Exercise 2, TFY4235 Computational physics

Martin Johnsrud

Introduction

This report documents the simulation of magnons, as described in the exercise text [1]. Using Heun's method, the Landau-Lifshitz-Gilbert equations for a classical, interacting ensemble of spins are integrated in time. The algorithm is first used to simulate one spin, for which the exact solution is known. Then, the formation of collective modes, magnons, are simulated, and difference between ferromagnetic and antiferromagnetic materials and magnetization is investigated.

Theory

Units

The Hamiltonian and the equations of motion as given in [1] gives natural units for this problem,

- Energy: $[\mathcal{H}] = [J]$
- Magnetic field and magnetization: $[\vec{B}] = [\vec{M}] = [J/\mu]$
- Anisotropy: $[d_z] = [J]$
- Time : $[t] = [\mu/\gamma J]$

The dynamical variables of the problem, $\vec{S}$, are assumed to be dimensionless, and normalized to $|\vec{S}| \in [0, 1]$. The defining dimensionful constants of the system are thus the coupling J, the magnetic moment μ and the gyromagnetic ratio γ. These units are used throughout the exercise, including in all figures. Physical values for a given, actual system is then dependent on its values for J, μ and γ. As $\vec{S}$ is dimensionless, J has units of energy and is given by $J = J' S_0^2$, where S_0 is the maximum value of the spins, and J' is the coupling appearing together with the dimensionful spins. In this paper as well as in the code it documents, all equations and quantities are given in these units, including the Hamiltonian (Equation 1) and the equations of motion (Equation 2 and Equation 3).

Indices

For easy implementation, the Hamiltonian can be written on index form and using the units as described above

$$\mathcal{H}(S; J, d_z, B) = -\frac{1}{2} J \sum_{\langle i,j \rangle, a} S_{i,a} S_{j,a} - d_z \sum_j (S_{j,3})^2 - \sum_{j,a} B_{j,a} S_{j,a}. \tag{1}$$

Here, $J \in \{-1, 0, 1\}$, $i \in \{1, ..., N\}$ is the site index, a is vector component index. The effective field can be written

$$H_{k,b} = -\frac{\partial \mathcal{H}}{\partial S_{k,b}} = \frac{1}{2} J \sum_{\langle i,j \rangle, a} (S_{i,a} \delta_{j,k} \delta_{a,b} + S_{j,a} \delta_{i,k} \delta_{a,b}) + 2 d_z \sum_j S_{j,3} \delta_{b,3} \delta_{j,k} + \sum_{j,a} B_{j,a} \delta_{k,b},$$

using the vector triple product identity $\vec{A} \times (\vec{B} \times \vec{C}) = (\vec{A} \cdot \vec{B})\vec{C} - (\vec{A} \cdot \vec{C})\vec{B}$. The first sum becomes

$$\frac{1}{2} \sum_{\langle i,j \rangle, a} (S_{i,a}\delta_{j,k}\delta_{a,b} + S_{j,a}\delta_{i,k}\delta_{a,b}) = \frac{1}{2} \sum_{\langle i,j \rangle} (S_{i,b}\delta_{j,k} + S_{j,b}\delta_{i,k}) = \frac{1}{2} \sum_{\langle j,i \rangle} 2S_{i,b}\delta_{j,k} = \sum_{j \in \mathrm{NN}_k} S_{j,b},$$

where NN_k are the set of nearest neighbours of lattice point k. The Landau-Lifshitz-Gilbert equation for the time evolution of the system is then

$$\frac{\mathrm{d}}{\mathrm{d}t} S_{j,a} = -\frac{1}{(1+\alpha^2)} \left[\sum_{bc} \varepsilon_{abc} S_{j,b} H_{j,c} + \alpha \sum_{b} \left(S_{j,b} S_{j,b} H_{j,a} - S_{j,b} H_{j,b} S_{j,a} \right) \right], \tag{2}$$

$$H_{k,b} = J \sum_{j \in \mathrm{NN}_k} S_{j,b} + 2 d_z S_{k,3} \delta_{k,3} + B_{k,b}. \tag{3}$$

Implementation

The main object of the simulation is a NumPy-array `S` of shape `(T, N, 3)`. This contains the components of each of the N spins at ecach of the T time steps. The function `integrate` then runs a loop, calling the implementation of Heun's method `heun_step`. The index notation laid out in the Theory section allows for straight forward implementation of the LLG equation using NumPy's `einsum`-function. For example, using a array `eijk[a, b, c]` for the Levi-Civita symbol,

$$\sum_{bc} \varepsilon_{abc} S_{j,b} H_{j,c}$$

becomes

```
np.einsum("...ac, ...c-> ...a", np.einsum("abc, ...b -> ...ac", eijk, S), H).
```

`LLG` takes as arguments `S` and the needed parameters.

