Report 9: Nuclear magnetic resonance simulation

Don Winter[1] **and Madhulan Suresh**[2]

[1]don.winter@rwth-aachen.de, Mat. No. 431380
[2]madhulan.suresh@rwth-aachen.de, Mat. No. 429481

Introduction

In this report we numerically solve the Bloch equations for bulk magnetization under the influence of decoherence and relaxation. We simulate the change in net magnetization of a sample induced by static and dynamic magnetic fields operated at resonance frequency and thereby make explicit use of the phenomenon of nuclear magnetic resonance. As usual, we first introduce the theoretical model followed by our simulation results and finally a concluding discussion.

Simulation model and method

If a previously non-magnetic piece of matter is exposed to a static magnetic field two observations can be made. First, if the induced magnetization of the material is measured one observes the magnetization vector to precess about the direction of the applied static field. If the strength of the field is increased, the precessing magnetization vector converges towards the field vector. However, the precession frequency, known as Larmor frequency, does not change. Another observation one can make is, once the external field is switched off the magnetization vector returns "slowly" back to the initial position before the field was applied. This return is caused by so-called *relaxation* and *decoherence* processes which happen with characteristic material time constants T_1 and T_2. Although these phanomena can only be understood quantum-mechanically, it turns out to model the time evolution of the magnetization vector a macroscopic description is sufficient. This was realized by Felix Bloch in 1946 who proposed a simple set of differential equations:

$$\frac{d\vec{M}}{dt} = \gamma \vec{M} \times \vec{B}(t), \tag{1}$$

where $\vec{M}$ is the magnetization vector exposed to an external magnetic field $\vec{B}(t)$. If one chooses a constant field $\vec{B} = const. = [0, 0, B_0]^T$, one obtains a set of coupled equations with solution $\vec{M}(t) = [M_x(0)\cos(\omega_0 t) + M_y(0)\sin(\omega_0 t), -M_x(0)\sin(\omega_0 t) + M_y(0)\cos(\omega_0 t), 0]^T$ for Larmor frequency $\omega_0 = \gamma B_0$. This solution shows precession in the xy-plane around the static magnetic field vector $\vec{B}$. If we further include a time-changing magnetic field in the xy-plane of strength h, we can additionally solve the equation of motion in the rotating frame of precession and obtain for the z-component of magnetization in this frame $M_z(t) = M_z(0)\cos(\gamma h t) - M_y(0)\sin(\gamma h t)$ for Rabi frequency γh. The additional dynamic field allows to freely rotate the magnetization vector $\vec{M}$ around the x-axis to arbitrary positions on the yz-unit circle by applying pulses at the Rabi frequency for various amounts of time t. However, after a pulse is applied we know from observation that the magnetization vector gradually returns to its initial position due to relaxation and decoherence. This we did not include in our model so far. To do so we can simply add another term at the end of Eq. 1

$$\frac{d\vec{M}}{dt} = \gamma \vec{M} \times \vec{B}(t) + \vec{\alpha} \odot \vec{M} = \gamma \begin{bmatrix} B_0 M_y - B_y(t) M_z \\ B_x(t) M_z - B_0 M_x \\ B_y(t) M_x - B_x(t) M_y \end{bmatrix} - \begin{bmatrix} M_x/T_2 \\ M_y/T_2 \\ (M_z - M_0)/T_1 \end{bmatrix}. \tag{2}$$

The solution to this equation simply adds an exponential decay term to our solution of Eq. 1. We choose an initial magnetization of zero along the the z-axis, i.e. $M_0 = 0$ and apply a constant field of strength B_0 along the z-axis and a changing field in the xy-plane. The solution for $1/T_1 = 0$ then becomes $\vec{M}(t) = e^{-t/T_2}[M_x(0)\cos(\omega_0 t) + M_y(0)\sin(\omega_0 t), -M_x(0)\sin(\omega_0 t) + M_y(0)\cos(\omega_0 t), 0]^T$, i.e. the x- and y-components gradually decay with characteristic time T_2 back to 0. In order to evolve $\vec{M}$ numerically in time, we rewrite Eq. 2 in matrix form:

