Facilitating Atmospheric Source Inversion via Deep Operator Network Surrogates

Mamikon Gulian, Joseph Hart, Indu Manickam, and Laura Swiler

Exceptional service in the national interest

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. SAND number: SAND2022-7855 C

Source Inversion / Identification in Climate Systems

- The identification of sources in climate systems is a vital problem for attribution and prediction of climate states.
- We cannot isolate sources in nature and climate simulators are expensive.
- Surrogate models enable the many-query algorithms required for inverse problems.
- Inversion for a source is an ill-posed problem so it is natural to treat it as a probabilistic problem.
- We utilize a probability model for the most probable source and uncertainty.
- Our framework identifies the source characteristics from an SO_2 plume by
  1. Calibrating deep operator network surrogates to the flow map,
  2. Setting up a Bayesian framework for a distribution over the forcing profiles,
  3. Optimizing to identify sources from sparse and noisy observations.
- The model we test on includes diffusion, wind, gravity, chemistry, and a volcano source with $O(10^5)$ degrees of freedom.
- Our results open the door for applications such as inverting for the Mt. Pinatubo source profile – the largest volcano eruption in the 20th century.

Basic Variables and Problem Statement

- **State:** represents **concentration** on the domain $\Omega = (0, L_1) \times (0, L_2)$, and maps

$$u : \Omega \times [0, T] \to \mathbb{R}, \quad (\mathbf{x}, t) \mapsto u(\mathbf{x}, t). \tag{1}$$

- **Forcing function:** represents injection of concentration on Ω, and maps

$$f : \Omega \times [0, T] \to \mathbb{R}, \quad (\mathbf{x}, t) \mapsto f(\mathbf{x}, t). \tag{2}$$

- **Forcing amplitude:** the forcing function is written as

$$f(\mathbf{x}, t) = z(t)F(\mathbf{x}), \tag{3}$$

  where F is a **known** spatial profile that is fixed in time, and z is a function in time that is decaying. We refer to $z(t)$ as the **forcing amplitude**.

- **Problem statement:** Given a training set of $M = \mathcal{O}(10)$ observed/simulated $\{(z^m, u^m)\}_{m=1}^M$, predict a forcing amplitude z from noisy and sparse observations of a concentration u not in the training set.

The SO_2 Plume Synthetic Model: Qualitative Properties

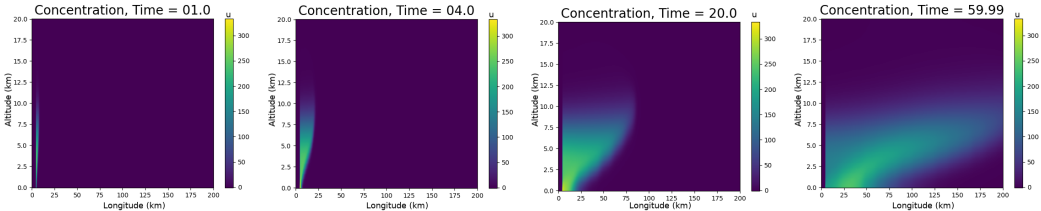

Figure 1: Illustration of the SO_2 concentration modeled by the Gaussian Plume synthetic model at various times.

- 2D model in longitude x_1 and altitude x_2.
- We have studied models in latitude and longitude, and plan to work in 3D.
- Based on Stockie's "The Mathematics of Atmospheric Dispersion Modeling" (SIAM Review 2011).
- Simulated numerically using a similar number of degrees of freedom in spacetime as a 3D Energy Exascale Earth System Model (E3SM) run for a single quantity of interest.

