

# Cognitive Modeling: Homework Assignment 3

## Stochastic Process Models and Bayesian Estimation

March 6, 2025

All answers and solutions to non-programming questions should be submitted to LMS as a **legible** write-up (either fully digital or a scan). The use of LLMs (e.g., ChatGPT) is **explicitly discouraged**, unless specified otherwise. All code should be committed to and merged into the **main** branch of your team's GitHub repository, unless specified otherwise. Your LMS submissions should contain a single ZIP file named according to the pattern:

- CogModel\_Assignment[#]-[TeamMember1Initials]-[TeamMember2Initials]

Only one team member should initiate the submission on LMS. A link to the GitHub repository should be clearly visible on the LMS as submission text.

### Problem 1: True-False Questions (5 points)

Mark all statements which are **FALSE**.

1. The solution of the stochastic integral  $\int_0^T \mu dW_t$  is  $\mu(W_T - W_0)$  and is a random variable itself.
2. The variance of a Wiener process with scale coefficient  $\sigma = 1$  at time  $t$  is  $t^2$ .
3. The standard Drift-Diffusion Model (DDM) assumes that evidence about a dominant alternative accumulates in discrete chunks over time.
4. The first passage time distribution has a closed-form probability density function, but its density can still be evaluated only numerically.
5. The Euler-Maruyama method can only be used to simulate linear stochastic systems.
6. For any Bayesian analysis, the prior will always have a smaller variance than the posterior.
7. In addition to good statistical practices, experimental validation of cognitive models is crucial for ensuring construct validity.
8. Markov chain Monte Carlo (MCMC) methods approximate a complex posterior distribution through a simpler, yet analytically tractable, distribution.
9. For most Bayesian problems, the more data we collect, the less influence does the prior exert on the resulting inferences.
10. The effective sample size (ESS) estimated from MCMC samplers differs from the total number of samples because the samples are not independent (i.e., exhibit non-zero autocorrelation).

## Problem 2: Diffusion Model Explorations (8 points)

As extensively discussed in class, the drift-diffusion model (DDM) generates two response time (RT) distributions, one for each boundary (i.e., lower and upper boundaries). This exercise asks you to first explore a somewhat counterintuitive question about the basic DDM: What differences between the means of the two RT distributions does the model predict?

To approach this question from a simulation-based perspective, you need to repeatedly solve the forward problem with different parameter configurations and collect the two summary statistics, namely, the two empirical means of the resulting RT distributions. First, choose a suitable configuration of the four parameters and vary only the drift rates within a reasonable range (e.g.,  $v \in [0.5 - 1.5]$ ) for a total of 25 different drift rates. Make sure that your parameterizations can generate a sufficient number of RTs for both distributions and you don't end up with the process only reaching the upper boundary. Second, for each of your parameter configurations, generate  $N = 2000$  synthetic observations and estimate the means of the two distributions. What do you observe regarding the mean difference? Describe and interpret your results. (4 points)

In a similar spirit (keeping all parameters fixed and varying one), explore the effects of each of the parameters on the means and standard deviations of the simulated RT distributions, quantify and describe your results. (4 points)

## Problem 3: Prior and Posterior Variance (4 points)

Show that the following identity holds for any given prior and posterior:

$$\text{Var}[\theta] = \mathbb{E}[\text{Var}[\theta | y]] + \text{Var}[\mathbb{E}[\theta | y]] \quad (1)$$

Clarification of terms:

1.  $\text{Var}[\theta]$  – Prior variance.
2.  $\mathbb{E}[\text{Var}[\theta | y]]$  – Expected posterior variance.
3.  $\text{Var}[\mathbb{E}[\theta | y]]$  – Variance of posterior mean.

## Problem 4: Normal-Normal Model (4 points)

This exercise mimics the approach we took for solving the Beta-Binomial Bayesian butter toast example using a Normal-Normal conjugate model with known variance (i.e., Normal prior, Normal likelihood). You would implement the model in Python (just as we did for the butter toast) and explore the effects of different prior configurations, so you don't necessarily need to analytically derive the results (conjugate models can be easily looked up on the web, as discussed in class). However, I recommend that you try it out using pen and paper first!

