MRI CA 1

question 1

part a

`question_1_a.m` contains the full code.
first define constants:

```
B0 = 1.5; % Tesla ( not used in rotating frame )
B1 = 0.05; % Gauss
B1 = B1 * 1e-04; % Gauss to Tesla
M0 = [0; 0; 1]; % Initial magnetization vector
B1_time = 7.35; % in ms
```

define the bloch equation in a function:

```
function dMdt = bloch_equations(t, M, B0, B1)
    gamma = 42.6e6; % Gyromagnetic ratio for protons in Hz/T
    B1 = [B1; 0; 0]; % converted B1 to rotating frame from book (B1i)
    B_eff = B1; % because no off resonance
    % eq 3.75 book
    dMdt = gamma * cross(M, B_eff); % Bloch equations in rotating frame
end
```

then define the ode equation and solve it (t is the step):

```
% Solve the ODE
[t, M] = ode45(@(t, M) bloch_equations(t, M, B0, B1), tspan, M0);
```

print the final values of Magnetization

```
equilibrium_M = [M(1, 1), M(1, 2), M(1, 3)]
final_M = [M(end, 1), M(end, 2), M(end, 3)]
```

3 types of plotting with quiver3, 2dplot and trajectory plot are done.

part b

`question_1_b.m` contains the full code.
same as part a, we just modify the `bloch_equations` function to calculate off resonance aswell.

```
% Bloch equations
function dMdt = bloch_equations(t, M, B0, B1)
    off_resonance = 1e3; % off resonance
    gamma = 42.6e6; % Gyromagnetic ratio for protons in Hz/T
    B1 = [B1; 0; 0]; % converted B1 to rotating frame from book (B1i)
    w0 = (gamma + off_resonance) * B0;
    w_rf = gamma * B0;
    delta_w = abs(w0 - w_rf);
```

```
    B_eff = B1 + (B0 - delta_w) / gamma * [0; 0; 1]; % because of off resonance eq
    % eq 3.75 book
    dMdt = gamma * cross(M, B_eff); % Bloch equations in rotating frame
end
```

question 2

part a

3 functions for magnetization rotation:

```
function Rx = Rx(flip)
Rx = [1 0 0; 0 cos(flip)  sin(flip); 0 -sin(flip)  cos(flip)];
end
```

```
function Ry = Ry(flip)
Ry = [cos(flip)  0 sin(flip); 0 1 0; -sin(flip) 0 cos(flip)];
end
```

```
function Rz = Rz(flip)
Rz = [cos(flip) -sin(flip) 0; sin(flip) cos(flip) 0; 0 0 1];
end
```

part b

function to define relaxation times with a matrix form

```
function [Mend] = bloch_relax(Mstart, T, M0, T1, T2)
Arelax = [exp(-T/T2) 0 0; ...
          0 exp(-T/T2) 0; ...
          0 0 exp(-T/T1)];
brecover = [0; 0; M0*(1-exp(-T/T1))];
Mend = Arelax*Mstart + brecover;
end
```

part c

`question_2_c.m` contains the code.
extra helper functions:
rotates the Magnetization based on a B1 field and B1 duration.

```
function [Mend] = bloch_rotate(Mstart, T, B)
GAMMA = 42.58; % kHz/mT
flip = 2*pi*GAMMA * norm(B) * T;
eta = acos(B(3) / (norm(B)+eps));
theta = atan2(B(2), B(1));
Mend = Rz(-theta)*Ry(-eta)*Rz(flip)*Ry(eta)*Rz(theta)* Mstart;
end
```

rotates the Magnetization based on a B1 field and B1 duration.

```
function [Mend] = bloch_rftip(Mstart, T, B1)
Mend = bloch_rotate(Mstart, T, [real(B1) imag(B1), 0]);
end
```

define constants:

```
% divide into thousand so each t is a ms
t = linspace(0, 1, 1000); % s
% total Magnetization at eq
M0 = 1;
% define T1 T2 in s
T1 = 1.;  T2 = .1; % s
% define M
M_equilibrium = [0, 0, M0].';

gammabar = 42.58; % kHz/mT
T = 1; % 1 ms pulse duration

% tau times
relaxation_time_1 = 15; %ms
relaxation_time_2 = 10; %ms
```

calculate B1 based on desired flip angle:

```
flip = 60;
B10 = flip*pi/180 / (2*pi*gammabar*T);
% for y axis
% B10 = (flip*pi/180 / (2*pi*gammabar*T)) * 1i
```

save the results in a cell array each time:

```
Magnetization_history = {};
Magnetization_cmt = {};
```

do relaxation times as needed

```
for It = 1:relaxation_time_1
    M_rel_1(:,It) = bloch_relax(M_after_60_x,t(It),M0,T1, T2);
end
```

do the same for the 45 degree flip.

part d

is essentially the same we just change these lines:

```
wait_time = 3;

% tau times
relaxation_time_1 = 15 + wait_time; %ms
relaxation_time_2 = 10 + wait_time; %ms
```

final value comparison:
part c:

```
M_after_relaxation_2 =

   -0.4412
   -0.0013
   -0.8589
```

part d:

```
M_after_relaxation_2 =

   -0.4281
   -0.0013
   -0.8534
```