Matlab Monday 02

Contents

- October 9th, 2019
- (a1) Let's first create a test data set with known parameters
- (b) Okay. Let's say we have this model and we have the time courses and we want to figure out the parameters! What should we do?
- (c) Once you have the parameters..

October 9th, 2019

Learning Objectives

(1) Visualizing 4D data

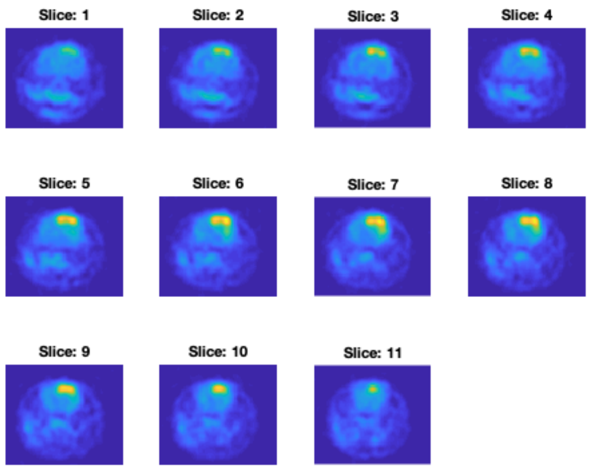
(2) Analyzing PET image data

(3) Curve fitting, parameter estimation

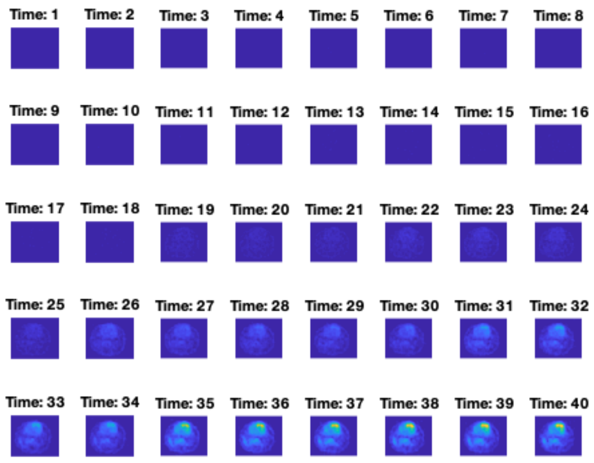
```
clear all; close all; clc;
load matlab_monday_02.mat
% The 4D array FDG_PET is an example dynamic (i.e. time resolved) PET image
% in a rat with a brain tumor.  The animal received an injection of 18-FDG.
% FDG is a tracer that closely resembles glucose.  When FDG is internalized by cells
% it gets stuck inside of the cells.  This results in an accumulation of 18-F, and
% thus our signal increases in areas with high glucose use.

% calculate image size
[sy, sx, sz, st] =  size(FDG_PET)
   % sy,sx,sz are the sizes in the x,y,and z direction
   % st is the number of time points this image volume was acquired.
```

```
Warning: Could not find appropriate
function on path loading function handle
/private/var/folders/6x/ht340mfx70gg9dzhsqnb78wh0000gp/T/Editor/LiveEditorEvaluationHelperESectionEval1c48ac2c.mlx>@(t)(a(1)*(t-tau)-a(2)-a(3)).*exp(e

sy =

    69


sx =

    81


sz =

    11


st =

    40
```

```
% Lets look at all of the slices
for z = 1:sz
    subplot(3,4,z)
    imagesc(FDG_PET(:,:,z,end),[0 max(FDG_PET(:))]); axis image; axis off;
    title(['Slice: ' num2str(z)])
end
```