Then, it first evaluates the first sum of Equation 2 as shown above. If $\alpha \neq 0$, it then evaluates the second sum. `LLG` calls `get_H`. This functions implements Equation 3, using NumPy's `roll`-function to sum over all nearest neighbours. This automatically implements periodic boundary conditions. `LLG` then returns `dtS`, a NumPy-array containing the time derivative of S. This implementation makes it easy to generalize to spins distributed in 2 or 3 dimensions, as is done in `3D.py`. The only change needed is to extend the shape of `S` to `(T, N, N, N, 3)`, and change `get_H` to sum over nearest neighbours in all dimensions.

The main loop is then executed by the function `integrate(f, S, h, step, args)`, which fills `S` by using the `step` function, for example `heun_step` to integrate the differential equation

$$f(S; \mathrm{args}) = \frac{\mathrm{d}S}{\mathrm{d}t},$$

with step length `h`. `args` is a tuple of arguments to be passed to `f`. This function is fully modular, and works with any array shape array `S`, as long as it has a first dimension for time, any integrator `step` of function `d`, as long as they accept the right arguments, and any functio `f`. The library `mayavi` is used to make 3D visualization of the spins, as in Figure 5 and to make videos of the time evolution of the spins. `ffmpeg` is then used to use the snapshots from `mayavi` to make a video. Videos of one spin, a chain of spins and a configuration of spins in 3D is included in the folder `plots`.

Results

Single spin

The first test of the simulation is to initialize a single spin, in a magnetic field $B = (0, 0, 1)$. This spin is given a slight tilt, with initial conditions $(\theta, \phi) = (0.5, 0)$. As the torque on the spin is

$$\vec{\tau} = -\frac{\mathrm{d}\vec{S}}{\mathrm{d}t} = \vec{S} \times \vec{B},$$

the expectation is that the spin will precess in a circle around the z-axis, with a Larmor frequency $\omega = -B$. [1] Figure 1 shows the x, y-components of this spin as a function time, together with the expected analytical result. The video `plots/test.mp4` shows a animation of the motion of an arrow representing the spin. On the right, the plot shows the error in each components. The error in the z component can then be tested checking the deviation from $\vec{S}^2 = 1$, which in this case was around 10^{-7}.

To analyze the convergence of the method, the simulation is run with different step lengths h, for the same simulation time $t_0 = 5$. The result is shown in Figure 2. As Heun's method is of higher order than Euler's method, it converges faster. It is, however, necessary to make 2 function calls when using Heun's method, while Euler's method only require one. This should make Euler's method twice as fast, which was observed. Euler's method ran at around 16 000 iterations per second, while Heun's method only ran at 8 000. The large gain in precision, however, makes Heun's method preferred for this application. Throughout this exercise, the step length used is $h = 0.01$, as it is deemed to give high enough precession.

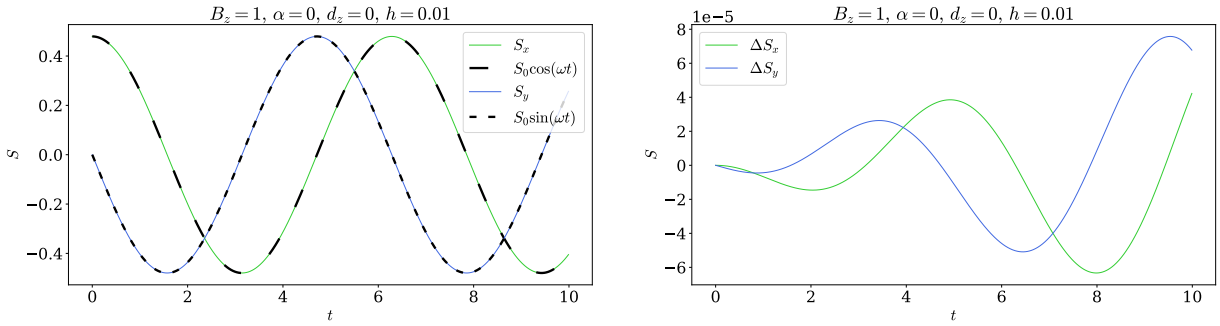

Figure 1: On the left, motion of the x and y component of the spin in a constant magnetic field, together with the analytical result. On the right, the difference between the simulated and the analytical result.