$$\frac{d\vec{M}}{dt} = \left(\gamma \begin{bmatrix} 0 & B_0 & -B_y(t) \\ -B_0 & 0 & B_x(t) \\ B_y(t) & B_x(t) & 0 \end{bmatrix} - \begin{bmatrix} 1/T_2 & 0 & 0 \\ 0 & 1/T_2 & 0 \\ 0 & 0 & 1/T_1 \end{bmatrix} \right) \begin{bmatrix} M_x \\ M_y \\ M_z \end{bmatrix} = \hat{H}\vec{M}. \tag{3}$$

This is a standard matrix differential equation with solution $\vec{M}(t) = e^{t\hat{H}}\vec{M}_0$. We use the symmetrical second order approximation $e^{t\hat{H}} = e^{t(\hat{B}+\hat{A})} \approx e^{t\hat{A}/2}e^{t\hat{B}}e^{t\hat{A}/2}$ and note the special structures of the two matrices. $\hat{B}$ is skew-symmetric and therefore its matrix exponential is unitary. $\hat{A}$ is diagonal, and therefore its exponential is simply the matrix of exponentiated diagonal elements. We can see that $\hat{B}$, having determinant 1 is the generator of unitary dynamics while $\hat{A}$ for parameters $1/T_1 \vee 1/T_2 \neq 0$ has determinant less than 1 and is therefore generator of non-unitary dynamics thus we can see that $||\vec{M}(t)|| \leq 1$ at all times. We expand $\hat{B}$ into exponential series, note that $\hat{B}^2 = -\Omega^2\hat{B}$ and can thus rewrite

$$e^{t\hat{B}} = \frac{1}{\Omega^2}\begin{bmatrix} B_x^2 + a(B_y^2+B_z^2) & bB_xB_y + dB_z & bB_xB_z - dB_y \\ bB_xB_y - dB_z & B_y^2 + a(B_x^2+B_z^2) & bB_yB_z + dB_x \\ bB_xB_z + dB_y & bB_yB_z - dB_x & B_z^2 + a(B_y^2+B_x^2) \end{bmatrix}, \tag{4}$$

for $\Omega = ||\vec{B}(t)||$, $a = \cos(\Omega t)$, $b = 1-\cos(\Omega t)$ and $c = \Omega \sin(\Omega t)$. Now, we trotterize the time-evolution operator $e^{t\hat{H}}$, i.e. we split it into small time-step operators which we then apply sequentially to the magnetization vector. As $\hat{B}$ is time dependent however, we must consider it to be constant within each small time step $\tau = T/m$ for large Trotter number m and small simulation time interval T. This is actually a physical condition, as experimentally we would also apply pulses step-wise and not truely continuously. Finally, our trotterized time-evolution reads

$$\vec{M}(T) = \hat{U}(T)\vec{M}(0) \approx \prod_{i=0}^{m-1} e^{\tau\hat{A}/2}e^{\tau\gamma\hat{B}(T-(i+1/2)\tau)}e^{\tau\hat{A}/2}\vec{M}(0). \tag{5}$$

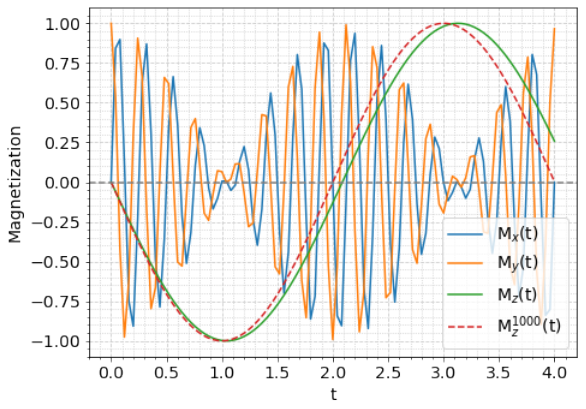
Our algorithm applies the terms in the product sequentially, starting with $e^{\tau\hat{A}/2}e^{\gamma\tau\hat{B}(\tau/2)}e^{\tau\hat{A}}$ to an initial magnetization $\vec{M}(0)$ and a certain specification of parameters in our model.