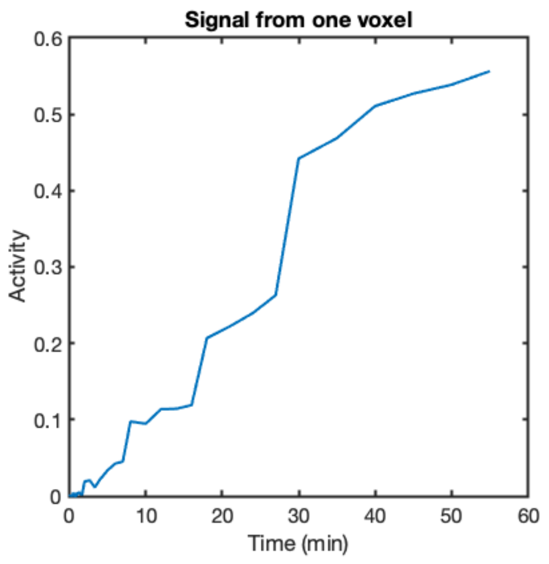


```
% Lets look at all of the time points
for z = 1:st
    subplot(5,8,z)
    imagesc(FDG_PET(:,:,6,z),[0 max(FDG_PET(:))]); axis image; axis off;
    title(['Time: ' num2str(z)])
end
```



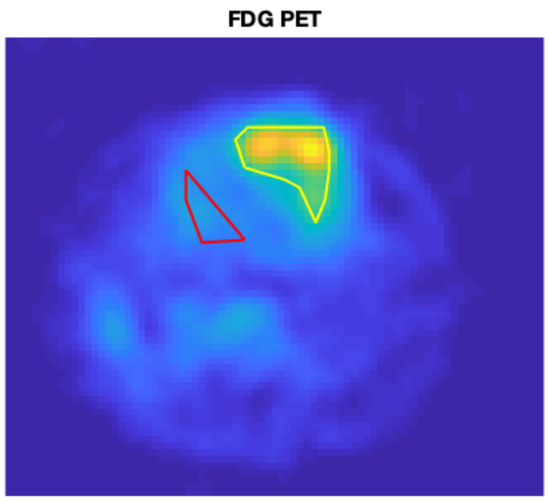
Lets just look the time course for an individual voxel.

```
figure
imagesc(FDG_PET(:,:,6,end),[0 max(FDG_PET(:))]); axis image; axis off;

plot(time_FDG,squeeze(FDG_PET(17,41,6,:)),'LineWidth',2); axis square;
xlabel('Time (min)','FontSize',20)
ylabel('Activity','FontSize',20)
title('Signal from one voxel','FontSize',20)
set(gca,'FontSize',15,'LineWidth',2)
```



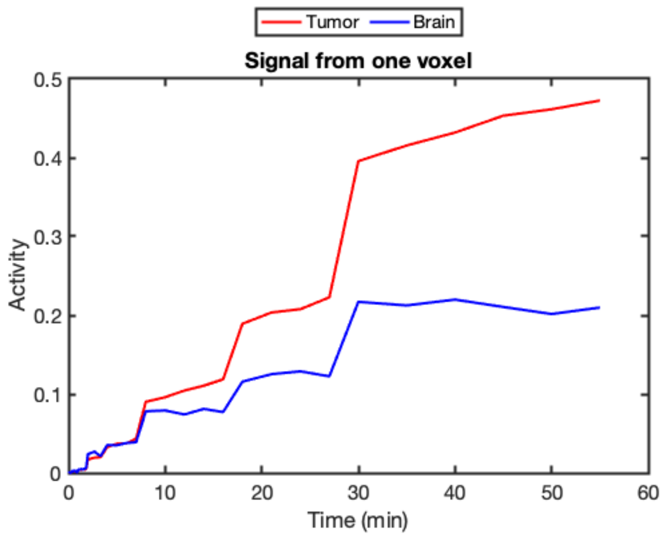
Lets look at the time course over the tumor and over healthy brain

```
imagesc(FDG_PET(:,:,6,end),[0 max(FDG_PET(:))]); axis image; axis off;

% place roi around brain
title('Draw ROI around brain','FontSize',20)
[brain_mask, brain_x,brain_y] = roipoly();

% place roi around  tumor
title('Draw ROI around tumor','FontSize',20)
[tumor_mask, tumor_x,tumor_y] = roipoly();
close all

% display rois
imagesc(FDG_PET(:,:,6,end),[0 max(FDG_PET(:))]); axis image; axis off;
axis image
colormap parula
title('FDG PET','FontSize',20)
xlabel('Position (mm)','FontSize',15)
ylabel('Position (mm)','FontSize',15)
set(gca,'FontSize',15)
hold on
plot(brain_x,brain_y,'r','LineWidth',2)
hold on
plot(tumor_x,tumor_y,'y','LineWidth',2)
hold off
```