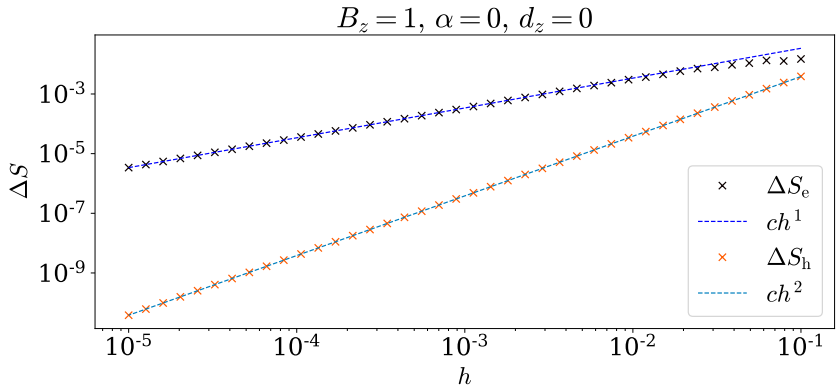

Figure 2: The error from the simulation using the euler method, ΔS_e and Heun's method, ΔS_h.

[1] `https://en.wikipedia.org/wiki/Larmor_precession`

When including $\alpha > 0$, one should expect the the oscillations to die away, with a lifetime given by

$$\tau = \frac{1}{\alpha\omega}.$$

Larger α should give a shorter lifetimes, and thus faster decay. Figure 3 shows this. Furthermore, we see that the amplitude is proportional to $\exp(-t/\tau)$.

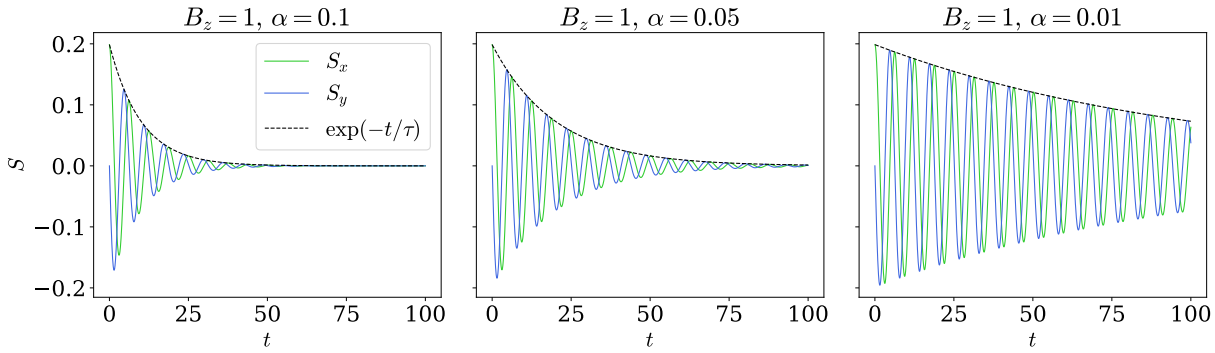

Figure 3: The decay of the oscillation of a single spin, for different values of α.

Spin chain

The simulation now includes several spins, in a ferromagnetic or antiferromagnetic coupling, depending on if $J = 1$ or $J = -1$, respectively, and a damping constant $\alpha > 0$. The spins are initialized in random directions.[2] Both the ferromagnet and antiferromagnetic systems settle into their respective ground states after some time. This is shown in Figure 4, where we see that the different systems have different ground states. In the ferromagnetic case, the lowest energy configuration is the alignment of all the spins, while in the antiferromagnetic case the spins are oppositely aligned. The final configurations of both systems are shown in Figure 5.

