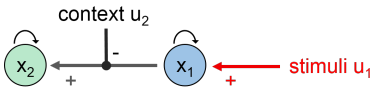
Requirements: Submit your solutions (code, answers and figures) for this exercise series in a Jupyter Notebook. Additionally, submit the Project.toml file for reproducibility.

Exercise 4.1: Programming a two region DCM (*19 points*)

In this exercise, you program your own DCM for fMRI. For this, consider the simple two region model used in the lecture:



As outlined in the lecture, the dynamics of the neural system are described by the bilinear model

$$\dot{x} = \frac{dx}{dt} = (A + u_2 B)x + Cu_1. \tag{1}$$

Here, x is a two-dimensional vector of neural states, A is the fixed connectivity matrix (size 2x2), B is the modulatory connectivity matrix (size 2x2). u_1 and u_2 are two input vectors of length 800: We model 80 seconds with a temporal resolution of $dt = 0.1s$. u_1 models a strong driving input pulse $u(t_{pulse}) = 5$ every seven seconds, while u_2 is a modulatory input which equals 1 for the period between 30 and 60 seconds and is zero otherwise. See Figure 1 (top) for an illustration of the input.

(a) (4 Points)
Write a function
`dxdt = single_step_neural(x, u,` P_{neural}`)` which calculates the right hand side of the differential equation (1). Here, $x = x(t)$ is the neural state vector and $u = u(t)$ the two-dimensional input vector at time t. P_{neural} is a dictionary containing the A, B and C parameters. We will assume that all of these have Gaussian priors, later. To render the dynamical system more stable we redefine the diagonal elements of the instantaneous connectivity matrix $CON = (A + u_2 B)$ to be $-0.5 \exp(a_{ii} + u_2 b_{ii})$. This assures that the neural self connections have negative weights, in line with the usually net inhibition within neural populations. Hence, the connectivity matrix for the simulation should be

$$\widetilde{CON} = \begin{pmatrix} -0.5\exp(a_{11} + u_2 b_{11}) & a_{12} + u_2 b_{12} \\ a_{21} + u_2 b_{21} & -0.5\exp(a_{22} + u_2 b_{22}) \end{pmatrix} \tag{2}$$

with the differential equation

$$\dot{x} = \frac{dx}{dt} = \widetilde{CON}x + Cu_1. \tag{3}$$

(b) (5 points)
Next, write a function `dhdt = single_step_hrf(h, x,` P_{HRF}`)` which computes the update for the hemodynamic model of a single region. Here, $h = h(t) = [s, f, v, q]$

is the hemodynamic state vector, and $x = x(t)$ is the neural state of the region at time t. P_{HRF} is a dictionary that contains the parameters of the hemodynamic model. Throughout this exercise we assume the key-value pairs of P_{HRF} are fixed to $\kappa = 0.64$, $\gamma = 0.32$, $\tau = 2$, $\alpha = 0.32$ and $E_0 = 0.4$. Note: If the parameters of the HRF need to be inferred, the fixed values in P_{HRF} are usually multiplied by a parameter that follows a log-normal distribution, e.g. $\kappa = P_{HRF}[\text{"}\kappa\text{"}] \cdot P_{neural}[\text{"}\kappa\text{"}]$. This parameter $P_{neural}[\text{"}\kappa\text{"}]$ is then estimated. If it is set to 0 then $\kappa = P_{HRF}[\text{"}\kappa\text{"}]$.

