Domain and Variables We consider a spatial domain $D = (i, j) : 1 \leq i \leq m, 1 \leq j \leq n$ consisting of $m \times n$ cells, where i and j are indices for the rows and columns respectively. The domain is partitioned into k different classes, where each class corresponds to a distinct Manning's coefficient value. Let's denote:

- The set of classes as $\mathcal{K} = 1, 2, \ldots, k$. This set is indexed by l.
- The Manning's coefficient is represented by the vector $\theta \in \mathbb{R}^k$, where each element θ_l represents the Manning's coefficient for class $l \in \mathcal{K}$.
  - the hypothetical (initial guess or sampled values) Manning's coefficient is noted $\hat{\theta}$,
  - the true (unknown) Manning's coefficient is noted θ^*.
- The streamflow at each cell is represented by the matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$, where $\mathbf{X}_{i,j}$ is the streamflow at cell (i, j).
  - The true (observed) streamflow is noted $\mathbf{X}^*$,
  - The streamflow computed from the hypothetical Manning's coefficient, using ParFlow, is noted $\hat{\mathbf{X}}$.
- Also we will denote ParFlow as f.

Problem Formulation Given the true streamflow $\mathbf{X}^*$, and the sinulator f, our goal is to estimate the posterior distribution $p(\theta|\mathbf{X}^*)$ of the Manning's coefficients. We aim to find the values of θ that minimize the difference between the simulated streamflow and the observed streamflow, i.e., minimize $|\mathbf{X}^* - f(\theta)|$, where $|\cdot|$ is a suitable norm.

More precisely, the objective is to determine the posterior distribution of the Manning's coefficient vector θ given the observed streamflow matrix $\mathbf{X}^*$:

$$p(\theta|\mathbf{X}^*) = \frac{p(\theta, \mathbf{X}^*)}{p(\mathbf{X}^*)} = \frac{p(\mathbf{X}^*|\theta) \cdot p(\theta)}{p(\mathbf{X}^*)} \tag{1}$$

where $p(\mathbf{X}^*|\theta)$ is the likelihood function, $p(\theta)$ is the prior distribution, and $p(\mathbf{X}^*)$ is the evidence. The evidence $p(\mathbf{X}^*)$ is the marginal probability of observing the streamflow $\mathbf{X}^*$, and is given by:

$$p(\mathbf{X}^*) = \int p(\mathbf{X}^*|\theta) \cdot p(\theta)\, d\theta \tag{2}$$

The denominator is basically normalizing the total probability to 1, so what we actually need is an efficient way of estimating $p(\mathbf{X}^*|\theta)$. Of course, the likelihood function is not explicitly available, as it results from the ParFlow model (of which we don't have an analytical form).

Question. Coudn't we use simpler methods to approximate the likelihood function by simulations (or is SBI exactly it?), such as: Approximate Bayesian Computation (ABC). In its most basic form, you simply simulate a dataset from a set of parameter points sampled from the prior distribution. Then you define identity with the true observations via a level of acceptable discrepancy. Or maybe we could compute the discrepancy for all simulated streamflows and transform these distances into a likelihood approximation using a kernel function.

Simulation-Based Inference (SBI) Say we choose Simulation-Based Inference (SBI) to estimate the posterior distribution $p(\theta|\mathbf{X}^*)$.

The goal it to leverage ML in order to tune the model (find actual Manning values, given original guesses), without the use of traditional parameter calibration methods that are computationally intractable due to the high dimensionality of the streamflow data and the complexity of the ParFlow simulator (important simulation time).

SBI Procedure The SBI procedure begins with a prior distribution $\pi(\theta)$. We use a uniform prior for each element of the Manning's coefficient vector:

$$\theta_l \sim \mathcal{U}\left(\frac{\hat{\theta}_l}{2}, 2\hat{\theta}_l\right) \quad \forall l \in \mathcal{K} \tag{3}$$

where $\mathcal{U}(a, b)$ denotes the uniform distribution on the interval $[a, b]$.

Then, we sample N (e.g., $N = 100$) parameter vectors $\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(N)}$ from the prior distribution $\pi(\theta)$. For each sampled parameter vector $\theta^{(i)}$, we compute the corresponding streamflow $\hat{\mathbf{X}}^{(i)} = f(\theta^{(i)})$ using the ParFlow simulator. These pairs $(\theta^{(i)}, \hat{\mathbf{X}}^{(i)})_{i=1}^{N}$ form our simulation dataset.

Using this dataset, we train a (NN-based) "density estimator", which we denote as $q_\phi(\theta|\mathbf{X})$, where ϕ represents the parameters of the neural network. This estimator approximates the posterior distribution: once trained, we can use it as $p(\theta|\mathbf{X}^*) \leftarrow q_\phi(\theta|\mathbf{X}^*)$.

Then we repeat the procedure: let $p_t(\theta|\mathbf{X}^*)$ denote the posterior distribution estimate at iteration t.

At iteration $t = 0$, we start with a prior $p_0(\theta) = \pi(\theta)$ as defined above.

As we iterate, the samples from the posterior distribution concentrate in regions of the parameter space that are more likely to produce streamflow patterns similar to the observed $\mathbf{X}^*$. This iterative refinement approach allows us to more efficiently explore the parameter space and obtain a more accurate estimate of the posterior distribution.

Neural Density Estimation The crux is to understand how the estimator $q_\phi(\theta|\mathbf{X})$ works. We start with the empirical distribution of the simulated data $(\theta^{(i)}, \hat{\mathbf{X}}^{(i)})_{i=1}^{N}$ and get to a relation with the observed data ... Do we try to approximate like $p(\theta|\hat{\mathbf{X}} \sim \mathbf{X}^*)$ or something similar?

Surrogates for Computational Efficiency Due to the computational cost of running the ParFlow simulator, we may introduce a surrogate model or emulator, denoted as $\tilde{f}$, which approximates the ParFlow simulator f. The surrogate model can be trained on a set of parameter-streamflow pairs $(\theta^{(i)}, \hat{\mathbf{X}}^{(i)})_{i=1}^{N}$ generated by the ParFlow simulator. The surrogate model allows us to (once trained) rapidly explore the parameter space without having to run the computationally expensive ParFlow simulator for each new parameter vector.