Simulation results

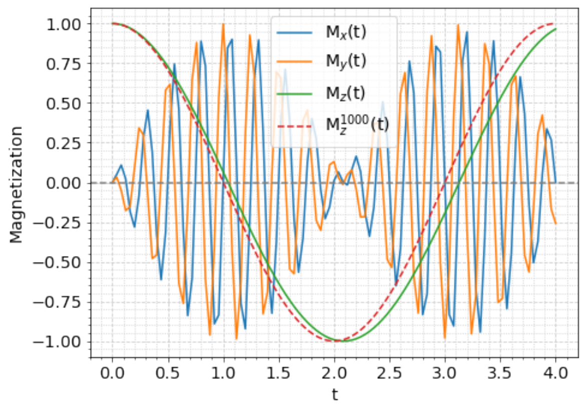
We run our algorithm for certain combinations of three initial conditions $\vec{M}_i(0)$, four different decay times $1/T_1 = j, 1/T_2 = k$ for $j,k = \{0,1\}$ and three different initial phase angles $\phi = 0, \pi/2, \pi/8$. The fixed parameters of the model are $f_0 = 4$, $f_1 = 1/4$, $B_0 = 2\pi f_0$, $h = 2\pi f_1$, $\gamma = 1$ and therefore $\omega_0 = B_0$. Furthermore, our magnetic field is always $\vec{B}(t) = [h\cos(\omega_0 t), -h\sin(\omega_0 t), B_0]^T$ and we use $m = 100$ time-steps for our simulation for a total time interval of $T = 4$.

No relaxation and decoherence ($T_1 = T_2 = \infty$)

In this case, starting from $\vec{M}(0) = [0,1,0]^T$ we expect a full CCW $\pi/2$-rotation around the x-axis after $t = 1$ due to the dynamic field, thus $\vec{M}(t=1) = [0,0,-1]^T$. Conversely, if we start with $\vec{M} = [0,0,1]^T$ we expect $M_z = -1$ after a π-rotation, i.e. after twice as long time (Fig. 1). For Trotter number $m = 100$ we see an error which increases with advancing simulation time. If we choose $m = 1000$, for which we plotted only the z-component for reference this error is starkly diminished. For the case where



(a) $\vec{M}(0) = (0,1,0)^T$



(b) $\vec{M}(0) = (0,0,1)^T$

Figure 1. Time evolution of magnetization vector components for no relaxation and decoherence for different initial magnitization components. M_z^{1000} is the z-component for 1000 simulation steps, i.e. higher temporal resolution.

$\vec{M}(0) = [1,0,0]^T$, i.e. in which the initial magnitization is aligned with the axis of the dynamic control field (Fig. 2) we see no effect on M_z and only M_x and M_y are changing due to precession around the z-axis due to the presence of the static field.

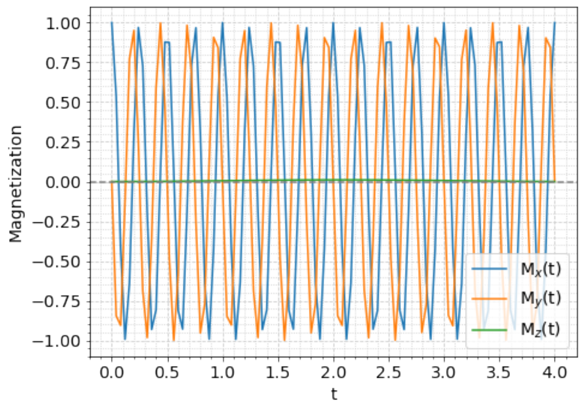
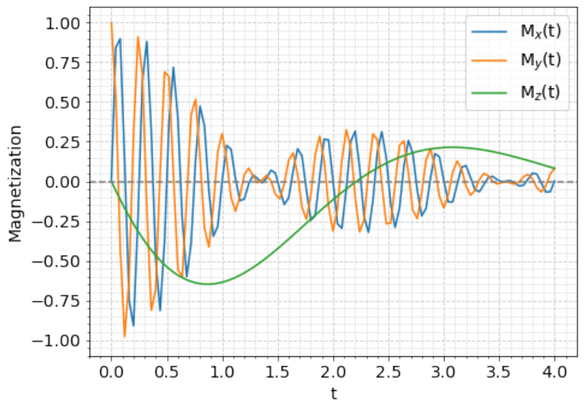


