Computational Physics - Exercise 7

Joel Schumacher, 309231, joel.schumacher@rwth-aachen.de

July 14, 2017

Nuclear magnetic resonance

Introduction

In this exercise the goal is to solve the Bloch equations, describing the time evolution of a nuclear magnetic moment $\vec{M}$ in a time-dependent magnetic field $\vec{B}(t)$.

$$\frac{dM_x(t)}{dt} = \gamma(\vec{M}(t) \times \vec{B}(t))_x - \frac{M_x(t)}{T_2}$$
$$\frac{dM_y(t)}{dt} = \gamma(\vec{M}(t) \times \vec{B}(t))_y - \frac{M_y(t)}{T_2}$$
$$\frac{dM_z(t)}{dt} = \gamma(\vec{M}(t) \times \vec{B}(t))_z - \frac{M_z(t) - M_0}{T_1}$$

with $\vec{B}(t) = \left(B^x(t), B^y(t), B^0\right)$ and $B^0 \gg |B^x(t)|, |B^y(t)|$, B^0 being a large static field. γ is the gyromagnetic ratio.

T_1 is then the spin-lattice relaxation corresponding to the time scale of the realignment of the spins (the alignment of $\vec{M}$ with the direction of B^0) and T_2 is the spin-spin relaxation corresponding to the decrease of the x-y component of $\vec{M}$.

For this system we expect the magnetic moments to precess around B^0 with the Larmor frequency $f = \frac{\gamma}{2\pi}B^0$ if they are not perfectly aligned. If the spins are excited resonantly with the Larmor frequency, they can be "turned" into our out of the x-y plane, which results in an oscillation between a parallel and an anti-parallel state.

In practical applications resonant excitation turns the magnetic moments into the x-y plane and their precession around the z-axis induce a current in a coil surrounding the sample, which is then measured.

Simulation model

The main approximation of our simulation is that the time-dependent magnetic field $\vec{B}$ is piecewise constant in short time intervals of length τ. If $\vec{B}$ is constant and relaxation is ignored the Bloch equation simplifies to:

$$\frac{d\vec{M}(t)}{dt} = \gamma \boldsymbol{B} \vec{M}$$

with:

$$\boldsymbol{B} = \begin{pmatrix} 0 & B^z & -B^y \\ -B^z & 0 & B^x \\ B^y & -B^x & 0 \end{pmatrix}$$

which can be formally solved with:

$$\vec{M}(t) = \exp(t\gamma\boldsymbol{B})\vec{M}(t=0)$$

with our piecewise approximation we can then evolve $\vec{M}(t=0)$ repeatedly for small time steps τ:

$$\vec{M}(t+\tau) = \exp\left(\tau\gamma\boldsymbol{B}(t+\tau/2)\right)\vec{M}(t)$$

to incorporate relaxation, $\boldsymbol{B}$ is modified in the following manner:

$$\boldsymbol{B} \to C/2 + \boldsymbol{B} + C/2$$

$$C = -\begin{pmatrix} \frac{1}{\gamma T_2} & 0 & 0 \\ 0 & \frac{1}{\gamma T_2} & 0 \\ 0 & 0 & \frac{1}{\gamma T_1} \end{pmatrix}$$

For our simulation the magnetic field will have the form:

$$\vec{B}(t) = \left(h\cos\left(\omega_0 t + \phi\right), -h\sin\left(\omega_0 t + \phi\right), B^0\right)$$

The parameters will be

$$\begin{aligned} f_0 &= 4 \\ f_1 &= 1/4 \\ B^0 &= 2\pi f_0 \\ h &= 2\pi f_1 \\ \gamma &= 1 \\ \omega_0 &= B^0 \end{aligned}$$

Simulation results

The time evolution for $\vec{M}(t=0) = (0,1,0)$, $\phi = 0$ and for different combinations of 0 and 1 for the inverse relaxation times can be found in the figures 1, 2, 3 and 4.

Plots with different starting orientations can be found in the figures 5, 6 and 7. Different choices of rotations are depicted in figure 8 and 9.

A comparison plot with it's parameters chosen equal to the example plot in the exercise slides is visible in 10.

