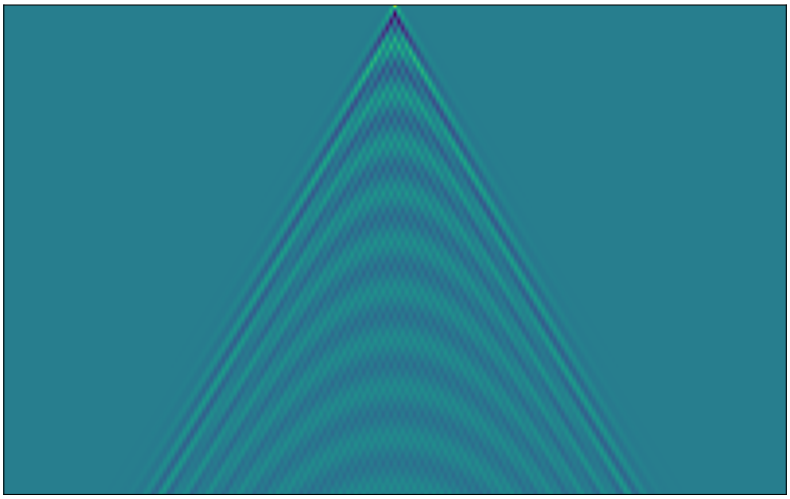
Numerical simulation of magnons

Candidate nr. 10037
Department of Physics, Norwegian University of Science and Technology, Trondheim
Norway
TFY4235 - Computational physics
(*Last updated on May 6, 2021*)

Abstract

We simulate a linear chain of spins through numerically solving the Landau-Lifschitz-Gilbert equation with Heun's method. The simple case of a single spin is compared to an analytical solution, and the more complicated systems are compared to known results in spin wave theory. Magnetic waves (magnons) propagating on the chain are demonstrated to emerge from this equation and their behaviour for different parameter values are discussed.



Contents

PART
I

Introduction

1 Theoretical background

The hamiltonian governing the dynamics of a chain of spins is

$$H = -\frac{1}{2}\sum_{j,k}^{N} J_{jk}\mathbf{S}_j \cdot \mathbf{S}_k - d_z \sum_{j=1}^{N}(S_{j,z})^2 - \mu \sum_{j=1}^{N} \mathbf{B}_j \cdot \mathbf{S}_j. \tag{1}$$

The time-dependence is governed by the Landau-Lifshitz-Gilbert equation (LLG)

$$\frac{\mathrm{d}\mathbf{S}_j}{\mathrm{d}t} = \frac{-\gamma}{\mu(1+\alpha^2)}\left[\mathbf{S}_j \times \mathbf{H}_j + \alpha \mathbf{S}_j \times (\mathbf{S}_j \times \mathbf{H}_j)\right] \tag{2}$$

where the *effective field* $\mathbf{H}_j$ for spin j is given by

$$\mathbf{H}_j = -\frac{\partial H}{\partial \mathbf{S}_j} + \boldsymbol{\xi}_j. \tag{3}$$

Proposition 1. In the absence of noise, $\boldsymbol{\xi}_j = 0$, the effective field in (3) is given by

$$\mathbf{H}_j = \sum_{i=1}^{N} J_{ij}\mathbf{S}_i + 2d_z S_{j,z}\mathbf{e}_z + \mu\mathbf{B}. \tag{4}$$

Proof. This can be shown by simply performing the derivatives

$$\begin{aligned}
-\frac{\partial H}{\partial \mathbf{S}_j} &= \frac{1}{2}\sum_{i,k}^{N} J_{ik}\frac{\partial}{\partial \mathbf{S}_j}(\mathbf{S}_i \cdot \mathbf{S}_k) + d_z \sum_{i=1}^{N}\frac{\partial}{\partial \mathbf{S}_j}(\mathbf{S}_i \cdot \mathbf{e}_z)^2 + \mu\sum_{i=1}^{N}\frac{\partial}{\partial \mathbf{S}_j}\mathbf{B}_i \cdot \mathbf{S}_i \\
&= \frac{1}{2}\sum_{i,k}^{N} J_{ik}\left(\delta_{ij}\mathbf{S}_k + \delta_{jk}\mathbf{S}_i\right) + d_z\sum_{i=1}^{N} 2\delta_{ij}(\mathbf{S}_i \cdot \mathbf{e}_z)\mathbf{e}_z + \mu\sum_{i=1}^{N}\mathbf{B}_i\delta_{ij} \\
&= \sum_{i=1}^{N} J_{ij}\mathbf{S}_i + 2d_z S_{j,z}\mathbf{e}_z + \mu\mathbf{B},
\end{aligned}$$

as advertised. □

We make further simplifications by only performing the sum in the effective field (4) over the set of nearest neighbours $\mathcal{N}_j$ of j. Using this expression we may rewrite the LLG in (2) as

$$\begin{aligned}
\frac{\mathrm{d}\mathbf{S}_j}{\mathrm{d}t} = \frac{-\gamma}{\mu(1+\alpha^2)}\Bigg[&\mathbf{S}_j \times \left(\sum_{i\in\mathcal{N}_j} J_{ij}\mathbf{S}_i + 2d_z S_{j,z}\mathbf{e}_z + \mu\mathbf{B}\right) \\
&+\alpha\mathbf{S}_j \times \left[\mathbf{S}_j \times \left(\sum_{i\in\mathcal{N}_j} J_{ij}\mathbf{S}_i + 2d_z S_{j,z}\mathbf{e}_z + \mu\mathbf{B}\right)\right]\Bigg].
\end{aligned} \tag{5}$$

It is the form of the LLG in equation 5 that we will use in the implementation.

PART II

Code overview

2 General structure

The main machinery in this code is found in `ode.py`. This contains a simple object oriented implementation of an ODE-solver. The child `MagnonSolver` of the `ODESolver` is used to solve the LLG, with the only difference being that it allows for an extra dimension of data corresponding to the spin index, and it unpacks parameters fed into the constructor which describes the hamiltonian of the system. To do a simulation of a system of spins, create an initial state of the system `S0`, given as a $n \times 3$ array where n is the number of spins. Next, specify the start and end time `t0`, `tN` and the step length `h`. Also, specify the parameters describing the problem `d`, `J`, `mu`, `B` and `alpha` in a dictionary `params`. To create an object for the solver, then use

```
solver = MagnonSolver(t0,S0,tN,h,"Heun",**params)
```

and do the integration by simply calling the object

```
T, S = solver(verbose = True)
```

which returns the array of the time steps, `T`, and the spins, `S`, at each time step with the first index representing the time step. The optional keyword `verbose` is default `false`, and outputs a progress-bar if `true`. To create an initial spin state with unit length one can use the function `initial_cond`, which takes as arguments the azimuthal and polar angle, θ and φ and outputs the point on the 2-sphere corresponding to these angles.