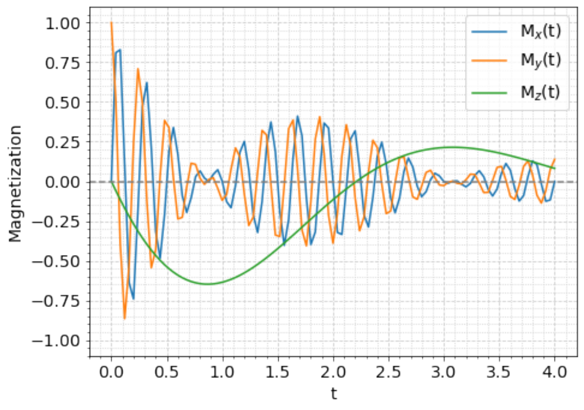
Figure 2. Time evolution for initial magnetization vector $\vec{M} = (1,0,0)^T$.

Different decoherence and relaxation scenarios

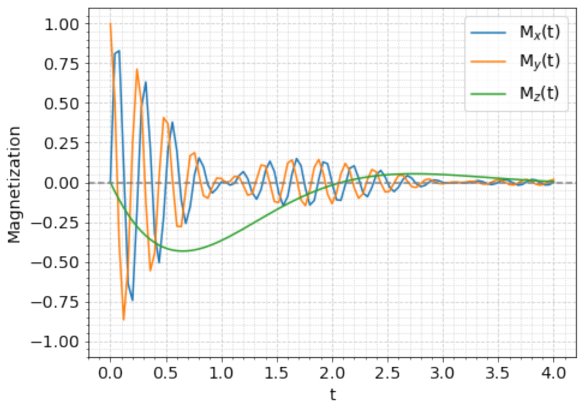
For all three different scenarios we use the initial magnetization $\vec{M}(0) = [0,1,0]^T$ in the xy-plane, to see T_1- and T_2-decay and in order to compare scenarios for the same initial condition. The simulation results can be seen in Fig. 3. We can clearly see the decaying effect which is similar for the scenarios in Figs. 3a–3b and strongest for the scenario in Fig. 3c, as expected.



(a) Only T_1



(b) Only T_2

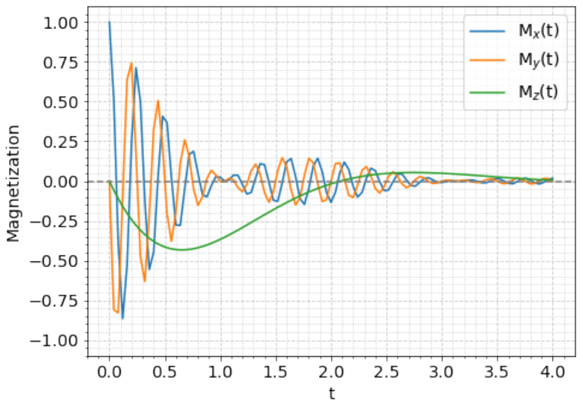


(c) Both T_1 and T_2

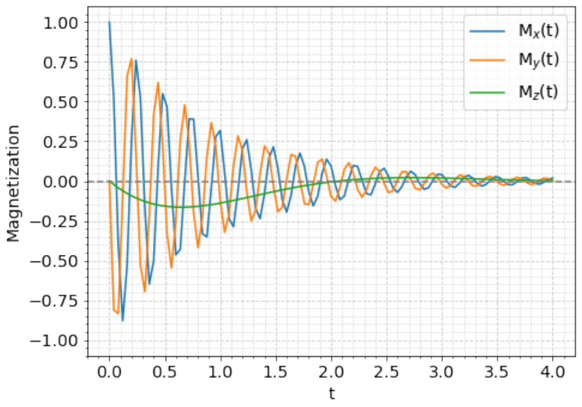
Figure 3. Time evolution of $\vec{M}(0) = (0,1,0)^T$ for three different decay scenarios.