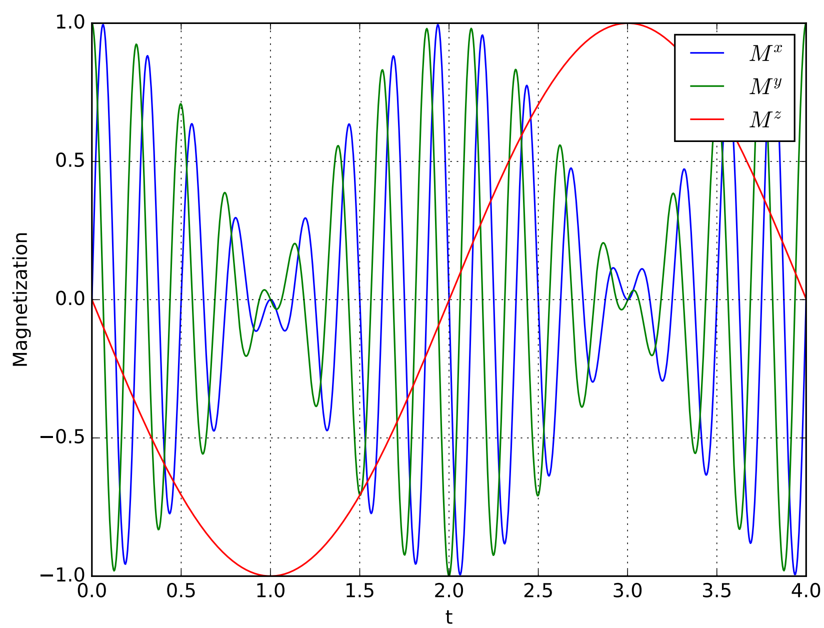

Figure 1: The time evolution of the magnetization with $1/T_1 = 0$, $1/T_2 = 0$, $\vec{M}(t=0) = (0,1,0)$, $\phi = 0$.

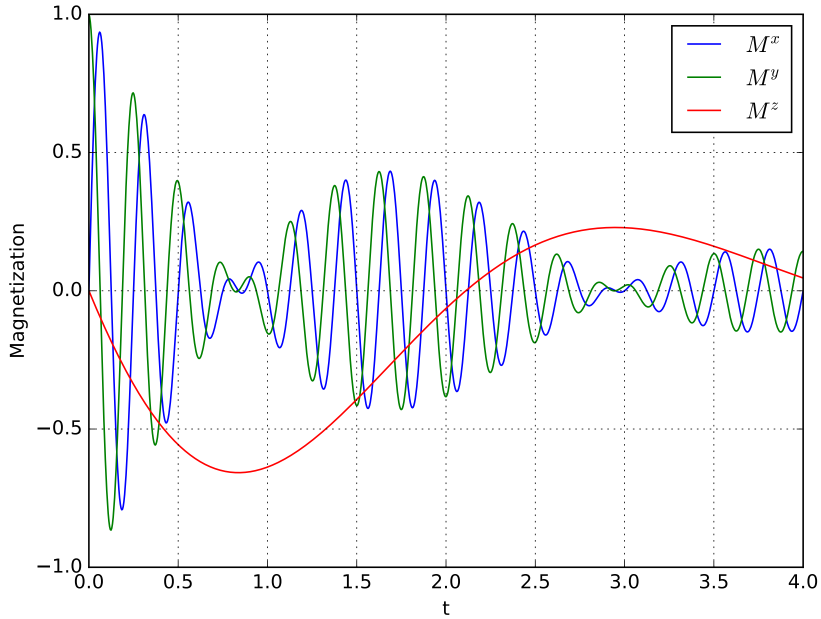

Figure 2: The time evolution of the magnetization with $1/T_1 = 0$, $1/T_2 = 1$, $\vec{M}(t=0) = (0,1,0)$, $\phi = 0$.