The four differential equations which define the hemodynamic update are (cf. Stephan et al., Neuroimage, 15:378, 2007)

$$\dot{s} = f_s(x, s, f) = x - \kappa s - \gamma(f - 1) \tag{4}$$

$$\dot{f} = f_f(s) = s \tag{5}$$

$$\dot{v} = f_v(f, v) = \frac{1}{\tau}\left(f - v^{\frac{1}{\alpha}}\right) \tag{6}$$

$$\dot{q} = f_q(f, v, q) = \frac{1}{\tau}\left(f\frac{1 - (1 - E_0)^{\frac{1}{f}}}{E_0} - v^{\frac{1}{\alpha}}\frac{q}{v}\right) \tag{7}$$

(c) (4 Points)
Then, write a function `y = compute_bold_signal(h,` P_{HRF}`)` which uses v and q (rows 3 and 4 in h) to compute the BOLD signal change y. The parameters of the BOLD model are also given in P_{HRF}. The BOLD signal equation and its parameters for different field strengths are provided in the lecture slides. Use the values for 3 Tesla, here. Note: The parameter ε is the only parameter of the BOLD signal equation which is sometimes estimated and multiplied with a log-normal parameter, as described above. Here, we assume it is fixed: $\varepsilon = 0.47$.

(d) (6 Points, incl. Figure)
Finally, write a function which emulates a simple Euler integrator `[y,h,x] = euler_integrate_dcm(U,` P_{neural}`,` P_{HRF}`,` x_0`,` h_0`)` and iterates over all time steps of the input to generate the neural (x), hemodynamic (h) and BOLD (y) traces. This is done by computing the functions you have written above in (a) and (b) at every time step and then adding this to the corresponding state vectors. The starting values are given by the fixed point of the system with input 0: $x_0 = x(t = 0) = [0, 0]'$ and $h_0 = h(t = 0) = [0, 1, 1, 1]'$ (h_0 has to be specified in each region). The BOLD signal equation can be applied to the final traces. For convenience, U is a dictionary with key-value pairs u (a 2x800 input as described above) and $dt = 0.1$. If you use the parameters $a_{1,1} = a_{2,2} = a_{1,2} = 0$, $a_{2,1} = 1$, $b_{1,1} = b_{2,2} = b_{1,2} = 0$, $b_{2,1} = -0.5$ and $C = \left(\begin{smallmatrix} 1 & 0 \\ 0 & 0 \end{smallmatrix}\right)$ (input u_1 targets region 1 with strength 1), you should get the traces shown in Figure 1.

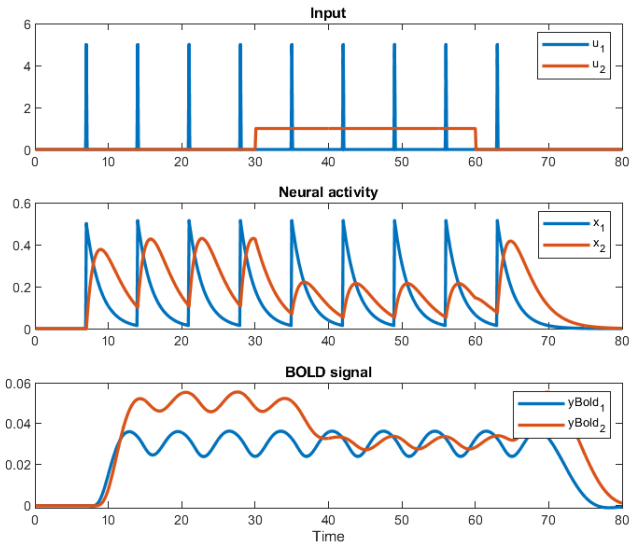


Figure 1: This figure shows the simulation results of the dynamical system you should program in this exercise. Top: Input traces, Middle: Neural activity, Bottom: BOLD signal (without noise).

Exercise 4.2: Exploring the effect of modulation (*5 points*)

In this exercise, you explore the model you have written above. In particular, you should create instances of the three models illustrated in Figure 2.

These models have modulatory B-matrices $B_1 = \left(\begin{smallmatrix} 0 & 0 \\ 0 & 0 \end{smallmatrix}\right)$, $B_2 = \left(\begin{smallmatrix} 0 & 0 \\ b & 0 \end{smallmatrix}\right)$ and $B_3 = \left(\begin{smallmatrix} 0 & 0 \\ 0 & b \end{smallmatrix}\right)$, respectively.

