Multiscale UppASD Tool

User's Guide

(version 1.0)

E. Méndez, E. Rosén, N. Ntallis, M. Pereiro and O. Eriksson

HTTP://PHYSICS.UU.SE/UPPASD

May 2020

μASD

MULTISCALE ATOMISTIC SPIN DYNAMICS

Contents

μASD

MULTISCALE ATOMISTIC SPIN DYNAMICS

1. General Overview

1.1 Introduction

This user guide describes the essential features of the Multiscale Atomistic Spin Dynamics program (μASD). The program μASD is a preprocessing tool for Uppsala Atomistic Spin Dynamics (UppASD) that allows user to setup the partitioned domain multiscale geometry. The ASD method and its implementation is described in the article by Skubic *et al.* [1]. The underlying theory is described in the articles by Antropov *et al.* [2] and García-Palacios and Lázaro [3]. An old, yet remarkably lucid overview is also given by Watson *et al.* [4]. A thorough review of the method and relevant applications is given in the book by Eriksson *et al.* [5]. The details of the multiscale framework, its capabilities, aims and limitations are covered in papers [6–8]. This documentation is a supplementary material for the main UppASD documentation [9] and covers only technical details related to implementation of the preprocessing tool and user input files. The system of units used in the program μASD are the same as in UppASD along with the common input files and for that information we refer to UppASD user guide [9]. In this guide, some information concerning the analysis of data generated by the code is also given.

The geometry, which is created by μASD, is partitioned into two regions: the continuum region, which is discretised using the finite-difference representation, and the atomistic region. Since the Landau-Lifshitz-Gilbert (LLG) equation of the continuum region has the same structure as the atomistic LLG equation, the same solver is used in both regions, i.e. finite-difference nodes are treated as atoms, however, with a different interaction. User has to be aware of this and if additional interactions/effects are added to the atomistic region, they have to be changed accordingly for the continuum region nodes as well. The examples, which are presented here, show how this is done for the classical exchange interaction.

In this document, user input parameters and some aspects of geometry generation are described by explaining the input files of several 1D and 2D problems.

1.2 Theoretical background

In the atomistic approach, the dynamic behaviour of magnetic moments of individual atoms is described by the atomistic Landau-Lifshitz-Gilbert-Slonczewski (LLGS) equation of motion [6–8]:

Equation 1 — The atomistic Landau-Lifshitz-Gilbert-Slonczewski equation.

$$\begin{aligned}\frac{\partial \mathbf{m}_i}{\partial t} &= -\beta_{\mathrm{L}}\mathbf{m}_i \times (\mathbf{h}_i + b_1\tau_i) - \alpha_{\mathrm{L}}\mathbf{m}_i \times (\mathbf{m}_i \times (\mathbf{h}_i + b_2\tau_i)),\\ \tau_i &= \mathbf{g}\cdot\sum_k \mathbf{s}_{ik}\mathbf{m}_k, \quad \mathbf{s}_{ik} = \mathbf{x}_k - \mathbf{x}_i,\\ \beta_{\mathrm{L}} &= \frac{\gamma}{\mu\,(1+\lambda^2)}, \quad \alpha_{\mathrm{L}} = \beta_{\mathrm{L}}\lambda,\end{aligned}$$

where γ is the gyromagnetic ratio, λ is the phenomenological Gilbert damping constant, $\mathbf{m}_i$ is the direction of spin magnetic moment with $|\mathbf{m}_i| = 1$, μ is the length of spin magnetic moment and $\mathbf{h}_i$ is the effective field. The spin-transfer torque (STT) is represented by the vector τ_i while b_1, b_2 and the components of the vector $\mathbf{g}$ are STT coefficients characteristic of the material. The vector $\mathbf{s}_{ik}$ is connecting atoms i and k (where $\mathbf{x}_k$ are atomic positions) and the summation is over atoms in a neighborhood of atom i, such that $k \neq i$. The following form of the effective field is considered:

Equation 2 — Effective field in the atomistic frame.

$$\mathbf{h}_i = \left(\sum_j J_{ij}\mathbf{m}_j\right) + \left(\sum_j \mathbf{D}_{ij}\times\mathbf{m}_j\right) + \mathbf{K}_{\mathrm{a}}\cdot\mathbf{m}_i + \mu\mathbf{h}_{\mathrm{e}},$$

where J_{ij} are constants of the Heisenberg exchange interaction between atoms i and j, vectors $\mathbf{D}_{ij}$ describe the Dzyaloshinskii-Moriya (DM) interaction, $\mathbf{K}_{\mathrm{a}}$ is the anisotropy tensor and $\mathbf{h}_{\mathrm{e}}$ is the external field. Summations are taken for all j, such that $j \neq i$, where any number of neighbours (i.e. non-nearest) can be included.

At the continuum scale, the magnetisation dynamics is modelled by the continuum version of the LLGS equation [6–8]:

Equation 3 — The Landau-Lifshitz-Gilbert-Slonczewski equation in the continuum.

$$\begin{aligned}\frac{\partial \mathbf{m}}{\partial t} &= -\beta_{\mathrm{L}}\mathbf{m}\times(\mathbf{h} + b_1\tau) - \alpha_{\mathrm{L}}\mathbf{m}\times(\mathbf{m}\times(\mathbf{h} + b_2\tau)),\\ \tau &= \mathbf{d}\cdot\nabla\mathbf{m},\end{aligned}$$

where $\mathbf{m}$ is the normalised magnetisation field ($|\mathbf{m}| = 1$) and β_{L} and α_{L} constants have the same meaning as in Eq. (1). In the case of the continuum approach, the following effective field is considered:

Equation 4 — Effective field in the continuum.

$$\mathbf{h} = \mathbf{A}_{\mathrm{e}} : \nabla\nabla\mathbf{m} + \nabla\cdot(\mathbf{D}_{\mathrm{e}}\times\mathbf{m}) + \mathbf{K}_{\mathrm{a}}\cdot\mathbf{m} + \mu\mathbf{h}_{\mathrm{e}},$$

where tensor $\mathbf{A}_\mathrm{e}$ contains the exchange interaction constants while the tensor $\mathbf{D}_\mathrm{e}$ includes the Dzyaloshinskii-Moriya interaction constants. The anisotropy tensor is represented by $\mathbf{K}_\mathrm{a}$ and $\mathbf{h}_\mathrm{e}$ is the external field. After a quick inspection of Eqs. (1) and (3), it is easy to see that both equations retain a similar mathematical form. It is straightforward to prove that both equations are invariant under the following transformations:

Equation 5 — Definition of interactions.

$$\mathbf{A}_\mathrm{e} = \frac{1}{2}\sum_{j,j\neq i} J_{ij}\mathbf{s}_{ij}\mathbf{s}_{ij},$$
$$\mathbf{D}_\mathrm{e} = \sum_{j,j\neq i} \mathbf{s}_{ij}\mathbf{D}_{ij},$$
$$\mathbf{d} = \mathbf{g}\cdot\sum_{k,k\neq i} \mathbf{s}_{ik}\mathbf{s}_{ik},$$

where $\mathbf{s}_{ij}$ is the vector connecting atoms i and j. Since the anisotropy term is local, the same anisotropy tensor $\mathbf{K}_\mathrm{a}$ is used in the continuum and the atomistic equations. The tensors $\mathbf{A}_\mathrm{e}$ and $\mathbf{D}_\mathrm{e}$ and the vector $\mathbf{d}$ are spatially homogeneous, i.e. these tensor/vector parameters do not depend on i.

