SAND2022-7855C

Facilitating Atmospheric Source Inversion via Deep Operator Network Surrogates

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

Sandia National Laboratories

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: SANDXXXX-XXXX XX

NNSA

U.S. DEPARTMENT OF ENERGY

dia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly ow
subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Source Inversion / Identification in Climate Systems

- Climate effects are caused by a combination of confounding sources which interact with the climate system through various feedbacks.
- The identification of sources in climate systems, be they natural or a result of human intervention, is a vital problem for attribution and prediction of climate states.
- Due to the inability to isolate sources in nature and the computational cost of climate simulators, surrogate models are required.
- Surrogate models enable the many-query algorithms required for exploration of the inverse problem of source identification.
- Because inversion for a source is an ill-posed problem, it is natural to treat it as a probabilistic problem, i.e., to construct a probability model that predicts the most probable source and quantifies uncertainty in the predicted source.
- In this talk, we will focus on a high-dimensional synthetic example with a view to applications involving E3SM and field data.

Summary of method and results

- Assuming the source represents injection of SO_2 and we observe concentratation, we introduce a framework to identify the source characteristics in time by:
  1. Calibrating deep operator network surrogates to the flow maps provided by an ensemble of simulations obtained by sources with different forcing time profiles,
  2. Setting up a Bayesian framework for a distribution over the forcing profiles,
  3. Using the trained surrogates and Bayesian framework together with automatic differentiation in an optimization framework to identify sources from sparse and noisy observations.
- We apply the framework to a synthetic model based on an advection-diffusion-reaction equation that includes effects of diffusion, wind, gravity, chemistry, and a volcano source and $O(10^5)$ degrees of freedom.
- The expressive and computationally efficient nature of the deep operator network surrogates allows for source identification in a complex system from comparatively limited data
- This opens the door for applications such as simulations/observations of Mt. Pinatubo eruption.

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 F and training data $(z^m, u^m)_{m=1}^M$ consisting of M known forcing amplitudes z^m and the corresponding observed concentration u^m, predict an unknown forcing amplitude z given noisy and sparse observations of a concentration u that is not in the training set.

The SO_2 Plume Synthetic Model

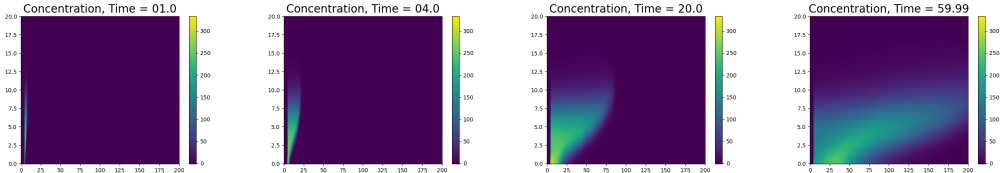

Figure 1: Illustration of the SO_2 concentration modeled by the Gaussian Plume synthetic model at various times.

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

The SO_2 Plume Synthetic Model

We consider a model for atmospheric transport of sulfur dioxide SO_2 in a rectangular region in space parametrized by longitudinal position x_1 and altitude x_2. We denote the concentration by $u(x_1, x_2, t)$ at time t. 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 κ is the diffusion coefficient,

$$\mathbf{v} = (v_1(x_1, x_2, t), 0)$$

describes the wind, with

$$\begin{aligned} v_1(x_1, x_2, t) = &\left(1 + 0.1 \cos\left(\frac{2\pi t}{60}\right)\right) \\ &\times \left(1 + 0.2 \cos\left(\frac{6\pi x}{40}\right) - 0.1 \sin\left(\frac{4\pi x}{40}\right)\right)\left(0.25 + 3.75 \sin\left(\frac{\pi x_2}{20}\right)\right). \end{aligned}$$

The SO_2 Plume Synthetic Model: $\frac{\partial u}{\partial t} - \kappa \Delta u + \mathbf{v} \cdot \nabla u - S\mathbf{e}_2 \cdot \nabla u = R(u) + f$

The term

$$S\mathbf{e}_2 = (0, S) = \left(0, \sqrt{\frac{8}{3} \frac{\rho_{\mathrm{SO}_2}}{\rho_{\mathrm{atmo}}} \frac{g}{C_s} r}\right)$$

describes the effect of the particles falling due to gravity with a terminal speed S, and

$$R(u) = -\gamma u$$

is the reaction function modeling chemistry with γ being the e-folding time. 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.