3 Remarks on performance

When doing the error-calculation in section 4.2 I found that my `python` implementation of the solver was pretty slow for the smallest time steps. This is probably due to the function being implemented in a quite general way so that it is easy to change the parameters of the problem. This is however not convenient when considering that we waste a lot of time calculating certain products which are later multiplied by zero and hence don't contribute. For instance, when using $h = 10^{-5}$ the naive `python` solution produces

```
S_0 = np.array([initial_cond(0.1,0.5)])
params = {'d':0,'J':0,'mu':1,'B':np.array([0,0,1.]),'alpha':0}
tN = 2*np.pi
spinsolver_heun = MagnonSolver(0,S_0,tN,1e-5,"Heun",**params)
%time spinsolver_heun();
```

```
CPU times:  user 2min 57s, sys:  507 ms, total:  2min 58s Wall time:  2min 57s
```

A large improvement is gained by compiling parts of the calculation with `numba`. However, trying to use `@jit` on methods in self defined objects turns out to be quite messy, so a large part of the main calulcations could not be just-in-time compiled in my case without rewriting the whole implementation. Running the same code as above with just-in-time

compilation used on only the function, $\mathbf{f}(t, \mathbf{y})$, defining the ode

$$\frac{\mathrm{d}\mathbf{y}}{\mathrm{d}t} = \mathbf{f}(t, \mathbf{y})$$

produces

```
CPU times:  user 10.5 s, sys:  624 ms, total:  11.2 s Wall time:  10.4 s
```

As a curiosity, I can also mentioned that only replacing `np.cross` by the self-made function `cross` shown below produces a speed-up by approximately a factor of 2.

```
def cross(A,B):
  return np.einsum('ijk,...j,...k',eijk,A,B)
```

This is based on the general expression

$$\mathbf{A} \times \mathbf{B} = \mathbf{e}_i \epsilon_{ijk} A_j B_k,$$

where ϵ_{ijk} is the Levi-Civita tensor, represented in python as simply a $3 \times 3 \times 3$ array:

```
eijk = np.zeros((3,3,3))
eijk[0, 1, 2] = eijk[1, 2, 0] = eijk[2, 0, 1] = 1
eijk[0, 2, 1] = eijk[2, 1, 0] = eijk[1, 0, 2] = -1
```

There is currently however not support for using `einsum` together with the just-in-time compiler from numba.

For the error analysis in particular, I wanted a faster implementation. Since this case only included 1 spin and only the effect of the **B**-field, I wrote a simple implementation in `julia` for this specific case. This implementation is albeit not general; changing the parameters requires writing a new implementation of **f**, and so on. However, it does perform extremely well compared to the `python`-version:

```
tN  = 2 * pi
S_0 = initial_cond(0.1,0.5)
@time magnon_integrate(S_0,0,tN,1e-5,f_llg,heun);
```

```
0.994903 seconds (16.34 M allocations:  1.124 GiB, 6.69% gc time)
```

Since we are mostly dealing with small systems in this project I found the `python` solution to be well suited and efficient enough. Moreover, it is easier to use than my implementation in `julia` which requires making new implementations of **f** for each specific case. At any rate, my time is more valuable than the computer's.

The program itself takes no reference to the actual physical values of the parameters in the model, and thus all results are given in natural units.

PART III

Results and discussion

4 Single spin

4.1 Precession of spin in uniform magnetic field

We simulate the time evolution of a single spin $\mathbf{S}$ in the presence of a uniform magnetic field $\mathbf{B} = (0, 0, B_0)^T$ in the $\mathbf{e}_z$-direction. The components of the spin at equidistant time steps during one period are shown in figure 1. As expected, we see that the spin precesses around the effective field $\mathbf{H}$, which in this case is given by

$$\mathbf{H} = -\frac{\partial H}{\partial \mathbf{S}} = \frac{\partial}{\partial \mathbf{S}} (\mu \mathbf{B} \cdot \mathbf{S}) = \mu \mathbf{B},$$

in the absence of damping and anisotropy.