The usual way of comparing the atomistic and the continuum models is by considering the continuum solution that coincides with the atomistic solution at all lattice points [10]. To find the difference between the models, the continuum solution is fixed and the asymptotic behaviour of the difference is found as a function of the atomistic lattice spacing. In the case of magnetism, a continuum magnetisation field $\mathbf{m}$ is considered in such a way that it coincides with the atomistic spin magnetic moments $\mathbf{m}_k$ at lattice positions, i.e. $\mathbf{m}(\mathbf{x}_k) = \mathbf{m}_k$, where $\mathbf{x}_k$ are atomic positions [7].

Most atomistic lattices of crystalline materials can be categorised as point-symmetric atomistic lattices, i.e. for each pair of atoms i and j, there exists an atom k which is of the same type as atom j, such that $\mathbf{s}_{ij} = -\mathbf{s}_{ik}$. The derivation of Eq. (5) can be found in Ref. [7], where it is shown that for the point-symmetric atomistic lattices, the atomistic and the continuum models are consistent with the error estimate

Equation 6 — Error estimation.

$$|\mathbf{h}_i - \mathbf{h}| = O\left(a^2\right),$$

as $a \to 0$, where a is the atomistic lattice spacing and the effective fields are indicated in Eqs. (2) and (4).

The proof of the consistency between the atomistic and the continuum spin-transfer torque terms, in Eqs. (1) and (3), respectively, can easily be derived by Taylor-expanding solution $\mathbf{m}(\mathbf{x}_i) = \mathbf{m}_i$, as was performed in Ref. [7]. This results in the following asymptotic behaviour:

Equation 7 — Asymptotic limit.

$$|\tau_i - \tau| = O\left(a^2\right).$$

In multiscale partitioned-domain atomistic-continuum coupling approach, the entire computational region is split into two subregions — the atomistic and the continuum domains. There is a "sharp" atomistic-continuum interface (i.e. a curve in two-dimensional systems, a surface in three-dimensional systems) between these two regions, as illustrated in Fig. 1.1.

The continuum region is discretised using the finite-difference method. The atomistic-continuum interface encapsulates a layer of finite-difference nodes and is linear between the neighbouring nodes. The coupling of the regions is implemented by constructing padding atoms and padding

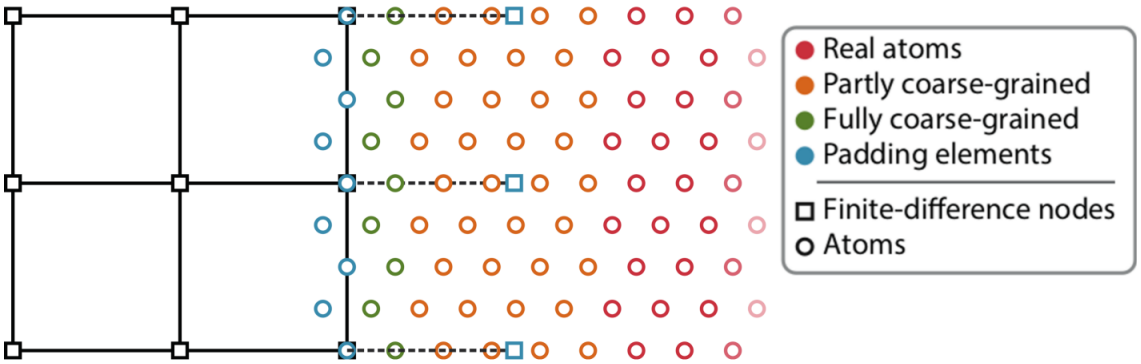

Figure 1.1: Schematic representation of the atomistic-continuum interface. The fully atomistic region is represented by real atoms while the fully micromagnetic region is represented by finite-difference nodes. In the interface, both regions mix each other.

nodes, which provide boundary conditions for the atomistic and the continuum regions, respectively. Real atoms interact with padding atoms, while the solution at padding atoms is obtained from the continuum region. Finite-difference nodes interact with padding nodes, while the solution at padding nodes is obtained from the atomistic region.

The solution at padding atoms is obtained by bilinear interpolation of the continuum solution with normalisation. The normalisation is introduced to preserve the length of spin magnetic moments. The solution at padding nodes is obtained by normalised weighted average of the atomistic solution inside the box with side Δx centred at the padding node, where Δx is the inter-node distance. For all atoms inside the box, the weight is assigned as the area of the intersection of the box with side a centred at the atom and the box with side Δx centred at the padding node. The normalisation is introduced to preserve the nodal length of the vector field solution.

To reduce the ghost-torque error for systems with non-nearest-neighbour atomistic interactions, the behaviour of atoms close to the atomistic-continuum interface is modified [7]. More specifically, J_{ij} and $\mathbf{D}_{ij}$ are modified for a finite set of atoms. Two types of modified atoms can be distinguished — the fully coarse-grained atoms, which interact only with the nearest neighbours, and the partly coarse-grained atoms, which also interact with non-nearest neighbours. The modification of J_{ij} and $\mathbf{D}_{ij}$ is performed such that the fully coarse-grained atoms completely isolate the atomistic-continuum interface. More details about the exact way of modifying the atoms can be found in Ref. [7].

To reduce the high-frequency wave-reflection from the atomistic-continuum interface, additional numerical damping is added to atoms close to the atomistic-continuum interface [6–8]. This damping acts as a low-pass filter for the waves travelling from the atomistic region to the continuum, as the solution is "attenuated" to an average solution within a certain window. Due to dispersive nature of the spin waves, the damping is non-linear and depends on time derivative of the solution. The analysis of the dynamics of the damping layer and the exact form of the modification can be found in [6].

The time stepping was performed simultaneously for all degrees of freedom, i.e. all finite-difference nodes and atoms. The solution at padding atoms/nodes at a certain time step is obtained from the continuum/atomistic solution of the same time step. Mid-point [11] and Depondt [12] methods were used to solve Eqs. (1) and (3) in time.

1.3 License

The μASD code is developed by the Division of Materials Theory, in the Department of Physics and Astronomy at the University of Uppsala, Sweden. The copyright of the code is help by the developers but the program is open for use and distribution according to the fXXX license. Further information concerning the license and contact information of the developers may be found on the UppASD webpage: `http://www.physics.uu.se/UppASD`. The current version of the code (1.0) is still under active development.

1.4 Installation

The source code is distributed along with documentation as part of UppASD and a growing set of examples. To install, perform the following actions.

