**, 53 (2014)

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[33] Haohua Wen and C. H. Woo, J. Nucl. Mater. **455**, 31 (2014)

[34] Pui-Wai Ma and S. L. Dudarev, Phys. Rev. B **90**, 024425 (2014)

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

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

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

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

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

[40] Z. G. Wang, Ch. Dufour, E. Paumier, and M. Toulemonde, J. Phys.: Condens Matter **6**, 6733 (1994)

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

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

[43] A. E. Sand, S. L. Dudarev, and K. Nordlund, EPL **103**, 46003 (2013)

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