A

```
%% Determine R1 of water as complementary fitting parameter

experiment = '251104_Demo';
folder = strcat('/Users/jschmoll/Documents/nmrdata/sept/', experiment,'/');%path to the NMR directory of the inversion recovery experiment
T1_exp = ''; %experiment number of the inversion recovery experiment
P.R1w = Inversion_Recovery(experiment, T1_exp); %performs the inversion recovery and stores water R1 rate in P.R1w

%% loading additional parameters for fitting:

P.water_CS = 4.699;
offsets = importdata('offsets.mat');
P.offsets = ((offsets(51:92)-4.7)*700)*2*pi;
P.ref_frequency=699.97; %B0 frequency of the spectrometer


P.T2_agarose=1.699e-05;
P.R2a=1/P.T2_agarose; %agarose parameters, determined previously via referenceMTpool script

P.R1a = 1; %R1 of MT-Pool 1 (agarose)
P.tp=5;  %duration of cw saturation
P.water=1; %relative population of water

P.kaw=112.4;
P.kwa=0.0744;
P.agarose = P.kwa/(P.kwa+P.kaw); %exchange kinetics of the agarose pool

w1=[100 170 250 330 500 650 780 1000]*2*pi; %saturation amplitudes
P.M0w=1; %relative Z magnetization

omega_a_ppm=5.3;
omega_a=(omega_a_ppm-P.water_CS)*P.ref_frequency*2*pi;
P.delta_w_w=P.offsets;
P.delta_w_a=P.offsets-omega_a; %chemcial shift center of the agarose pool


integral_water_real_all = integral_water_real_all(:,51:92); %if two-sided experiment was recorded: subset the upfield offsets for fitting
```

Background parameters from previous CONDENSE-MT analysis in absence of the condensates

B

```
%% Condense-MT fitting

signal_ideal=@(fit) [function_2MT(P,w1(1),fit),...
                     function_2MT(P,w1(2),fit),...
                     function_2MT(P,w1(3),fit),...
                     function_2MT(P,w1(4),fit),...
                     function_2MT(P,w1(5),fit),...
                     function_2MT(P,w1(6),fit),...
                     function_2MT(P,w1(7),fit),...
                     function_2MT(P,w1(8),fit)]'; %function for 2 independent MT-pools


error=@(fit) norm(integral_water_real_all-signal_ideal(fit)); % error between experimental data and analytical expression
fit0=[150 0.10 1e-5 7.5]; %starting parameters for optimization
options=optimset('MaxFunEvals', 100000, 'MaxIter',100000, 'Display', 'on', 'TolX', 1e-0012,'PlotFcns',@optimplotfval); %settings for fminsearch
fit=fminsearch(error,fit0,options); %optimization
result = error(fit); %resulting parameters after optimization

%% Parameter uncertainities via jacobian first-order derivative matrix
dof=length(integral_water_real_all(1,:))-2;

for i=1:length(integral_water_real_all(:,1))
    fitted_data(i,:)=function_2MT(P,w1(i),fit); %predicted data using best-fit parameters
end

sdr=sqrt(sum((integral_water_real_all-fitted_data).^2)/dof);
J=jacobianest(signal_ideal,fit); %numerically deriving first-order jacovian matrix

for j=1:length(fit)
variance(:,j)=sdr.^2*inv(J(:,j)'*J(:,j)); %variance - covarance matrix
sigma = sqrt(diag(variance(:,j)))'; %parameter uncertainty from diagonal elements of jacobian matrix
sigma_average(j)=sqrt(trace(sigma^2)/length(sigma));
end

P.droplets = fit(2)/(fit(2)+fit(1)); %extract the rel. population of condensed RNA-protons according to proton exchange kinetics
error_droplets = P.droplets*(sigma_average(2)/fit(2)+sigma_average(1)/fit(1));
```

C

```
%% plotting of experimental data and model predictiona after computational optimization

cmap = [linspace(0, 1, 8)', linspace(0, 0.0, 8)', linspace(1, 0, 8)'];
marker =  ['*','o', '<', '+', '>', 'x', '^', 'v'];
offsets = P.offsets;
legend_entries = gobjects(1,length(w1));
irradiation_strengths = {'100 Hz', '170 Hz', '250 Hz', '330 Hz', '500 Hz', '650 Hz', '780 Hz', '1000 Hz'};

figure;
hold on
set(gcf, 'Units', 'inches', 'Position', [0, 0, 7, 4]); % figure dimensions

for i = 1:8
    semilogx(offsets/(2*pi)/P.ref_frequency + P.water_CS,function_2MT(P, w1(i), fit), '--','Color', [0, 0, 0], 'LineWidth', 1.5);
    hold on;
    legend_entries(i) = scatter(offsets/(2*pi)/P.ref_frequency + P.water_CS, ...
    integral_water_real_all(i,:), 50, 'Marker', 'o', 'MarkerEdgeColor', cmap(i,:), 'LineWidth',1);
    set(gca, 'xdir', 'reverse');
end

eqn = string({"R2w (Hz) = "+ sprintf(['%.', num2str(2), 'f'], fit(4)) + "±" + sprintf(['%.', num2str(2), 'f'], sigma_average(4)), ...
    "K(condensate-H2O) (Hz) = " + sprintf(['%.', num2str(1), 'f'], fit(1)) + "±" + sprintf(['%.', num2str(1), 'f'], sigma_average(1)), ...
    "K(H2O-condensate) (Hz) = " + sprintf(['%.', num2str(4), 'f'], fit(2)) + "±" + sprintf(['%.', num2str(4), 'f'], sigma_average(2)),...
    "Population MT pool = " + sprintf(['%.', num2str(5), 'f'], P.droplets) + "±" + sprintf(['%.', num2str(5), 'f'], error_droplets),...
    "T2 MT pool (us) = " + sprintf(['%.', num2str(2), 'f'], fit(3)*1e6) + "±" + sprintf(['%.', num2str(2), 'f'], sigma_average(3)*1e6), ...
    "Fit deviation: " + result});

text(-20,0.5,eqn,"HorizontalAlignment","left","VerticalAlignment","top", 'FontSize',12)
ax = gca;
ax.LineWidth = 2;
ax.FontSize = 15;
xlim([-90 5])
ylim([0 1.01])
xlabel('offset of irradiation', FontSize=18)
ylabel('rel. integral (H2O)', FontSize=18)
legend(legend_entries, irradiation_strengths, Location='southeast', box='off', FontSize=13)
```