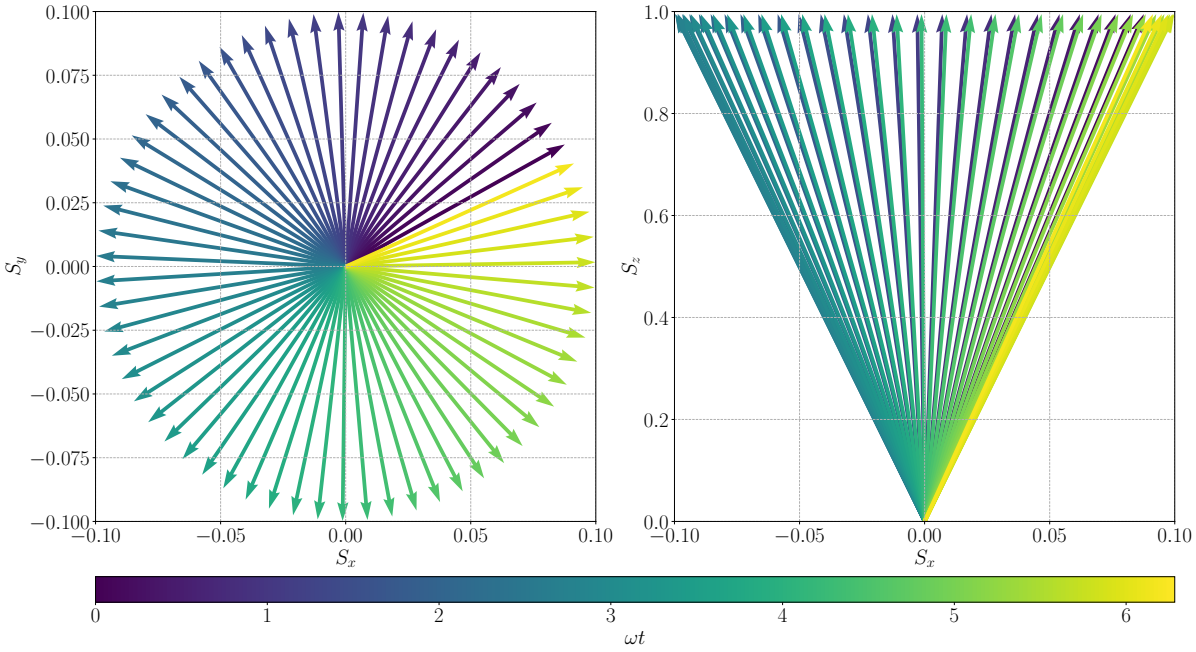


Figure 1: The figure shows the x and y component of the spin during one period in a uniform magnetic field without further interactions.

In this particular case, we can easily find an analytical solution to compare with. The LLG-equation reads

$$\partial_t \mathbf{S} = -\gamma \mathbf{S} \times \mathbf{B}.$$

As $\mathbf{B} = (0, 0, B_0)^T$,

$$\mathbf{S} \times \mathbf{B} = (S_y \mathbf{e}_x - S_x \mathbf{e}_y) B_0.$$

Thus, we have two equations

$$\begin{cases} \partial_t S_x = -\gamma B_0 S_y \\ \partial_t S_y = \gamma B_0 S_x. \end{cases} \tag{6}$$

which are easily solved by differentiating both with respect to t, and then substituting the first order derivatives on the right hand side by the corresponding expressions in 6.

$$\begin{cases} \partial_t^2 S_x = -\gamma B_0 \partial_t S_y \\ \partial_t^2 S_y = \gamma B_0 \partial_t S_x. \end{cases} \tag{7}$$

This yields the two equations

$$\ddot{S}_x = -\left(\gamma B_0\right)^2 S_x \quad ; \quad \ddot{S}_y = -\left(\gamma B_0\right)^2 S_y, \tag{8}$$

which have solutions

$$S_x(t) = S_x(0)\cos\left(\omega t\right) - S_y(0)\sin\left(\omega t\right) \tag{9}$$
$$S_y(t) = S_y(0)\cos\left(\omega t\right) + S_x(0)\sin\left(\omega t\right), \tag{10}$$

with the frequency $\omega = \gamma B_0$. When comparing the exact solution with the numerical estimate obtained through integrating the LLG-equation with Heun's method, the trajectories of S_x and S_y are as shown on the left hand plot in figure 2.

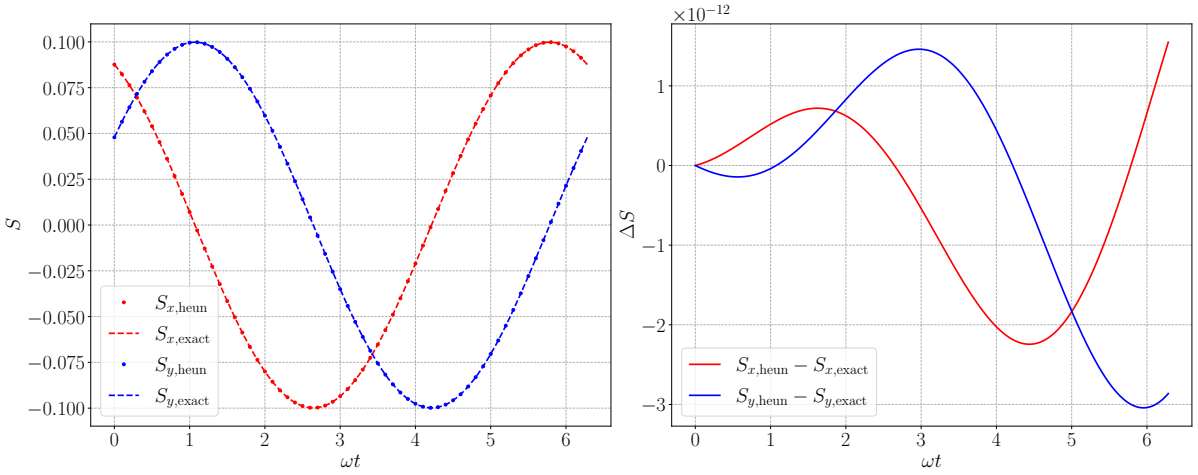


Figure 2: The plot on the left hand side shows the exact solution given in (9) and (10) compared with the numerical solution sampled at every tenth step to be able to distinguish the paths. The plot on the right hand side shows the difference between the exact solutions given in (9) and (10), and the numerical solutions over one period.

It is perhaps more enlightening to consider the pointwise difference of the exact and numerical solution. This is shown on the right hand side of figure 2. The figure shows that the deviations locally oscillate, and that the absolute global deviations increase with time as we'd expect. Further considerations on the deviations are considered in 4.2.

4.2 Error analysis

For the error analysis, I choose 10 logarithmically spaced step-lengths from 10^{-5} to 10^{-1}. As a measure of the *global error*, that is the accumulated error over one period, I use the absolute difference between the numerical solutions and the corresponding known solutions in (9) and (10). The error as a function of step length is shown in figure 3. Since Heun's method is of second order, we would expect its slope in a log-log plot to be