The SO_2 Plume Synthetic Model: Equations and Formulae

- We take $L_1 = 200$ (km), $L_2 = 20$ (km), and generate u as the solution to the equations

$$\begin{aligned} &\frac{\partial u}{\partial t} - \kappa \Delta u + \mathbf{v} \cdot \nabla u - S\mathbf{e}_2 \cdot \nabla u = R(u) + f && \text{on } \Omega \times [0,T] \\ &\nabla u \cdot \mathbf{n} = 0 && \text{on } \partial\Omega \times [0,T] \\ &u = 0 && \text{on } \Omega \times \{0\} \end{aligned}$$

  where $\kappa \Delta u$ represents diffusion, $\mathbf{v} \cdot \nabla u$ represents wind drift, $S\mathbf{e}_2 \cdot \nabla u$ represents gravity, $R(u) = -\gamma u$ represents reaction, and f the forcing.

- We generate data using the forcing term

$$f(t, x_1, y_2) = z(t) \exp\left(-100(x_1 - 5)^2\right) \exp\left(-\frac{(x_2 - 5)^2}{16}\right)$$

  with forcing amplitude

$$z(t) = \lambda_1 \exp\left(-\lambda_2 t\right) \tag{4}$$

  to model SO_2 injection. λ_1 and λ_2 are user controlled.

Surrogate Model Problem Statement

- State vector: discretized over a spatial grid of $d = \mathcal{O}(10^5)$ points in Ω and N times, the state u is represented by a series of vectors

$$\mathbf{u}_n \in \mathbb{R}^d, \quad n = 0, 1, 2, ..., N. \tag{5}$$

- Forcing amplitude vector: discretized at the same times as the state vector, z is represented by a vector

$$\mathbf{z} \in \mathbb{R}^N. \tag{6}$$

- **Surrogate model problem statement**: we seek a data-driven approximation to

$$B : (\mathbf{u}_0, \mathbf{z}) \mapsto \{\mathbf{u}_n\}_{n=1}^N \tag{7}$$

that is trained using data generated by solving the above differential equations.

- Evaluation of $B(\cdot, \mathbf{z})$ will allow for inverting for $\mathbf{z}$ given observations of the state.
- We have small ensemble $\{(z^m, u^m)\}_{m=1}^M$, so rather than approximate B directly, **we construct a surrogate for the flow map**

$$A : (\mathbf{u}_n, z_n) \mapsto \mathbf{u}_{n+1}, \quad n = 0, 1, 2, ..., N-1. \tag{8}$$

Then, $B(\cdot, \mathbf{z})$ is given by compositions of $A(\cdot, z_n)$.

Ensemble of (forcing amplitide, solutions) to train the surrogate

- Given M initial conditions/forcing functions and concentrations:

$$\mathbf{u}_0^m, \mathbf{u}_1^m, ..., \mathbf{u}_N^m \in \mathbb{R}^d \quad \text{and} \quad \mathbf{z}^m \in \mathbb{R}^N, \quad m = 1, ..., M. \tag{9}$$

- **Superscript** m indexes different pairs of forcing and simulated concentration
- For each m, the **subscript** n indexes different time steps of the concentration.
- In the examples presented here, we take $\mathbf{u}_0^m = 0$ for each m, and vary $\mathbf{z}$.

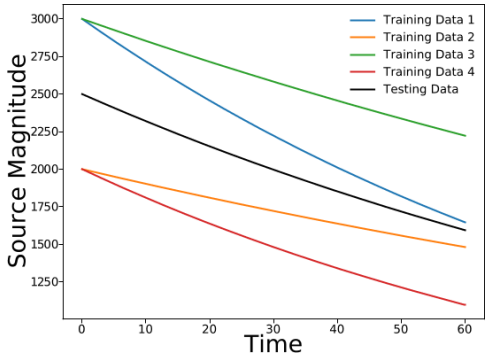

Figure 2: Ensemble of forcing amplitudes used to generate data.
The training data is generated with $\lambda_1 \in \{2000, 3000\}$ and $\lambda_2 \in \{0.005, 0.01\}$.
The testing data is generated using $(\lambda_1, \lambda_2) = (2500, 0.0075)$.
Recall $z(t) = \lambda_1 \exp(-\lambda_2 t)$