Surrogate Model Problem Statement

- State vector: discretized over a spatial grid of d 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 the map

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

that is trained using data generated by solving the above differential equations. Evaluation of this map will allow for inverting for $\mathbf{z}$ given observations of the state.

- **Rather than approximate this map directly**, we construct a surrogate for the flow map

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

Then, B is given by repeatedly composing A with itself.

Ensemble of solutions to calibrate surrogate

- Given M initial conditions and corresponding forcing functions, the discretized state vectors and forcing function vectors are written as

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

- Thus, the superscript m indexes different pairs of forcing amplitude and concentration, while for each m the subscript n indexes different times of the concentration.
- In the examples presented here, we take $\mathbf{u}_0^m = 0$, and vary $\mathbf{z}$.

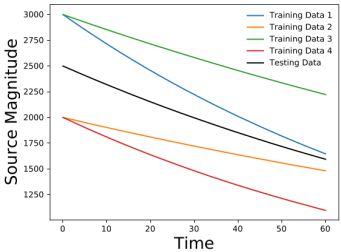

Figure 2: Ensemble of exponentially decay forcing amplitudes used to generate data for training and testing. The training data is generated using combinations of $\lambda_1 \in \{2000, 3000\}$ and $\lambda_2 \in \{0.005, 0.01\}$, and the test data is generated using $(\lambda_1, \lambda_2) = (2500, 0.0075)$.

PCA for dimension reduction

- PCA dimension reduction: given a target dimension r, we assemble the data and perform an SVD decomposition as

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

- Here, the left-hand side represents the $d \times MN$ matrix of all N state vectors corresponding to the M solutions, U_r represents the $d \times r$ truncated matrix of left singular vectors, Σ_r represents the diagonal matrix of first r singular values, and V represents the $MN \times r$ represents the truncated matrix of right singular vectors.
- Then

$$\begin{aligned}\left[\mathbf{c}_1^1|\dots|\mathbf{c}_N^1|\mathbf{c}_1^2|\dots|\mathbf{c}_N^2|\dots|\mathbf{c}_1^M|\dots|\mathbf{c}_N^M|\right] \\ = U_r^\top \left[\mathbf{u}_1^1|\dots|\mathbf{u}_N^1|\mathbf{u}_1^2|\dots|\mathbf{u}_N^2|\dots|\mathbf{u}_1^M|\dots|\mathbf{u}_N^M|\right]\end{aligned} \tag{11}$$

  gives the assembly of r-dimensional reduced state vectors $\mathbf{c}_n^m$ corresponding to $\mathbf{u}_n^m$.
- In other words, the $\mathbf{c}_n^m$ represent the coefficients of the state $\mathbf{u}_n^m$ in the PCA basis.

Operator Learning Setup

- PCA projection and reconstruction: left multiplication by $U_r^\top$ of a matrix Q with d rows is referred to as "PCA Projection":

$$\text{PCA Projection}\left(\ \text{Q}\ \right) = U_r^\top Q. \tag{12}$$

- Left multiplication by U of a matrix Q with r rows is referred to as "PCA reconstruction":

$$\text{PCA Reconstruction}\left(\ \text{Q}\ \right) = U_r Q. \tag{13}$$

- Operator learning setup in reduced coordinates: we seek A_{red} such that

$$A \approx \text{PCA reconstruction} \circ A_{\text{red}} \circ \text{PCA projection}, \tag{14}$$

i.e., so that the following diagram approximately commutes:

$$\begin{array}{ccc} u_n^m & \xrightarrow{A(\cdot,\mathbf{z})} & u_{n+1}^m \\ \text{PCA projection}\ \downarrow & & \uparrow\ \text{PCA reconstruction} \\ c_n^m & \xrightarrow[A_{\text{red}}(\cdot,\mathbf{z})]{} & c_{n+1}^m \end{array} \tag{15}$$