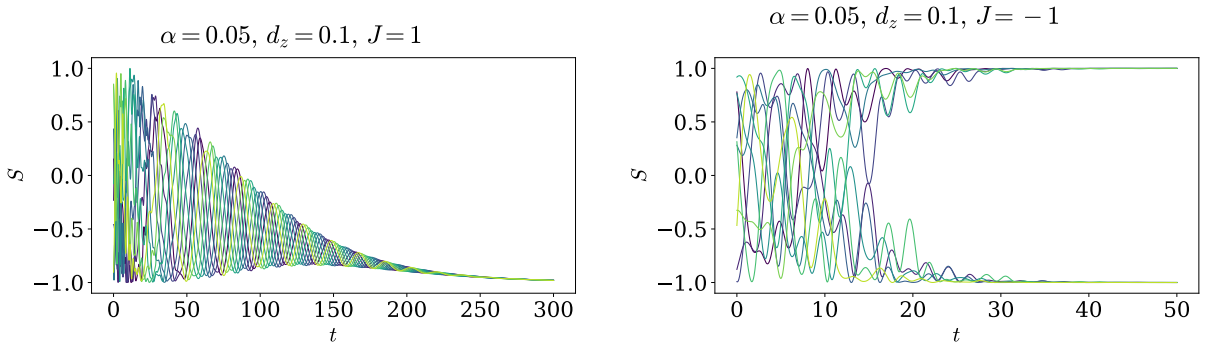

Figure 4: The z-component of all the spins in a ferromagnetic (left) and antiferromagnetic(right) spin chain. As $\alpha > 0$, the states settle down into their ground state.

[2]The positions are drawn uniformly from $(\phi, \theta) \in [0, 2\pi) \times [0, \pi)$. This means that the starting positions will *not* be uniformly distributed on the unit sphere, as the area of a small square $\Delta\phi\Delta\theta$ varies over the surface, however this has no effect on the conclusinos here.

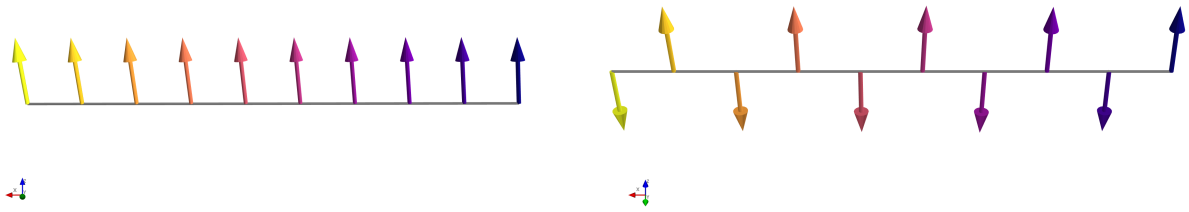

Figure 5: The ground state of ferromagnetic (left) and antiferromagnetic (right) spin chains.

When the coupling is turned of, $J = 0$, $\alpha = 0$ but $d_z > 0$ and the spins are initialized randomly, they will all precess around the x-axis, with different frequencies, as shown in Figure 6. This is because it is now an uncoupled system, where each spin follows

$$\frac{\mathrm{d}\vec{S}}{\mathrm{d}t} = -2d_z S_z \, \vec{S} \times \hat{z},$$

i.e. circular precession around the z-axis with frequency proportional to the z-component of the spin. If only one spin is tilted, it will precess alone, as shown in to the left in Figure 7. However, if the coupling is turned on again, as shown on the right in Figure 7, the disturbance will ripple through the chain, in a wave. This happens as the tilted spin makes it energetically advantageous for its neighbours to tilt towards it, starting a chain reaction that propagate through the chain. This implementation uses, as mentioned earlier, a periodic boundary condition. This means that the wave continues through the wall, and appears on the other side, thus approximating an infinite system.

We can see that the vibrations in the chain is dominated by high frequency oscillations. By turning back on the damping, as shown in Figure 8 on the left, the energetically costly high frequencies die out fast, and we are left with a fundamental frequencies of the system. This behavior is evident in the video `plots/gs_f.mp4`. The plot on the right in Figure 8 illustrates the average of the x-components of all the spins. This follows the long-term pattern immediately. The dotted line is curve-fitted to the function $f(t; A, \omega, \tau) = A\cos(\omega t)e^{-t/\tau}$. Another example of how the organized, long wavelength behavior emerges from a random initial configuration is shown in the last page of this report, in Figure 11, where 50 spins in a ferromagnetic system is simulated.