PCA for dimension reduction

- PCA dimension reduction: given a target dimension $r = \mathcal{O}(10)$, we assemble the $\mathcal{O}(10^5)$-dimensional concentration data and perform an SVD decomposition as

$$\left[\mathbf{u}_1^1|\dots|\mathbf{u}_N^1|\mathbf{u}_1^2|\dots|\mathbf{u}_N^2|\dots|\mathbf{u}_1^M|\dots|\mathbf{u}_N^M|\right] \approx U_r \Sigma_r V_r^\top. \tag{10}$$

- Then we obtain the assembly of r-dimensional reduced state vectors $\mathbf{c}_n^m$:

$$\left[\mathbf{c}_1^1|\dots|\mathbf{c}_N^1|\mathbf{c}_1^2|\dots|\mathbf{c}_N^2|\dots|\mathbf{c}_1^M|\dots|\mathbf{c}_N^M|\right] = U_r^\top \left[\mathbf{u}_1^1|\dots|\mathbf{u}_N^1|\mathbf{u}_1^2|\dots|\mathbf{u}_N^2|\dots|\mathbf{u}_1^M|\dots|\mathbf{u}_N^M|\right]$$

- In other words, the $\mathbf{c}_n^m$ represent the coefficients of the state $\mathbf{u}_n^m$ in the PCA basis.
- We seek A_{red} such that $A \approx U_r \circ A_{\text{red}} \circ U_r^\top$ i.e., so that the following diagram approximately commutes:

$$\begin{array}{ccc} \mathbf{u}_n^m & \xrightarrow{A(\cdot, z_n)} & \mathbf{u}_{n+1}^m \\ \text{PCA projection} = U_r^\top \big\downarrow & & \big\uparrow \text{PCA reconstruction} = U_r \\ \mathbf{c}_n^m & \xrightarrow[A_{\text{red}}(\cdot, z_n)]{} & \mathbf{c}_{n+1}^m \end{array} \tag{11}$$

- **We refer to A_{red} as the surrogate flow map between reduced spaces.**

Deep Neural Network Operator Surrogate for A_{red}

- We consider a family of neural networks $\mathcal{NN} : \mathbb{R}^r \times \mathbb{R} \to \mathbb{R}^r$ consisting of L hidden layers of width r composed with a final linear layer:

$$\mathcal{NN}(\mathbf{c}, z; \boldsymbol{\xi}) = W \circ \boldsymbol{\Phi}(\mathbf{c}, z; \boldsymbol{\xi}^{\mathrm{H}}), \tag{12}$$

  W and $\boldsymbol{\xi}^{\mathrm{H}}$ are the parameters corresponding to the final linear layer and the hidden layers, respectively; their union is $\boldsymbol{\xi}$.

- Given a neural network architecture for $\mathcal{NN}$, we model

$$\mathbf{c}_{n+1} = A_{\text{red}}(\mathbf{c}_n, z_n; \boldsymbol{\xi}) = \mathbf{c}_n + \Delta t \mathcal{NN}(\mathbf{c}_n, z_n; \boldsymbol{\xi}), \quad n = 1, 2, ... \tag{13}$$

  where Δt is the time difference between steps n and $n+1$.

- This amounts to a ResNet-like skip connection for the final output of the DNN that is informed by the time step Δt and is suggested by the forward Euler discretization.

- Since our networks have 4 or less layers, we use a plain neural network architecture

$$\boldsymbol{\Phi}^{\text{plain}} = \boldsymbol{\sigma} \circ T_L \circ \cdots \circ \boldsymbol{\sigma} \circ T_1 \tag{14}$$

  where $\boldsymbol{\Phi}$ is the vector of the r functions Φ_i, $\boldsymbol{\sigma}$ the vector of r copies of σ.