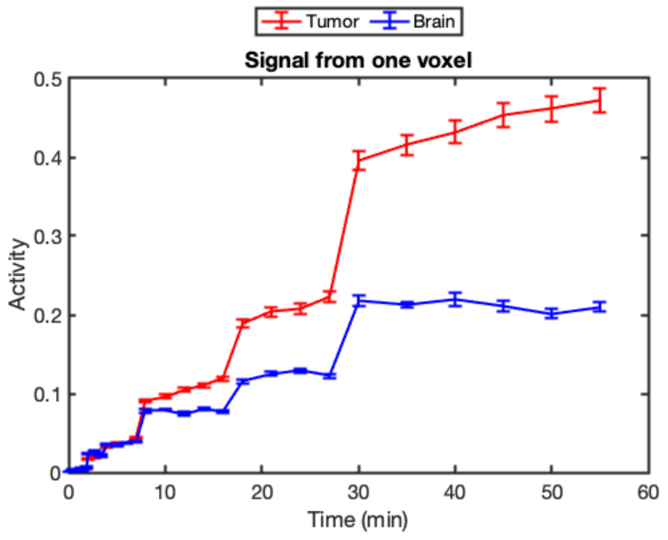
calculate the mean value at each time point

```
for t = 1:st % loop through each time point
    d = FDG_PET(:,:,6,t);  % d is just a temporary variable;

    % the below code calculates the mean value of elements within the tumor mask;
    tumor_tc(t,1) = mean(d(tumor_mask(:)));
    % calculates the 95% confidence interval +/- 1.96*standard_deviation/sqrt(n)
    tumor_tc(t,2) = 1.96*std(d(tumor_mask(:)))/sqrt(sum(tumor_mask(:)));


    % repeats this for brain
    brain_tc(t,1) = mean(d(brain_mask(:)));
    brain_tc(t,2) = 1.96*std(d(brain_mask(:)))/sqrt(sum(brain_mask(:)));

end
plot(time_FDG,tumor_tc(:,1),'-r',time_FDG,brain_tc(:,1),'-b','LineWidth',2)
xlabel('Time (min)','FontSize',20)
ylabel('Activity','FontSize',20)
title('Signal from one voxel','FontSize',20)
set(gca,'FontSize',15,'LineWidth',2)
legend('Tumor','Brain','Location','Northoutside','Orientation','horizontal')
```



can we add the confidence intervals to this plot? errorbar([Time vector], [mean value vector],[confidence interval])

```
errorbar(time_FDG,tumor_tc(:,1),tumor_tc(:,2),'-r','LineWidth',2);
hold on
errorbar(time_FDG,brain_tc(:,1),brain_tc(:,2),'-b','LineWidth',2);
hold off;
xlabel('Time (min)','FontSize',20)
ylabel('Activity','FontSize',20)
title('Signal from one voxel','FontSize',20)
set(gca,'FontSize',15,'LineWidth',2)
legend('Tumor','Brain','Location','Northoutside','Orientation','horizontal')
```