The left plot in Figure 9 shows the same situation, but with a antiferromagnetic coupling $J = -1$. This system starts in a much higher energy state, and thus becomes highly excited. However, the excitation dies out much faster than in the ferromagnetic case. The right plot shows the same system, but starting in the ground state, except one tilted spin. Here aswell, the excitation dies out much faster, without a surviving oscillation.

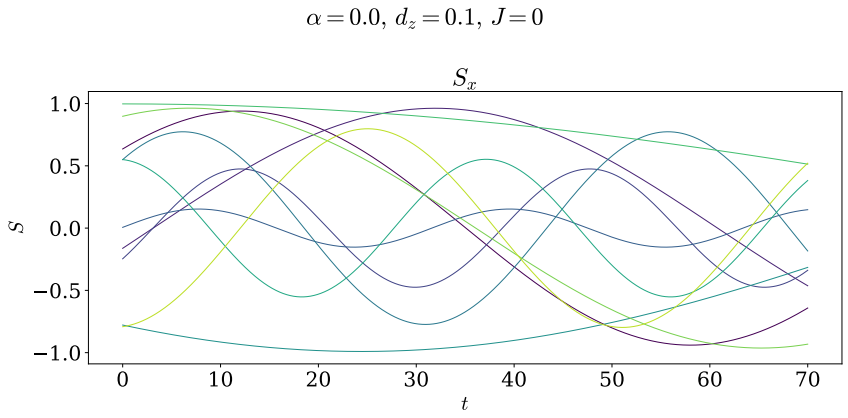

Figure 6: The motion of the x-component of the spin of all spins in a chain, with anisotropy $d_z = 0.1$. The frequency depends on z-component of the spin, which is why it varies.