The prior distribution for the mean is given by ( $\mu_0$  and  $\sigma_0$  are known *hyperparameters* selected by the modeler):

$$p(\mu) = \mathcal{N}(\mu | \mu_0, \sigma_0) \quad (2)$$

The likelihood distribution is given by (the variance  $\sigma^2$  is assumed to be known):

$$p(y | \mu, \sigma^2) = \mathcal{N}(y | \mu, \sigma^2) \quad (3)$$

Come up with an entertaining use case for the model, conjure up (i.e., simulate) a modest number of “data points”, and compute the posterior given a self-chosen configuration for the prior. Visualize the difference between prior and posterior using histograms, as we did in class (4 points).

### Problem 5: Simple Bayesian Regression with Stan (6 points)

For this exercise, you will write a probabilistic program in **Stan** that can sample from the posterior  $p(\alpha, \beta \mid \mathcal{D})$  of a simple (linear) Bayesian regression given a data set  $\mathcal{D} = \{x_n, y_n\}_{n=1}^N$ . The program should represent the following generative specification:

$$\begin{aligned}\sigma^2 &\sim \text{Inv-Gamma}(1, 1) \\ \alpha &\sim \mathcal{N}(0, 10) \\ \beta &\sim \mathcal{N}(0, 10) \\ y_n &\sim \mathcal{N}(\alpha + \beta x_n, \sigma^2) \quad \text{for } n = 1, \dots, N,\end{aligned}$$

which essentially encodes the assumption that an outcome  $y$  is predicted from a covariate  $x$  via:

$$y_n = \alpha + \beta x_n + \epsilon_n \quad \text{with} \quad \epsilon_n \sim \mathcal{N}(0, \sigma^2) \quad (4)$$

First, simulate a data set with fixed intercept ( $\alpha$ ), slope ( $\beta$ ), and noise ( $\sigma$ ) parameters of your choosing, as well as a pre-set number of observations  $N$ . For instance, your simulation could look something like this:

```
N = 100
alpha = 2.3
sigma = 2.
slope = 4.
x = np.random.normal(size=N)
y = alpha + slope * x + sigma * np.random.normal(size=N)
```

Then, pass the data to your program, inspect convergence and efficiency diagnostics, and summarize your inferential results both numerically and graphically.

- How accurate are the posterior means and how much uncertainty is left? (4 points)
- Repeat the analysis with ten times as many observations and report what happens to the precision and uncertainty (2 points).

### Problem 6: Estimating the Drift-Diffusion Model (8 points)

In this exercise, we will perform Bayesian estimation of the Drift-Diffusion Model (DDM) given an actual data set. Load the data `sample_response_times.csv`, which contains three columns: **rt** (the response times), **choice** (the categorical choices), and **condition** (a condition indicator). There are  $N = 300$  rows in total, corresponding to 300 trials in an experiment.

There were two conditions in this experiment - an easy one (participants had to classify a familiar face in a crowd of 10 people) and a difficult one (participants had to classify a face in a crowd of 100 people). Unfortunately, a colleague of yours has lost the *codebook* and no longer knows which indicator in the column **condition** corresponds to which condition.

This is where you step in. Given your knowledge of the interpretation of four key diffusion model parameters, you offer to perform a model-based analysis and disentangle the conditions by estimating the parameters of the diffusion model. Which parameter best reflects task difficulty (i.e., amount of information to be processed)? Indeed, once you answer this question, you automatically know

which parameter you need to vary between the conditions to answer the question. Fortunately for you, a template script is already provided.

Your task is to complete the Stan program provided in `diffusion_model.stan`, solve the inverse problem from the data, and determine which indicator corresponds to the unknown **difficult condition** based on your estimates (6 points). Along the way, you also need to ensure computational faithfulness by reporting and interpreting convergence and efficiency diagnostics (2 points).