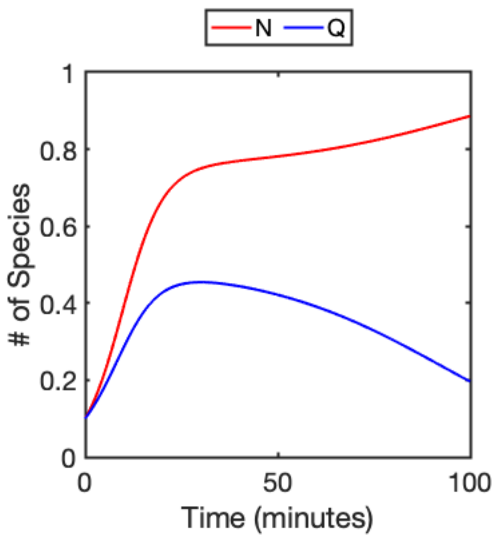


(2) Curve fitting example

```
clear all; close all; clc;
```

(a1) Let's first create a test data set with known parameters

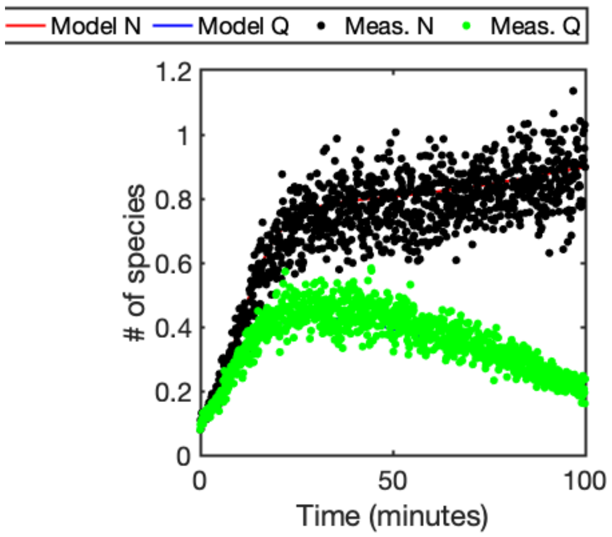
```
%{

dN/dt = param(1)*N*(1-N)  -param(2)*Q*N
(1)        (2)               (3)
(1) = change in N over time
(2) = grows logistically
(3) = Dies as an interaction with species Q

dQ/dt = param(3)*Q - param(4)*Q*N;
(4)      (5)           (6)
(4) = change in Q over time
(5) = grows exponetially
(6) = dies as an interaction with species N

%}

param_truth = [.2 .1 .15 .2];  %what are the true values of these parameters
ICs = [.1 .1];  % Initial conditions for N and Q

% (a2) Use ODE45 to solve this ode at time points "time"
time = 0:.1:100;

   [~, P] = ode45(@(t,y) two_species(t,param_truth,y),time,ICs);
%                   (1)      (2)                        (3)   (4)
%{
(0) First create a matlab function called "two_species.m"  See attached
function
(1) In ODE45 note that this function will be a function of both "t" and
"y".  These are variables used internally in ODE45.  ODE45 will pass the
current timepoint "t" and the current initial condition "y"

(2)  Place function here. t, and y do not need to be assigned prior.
params_truth does need to be assigned.
(3) This is your time vector (when do you want to have the solution for
this ode?
(4) this is the initial condition array

   %}

plot(time,P(:,1),'-r',time,P(:,2),'b','LineWidth',2)
xlabel('Time (minutes)','FontSize',20)
ylabel('# of Species','FontSize',20)
set(gca,'FontSize',20,'LineWidth',2)
axis square;
legend('N','Q','Orientation','horizontal','location','northoutside')
```



(b) Okay. Let's say we have this model and we have the time courses and we want to figure out the parameters! What should we do?