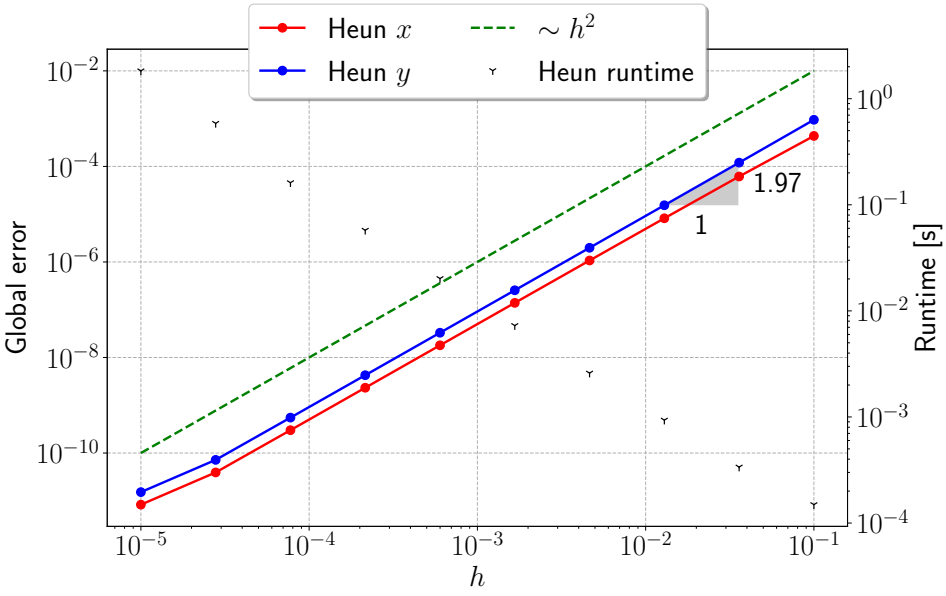


Figure 3: Error as a function of step length for Heun's method.

roughly 2. Comparing to the line $\sim h^2$ shows that this is the case, at least for reasonably large h. In figure 3 we notice that the error line seems to have a kink for very low step lengths. This can be understood by the fact that the round-off error done by the machine pollutes the numerical calculation. To have a rough estimate of the pollution for $h = 1 \cdot 10^{-5}$, consider the following: We do roughly 60 000 steps on one period. If the round-off error in each step is $\approx 10^{-15}$ we would expect an accumulated error of $\approx 6 \cdot 10^{-11}$, which indeed is comparable to the error at $h = 10^{-5}$ in figure 3. This accounts for the kink in the error at $h = 10^{-5}$.

In figure 4 we have plotted the errors for Euler's method for the same step lengths. This plot clearly demonstrates the well known fact that Euler's method is of order 1. Notice that the error never becomes so small as to be affected by numerical round-off error done by the machine. Also, note that the runtimes of Euler's method are always lower than the corresponding runtime of Heun's method. This is also as expected as there is one more function call involved in each step of the former method.

Remark 1. Going even further down in step length for Heun's method would result in errors eventually starting to increase for decreasing step lengths as the numerical round-off error becomes larger than the numerical error. To find the ideal step length, one should examine the numerical error and find the minimum as a function of step length.

One could also do this analytically: What we seek is an upper bound for the global error as a function of h, $e(h)$, given that we can only calculate a number in the computer up

to some number δ, of the order of the machine round off error. Including the round-off error is not entirely straight forward, but roughly speaking, one can derive an expression along the lines of

$$e(h;\delta) \leq M_1 h^p + \frac{M_2}{h}\delta,$$

where p is the order of the method, M_1 and M_2 are constants depending on the function $\mathbf{f}$ defining the ODE, its derivatives and the length of the integration interval. The global truncation error of the method is $\mathcal{O}(h^p)$ while the error introduced by round-off in the computer must scale approximately like $\frac{\delta}{h}$, since $h \sim N^{-1}$. If one manages to find the constants M_1 and M_2 (or some bounds of them), one can approximate the ideal step length by minimizing the error with respect to h and get

$$h_{\text{ideal}} \approx \left(\frac{M_2\delta}{pM_1}\right)^{1/(p+1)}.$$

This is however not embarked on here, as step lengths $\in (10^{-3}, 10^{-2})$ give more than good enough results for us to study the physical properties of the system of spins.

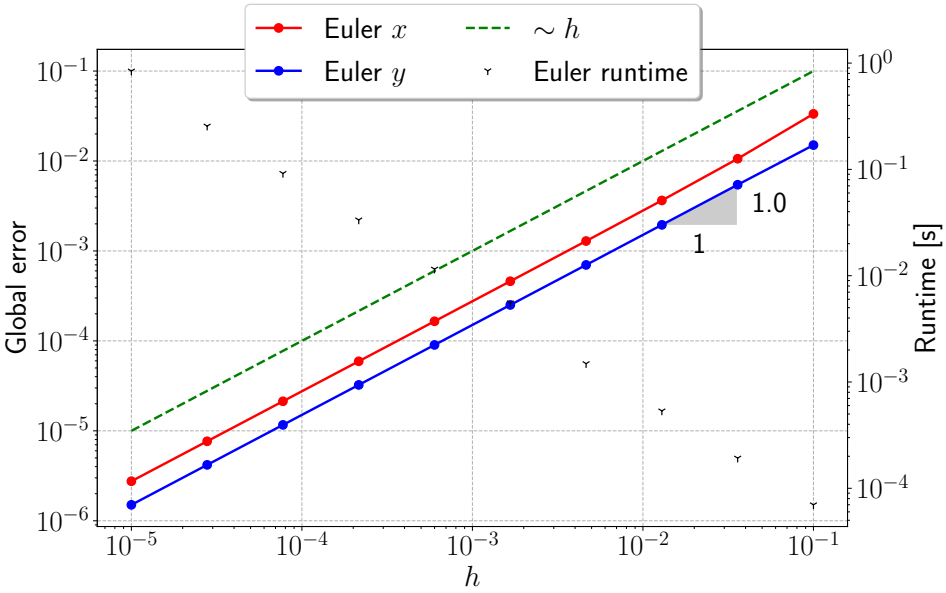


Figure 4: Error as a function of step length for Euler's method.

4.3 Damped precession of spin in magnetic field

Next, we include damping in the model. This amounts to setting $\alpha \neq 0$. The plots of the trajectories in the x and y direction for three different damped cases is shown in figure 5. The damping lifetime $\tau = \frac{1}{\alpha\omega}$ is fitted to the plots by plotting

$$\pm\sqrt{S_x^2 + S_y^2}\exp\left(-\alpha\omega t\right),$$

which is approximately the shape of the envelope in the presence of damping.