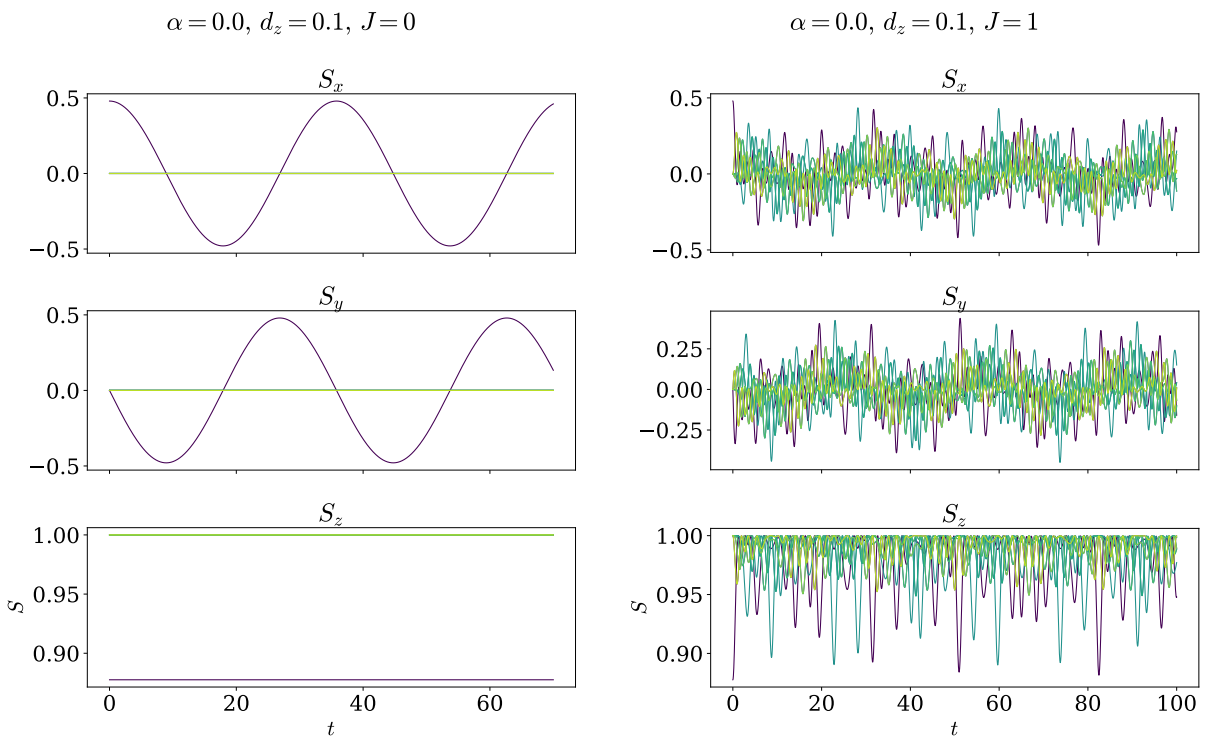

Figure 7: The motion of the different components of each spin in a chains, with anisotropy $d_z = 0.1$. On the left, $J = 1$, on the right $J = 1$.

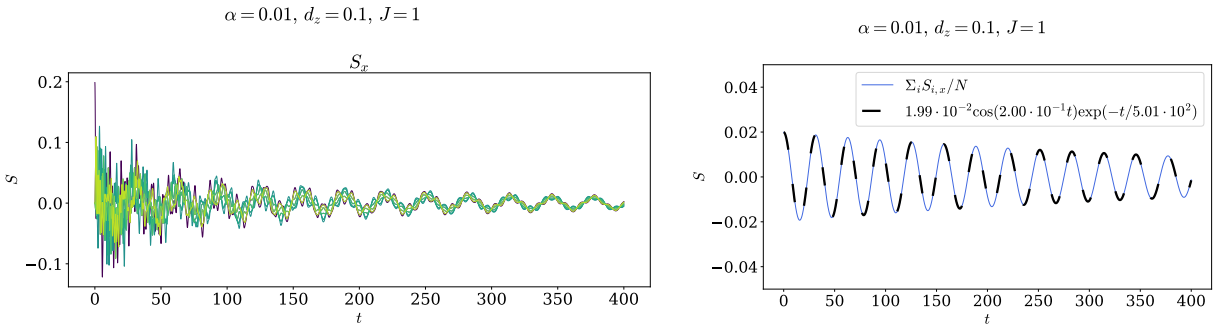

Figure 8: On the left, the x-coords all spins in a chain. On the right, the average of all the x-components, and a fitted function.