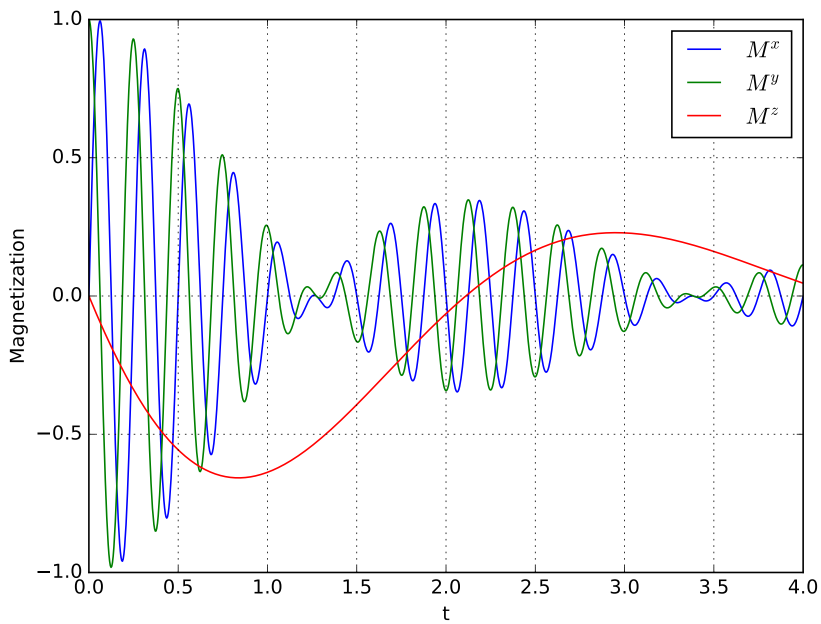

Figure 3: The time evolution of the magnetization with $1/T_1 = 1$, $1/T_2 = 0$, $\vec{M}(t=0) = (0,1,0)$, $\phi = 0$.

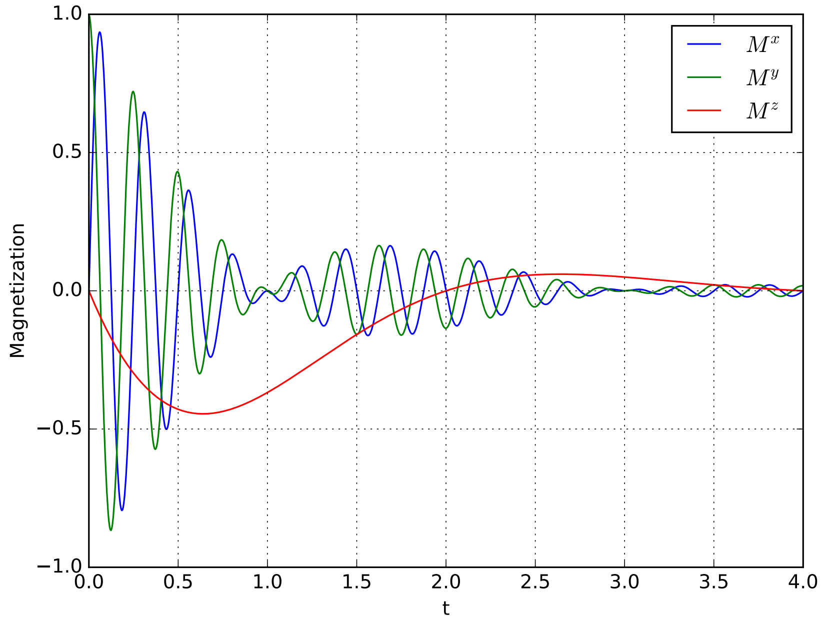

Figure 4: The time evolution of the magnetization with $1/T_1 = 1$, $1/T_2 = 1$, $\vec{M}(t = 0) = (0, 1, 0)$, $\phi = 0$.

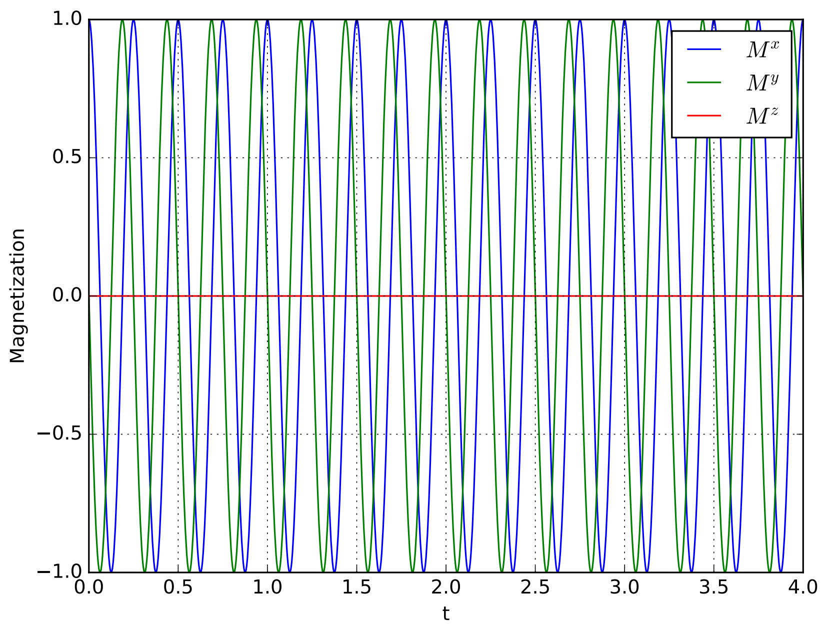

Figure 5: The time evolution of the magnetization with $1/T_1 = 0$, $1/T_2 = 0$, $\vec{M}(t = 0) = (1, 0, 0)$, $\phi = 0$.