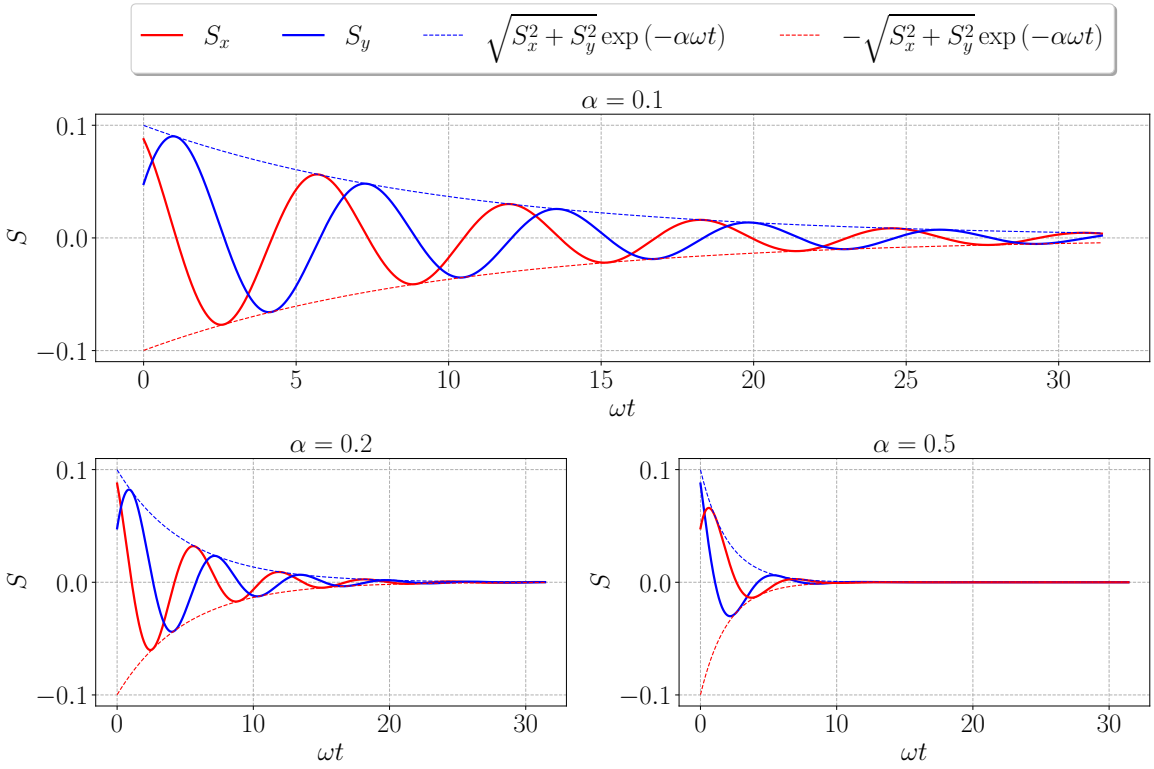


Figure 5: Three different damped precessions.

5 The spin chain

5.1 Ground states

In this section we initialise 10 spins in random directions and simulate the time evolution for $J < 0$ and $J > 0$. The plots of the z-component of each of the spins over time is shown in figure 6. What is evident from these plots is that a positive coupling constant J tends to make the spins align with their neighbours, while the negative constant tends to make them oppose them. This is a fact that is readily seen from inspecting the hamiltonian in (1), since a positive J makes the maximum of $\sum_{j,k} \mathbf{S}_j \cdot \mathbf{S}_k$ most energetically favourable, while a negative one makes its minimum most favourable. These two cases correspond to a ferromagnetic and anti-ferromagnetic system respectively.

The reason for this being apparent when only plotting the S_z-component is the non-zero anisotropy-constant d_z. As seen from the hamiltonian, setting this positive will favour spins along either $+\mathbf{e}_z$ or $-\mathbf{e}_z$. This is also the reason that approximately half of the spins in the case of $J < 0$ being directed along $\mathbf{e}_z$ while the others are directed along $-\mathbf{e}_z$.

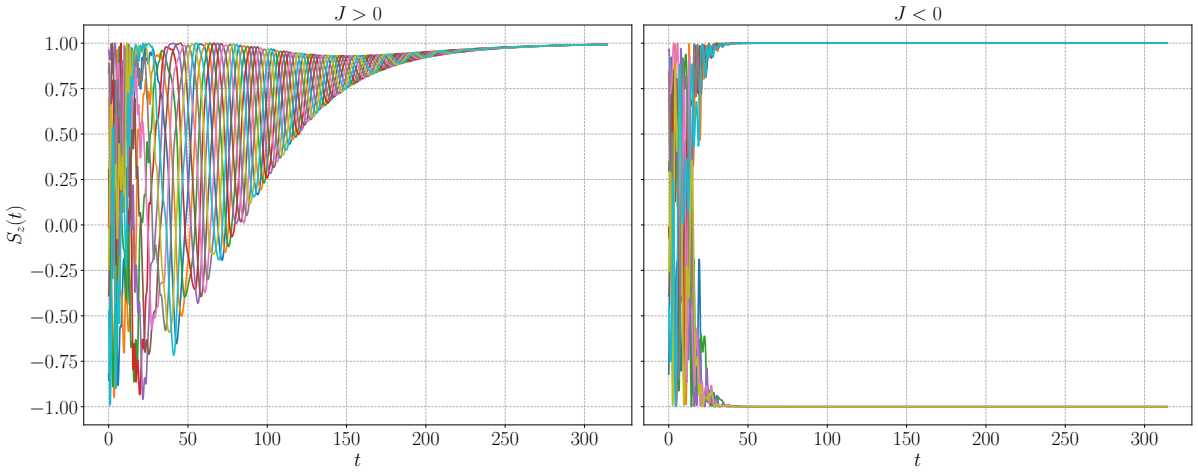


Figure 6: The figure shows the plot of the z-component of 10 spins in a system of positive and negative coupling constant.