Prediction, Loss, and Training

- For $p \geq 1$, we define the prediction in reduced space after p timesteps as

$$A_{\text{red}}^{[p]}(\mathbf{c}, [z_1, ..., z_p]; \boldsymbol{\xi}) = A_{\text{red}}(\cdot, z_p; \boldsymbol{\xi}) \circ \ldots \circ A_{\text{red}}(\cdot, z_1; \boldsymbol{\xi})(\mathbf{c}). \tag{15}$$

- The predicted concentration at timestep p is given by

$$\mathbf{u}_p^{m,*} = U_r \, A_{\text{red}}^{[p]}(\mathbf{c}_0^m, [z_1, ..., z_p]; \boldsymbol{\xi}) \tag{16}$$

- Multistep Loss: given forcings $\mathbf{z}^m$ and corresponding solutions $[\mathbf{c}_0^m, \mathbf{c}_1^m, ..., \mathbf{c}_N^m]$ in reduced space for $m = 1, 2, ..., M$,

$$\text{Loss}(\boldsymbol{\xi}) = \sum_{m=1}^{M} \sum_{n=0}^{N-1} \sum_{p=1}^{P} \left\| \mathbf{c}_{n+p}^m - A_{\text{red}}^{[p]}(\mathbf{c}_n^m, \mathbf{z}; \boldsymbol{\xi}) \right\|_{\ell^2}^2 \tag{17}$$

- The DNN is initialized using default Glorot initialization, and trained using the `adam` gradient descent optimizer with an exponentially decaying learning rate schedule.

Surrogate Flow Map Training Example

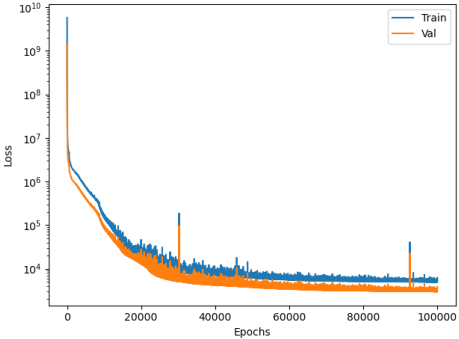

Figure 3: Loss vs number of steps of `adam` optimizer (Epochs) for an ensemble of 3 forcing, concentration pairs (Train). The same loss function is monitored for a different forcing, concentration pair (Val) to watch for overfitting.

- This was performed by training in a reduced space of $r = 56$ dimensions.
- There are many options for the selection of data, DNN architectures, and hyperparameters related to training.
- With appropriate choices, we are able to achieve $O(1\%)$ relative ℓ^2 reconstruction error in the predicted flow using the trained surrogate.

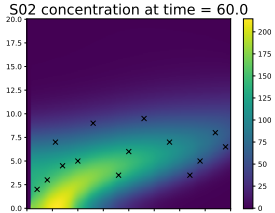

Figure 4: Sparsely scattered observations of the concentration, expressed by an observation operator $\mathcal{O} : \mathbb{R}^d \to \mathbb{R}^q$, where $q \sim 10$ is the number of sensors.

- Assuming observed data $\mathcal{D} = [\mathcal{O}(\mathbf{u}_0), ..., \mathcal{O}(\mathbf{u}_N)]$ is contaminated with mean zero Gaussian noise with a covariance matrix $\Gamma_{\text{noise}} \in \mathbb{R}^{q \times q}$, the likelihood function is

$$\pi_{\text{like}}(\mathbf{z}|\mathcal{D}) \propto \exp\left(-\frac{1}{2}\sum_{n=1}^{N}\left\|\mathcal{O}(\mathbf{u}_n) - \mathcal{O}\left(U_r\, A_{\text{red}}^{[n]}\left(U_r^\top \mathbf{u}_0^m, \mathbf{z}\right)\right)\right\|^2_{\Gamma_{\text{noise}}^{-1}}\right) \tag{18}$$