(a) (1 Point) Run the model without modulation (B_1) first.

(b) (4 Points, incl. Figure) Next explore the changes in the activity x and the BOLD signal y of region 2 when you change the value of b in the matrices B_2 and B_3. For

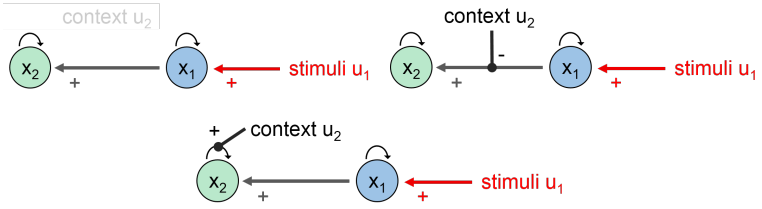


Figure 2: Three different two region DCMs.

example use $b = [-1, -0.5, 0, 0.5, 1]$. What do you observe when you look at the resulting neural and BOLD traces? *Hint:* Remember how we have ensured that the diagonal elements of $CON = (A + u_2 B)$ are always negative.

Exercise 4.3: Maximum likelihood and maximum a posteriori (MAP) estimates (*11 points*)

In this third part, you will generate noisy traces and then compute a maximum likelihood and maximum a posteriori estimates for four different priors. The goal of this exercise is to investigate how the prior can influence the parameter estimates. For simplicity, we restrict ourselves to model 2 above and only estimate five parameters $\pi = \{a_{11}, a_{21}, a_{22}, b_{21}, c_{11}\}$. We assume that all other parameters, including the hemodynamic parameters, are fixed.

(a) (2 Points) Generate a noisy BOLD trace $\hat{y} = y + \epsilon$ from model 2 with the same parameter setting as in exercise 4.1. ϵ is a Gaussian noise with $\sigma_\epsilon = \sigma_y$, i.e. we set the standard deviation of the noise to be the same as the standard deviation of the signal. You can use the Julia method `randn()` to create the noise. Note: For simplicity, use the same noise level for both regions.

(b) (2 Points) Write a function that computes the log-likelihood $L = \log(p(\hat{y}|\pi))$ of the model for a given set of connectivity parameters. For this, assume that you know the standard deviation of the noise. You can simply use the value you used for the generation of the data.

(c) (3 Points) Write a function that computes the value of the log joint distribution $log(p(\hat{y}|\pi)p(\pi))$ of the model for a given set of connectivity parameters. Here, you need to define a prior $p(\pi)$ over the parameters. Set the mean of the Gaussian prior to 0 for all five parameters you estimate and the corresponding variances to $\sigma^2 = 0.1$. As in (b) do not estimate the noise, but set it to the true value, for simplicity. Note: For numerical reasons it is better if you work directly in log space. It is not difficult to derive the equations for the log of a multidimensional normal distribution which can be used in the code. This also speeds up your code.

(d) (4 Points) Now, use these two functions and an optimizer such as `Optim.jl` to estimate maximum likelihood and MAP estimates given $\hat{y}$. For the MAP estimates, use four different sets of priors with $\sigma_1^2 = 10$, $\sigma_2^2 = 10^{-1}$, $\sigma_3^2 = 10^{-3}$ and $\sigma_4^2 = 10^{-5}$. Write out the estimated parameters a_{11}, a_{21}, a_{22}, b_{21} and c_{11} and the log-likelihood L at the estimated parameters for all five inversions.

Compare how changing the prior changes the parameters. Does a tighter prior, i.e. smaller variance, always lead to more shrinkage, i.e. lower parameter values, in all parameters?

Also have a look at the log-likelihood evaluated at the MAP and compare it to the log-likelihood evaluated at the maximum likelihood estimate.

- Exercises are solved in groups of 2–3.
- If you have questions, contact us via *Moodle*.