5.2 The magnon

5.2.1 Uncoupled system

We initialise the system of 10 spins with random orientations in space and set $J = 0$, $d_z = 1$, $\alpha = 0$ to demonstrate that all the spins precess in time. The trajectories of the x and y components of the spins is plotted in figure 7

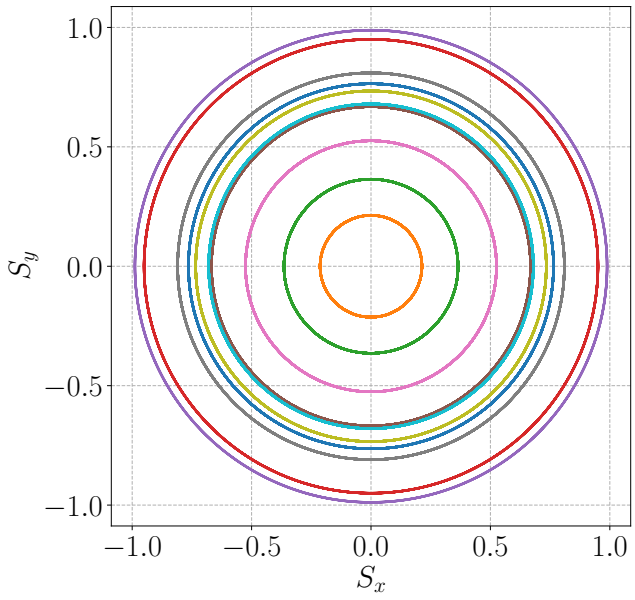


Figure 7: Trajectories of the x and y components of the 10 randomly initialised spins.

Next, we initialise all the spins along the z direction except for one of them which we tilt slightly. Since there is no coupling between the spins, the precession of the first spin will not affect the others. Furthermore, when $\mathbf{S} = S\mathbf{e}_z$, the right hand side of the LLG (2) will be zero since the effective field is $\parallel \mathbf{e}_z$. As expected, this is also what is observed when simulating the system. This is shown in figure 8.

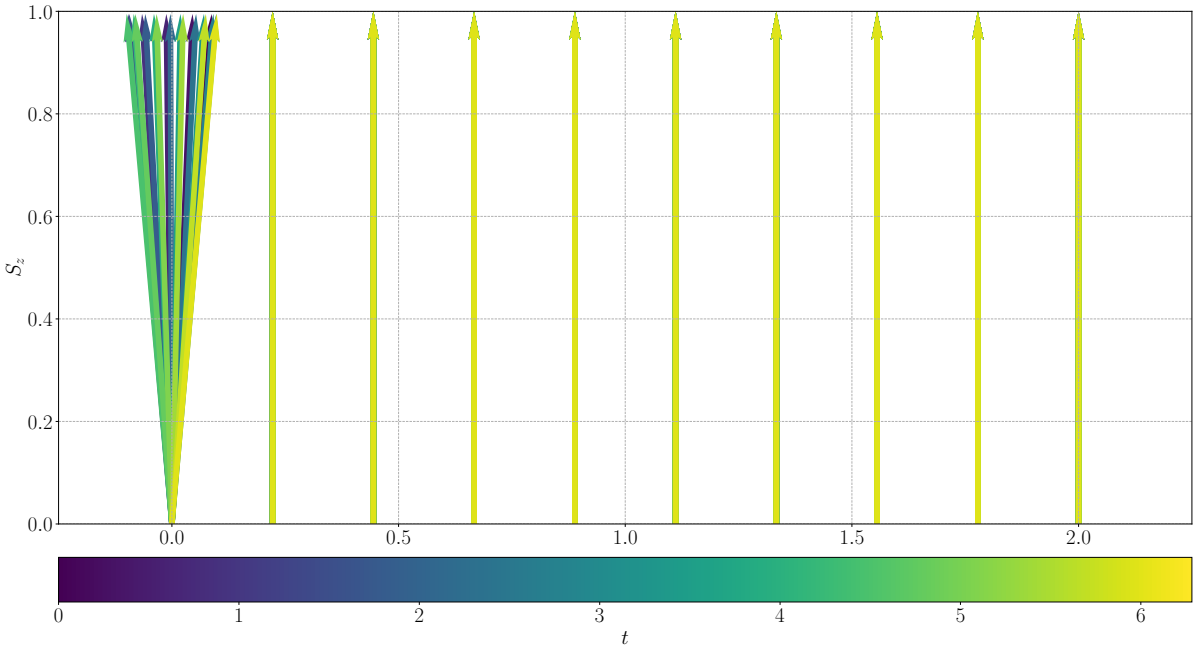


Figure 8: The figure shows uncoupled oscillations of the spin chain.

5.2.2 Coupled system, $J > 0$

If we repeat the procedure used in the previous section except that we couple the spins with $J > 0$, we will observe that all the spins will start to precess, even if they are initialised along the z-axis. Since the time evolution is hard to catch on a 2d plot I have made a video showing it in the file `coupled_spins.mp4` and in 3D in the file `coupled_spins_3d.mp4`. It is also shown in figure 10 as a heat map. What is apparent here is that the neighbouring spins couple to each other. Since we are dealing with a finite system, I have chosen to use periodic boundary conditions, so that the first spin affects the last one and vice versa. Implementing this in python requires no effort at all as we can access the left hand neighbour of spin `i` with `i-1` regardless of the value of `i`, and the right hand neighbour with `(i+1)%n` where `n` is the number of spins. This looks a bit artificial in the video, but it corresponds physically to arranging the linear chain in a circle. Another choice of boundary conditions is to simply say that the first and last spin only has *one* neighbour.

Remark 2 (Simulating an infinite system). If we are to simulate an infinite system we obviously cannot use a discrete model. However, the simulations above suggests that in the limit of *many* spins we can describe the chain as a continuous wave. This suggests that there is a continuum a version of the system in this regime. Then we would have to associate to every index j a position in space

$$\mathbf{S}_j(t) \to \mathbf{S}(\mathbf{x}, t) \quad ; \quad j \to \mathbf{x},$$

where $\mathbf{x}$ is the position of spin j. For simplicity, we assume the spins are distributed on a 1-dimensional lattice of lattice spacing a. If we approximate the second derivative by the finite differences

$$\frac{\partial^2 \mathbf{S}}{\partial x^2}(x,t) = \frac{\mathbf{S}(x-a,t) - 2\mathbf{S}(x,t) + \mathbf{S}(x+a,t)}{a^2} + \mathcal{O}(a^2)$$

we see that the nearest neighbour sum in (5) can be expressed as

$$\mathbf{S}_{j-1}(t) + \mathbf{S}_{j+1}(t) \to \mathbf{S}(x-a,t) + \mathbf{S}(x+a,t) = 2\mathbf{S}(x,t) + a^2 \frac{\partial^2 \mathbf{S}}{\partial x^2}(x,t) + \mathcal{O}(a^4).$$