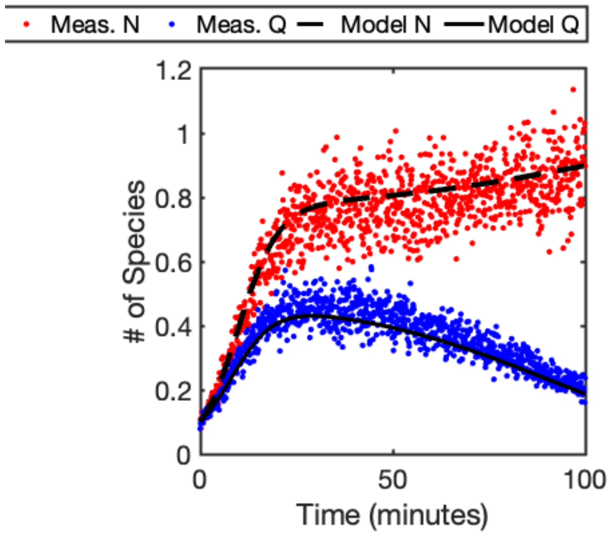
```
x_guess = .1*ones(4,1); % initial guess for parameters
ICs = [.1 .1];  % Initial conditions for N and Q
LB = zeros(4,1);  % assume it is a bounded optimization problem. Lower bounds
UB = ones(4,1);  % upper bounds
options = optimoptions(@lsqnonlin,'Display','iter','FunctionTolerance',1e-8);

noise_data(:,1) = P(:,1).*random('Normal',1,0.1,[length(time),1]);
noise_data(:,2) = P(:,2).*random('Normal',1,0.1,[length(time),1]);

[params_fit] = lsqnonlin('two_species_fit',x_guess,LB,UB,options,ICs,time,noise_data);
%                        (1)              (2)      (3)(4)  (5)   [    (6) ]
%   (1) Function name
%   (2)   initial guess
%   (3/4)  Lower and upper bounds
%   (5)  Options
%   (6)   any other variables you need to run the function in (1)
```

```
                                         Norm of      First-order
 Iteration  Func-count     f(x)          step          optimality
     0          5     3.29801e+07                      5.38e+08
     1         10     1.30345e+06      0.0324822       6.74e+07
     2         15           74785      0.0115989       8.13e+06
     3         20         4880.36     0.00541682       1.03e+06
     4         25         417.007     0.00283544       1.31e+05
     5         30         72.0089     0.00218545       1.85e+04
     6         35         30.8355     0.00844758       4.02e+03
     7         40         30.2184      0.0267929       4.98e+03
     8         45         14.6221     0.00669823            829
     9         50         10.9369      0.0133965            166
    10         55         8.58653      0.0267929             59
    11         60         7.07041      0.0535859           59.5
    12         65         6.74007      0.0649962           21.5
    13         70         6.72186      0.0198848           14.7
    14         75         6.71037      0.0262079           12.3
    15         80         6.70225      0.0265257           16.4
    16         85         6.69449      0.0154985           7.38
    17         90         6.69449      0.0254806           7.38
    18         95         6.69198     0.00637016           1.43
    19        100         6.68892      0.0127403           7.25
    20        105         6.68753      0.0184689           20.7
    21        110         6.68315     0.00489992           1.48
    22        115         6.68206     0.00923447           6.85
    23        120         6.68107     0.00734179           4.91
    24        125         6.68056       0.005935           3.59
    25        130         6.68034      0.0040072           1.78
    26        135         6.68027     0.00266942           0.85
    27        140         6.68027     0.00103172          0.132
    28        145         6.68027    0.000214889        0.00644
    29        150         6.68027    1.13567e-06       9.53e-06

Local minimum possible.

lsqnonlin stopped because the final change in the sum of squares relative to
its initial value is less than the value of the function tolerance.
```



(c) Once you have the parameters..

```
%{
Once you have the parameters you can now run the forward model again and
plot your model with the optimal parameters
%}
   [~, P] = ode45(@(t,y) two_species(t,params_fit,y),time,ICs);


plot(time,noise_data(:,1),'.r',time,noise_data(:,2),'.b',time,P(:,1),'--k',time,P(:,2),'-k','LineWidth',3,'MarkerSize',10)
xlabel('Time (minutes)','FontSize',20)
ylabel('# of Species','FontSize',20)
set(gca,'FontSize',20,'LineWidth',2)
axis square;
legend('Meas. N','Meas. Q','Model N','Model Q','orientation','horizontal','location','Northoutside')
```



Published with MATLAB® R2019a