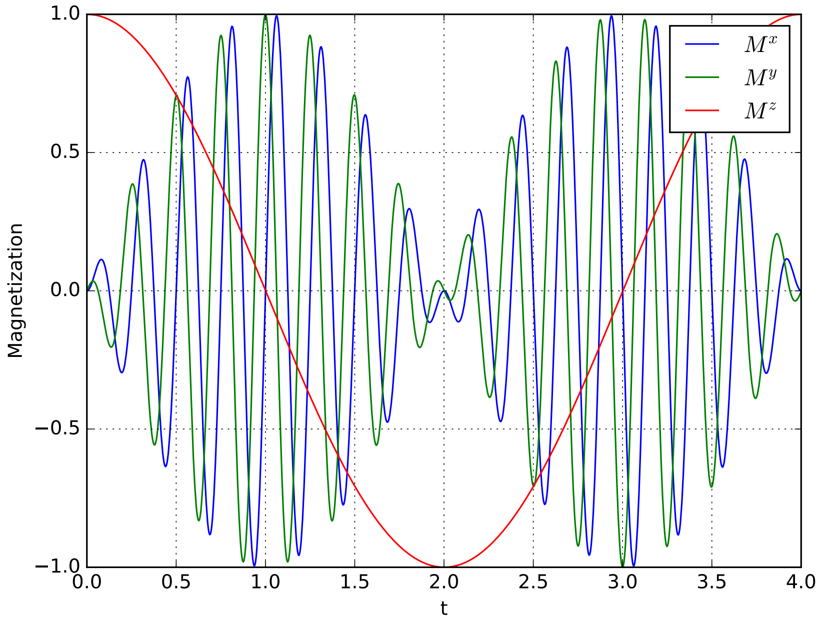

Figure 6: The time evolution of the magnetization with $1/T_1 = 0$, $1/T_2 = 0$, $\vec{M}(t = 0) = (0, 0, 1)$, $\phi = 0$.

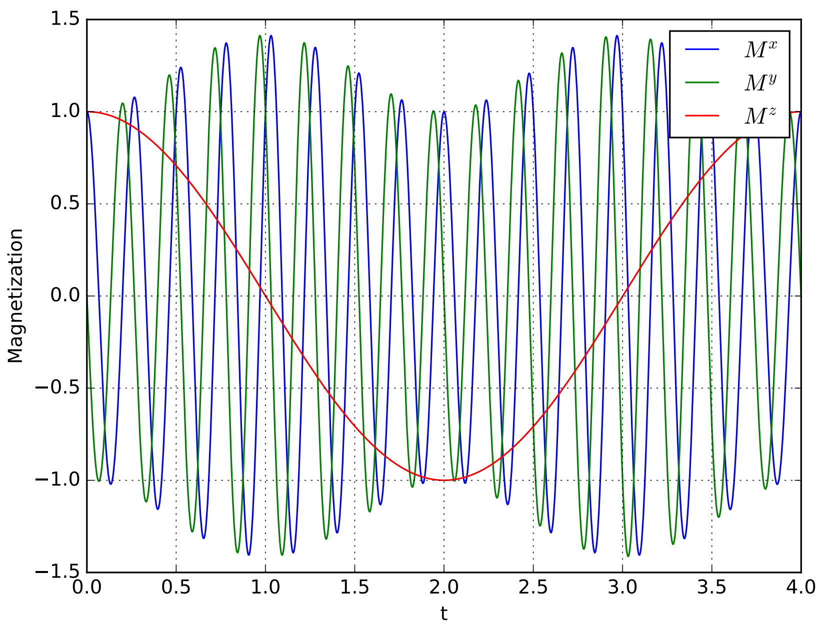

Figure 7: The time evolution of the magnetization with $1/T_1 = 0$, $1/T_2 = 0$, $\vec{M}(t = 0) = (1, 0, 1)$, $\phi = 0$.