1. Obtain the code, by extracting the downloaded archive (or by pulling from the git repository)

   ```
   tar xvzf  UppASD_dist.tar.gz
   ```

   or in the case of git

   ```
   git clone github:uppasd
   ```

2. Generate the dependencies needed for compiling the code

   ```
   make deps
   ```

3. (Optional) Perform a system check for available compiler profiles

   ```
   make probe
   ```

4. Compile the code with the selected compiler profile

   ```
   make <profile>
   ```

   where `<profile>`is the name of the profile, i.e. `ifort`, `ifort-cuda`, `gfortran`, `gfortran-osx`, and so on. Eg. `make ifort`

5. (Optional) Test the compiled program against a selection of realistic runs

   ```
   make tests
   ```

In addition to the source files, the μASD code as part of UppASD distribution also contains several examples (in the directory `examples_revision_controlled/multiscale`), documentation, including this file (in `docs/`) and routines and reference data (`codeTester/`) for validating the installation of the μASD code as part of the UppASD program.

1.5 Principles of the Code

When run, μASD essentially goes through three stages as in UppASD:

1. Initialization: the system is set up.
2. Initial phase: an optional stage in which the system is brought into thermal equilibrium, with limited data sampling.
3. Measurement phase: the system is evolved in time, with complete data sampling being made.

During the initialization phase, all the parameters necessary to describe the system of interest, such as its geometry, dimensions, exchange couplings and boundary conditions, are set up. In addition, the initial phase also sets the simulation parameters, such as the number of simulation steps to record data over, which SDE solver to use, and the temperature at which the simulation should be run.

The initial phase, which is optional, is typically performed in order to bring the system into thermal equilibrium, so that the data recorded in the measurement phase is for a thermalized system. Obviously, if one is interested in out-of-equilibrium dynamics, then there is no need to perform this phase. In the current version of the code, the initial phase can only be performed using Spin Dynamics (SD).

During the measurement phase, the data sampling is performed. Simulations can be run only in SD mode.

μASD

MULTISCALE ATOMISTIC SPIN DYNAMICS

2. Input files

μASD accepts some parameters from command line, but the entire setup is specified in a single file, called `multiscale.conf` (if not specified otherwise), that should be placed in the working directory. The output is written in a separate folder called `output`. As FORTRAN provides no standard way of creating a directory, the user is responsible of creating it.

2.1 Configuration file

The configuration is contained in a simple text file. There are two kind of parameters; most of them are variables, specified by a name followed by a value (the value may consist of various words, for example, when the variable is a vector or a list).

The other kind of parameter is the list. Every line after the list header that matches the format of its entries defines an element of the list. Lists accept the optional `filename` tag at the end of their header, that must always be followed by file name. If this tag is present, entries of the list are read from the specified file instead of the main configuration file. This is recommended for large lists, as it makes reading the configuration easier.

Text after the symbol # is ignored, allowing to create comments or, if desired, UNIX shebang headers, to make a configuration file executable.

This is the list of parameters accepted by the configuration file:

atomistic_shape [file <filename>]

Mandatory, list. The shapes that define the atomistic region.This field can be followed by one or many shape specifiers, see subsection 2.1.1.

format (P | B | M)

Optional, default is set to P. If P, `*.dat` files are saved as plain text files. B specifies `*.dat` binary files and M represents MATLAB-compatible `*.m` files. Plaintext and binary files are valid UppASD input files. Plaintext is easier to work with but generates large files. Binary files are compact and faster to read but hard to modify. MATLAB format is not valid as UppASD input, but is useful for debugging and visualisation purposes.

coarse_grained_width <nonnegative real>

Mandatory. Specifies the width of the fully coarse-grained region, the width is measured

from the padding atoms. All the atoms closer than the parameter to a padding atom are considered as coarse-grained.

damping_band_width <positive real>

Mandatory. Specifies the width of the damping band. The width is measured from the padding atoms. If 0 is specified, no damping band will be constructed.

damping_band_window_size <width> <height> <depth>

Mandatory. Specifies the averaging window size for the damping band. The value of this parameter should depend of the dimensionality, the lattice and the interaction distance. There is a lower bound for this window size, as if there are less than 3,6 or 10 atoms for dimensionality 1,2 and 3 respectively, the calculation of the damping coefficients would involve inverting a singular matrix. MUASD generates NaN's in the damping band coefficients file in this case.

dimension (1 | 2 | 3)

Mandatory. Specifies the dimensionality of the simulation.

exchange_atoms [unitcell] [file <filename>]

Mandatory, list. The exchange between the atoms. See exchange specifiers subsection 2.1.2.

exchange_coarse [unitcell] [file <filename>]

Mandatory, list. The exchange between the fully coarse-grained atoms. See exchange specifiers subsection 2.1.2.

finitediff_boxes <x> <y> <z>

Mandatory. The number of finite-difference boxes in the direction of each dimension. A box in three dimensions is a cube that have eight finite-difference grid points as its corners. This parameter is used together with the universe size to define Δx.

continuous_exchange_coef <real>

Mandatory. Exchange parameter for finite-difference region.

continuous_moment_magnitude <real>

Mandatory. The magnitude of the moments in the continuous region. The moments of these nodes are initialised to $(\text{magnitude}, 0, 0)$. In order to set up different initial moments, the user can to modify `output/moments.dat`.

hole_shape [file <filename>]

Optional, list. Defines holes in the atomistic region. This field can be followed by one or many shape specifiers, see subsection 2.1.1.

atom_lattice_spacing <positive real>

Mandatory. This parameter is used to approximate the side of the Wigner-Seitz cell around each atom. Each atom is enclosed by a cube of side the value of this parameter. The approximation is used to calculate weights for averaging padding atoms and interface nodes and in the coefficients used in the damping band. The optimal value depends on the lattice.

padding_width <positive real>

Mandatory. Specifies the width of the padding region, the width is measured from the boundary. Padding atoms serve as reference while placing fully coarse-grained and partially coarse-grained atoms, as well as damping atoms. See section 4.2 for more details. The value of this parameter should be the interaction radius of fully coarse-grained atoms, plus a small value to prevent numerical issues. This value ensures that all fully coarse-grained atoms have a padding atom for all possible interaction they could have.

part_coarse_grained_width <nonnegative real>

Mandatory. Specifies the width of the partially coarse-grained region. Those atoms that are closer to a padding atom than the distance `coarse_grained_width` + `part_coarse_grained_width` is considered to be partially coarse-grained.

periodic_boundary (T | F) (T | F) (T | F)

Mandatory. Specifies if the boundary is periodic or not, T indicates a periodic boundary, F indicates an open boundary.

universe_size <width> <height> <depth>
Mandatory. Specifies the size of the space that the simulation takes place in.

unitcell_atoms <width> <height> <depth> [file <filename>]
Mandatory, list. The width, height and depth are the size of the unit cell. Then a list of the atoms should follow. See Unit cell atoms in subsection 2.1.3.

anisotropy <anisotropy parameters> [<anisotropy parameters>]
Optional, list, repeatable. Specifies regions affected by an anisotropy. This line must be followed by shape specifiers (subsection 2.1.1). The `anisotropy parameters` are defined as:
`(uniaxial|cubic|both <ratio>) K1 K2 ex ey ez`
The first parameter is the anisotropy type. `K1` and `K2` are two real coefficients and `ex ey ez` is the direction vector $\vec{e}$. It must be specified twice if the flag `mult_axis` is present in `inpsd.dat`. This parameters correspond to what appears in UppASD's manual, please refer to it for more details. [9] This option is offered as a convenient manner to specify anisotropies, but inputed values are kept unchanged in the output file. If two regions of different anisotropies overlap, the one specified first will take precedence.