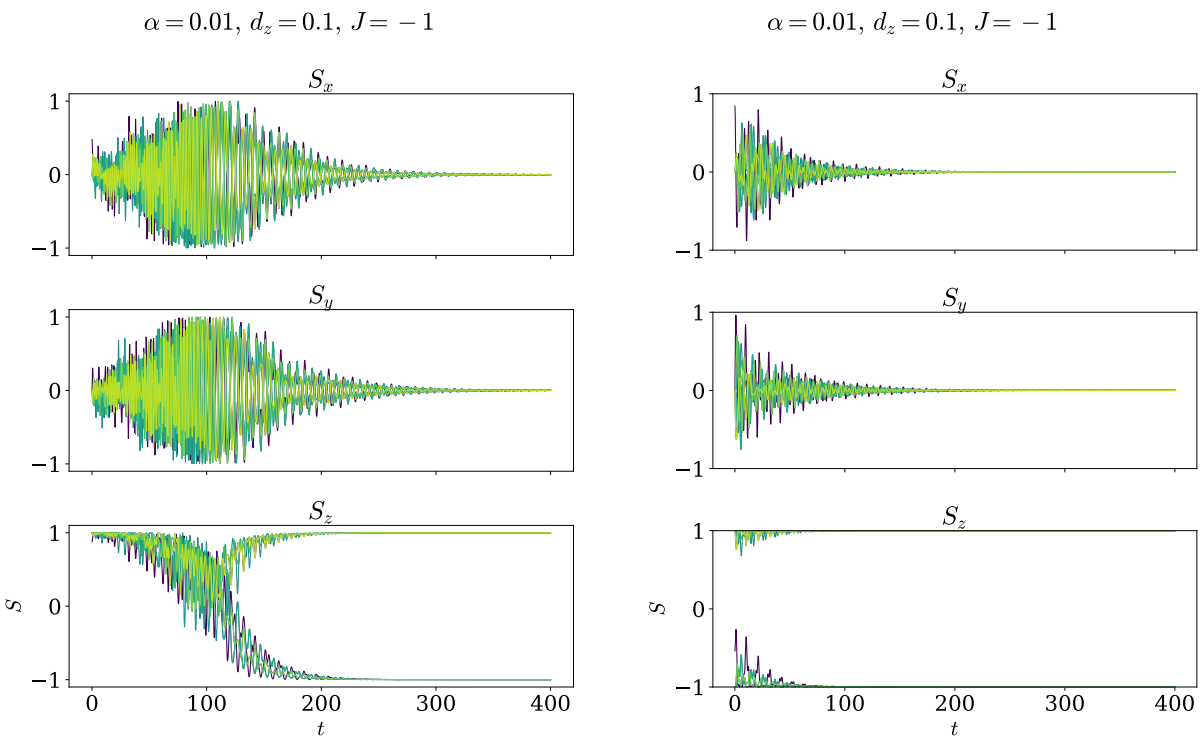

Figure 9: The components of the spins in an antiferromagnetic spin chain. The left starts with all spins almost aligned, the right almost in the groundstate for the anti-ferromagnet.

The magnetization of the state is, in the units previously described, given by

$$M_a = \frac{1}{N} \sum_j S_{j,a}.$$

Thus, the magnetization of the ground state is in ferromagnetic case $\vec{M} = (0, 0, 1)$, while in the antiferromagnetic case $\vec{M} = (0, 0, 0)$. This means that a disturbance in the ground state of the ferromagnet, like the sustained magnon oscillation we showed earlier, will result in a sustained loss of magnetization. This is due to the fact that it is the state of maximum magnetization. This is showed in Figure 10. Here, the magnetization quickly approaches the ground state value as the energy is dissipated. However, as the system settles in to the mode of sustained oscillation, the ferromagnetic on the left retains some loss in magnetization, while the anti-ferromagnet reaches the ground state much faster, and has no sustained difference in magnetization.

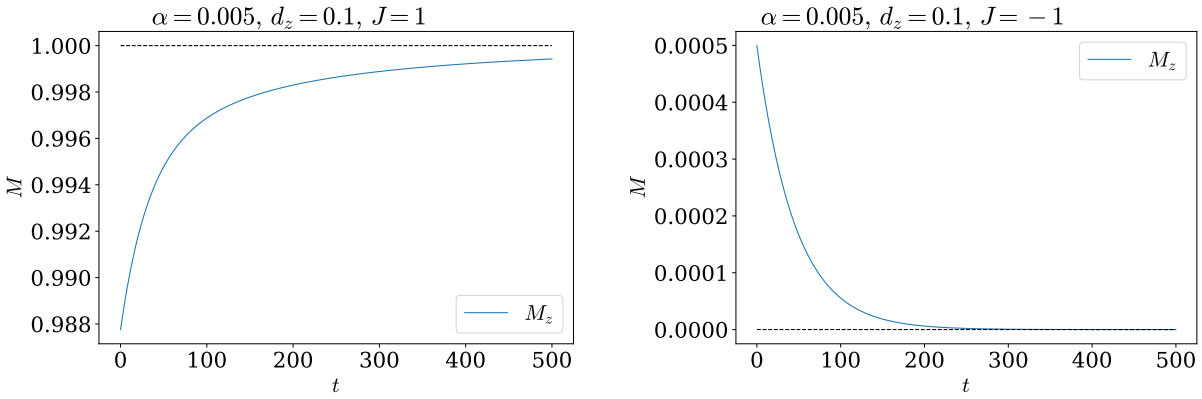

Figure 10: The magnetization as a function of time, for a ferromagnet (right) and an anti-ferromagnet. Both systems are initiated with one spin slightly away from the ground state.

Conclusion

This report has documented the simulation of the classical Heisenberg model. By simulating a single spin, we have demonstrated that the expected analytical result is recreated, and the convergence properties of Heun's method. Ferromagnetic and antiferromagnetic states have different ground states, and antiferromagnetic reaches its ground state much faster through dissipation than its ferromagnetic counterpart. Collective modes, magnons, is shown to exist in ferromagnetic materials. These modes are clearly visible in systems which are disturbed from the ground state, and has some dissipation. They affect the magnetization of the material, as they disturb it away from its highest magnetization state, the ground state.