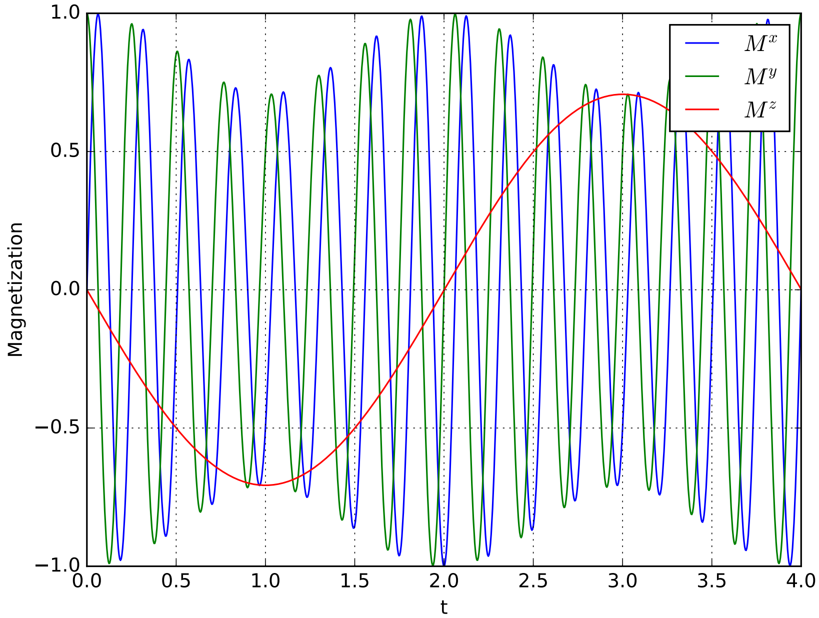

Figure 8: The time evolution of the magnetization with $1/T_1 = 0$, $1/T_2 = 0$, $\vec{M}(t=0) = (0, 1, 0)$, $\phi = \pi/4$.

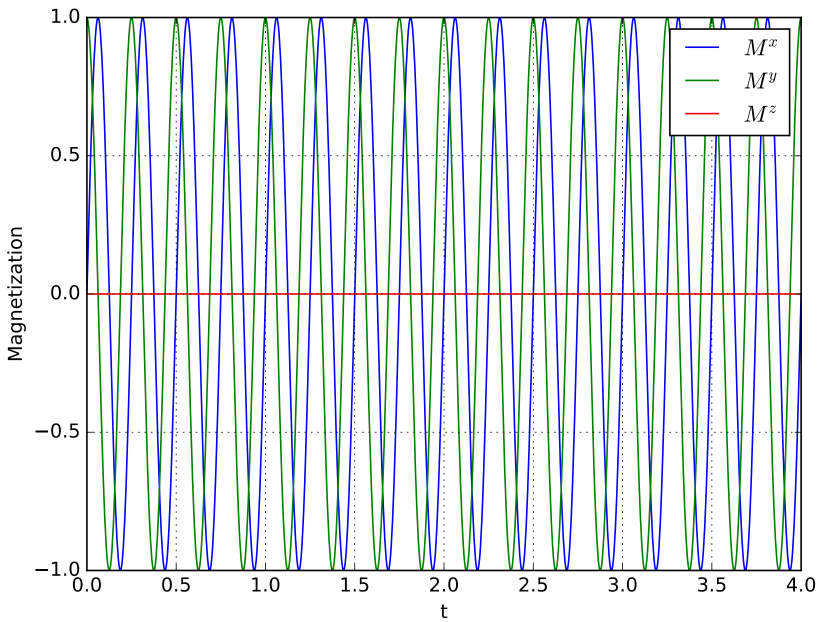

Figure 9: The time evolution of the magnetization with $1/T_1 = 0$, $1/T_2 = 0$, $\vec{M}(t=0) = (0, 1, 0)$, $\phi = \pi/2$.