Different initial phase angles and initial magnetizations

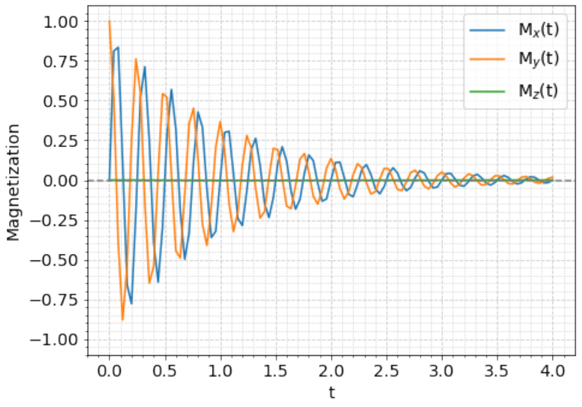
Finally, we plot the evolutions for different initial phase angles and magnetizations. The results are shown in the two columns of Fig. 4 for two different initial phase angles ϕ. For these plots we only considered the case in which we have both decoherence and relaxation. We can see that for an initial angle of $\phi = \pi/2$ in Fig. 4 (left column) the role of the M_x and M_y components



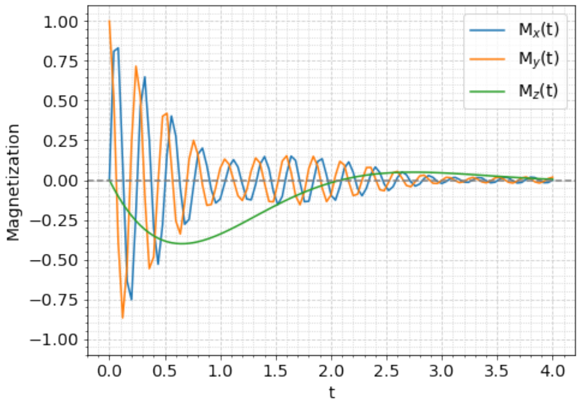
(a) $\vec{M}(0) = (1,0,0)^T$



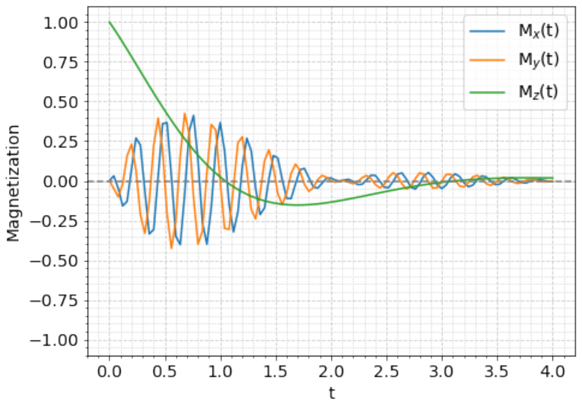
(b) $\vec{M}(0) = (1,0,0)^T$



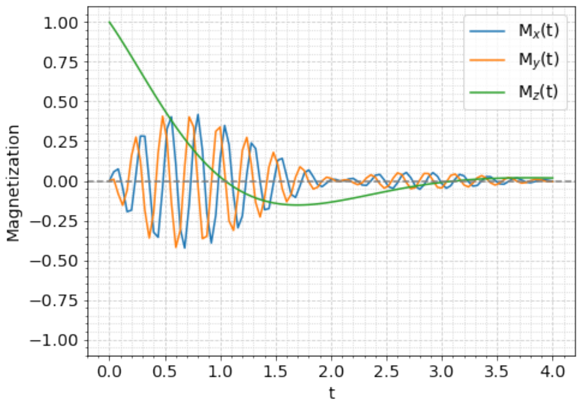
(c) $\vec{M}(0) = (0,1,0)^T$



(d) $\vec{M}(0) = (0,1,0)^T$



(e) $\vec{M}(0) = (0,0,1)^T$



(f) $\vec{M}(0) = (0,0,1)^T$

Figure 4. Time evolution for three different initial magnetizations for initial phase $\phi = \pi/2$ in the left column and $\phi = \pi/8$ in the right column.