- Assuming a Gaussian distribution $\pi_{\text{prior}}(\mathbf{z})$ with mean $\overline{\mathbf{z}}$ and covariance $\Gamma_{\text{prior}} \in \mathbb{R}^{k \times k}$, the posterior distribution is then given by $\pi_{\text{post}}(\mathbf{z}|\mathcal{D}) \propto \pi_{\text{like}}(\mathbf{z}|\mathcal{D})\pi_{\text{prior}}(\mathbf{z})$.
- The MAP point $\mathbf{z}$ for which the posterior PDF π_{post} found via

$$\min_{\mathbf{z}\in\mathbb{R}^N} J(\mathbf{z}) = \min_{\mathbf{z}\in\mathbb{R}^N} \frac{1}{2}\sum_{n=1}^{N}\left\|\mathcal{O}(\mathbf{u}_n) - \mathcal{O}\left(U_r\, A_{\text{red}}^{[n]}\left(U_r^\top \mathbf{u}_0^m, \mathbf{z}\right)\right)\right\|^2_{\Gamma_{\text{noise}}^{-1}} + \frac{1}{2}\left\|\mathbf{z} - \overline{\mathbf{z}}\right\|^2_{\Gamma_{\text{prior}}^{-1}} \tag{19}$$

Optimization, codebase, and posterior sampling algorithm

- We minimize $J(\mathbf{z})$ to find $\mathbf{z}_{\text{MAP}}$ using a truncated CG trust region algorithm in the Rapid Optimization library (ROL), part of the Trilinos project developed by SNL.
- The gradient is computed by solving the adjoint equation corresponding to the discrete time stepping algorithm.
- The DNN Jacobians are computed using the automatic differentiation tools in Tensorflow, which then interfaces with ROL via PyROL.
- A Gauss-Newton Hessian approximation is used to leverage DNN Jacobians for an efficient Hessian approximation.
- To sample posterior, use Laplace approximation: samples from π_{post} generated by assuming that π_{post} is the PDF for a Gaussian distribution the mean of which is $\mathbf{z}_{\text{MAP}}$ and with covariance given by the inverse Hessian of J evaluated at $\mathbf{z}_{\text{MAP}}$.

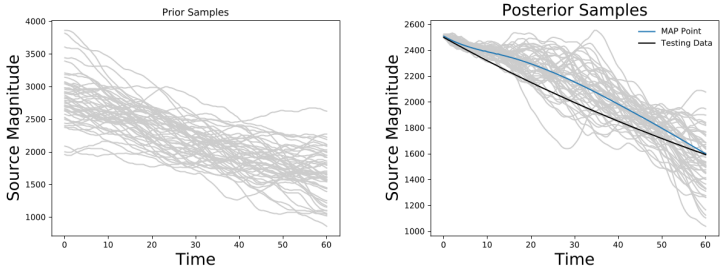

Figure 5: Prior (left) and posterior (right) distribution over $\mathbf{z}$, predicted using the observations illustrated in Figure 4 and the data shown in Figure 2. Grey curves are samples from distributions.

- Observations u are contaminated with multiplicative ($\sim 2\%$) Gaussian noise.
- An ensemble of 4 simulations was used for surrogate, and the wall-time for this complete run was roughly 1 hour.
- With a few noisy observations, the predicted $\mathbf{z}_{\text{MAP}}$ shows significant improvement over the prior $\overline{\mathbf{z}}$ with decreased uncertainty.

Conclusions and Future work

- The method provides efficient and reasonably accurate source inversion and UQ given scattered, noisy observations and utilizing a surrogate model trained on only a limited ensemble of simulations.
- We plan to explore the effect of increasing data and noise on the source prediction fidelity.
- There are a vast array of options for all stages of constructing the surrogate operator.
- We are developing an optimization suite to automate the architecture, loss, and training hyperparameters.
- Proceed to study 3D simulation datasets, including E3SM simulations of the Mt. Pinatubo eruption.