References

[1] NTNU, Institutt for Fysikk. *Exercise 2, TFY4235 Computational Physics*. 2021.

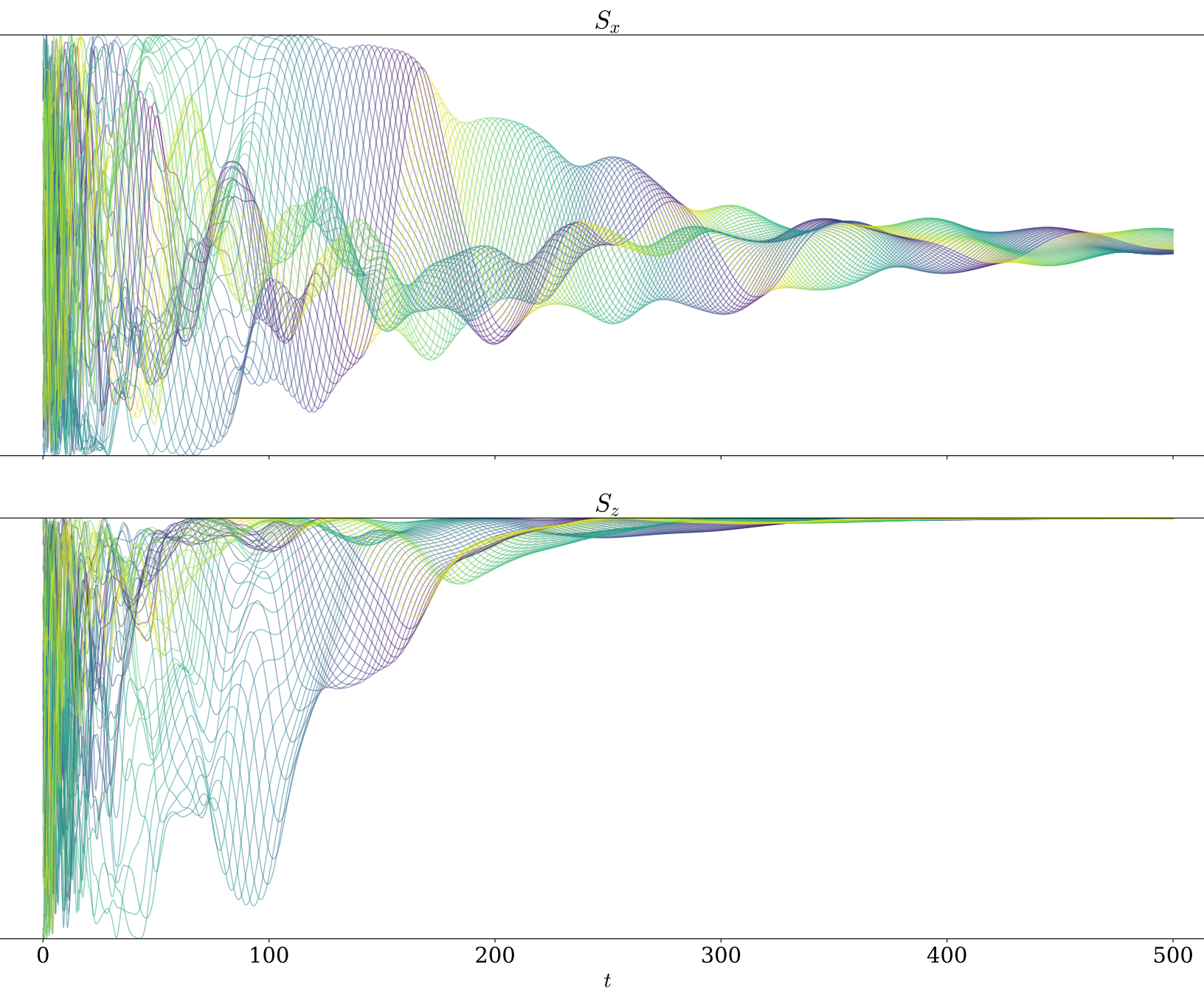

Figure 11: 50 spins simulated over 50 000 steps. Chaos quickly gives away for collective behavior. The same system is also illustrated by the video plots/bonus.mp4