In the absence of damping one may insert this into LLG (5) and then obtain

$$\frac{\partial \mathbf{S}(x,t)}{\partial t} = -\frac{\gamma a^2 J}{\mu} \mathbf{S}(x,t) \times \frac{\partial^2 \mathbf{S}(x,t)}{\partial x^2} + \mathcal{O}(a^4).$$

As we want to consider the limit of $a \to 0$, we are led to define a new time $\tau = a^2 t/J$. Then taking the limit as $a \to 0$ we obtain

$$\frac{\partial \mathbf{S}(x,\tau)}{\partial \tau} = -\frac{\gamma}{\mu} \mathbf{S}(x,\tau) \times \frac{\partial^2 \mathbf{S}(x,\tau)}{\partial x^2},$$

where we have to restrict our attention to phenomena with time scales $\mathcal{O}(1/a^2)$. Hence, in the limit of small lattice spacings $\|\mathbf{a}\| \to 0$ it should be possible to recast the LLG equation in a continuous form, as a PDE rather than an ODE, presumably also in a 3-dimensional lattice. A self contained discussion of this can be found in [Lak11]. To see the connection between this system and continuous waves qualitatively we simulate coupled system of 201 spins and excite the spin at the centre of the system. The heatmap displaying the time evolution is shown in figure 9.

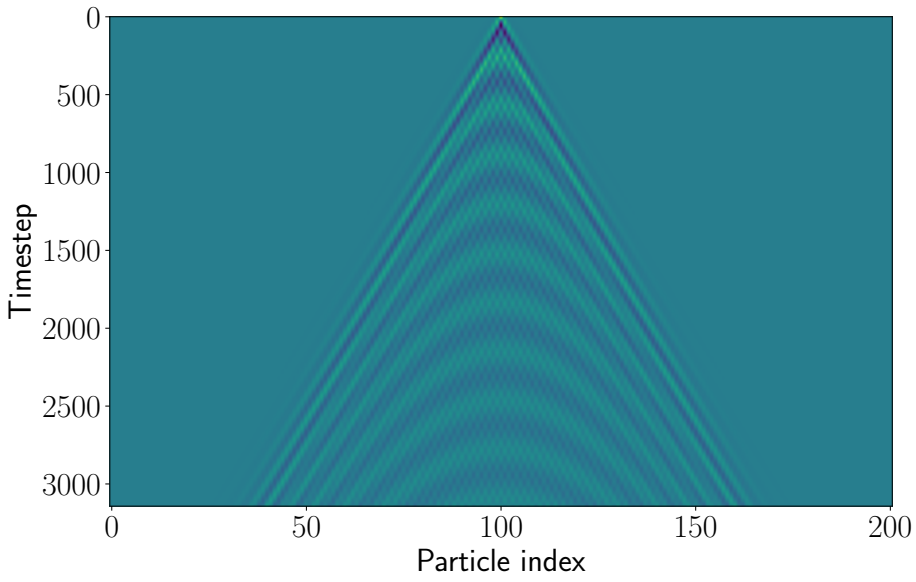


Figure 9: The figure shows the x-component of coupled oscillations of 201 spins.

5.2.3 Damped, coupled system

Based on the results from including damping on one spin and the behaviour of the magnon in the previous section we would expect the damped coupled system to show a collective wave motion whose amplitude eventually dies out. This is exactly what is observed in

the video in file `coupled_spins_damped.mp4` showing the time evolution of the system when we set $\alpha = 0.1$. A heat map illustrating the same time evolution only with slightly less damping is shown in figure 10. Here it is evident that the oscillations die out in the case of $\alpha = 0.05$ while they continue to propagate when $\alpha = 0$. Notice also that due to the periodic boundary conditions, there are two magnons propagating through the system. This is observed in the plot as the zigzag pattern, while in the video it is clear from the start that the initial perturbation also affects the last spin in the chain.

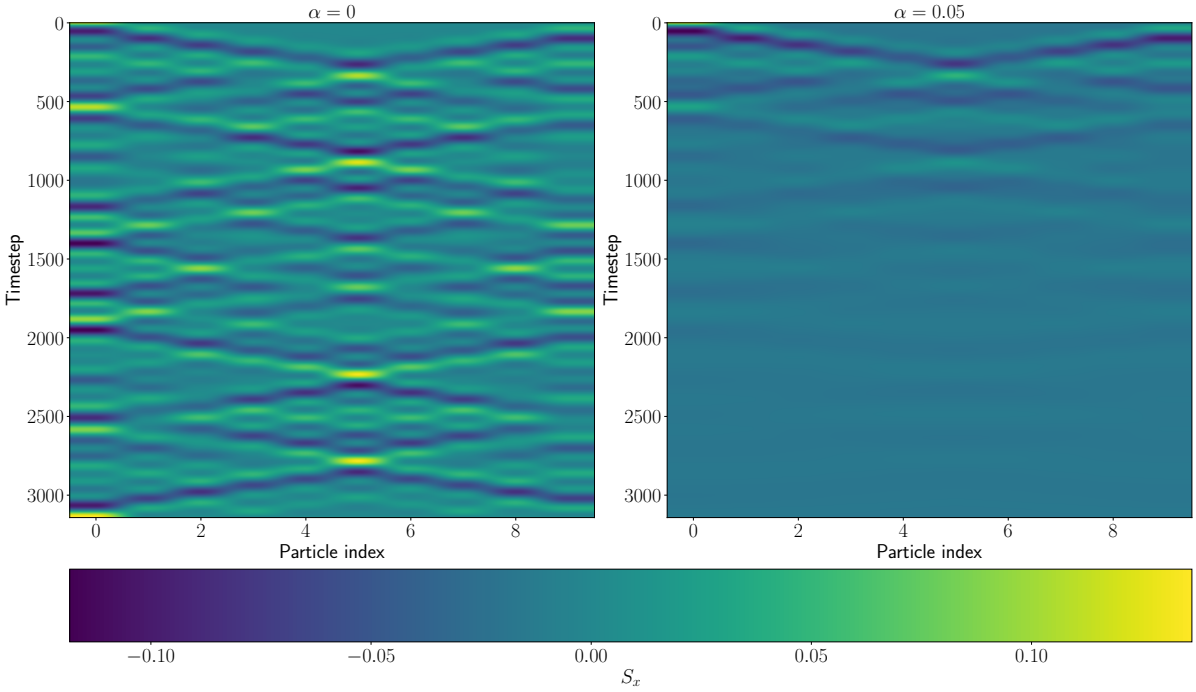


Figure 10: The figure shows the amplitudes of coupled oscillations of the spins for $\alpha = 0$ and $\alpha = 0.05$ given the same initial conditions.