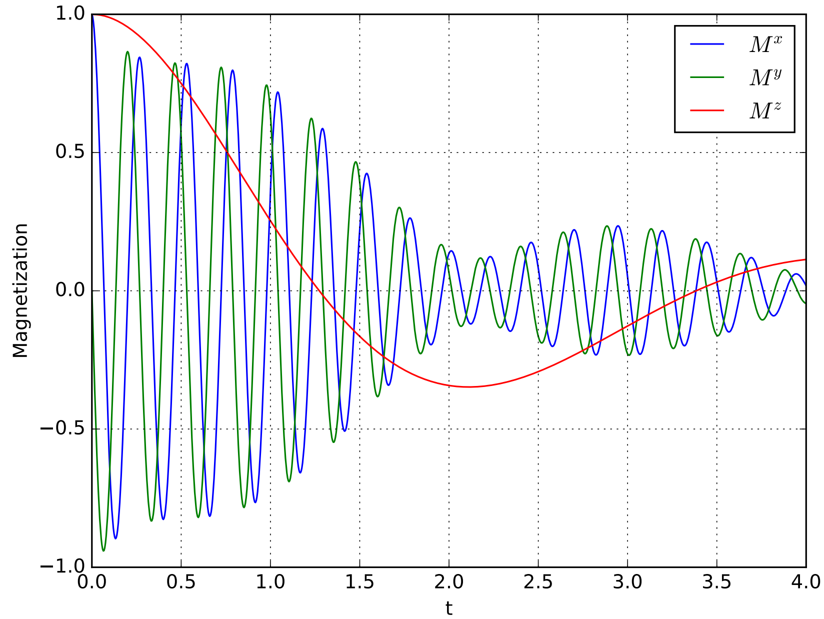

Figure 10: The time evolution of the magnetization with $1/T_1 = 0$, $1/T_2 = 1$, $\vec{M}(t = 0) = (1, 0, 0)$, $\phi = 0$.

Discussion

In the first four plots it can be observed how the finite relaxation times increase the decay speed of the oscillation.

For a starting orientation of $\vec{M}(t = 0) = (1, 0, 0)$ no beat in the x and y component can be observed since the rotation of the magnetic field is around the x-axis. The x-component is constant. For $\vec{M}(t = 0) = (0, 0, 1)$ we see the same plot as for $\vec{M}(t = 0) = (0, 1, 0)$, but with a phase difference since the same process (rotation around x-axis) is essentially observed but starting from a different point in time. For $\vec{M}(t = 0) = (1, 0, 1)$ we see the beat never reaching zero and no full alignment of the z-direction of the spin. The comparison plot matches well with the example plot from the exercise. In conclusion nuclear magnetic resonance can be simulated using the already known tools to us in the form of partial linearization and formally solving differential equations with matrix exponentials. This procedure is a simple and computationally inexpensive algorithm with reasonable accuracy that is acceptable even with big time steps.

Appendix

```
import math
import numpy as np
import matplotlib.pyplot as plt

invT1 = 0
invT2 = 0
M0 = [0, 1, 0]
phi = 0 # in units of pi

######

f0 = 4
f1 = 1/4
B0 = 2*math.pi*f0
```

```
h = 2*math.pi*f1
gamma = 1
omega0 = B0

t_min = 0
t_max = 4
t_range = t_max - t_min
steps = 1000
tau = t_range / steps

def _B(t):
    arg = omega0*t + math.pi*phi
    return np.asarray([h*math.cos(arg), -h*math.sin(arg), B0])

def _expC():
    expT1 = math.exp(-tau/2 * invT1)
    expT2 = math.exp(-tau/2 * invT2)
    return np.diag([expT2, expT2, expT1])

def _expB(t, B):
    omegaSqr = B[0]*B[0] + B[1]*B[1] + B[2]*B[2]
    omega = math.sqrt(omegaSqr)
    BB = np.outer(B, B)
    c = math.cos(omega*t)
    s = math.sin(omega*t)

    ret = BB * (1 - c)
    ret += np.eye(3) * omega * omega * c
    v = omega * s * B
    # essentially: M_ij = eps_ijk * v_k (eps being the levi-civita symbol)
    ret += np.asarray([[0,v[2],-v[1]], [-v[2],0,v[0]], [v[1],-v[0],0]])

    ret /= omegaSqr

    return ret

def expB(t):
    C = _expC()
    return C @ _expB(tau, _B(t + tau/2)) @ C

t = [t_min]
M = [np.asarray(M0)]
for i in range(steps):
    M.append(np.dot(expB(t[-1]), M[-1]))
    t.append(t[-1] + tau)
M = np.asarray(M)

plt.plot(t, M[:,0], label="$M^x$")
plt.plot(t, M[:,1], label="$M^y$")
plt.plot(t, M[:,2], label="$M^z$")
plt.xlabel("t")
plt.ylabel("Magnetization")
plt.xlim(t_min, t_max)
plt.ylim(-1,1)
plt.legend()
plt.grid()
```

```
plt.savefig("invT1={},invT2={},M0={}{}{},phi={}.pdf".format(invT1, invT2, M0
    [0], M0[1], M0[2], phi))
plt.show()
```