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

Deep Neural Network Operator Surrogate

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

$$\mathcal{NN}(\mathbf{c}, \mathbf{z}; \boldsymbol{\xi}) = W \circ \boldsymbol{\Phi}(\mathbf{c}, \mathbf{z}; \boldsymbol{\xi}^{\mathrm{H}}), \tag{16}$$

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

- We make use of a given neural network architecture, but instead of modeling $\mathbf{c}_{n+1} = \mathcal{NN}(\mathbf{c}_n, \mathbf{z})$, we have

$$\mathbf{c}_{n+1} = A_{\mathrm{red}}(\mathbf{c}_{n+1}, \mathbf{z}) = \mathbf{c}_n + \Delta t \mathcal{NN}(\mathbf{c}_n, \mathbf{z}), \quad n = 1, 2, ... \tag{17}$$

  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 aren't very deep, we use a plain DNN architecture:

$$\boldsymbol{\Phi}^{\mathrm{plain}} = \boldsymbol{\sigma} \circ T_L \circ \cdots \circ \boldsymbol{\sigma} \circ T_1 \tag{18}$$

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

Loss, Training, and Prediction

- 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 $m = 1, 2, ..., M$,

$$\text{Loss}(\boldsymbol{\xi}) = \sum_{m=1}^{M} \sum_{n=0}^{N-1} \sum_{p=1}^{P} \|\mathbf{c}_{n+p}^m - [\mathcal{NN}(\cdot, \mathbf{z}; \boldsymbol{\xi})]^p(\mathbf{c}_n^m, \mathbf{z})\|_{\ell^2}^2 \tag{19}$$

- The DNN is initialized using default Glorot initialization, and trained using the `adam` gradient descent optimizer with an exponentially decaying learning rate schedule.
- Prediction: given forcing vector $\mathbf{z}$, the prediction in reduced space is,

$$\mathbf{c}_n^{m,*} \approx [\mathcal{NN}(\cdot, \mathbf{z}; \boldsymbol{\xi})]^n \, \mathbf{c}_0^m \tag{20}$$

Then, the predicted state is given by

$$\mathbf{u}_n^{m,*} = \text{PCA Reconstruction}\,(\mathbf{c}_n^{m,*})\,. \tag{21}$$

Surrogate Flow Map Training Results

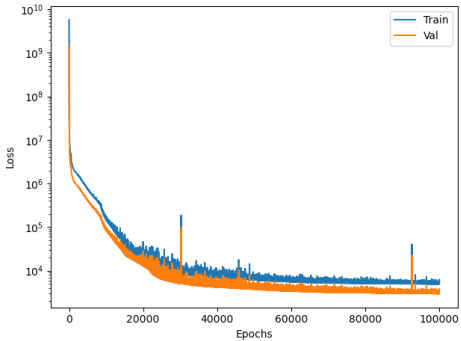

Figure 3: Loss vs number of steps of `adam` optimizer (Epochs) for an ensemble of 3 forcing, concentration pairs (Train). The same loss function is monitored for a different forcing, concentration pair (Val) to watch for overfitting.

There are many options for the selection of Δt, 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.

Bayesian Model

- Observation operator: $\mathcal{O} : \mathbb{R}^d \to \mathbb{R}^q$, where q is the number of locations where data is observed in space at each time step
- Assume that observed data is contaminated with mean zero Gaussian noise with a covariance matrix $\Gamma_{\text{noise}} \in \mathbb{R}^{q \times q}$. The likelihood function is defined as the difference between the DNN prediction $[\mathcal{NN}(\cdot, \mathbf{z}; \boldsymbol{\xi})]^n \mathbf{u}_0$ and the observed data

$$\mathcal{D} = [\mathcal{O}\mathbf{u}_0, \mathcal{O}\mathbf{u}_1, ..., \mathcal{O}\mathbf{u}_N] \tag{22}$$

in the noise model weighted inner product,

$$\pi_{\text{like}}(\mathbf{z}|\mathcal{D}) \propto \exp\left(-\frac{1}{2}\sum_{n=1}^{N} \|\mathcal{O}\mathbf{u}_n - \mathcal{O}\left[\mathcal{NN}(\cdot, \mathbf{z}; \boldsymbol{\xi})\right]^n \mathbf{u}_0\|^2_{\Gamma^{-1}_{\text{noise}}}\right) \tag{23}$$

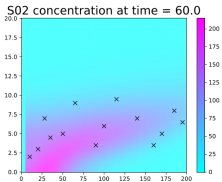

Figure 4: Sparsely scattered observations of the concentration at a given timestep, from which the source $\mathbf{z}$ is inferred.