2.1.1 Shape specifiers

`hole_shape`, `atomistic_shape` and `anisotropy` can be followed by a list of shapes. These are the recognised shapes:

box <x> <y> <z> <width> <height> <depth>
Specifies a box with the same dimension as the space. The parameters X Y and Z determine where the corner closest to the origin of the coordinate system will be placed. Width Height and Depth determine the extension of the box in the X, Y and Z directions, respectively. Even if the dimension is less than 3, the user must specify all 3 coordinates. The box considers all the coordinates when it decides if a point is inside it or not.

sphere <x> <y> <z> <radius>
Specifies a sphere of the given radius with centre in (X,Y,Z). As with the box, the user must specify all 3 coordinates regardless of the dimension.

2.1.2 Exchange specifiers

The syntax for the exchange list header is the following:

`exchange_atoms [unitcell] [file <filename>]`

This header can be followed by one or more inter-atomic exchange specifiers. If the flag `file` is present, the specifiers will be read from the given file. Each specifier should be in a separate line, following this format:

`<Atom type 1> <Atom type 2> <x> <y> <z> <exchange>`

The `Atom type` fields serve to filter which type of atoms each rule affects. The `exchange` field specifies the exchange value in mRy. The meaning of `x`,`y` and `z` depends on the flag `unit cell`

If the flag `unitcell` is not present in the header, `x`,`y` and `z` form a direction vector, representing the relative position between two atoms. For example:

```
1 exchange_atoms
2 1 1  0.5  0.0 0.0 1.23
3 1 1  0.0  0.4 0.0 2.22
4 1 1 -0.5  0.0 0.0 1.23
5 1 1  0.0 -0.4 0.0 2.22
```

In this setup, atoms of type 1 are at a distance of 0.5 units each other in the X axis and are aligned with respect to the Y axis will have an exchange parameter of 1.23. Those aligned with respect with the X axis and whose distance is 0.4 will have an exchange parameter of 2.22.

If the `unitcell` tag is present, the exchange is specified depending on the relative position of the unit cells in which each atom is, instead of the atom position. For example:

```
exchange_atoms unitcell
1 1   0  0 0 2.22
1 1   1  0 0 1.23
1 1  -1  0 0 1.23
1 1   0  1 0 1.23
1 1   0 -1 0 1.23
```

In this setup, atoms that are in the same unit cell receive an exchange parameter of 2.22. Atoms in neighbouring unit cells (using 4-connected neighbourhoods) receive an exchange constant of 1.23. Note that the offsets in this case are integer-valued.

2.1.3 Unit cell atoms

The syntax for the unit cell header is as follows:

`unitcell_atoms <width> <height> <depth> [file <filename>]`

The `width`, `height` and `depth` values refer to the size of the unit cell. Following this header the user can describe how atoms in the unit cell are.

Each atom is defined by one line in this format:

`<Atom type> <x> <y> <z> [/] <moment magnitude> <mx> <my> <mz>`

`Atom type` indicates the type of the atom, used in the exchange specification (see subsection 2.1.2). `x`, `y` and `z` are the coordinates of the atom inside the unit cell. These coordinates are in the interval [0,1]. The user should avoid placing atoms close to the corners of the cell. This is because otherwise atoms could be placed at the atomistic-continuum boundary, which should not happen. μASD will automatically ignore atoms close to the edges, but it may generate unexpected results if the user is not aware of this. Optionally, the user may type a slash(`/`) after the coordinates. This helps with readability of the line. The value of `moment magnitude` is length of the magnetic moment vector. Parameters `mx`, `my` and `mz` are components of a vector parallel to the moment of the atom. There is a constrain in the norm of the vector specified by the user, it can never be shorter than 10^{-3} to prevent numerical issues.

In order to calculate the magnetic moment of the atom, the vector (`mx`, `my`, `mz`) is normalised and multiplied by `moment magnitude`.

The whole generation process is discussed in section 4.1.

2.2 Command line

In the command line, the typical help (`--help` or `-h`) and version (`--version` or `-v`) options are available. Furthermore, it is possible to get a brief explanation of the input file fields using the help option followed by one or several parameter names. For example:

```
$ ./muasd -h dimension universe_size
option dimension
   Syntax: dimension (1 | 2 | 3)
   Specifies the dimensionality of the simulation.
option universe_size
   Syntax: universe_size <x> <y> <z>
   Specifies the size of the space that the simulation takes place in.

```

It is also possible to select an alternative configuration file (other than `multiscale.conf`). This is done by passing the name of the configuration file as an argument.

2.3 CheckPositions

CheckPositions is a MATLAB program bundled with μASD. After running μASD you can change your working directory to `./output` in MATLAB and run there checkPositions. In order to run checkPositions from any directory, use the command `addpath('path/to/checkpositions');`.

To run checkPositions it is required that the format of the output is set to MATLAB (`format m`, See section 2.1).

In 1D, checkPositions adds an offset in the `Y` coordinate of the plot for different kinds of elements. This eases reading atom indices and prevents atoms from occluding each other. See an example in Figure 2.1

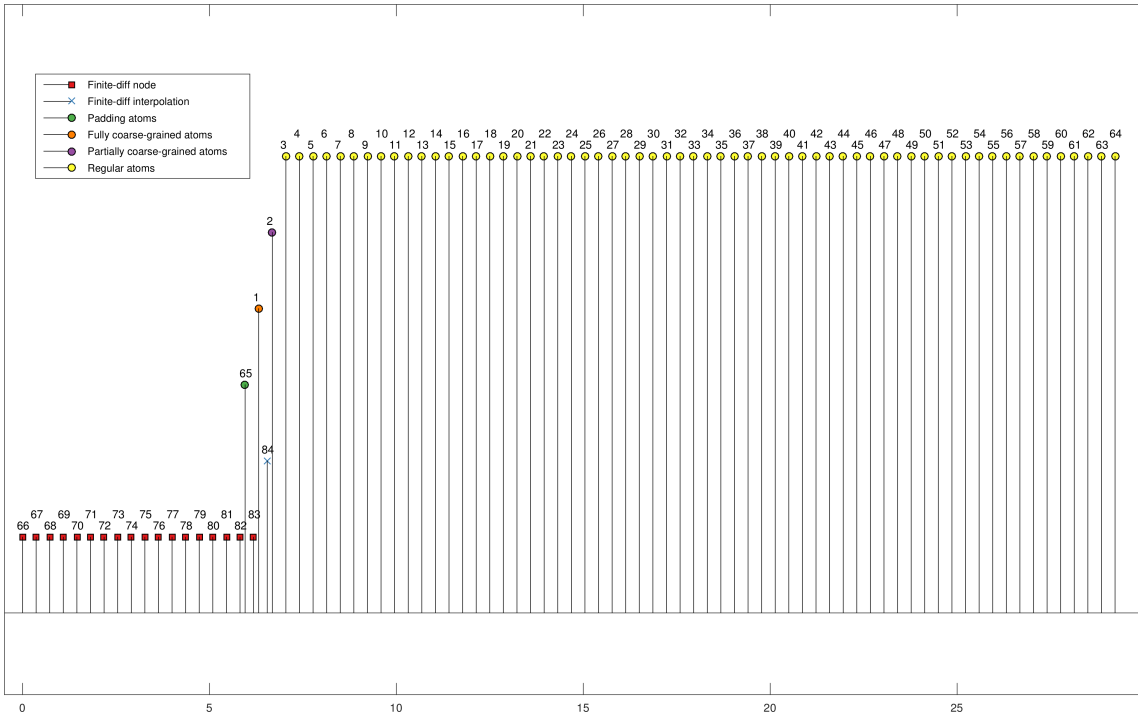

Figure 2.1: 1D Example, checkPositions plot.