5.2.4 Antiferromagnetic coupling, $J < 0$

The case of $J < 0$ is very similar to the case considered in 5.1. The video of the time evolution corresponding to this case can be found in the file `coupled_spins_anti.mp4`. As one spin is tilted, the neighbours tend to oscillate in phase with their neighbours as opposed to the ferromagnetic case where the spins are repelled by their neighbours.

5.2.5 Magnetisation

Definition 1. We will refer to the *magnetisation* of the system as the mean value of the z-component of each spin in the system,

$$M := \frac{1}{N} \sum_{i=1}^{N} S_{i,z}$$

In section 5.1 there is a net magnetisation only in the case where $J > 0$, whereas it is zero when $J < 0$. In the first case the magnetisation approaches 1 as the system approaches

an equilibrium state. For the case of $J < 0$, half of the spins will eventually align along $+\mathbf{e}_z$ and the other half along $-\mathbf{e}_z$ as explained in section 5.1, giving a net magnetisation of 0. We visualise this behaviour by plotting the magnetisation as a function of time for the two cases. This is shown in figure 11.

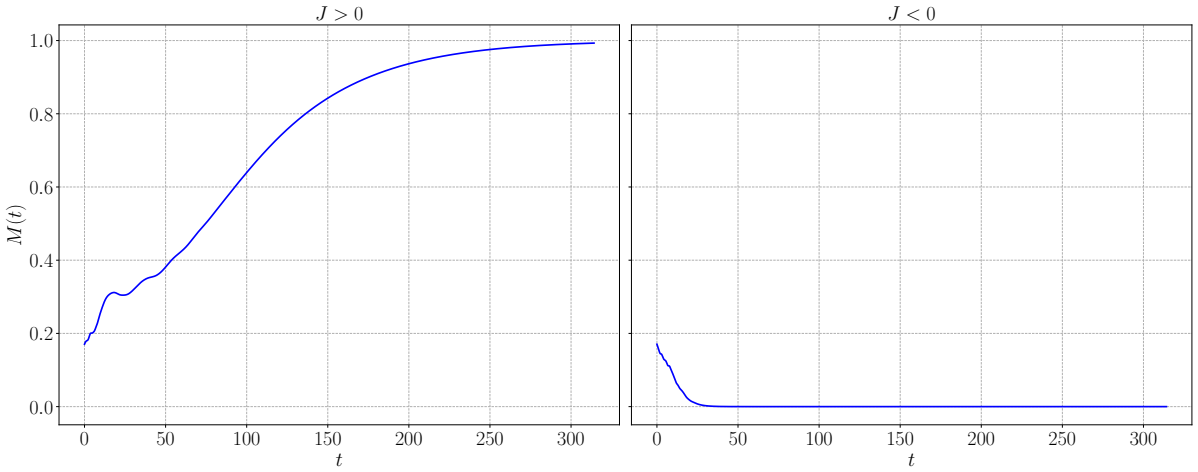


Figure 11: Magnetisation as a function of time for a ferromagnetic and anti-ferromagnetic chain of 10 spins, initialised randomly.

We plot the same time evolution of M for the ferromagnetic coupled system shown in the left hand side of figure 10. This is shown in figure 12. It is evident from this plot that the magnetisation of the coupled system to a good approximation is constant, despite the z-component of each spin varying in time. The coupling introduces — as we have seen — propagating excitations along the chain: magnons. This demonstration shows that the magnons cause the magnetisation of the system to decrease.

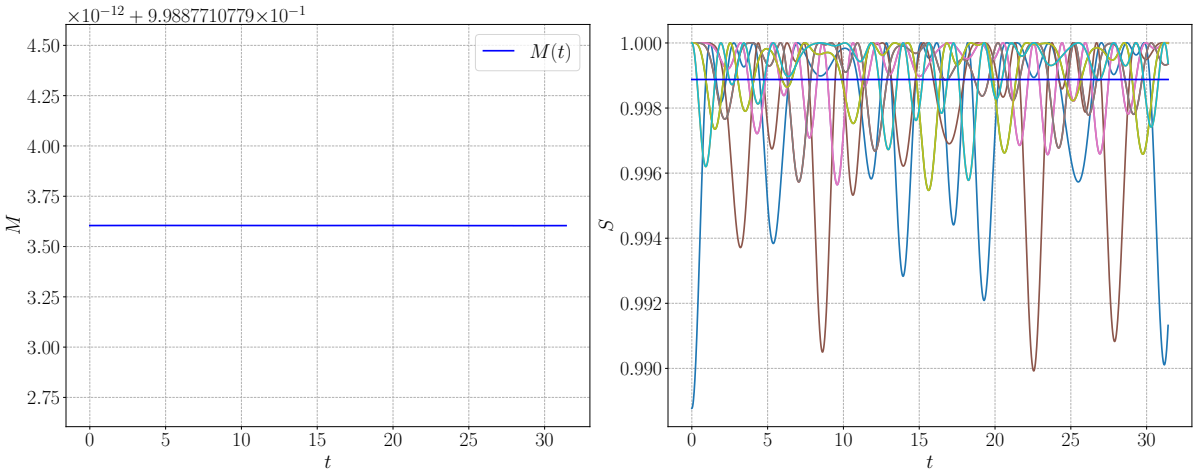


Figure 12: Magnetisation as a function of time for a ferromagnetic chain of 10 spins.

PART
IV

Conclusion

We have simulated the time evolution of various spin chains governed by a Hamiltonian including a variety of interactions. When there is non-zero coupling between the spins the local excitations, represented as tilting of the spins, propagate along the chain like a wave. This demonstrates the emergence of *magnons*, which - much like phonons - are quantised waves. These excitations resemble real wave behaviour formally only in the limit of infinitely many spins, but the qualitative wave behaviour is seen to be captured within this simple model including only a small number of spins.

References

[Lak11] M. Lakshmanan. The fascinating world of the Landau-Lifshitz-Gilbert equation: An overview, mar 2011.