have changed. In reference to Fig. 3, we see the same behavior in 3c but for initial magnetization along the x-axis instead of y-axis (Fig. 4a). It turns out that the phase shift has the effect of CW rotating both the x- and y-basis vectors of $\vec{M}$ in the xy-plane by 90°. For this reason we furthermore observe not much difference for the time evolution starting with magnetization in the z-direction (Figs. 4e and 4f). As the axis of rotation in the xy-plane is the z-axis, the z-component of the magnetization itself is invariant under such a transformation and is thereby not affected by the phase shift. We only see a difference in the time evolution of the x- and y-components as those are affected by the phase shift.

Discussion

In summary, we developed an algorithm for simulating the time evolution of the magnetization vector of a material subjected to static and dynamic magnetic fields by making use of the Bloch equations in matrix form and the product formula approach. We showed our simulation results for different initial magnetizations, relaxation and decoherence scenarios and initial phase angles and explained the consequences. Furthermore, we observed for $m = 100$ an error which increased with advancing simulation time, which we can however reduce by choosing smaller time steps. We showed this to be the case for $m = 1000$. Overall, our simulation results match very well with our expectation for every case. We further notice that the algorithm produces (sub)-unitary time evolution and is therefore unconditionally stable.

Appendix

Common libraries and parameters

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib
font = {'size': 14}
matplotlib.rc('font', **font)

# parameters
gamma = 1
f_0 = 4
f_1 = 1/4
h = 2*np.pi * f_1 # field strength
B_0 = 2*np.pi * f_0 # Z-component of B-field
omega_0 = B_0 # resonance frequency

m = 100 # time steps
T = 4 # simulation time
tau = T / m # time step size
ts = np.arange(0,T+tau,tau) # time range
```

Simulation

```
from common import *

# T1, T2 decay step unitary
T_1_inv, T_2_inv = 1, 1
C = np.exp( - 0.5 * tau * np.array([T_2_inv, T_2_inv, T_1_inv]) )

Ms = [] # track M vector per time step
M = np.array([0,0,1]) # inital condition
Ms.append(M) # save initial M

# initial B-field phase offset
phi = np.pi / 8

for i in range(0,m):

    # time for B field update
    t = (i+0.5)*tau

    # update B field
    Bx = h * np.cos(omega_0 * t + phi)
    By = -h * np.sin(omega_0 * t + phi)
    Bz = B_0

    # precompute factors
    omega2 = Bx**2 + By**2 + Bz**2
    omega = np.sqrt(omega2)
    a = np.cos(omega * tau * gamma)
    b = 1 - a
    c = omega * np.sin(omega * tau * gamma)

    # precompute field products
    BxBy = Bx * By
    BxBz = Bx * Bz
    ByBz = By * Bz
    Bx2 = Bx**2
    By2 = By**2
    Bz2 = Bz**2

    # time step unitary
    U = 1/omega2 * np.array([[Bx2 + a*(By2 + Bz2), b*BxBy + c*Bz,            b*BxBz - c*By],
                             [b*BxBy - c*Bz,       By2 + a*(Bx2 + Bz2),      b*ByBz + c*Bx],
                             [b*BxBz + c*By,       b*ByBz - c*Bx,      Bz2 + a*(By2 + Bx2)]])

    # update magnetization
    M = C * M
    M = U @ M
    M = C * M

    # save magnetization
    Ms.append(M)

# Plotting
plt.plot(ts,Ms);
```