μASD

MULTISCALE ATOMISTIC SPIN DYNAMICS

3. Basic examples

Some typical examples of simulations using the μASD software is discussed here in rather great detail.

3.1 1D strip

This example shows how to set up a 1D strip with an atomistic and a continuum region.

```
dimension 1
coarse_grained_width 0.37
part_coarse_grained_width 0.37
damping_band_width 5.9
padding_width 0.37
periodic_boundary F F F
universe_size 29.4516 0.0 0.0
finitediff_boxes 81 1 1
atom_lattice_spacing 0.3636
continuous_exchange_coef 0.3134398421312568
continuous_moment_magnitude 5.0
damping_band_window_size 4.3642 4.3642 4.3642

format m

unitcell_atoms 0.3636 1.0 1.0
1 0.5 0.0 0.0 5.0 1.0 0.0 0.0

atomistic_shape
sphere 29.4516 0.0 0.0 23.27

exchange_atoms
1 1  0.3636 0.0 0.0 2.3708629550495477
1 1 -0.3636 0.0 0.0 2.3708629550495477

exchange_coarse
1 1  0.3636 0.0 0.0 2.3708629550495477
1 1 -0.3636 0.0 0.0 2.3708629550495477
```

The first line specifies the dimensionality, this is a 1D simulation.
Lines 2,3,4 and 5 let the user specify the thickness of each atomistic layer.
Line 6 specifies that periodic boundary conditions are not used for either direction.
Line 7 sets the size of the universe. Only the first coordinate is considered, as the example is 1-dimensional.
Line 8 defines the number of boxes in which the universe will be split. μASD first covers the whole space with finite difference boxes, then turn some of those boxes inside atomistic boxes. Later finite difference nodes are placed in the corners of finite difference boxes and the unit cell is tiled inside atomistic boxes. The finite difference lattice spacing is the element-wise division between `universe_size` and `finitediff_boxes`, in this case 0.3636.
Line 9 sets the atomistic lattice spacing. As the unit cell contains only one atom, the interaction distance will also be 0.3636 (specified in line 10). For more details on how this parameter is used see section 4.4.
Line 10 specifies the exchange parameter for the finite-difference region. Note that this is a parameter in the PDE, i.e. nodes interact with the value scaled by Δx^2.
Line 11 establishes the magnetic moment magnitude in the finite-difference region. Remember that all magnetic moments are pointing towards $+x$ unless `moments.dat` is modified.
Line 12 defines an averaging window size for calculating the damping band coefficients.
Line 14 will cause the output to be compatible with GNU Octave/MATLAB. This is required for plotting the setup checkPositions.
Line 16 starts the `unitcell_atoms` list. Its parameter is the dimension of the cell, in this case 0.3636, same as the finite-difference lattice spacing. The next line adds one atom to the unit cell, of type 1, located at $x = 0.5$ and magnetic moment magnitude 5 towards $+x$.
Line 19 contains the header of the `atomistic_shape` list. In line 20 we have the single element that this list holds, in this case a sphere centred in $(29.4516, 0, 0)$ and of radius 23.2741. The centre corresponds to the universe size, so it will be in one end of the space. The radius spans $23.27/0.3636$, a bit more than 63 unit cells.
Line 22 defines the interactions between atoms. We are using position offsets instead of unit cell offsets in this case. Lines 23 and 24 specify interaction rules from an atom A to an atom B. Note that the interactions are not symmetric. The first two numbers select the type of atoms A and B respectively. The next three numbers filter atoms by their relative positions. Let's call these numbers d_x,d_y and d_z, then, only if $B - A \approx (d_x, d_y, d_z)$ the interaction will occur. The last value is the exchange parameter, in mRyd.
Lines 26, 27 and 28 work in the same way. These lines represent the exchange used for fully coarse-grained atoms.

3.2 2D example

This example shows how to set up a 2D simulation with an atomistic and a continuum region.

```
1  dimension 2
2  coarse_grained_width 1.1
3  part_coarse_grained_width 1.1
4  damping_band_width 0.0
5  padding_width 1.1
6  periodic_boundary F F F
7  universe_size 40.0 25.0 1.0
8  finitediff_boxes 10 6 1
9  atom_lattice_spacing 1.0
10 damping_band_window_size 12.0 12.0 12.0
11 continuous_exchange_coef 1.0
12 continuous_moment_magnitude 1.0
13 format m
```

```
14
15 atomistic_shape
16 box 25.0 -1.0 -1.0 100.0 25.0 2.0
17 box 19.5 -1.0 -1.0 5.0 10.1 2.0
18
19 unitcell_atoms 1.0 1.0 1.0
20 1 0.5 0.5 0.0 1.0 1.0 0.0 0.0
21
22 exchange_atoms
23 1 1  1.0  0.0 0.0 1.0
24 1 1 -1.0  0.0 0.0 1.0
25 1 1  0.0  1.0 0.0 1.0
26 1 1  0.0 -1.0 0.0 1.0
27
28 exchange_coarse
29 1 1  1.0  0.0 0.0 1.0
30 1 1 -1.0  0.0 0.0 1.0
31 1 1  0.0  1.0 0.0 1.0
32 1 1  0.0 -1.0 0.0 1.0
```

See in Figure 3.1-a) how the setup will be generated. The squares in the figure are finite-difference nodes, the crosses are also finite-difference nodes but there moments will be interpolated from the atoms that surround them. Thus those interpolated nodes work as boundary conditions for the other finite-difference nodes. The yellow points are real atoms, inside the atomistic region. As we approach the continuum we see purple (partially coarse-grained) atoms. There is one layer of them, since `part_coarse_grained_width` roughly exceeds the atomistic lattice spacing. The next row, in orange, represents fully coarse-grained, again only one layer is generated. The green layer is already outside the atomistic region and represents padding atoms. Those atoms are interpolated by their closest four finite difference nodes (red squares in the picture). Inside the atomistic region there is one layer of finite difference interpolation nodes. These nodes are interpolated with their surrounding atoms, see more details in section 4.3.

If boundary condition is used along the vertical direction then the setup becomes as in Figure 3.1-b). We can see that the continuum nodes disappear at the top of the setup, to avoid duplicate nodes. The continuum nodes that are at the bottom of the setup will be linked to those nodes that are closest to the top boundary in this setup.

