Instructions for the Huang et al. assignment

Submission Details

You should submit two, stand-alone files (last_first_huang.m such as bening_sarah_huang.m). Label all figures, axes, and units, and comment your code. Submit your implementation write up as a PDF.

Required Outputs

- Reproduce Figures 4, 5, and 6b
- Report your fitted parameter values. Include a brief discussion in the write-up comparing your parameter values to those found in the original paper (in the "Conclusions" section).

Recommended Workflow

Setting up the code for this implementation starts with a bit of mental gymnastics: we fit what? From where? With what data? I recommend the following general workflow. For each of the 3 systems that you have to model:

- Write out the ODEs that describe it, working from Figure 2. Based on this, write a function that can be solved by ode15s (or ode45) for a given set of parameters.
- Write a function that can be solved by nlinfit, which takes a set of parameters and feeds them into the ODE solver function. Do not use the "fit" function for the nonlinear fitting in this assignment; please only use "nlinfit".
    - In this function, you can add the CBV(Cwb(t)) term for the striatum and cerebellum models (this term is in equations 1, 3)
- It may be helpful to write down what data and parameters you'll use as inputs, what data or parameters will be your outputs, and drawing how your main function, the nlinfit model function, and ODE solver ode function interact with each other and the data and parameters you identified.

Reminder/Example: passing extra things into ode15s/ode45 and nlinfit

For ODE solvers:

```
function main
[T_out, C_out] = ode45 (@ODE_fun, tspan, c0, options, extra1, extra2,…)
end

function [C_dot] = ODE_fun(t, c, extra1, extra2,…)
% stuff
end
```

If you don't want to set any options, you can replace options with [ ] as an input.

For nlinfit:

```
function main
Varnam = ***input variable defined with values***;
[param_estimates] = nlinfit(X, Y, @(A, B)modfun(A, B, Varnam), param_guesses);
end

function [Pred_Y] = modfun(params, X, things_you_passed_in_like_varnam)
% stuff
end
```

More specific instructions:

- We have provided the experimental data points for Figs 3, 4, 5, and 6 in tabular form (file names, columns should be self-explanatory). You can load these into Matlab or copy and paste them into your code. When we run your code, these Excel files will be in the working directory if you decide to load them in your function.
- For initial parameter guesses, you can use the averages reported in the paper (in Tables 1, 2)
- Parameters should only be allowed to be positive
    - One way to enforce this is take their absolute value in the ODE function before using them. This way, you might still get a negative parameter from nlinfit, but you can assume that using the positive value gives the same results (so you can just report the positive value).
- Initial conditions for all ODEs should be 0's.
- The plasma F18 concentration data is always determined empirically! E.g. don't try to model it or write an ODE that describes it.
    - Since you only have experimental data for a few discrete time points, use the MATLAB function interp1 to estimate the experimental data for any given time point.
    - *This can be one of the tricky parts of this implementation. How do you have to manipulate these different time courses so that you can do the fitting and comparison?*

- For Figure 4: start from the plasma total F18 concentration time course derived from Figure 3.
- For Figure 5: No need for nonlinear fitting for this figure, just solve the plasma ODEs using the plasma total F18 concentration time course from Figure 5 and the fit parameters found while plotting Figure 4.
- For Figure 6b:
    - Use plasma F18 concentration from Figure 5
        - You'll need to calculate the whole blood F18 concentration that corresponds to this F18 time course.
            - Fit the whole blood and plasma F18 data from Figure 3 to Equation 9, in order to get estimates for p1, p2. Then use those to calculate Fig5 whole blood F18.
    - Don't model plasma 3-OMFD or plasma FDOPA with ODEs
        - For the striatum, you only want 3 ODEs (for c1, c2, c3); when modeling the cerebellum, you only want 2 ODEs (for cp1, cp2)
        - Whenever you need plasma 3-OMFD or plasma FDOPA concentrations, interpolate from the plasma 3-OMFD and plasma FDOPA concentration time courses you found for Figure 5.

Possibly helpful tips:

- Fig 5 and 6 troubleshooting
  - If you're unsure about your results for Figure 4 and you need some parameters [kb12 kb2 kb3] to try while writing your code for Figures 5 and 6, use those from patient 10 in Table 1. They won't give as good of a fit to the original figure, but they should be close.
- Whole blood partition coefficients
  - We want you to go through the process to see how they generated these coefficients. You will definitely get different partition coefficients than those they reported on page 903, again because of the model's sensitivity to input data.
  - However, this should not impact your results. You can try this out if you want – use (1,1) as your partition coefficients. This should not result in significant differences in your striatum and cerebellum parameters, or in the fit curves.
- Global variables
  - If you use nested functions (see 2015 Matlab workshop slides), you can have functions that can access variables that weren't directly passed to them. When you use variables like this, they'll turn aqua in MATLAB. This could be useful for accessing F18 concentrations, for example, since you have to use those values for multiple functions.
- Efficient homeworking/troubleshooting
  - Rather than running your entire code as you work on this, when you finish a figure, save those outputs to a .mat file that you can load and use as inputs to the next part of the implementation and comment out the part you completed. This will likely save you time. Remember to uncomment before submitting!
- Interpolation within ODEs
  - Within each of my ODE functions, I use the first parameter t to interpolate the appropriate F18 or 3-OMFD value for that timestep. I've heard from a couple people that results improve when you do this.

This means that your ODE functions should look something like this:

```
function dydt = cerebellumODEs(t,y,p)
c_omfd = interp1(comfd(:,1),comfd(:,2),t);
c_fd = (something)
p=abs(p); % params: Kp1, kp2
d_cp1 = (something);
d_cp2 = (something);
dydt=[d_cp1;d_cp2];
end
```

For reference, here are example figures from a working implementation. Some of the lines in Fig 5 are a little lumpy, probably because of the linear interpolation method – this is OK for our purposes.