The green strip of atoms that appeared at the top-centre of the setup is made of padding atoms, these appeared due to the orange (fully coarse-grained) atoms at the bottom boundary.

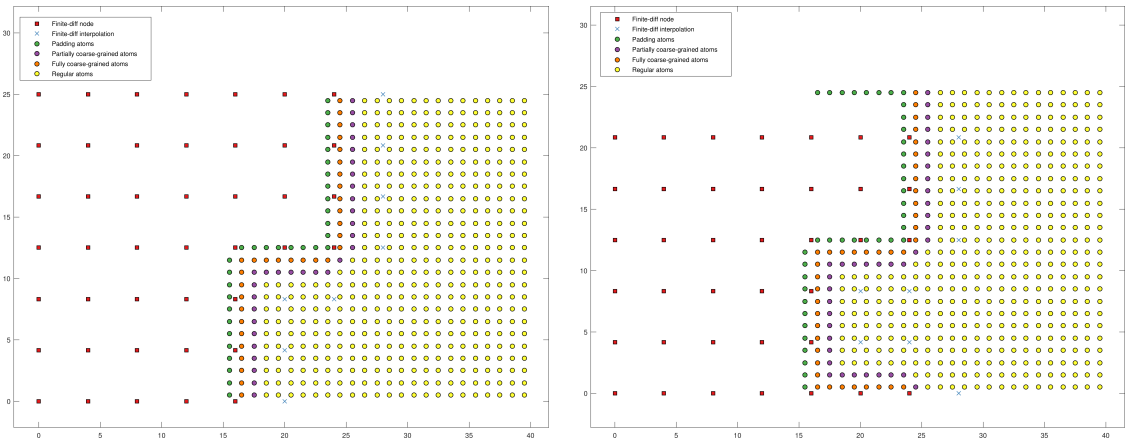

Figure 3.1: a) 2D example with open boundary conditions (left panel). b) 2D example with periodic boundary condition in the vertical direction (right panel).

3.3 3D example

This example shows a 3D simulation with an atomistic spherical region in the centre, surrounded by the continuum region. In Fig. 3.2 is shown a 3D distribution of the atoms in the interval $[25,50]$ along the x, y and z cartesian coordinates.

```
1  dimension 3
2  coarse_grained_width 1.1
3  part_coarse_grained_width 1.1
4  damping_band_width 5.9
5  padding_width 1.1
6  periodic_boundary F F F
7  universe_size 50 50 50
8  finitediff_boxes 25 25 25
9  atom_lattice_spacing 1.0
10 damping_band_window_size 6.1 6.1 6.1
11 continuous_exchange_coef 0.2936
12 continuous_moment_magnitude 7.63
13 format M
14
15 atomistic_shape
16 sphere 25.0 25.0 25.0 15.0
17
18 hole_shape
19 sphere 25.0 25.0 25.0 5.0
20
21 unitcell_atoms 1.0 1.0 1.0
22 1 0.5 0.5 0.5 5.0 1.0 0.1 0.2
23
24 exchange_atoms
25 1 1  1.0 0.0 0.0 0.2936
26 1 1 -1.0 0.0 0.0 0.2936
27 1 1  0.0 1.0 0.0 0.2936
28 1 1  0.0 -1.0 0.0 0.2936
29 1 1  0.0 0.0 -1.0 0.2936
30 1 1  0.0 0.0 1.0 0.2936
31
32 exchange_coarse
33 1 1  1.0 0.0 0.0 0.2936
34 1 1 -1.0 0.0 0.0 0.2936
35 1 1  0.0 1.0 0.0 0.2936
36 1 1  0.0 -1.0 0.0 0.2936
37 1 1  0.0 0.0 -1.0 0.2936
38 1 1  0.0 0.0 1.0 0.2936
```

This example also includes a hole in the middle of the space. The hole is specified in lines 15 and 16 as a sphere of centre $(25,25,25)$ and radius 5.0. Also notice that the interaction rules for the Z axis have been added for both coarse-grained atoms (from line 32) and regular atoms (from line 24). The rest of the setup is very similar to the previous one in section 3.2.

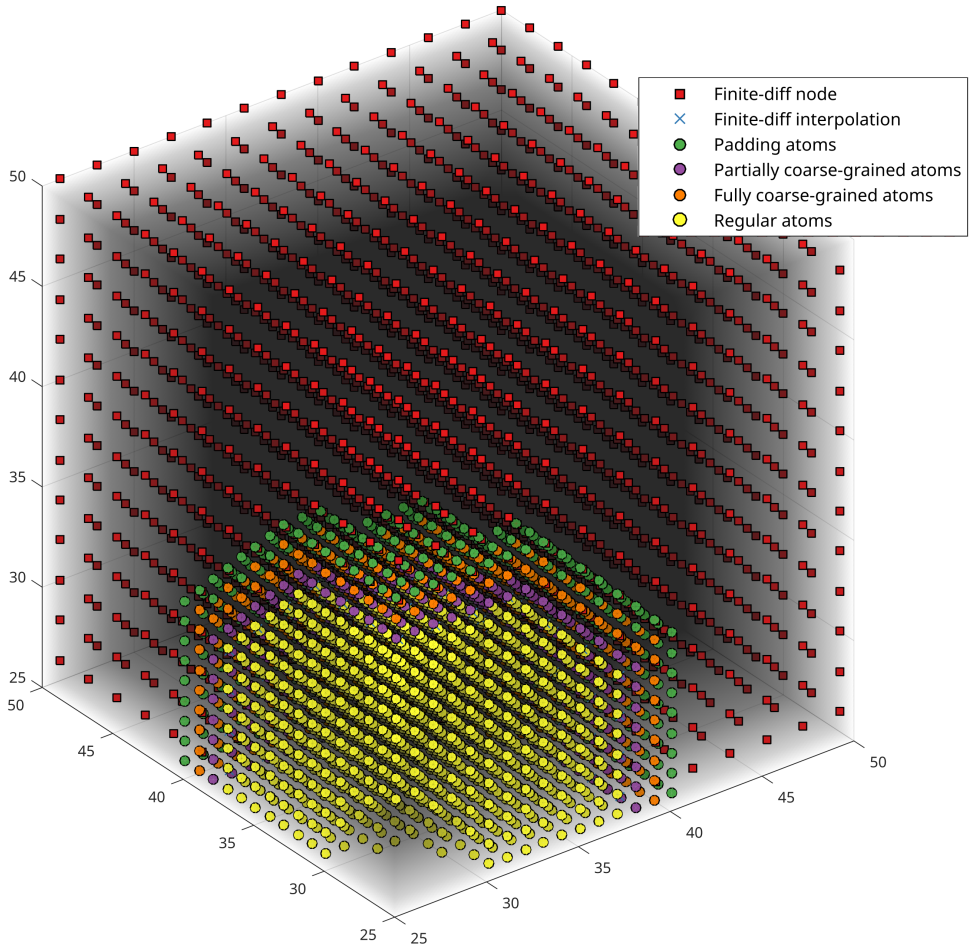

Figure 3.2: 3D example. For better visualisation only atoms in the interval $[25, 50]$ are plotted.

μASD

MULTISCALE ATOMISTIC SPIN DYNAMICS

4. Implementation details

4.1 System generation

At first, the finite-difference mesh is created. The boxes within the mesh are marked as either atomistic or continuum. A box is marked as continuum if one of its corners is inside an atomistic shape. Thereafter the atoms are generated from the unit cell. The unit cell is repeated over the entire universe and atoms that are inside atomistic boxes are kept, while other atoms are disregarded. Note that there is a small region between atomistic and the continuum domains where atoms are always discarded. This is done to avoid placing atoms exactly on the boundary between the atomistic region and the continuum region. This should not be allowed because, as continuum nodes are always placed in the boundary, atoms overlapping with nodes may appear. This would generate a region that is described by both models, which may not perfectly agree in the result.

The next step is generating padding atoms. These atoms work as boundary conditions for the atomistic part. They are generated exactly as the real atoms, however they are added to the system if they are in the continuum region and if the distance to the closest regular atom is small enough. This distance can be specified in the input file using the keyword `padding_width`.

Related routines

```
multiscale_setup ..... multiscale_setup.f90
├─ createFiniteDiffMesh ..... finitedifference.f90
│  └─ (auxiliar functions) ..... finitedifference.f90
└─ createAtoms ..... multiscale_setup.f90
   ├─ generateAtoms ..... atomgenerator.f90
   ├─ generatePaddingAtoms ..... atomgenerator.f90
   ├─ extractFromUnitcellLocation ..... atomgenerator.f90
   ├─ generateFiniteDifferenceAtoms ..... atomgenerator.f90
   └─ (auxiliar functions) ..... atomgenerator.f90
```

4.2 Atom linking

The non-padding atoms are divided into three separate regions, the coarse-grained atoms, the partially coarse-grained atoms and the real atoms. The coarse-grained atoms are selected from the generated atoms if their distance to a padding atom is less than `coarse_grained_width`. The partially coarse-grained atoms are selected by choosing those atoms that are closer to a padding atom than `part_coarse_grained_width` but are not closer than `coarse_grained_width`. The remaining atoms are the real atoms. For details about these three atom categories see [7].

After the regions have been initialised, the linking takes place. The coarse-grained atoms are linked only to their nearest neighbours using an exchange derived from `exchange_coarse`. The linkage in real atoms is done as specified by the user with `exchange_atoms`. Partially coarse-grained atoms receive their exchange coefficients in function of their position and the exchange used in fully coarse-grained and real atoms, thus providing a smooth transition between these two regions [7].

Related routines

```
multiscale_setup ..... multiscale_setup.f90
├─ createRegions ..... multiscale_setup.f90
└─ setupLinks ..... multiscale_setup.f90
   ├─ createFiniteDiffLinks ..... finitedifference.f90
   └─ createExchangeMatrix ..... atomlinker.f90
      └─ (auxiliar functions) ..... atomlinker.f90
```

4.3 Interpolation

There are finite-difference nodes placed inside the atomistic region, these are only used as boundary conditions for the continuum part. These nodes appear as a blue cross in checkPositions (see Figure 3.1-a)). The magnetic moment for these nodes is obtained by interpolation of the atoms that surround them. The interpolation is just a weighted average of the atomistic values. To calculate the weights for the atoms, a bounding box, of side the finite-difference spacing, is placed around the finite-difference node. The volume of the intersection between the atom's Wigner-Seitz cell and the bounding box is taken as the weight.

The intersection between the bounding box and the Wigner-Seitz cells is done by an iterative partitioning method: Let B be a box of equal to the bounding box. If not all corners of B have as one of their closest neighbours the same node, the box is split in every dimension, yielding 2 subboxes in 1D, 4 in 2D and 8 in 3D. The process is repeated taking each subbox as B, until all nodes share a nearest-neighbour or a depth limit is reached. The volume of the subboxes is finally divided and accumulated for each closest neighbours of each corner. This approximation is accurate with a small depth limit, but is still the slowest point in μASD for a large enough interpolation.

The padding atoms are interpolated from the finite difference-nodes that enclose them in a box. That is, the corners of the finite-difference box in which the padding atom is located.

Related routines

```
multiscale_setup ..... multiscale_setup.f90
└─ setupInterpolationWeights ..... multiscale_setup.f90
   ├─ createFiniteDiffInterpolationLinks ..... atomlinker.f90
   │  └─ iterativeAreaCoefficients ..... areacoefficients.f90
   └─ createPaddingInterpolationWeights ..... atomlinker.f90
```

4.4 Damping band

The damping band is a low-pass filter that damps high frequency perturbations travelling from the atomistic to the continuum region. The damping band can be applied as an additional term in the effective field in the following way:

Equation 8 — Damping band in the LLG equation.

$$\frac{\partial \vec{m}}{\partial t} = \beta_L \vec{m}_i \times \vec{H}_i - \alpha_L \vec{m}_i \times \left(\vec{m}_i \times \vec{H}_i\right) - \gamma_{Di} \vec{m}_i \times \left(\vec{m}_i \times \vec{m}_{Ai}\right),$$

$$p_{ij} = \left(\frac{r_i^r}{r_i^r + r_j^p - a}\right)^2,$$

$$\gamma_{Di} = g_D \cdot p_{ij} \sqrt{\left|\frac{d}{dt}\vec{m}\right|}.$$

For more details refer to the paper of the method [7]. The coefficient g_D is a parameter (`dband`) the user can directly specify in UppASD, in the file `inpsd.dat`. The vector m_{A_i} is the normalized result of a weighted sum of neighbouring vectors. The position-dependent term p_{ij} and the coefficients for calculating m_{A_i} are evaluated in μASD and written into the file `dampingband.dat`.

In a similar way as the interpolation, damping band coefficients depend on the Wigner-Seitz cell area. In this case finite-difference nodes are also assigned an area, since the interpolation involves

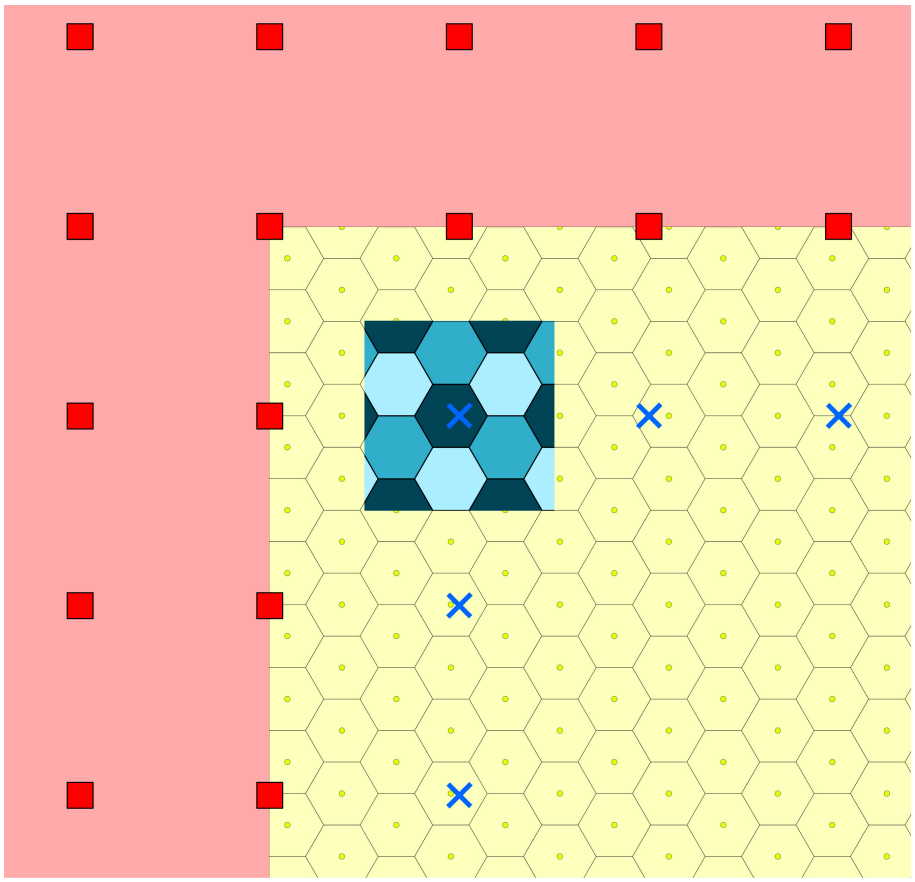

Figure 4.1: Two-dimensional bcc lattice. In patterned blue, area that each atom contributes to the interpolation node in the center. In yellow atoms and their Wigner-Seitz (Voronoi) cell.

magnetic moments from both domains (see Fig. 4.1). However, the number of atoms affected by the damping band is larger than in the interpolation, and the computation of voronoi areas as described in the interpolation is computationally too expensive. In order to reduce calculation time, μASD approximates the areas, taking a square of side `atom_lattice_spacing` for each atom instead of its voronoi cell (see Fig. 4.2). Continuum nodes are circumscribed by boxes of side Δx. This approximation comes at a cost: not all lattices are ideally represented for the damping band. Cubic lattices are resolved almost perfectly, while some lattices cause holes or intersections between boxes, introducing some error.

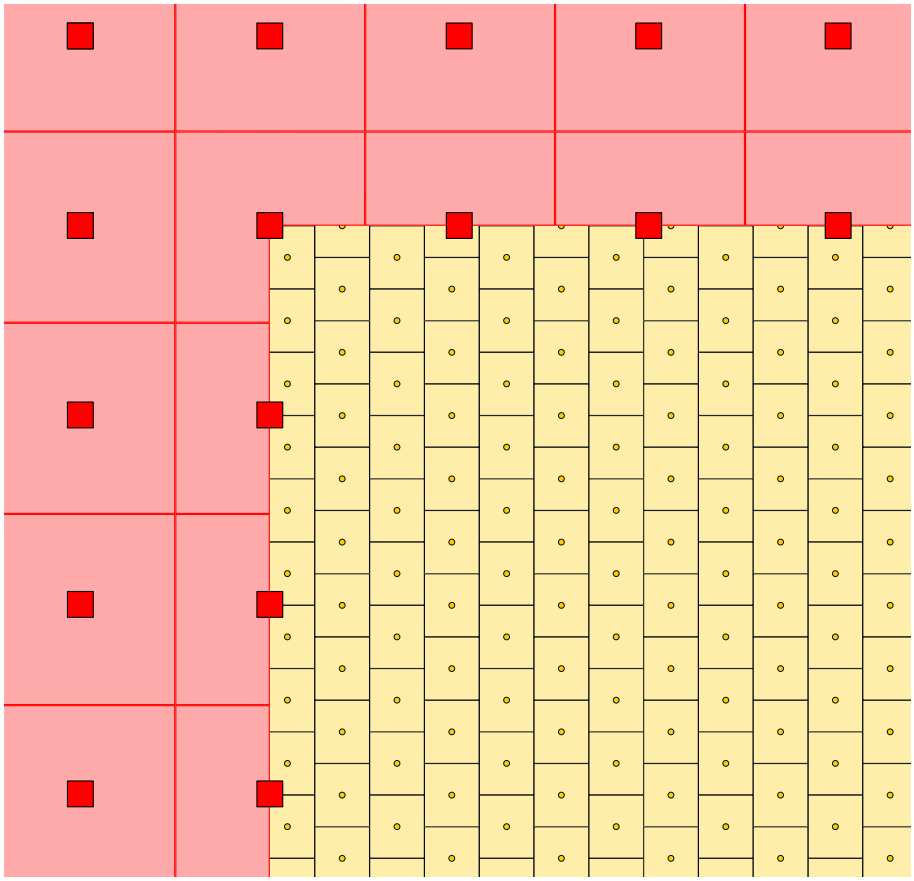

Figure 4.2: Boxes approximation in the same setup as Figure 4.1. In yellow, boxes approximating Wigner-Seitz cells for each atom. In red areas corresponding to each finite-difference node. Note that atomistic and continuum cells never have effect outside their own domain.

Related routines

```
multiscale_setup ..... multiscale_setup.f90
└─ setupDampingBand ..... multiscale_setup.f90
   ├─ dampingPositionalCoefficients ..... dampingband.f90
   └─ dampingAreaCoeff ..... dampingband.f90
      └─ boxesAreaCoefficients ..... areacoefficients.f90
```

μASD

MULTISCALE ATOMISTIC SPIN DYNAMICS

Bibliography

[1] B. Skubic, J. Hellsvik, L. Nordström, and O. Eriksson
J. Phys.: Condens. Matter **20**, 315203 (2008).

[2] V. P. Antropov, M. I. Katsnelson, B. N. Harmon, M. van Schilfgaarde, and D. Kusnezov
Phys. Rev. B **54**, 1019 (1996).

[3] J. L. García-Palacios and F. J. Lázaro
Phys. Rev. B **55**, 1006 (1997).

[4] R. E. Watson, M. Blume, and G. H. Vineyard
Phys. Rev. **181**, 811 (1969).

[5] O. Eriksson, A. Bergman, L. Bergqvist, and J. Hellsvik
Atomistic Spin Dynamics - Foundations and Applications.
Oxford University Press, Oxford, 1st edition (2017).

[6] M. Poluektov, O. Eriksson, and G. Kreiss
Commun. Comput. Phys. **20**, 969 (2016).

[7] M. Poluektov, O. Eriksson, and G. Kreiss
Comput. Methods Appl. Mech. Engrg. **329**, 219 (2018).

[8] É. Méndez, M. Poluektov, G. Kreiss, O. Eriksson, and M. Pereiro
Phys. Rev. Research **2**, 013092 (2020).

[9] See *Uppsala Atomistic Spin Dynamics User Guide.*

[10] M. Luskin and C. Ortner
Acta Numerica **22**, 397 (2013).

[11] M. d'Aquino, C. Serpico, and G. Miano
J. Comp. Phys. **209**, 730 (2005).

[12] Ph. Depondt and F. G. Mertens
J. Phys.: Condens. Matter **21**, 336005 (2009).

μASD

MULTISCALE ATOMISTIC SPIN DYNAMICS

Index