Bayesian Model

- Prior distribution: assume a Gaussian distribution for the source $\mathbf{z}$ as a prior with mean $\overline{\mathbf{z}}$ and covariance $\Gamma_{\text{prior}} \in \mathbb{R}^{k \times k}$:

$$\pi_{\text{prior}}(\mathbf{z}) \propto \exp\left(-\frac{1}{2}\|\mathbf{z}-\overline{\mathbf{z}}\|^2_{\Gamma^{-1}_{\text{prior}}}\right) \tag{24}$$

- Bayesian inverse problem to estimate a source $\mathbf{z}$,

$$\pi_{\text{post}}(\mathbf{z}|\mathcal{D}) \propto \pi_{\text{like}}(\mathbf{z}|\mathcal{D})\pi_{\text{prior}}(\mathbf{z}), \tag{25}$$

  where π_{post} is the posterior probability distribution for $\mathbf{z}$ given the prior distribution π_{prior} for $\mathbf{z}$ and the likelihood function π_{like}.

- Maximum a posteriori probability (MAP) point: the point $\mathbf{z}$ for which the posterior PDF π_{post} attains its maximum value. It may be computed by minimizing the negative log of π_{post}, or equivalently solving

$$\min_{\mathbf{z}\in\mathbb{R}^k} J(\mathbf{z}) = \frac{1}{2}\sum_{n=1}^{N} \|\mathcal{O}\mathbf{u}_n - \mathcal{O}\left[\mathcal{NN}(\cdot,\mathbf{z};\boldsymbol{\xi})\right]^n \mathbf{u}_0\|^2_{\Gamma^{-1}_{\text{noise}}} + \frac{1}{2}\|\mathbf{z}-\overline{\mathbf{z}}\|^2_{\Gamma^{-1}_{\text{prior}}} \tag{26}$$

Optimization and 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.
- The gradient is computed by solving the adjoint equation corresponding to the discrete time stepping algorithm
- The Neural Network Jacobians are computed using the automatic differentiation tools in Tensorflow, and passed to ROL.
- A Gauss-Newton Hessian approximation is used to leverage for the Neural Network Jacobians for an efficient Hessian approximation.
- We approximate samples from π_{post} 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 $\mathbf{z}_{\text{MAP}}$.

Summary of Model and Optimization

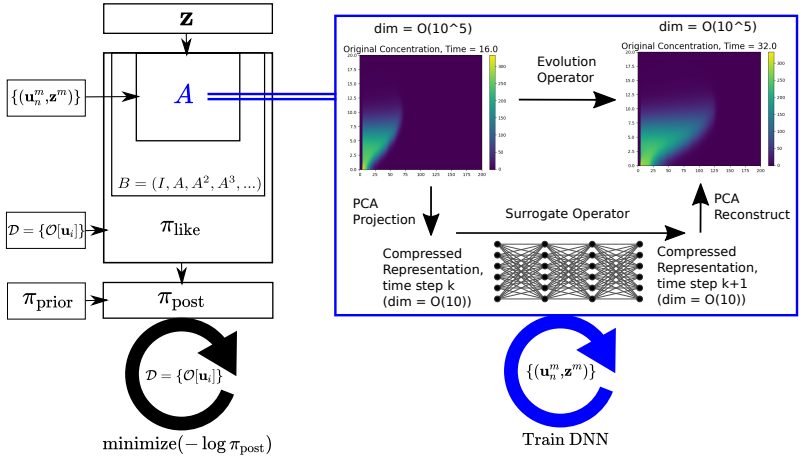

Figure 5: Diagram of the complete Bayesian model for evaluating posterior probability of $\mathbf{z}$ given noisy, scattered observations utilizing a surrogate model for the flow map that is informed by an ensemble of simulations.

Results

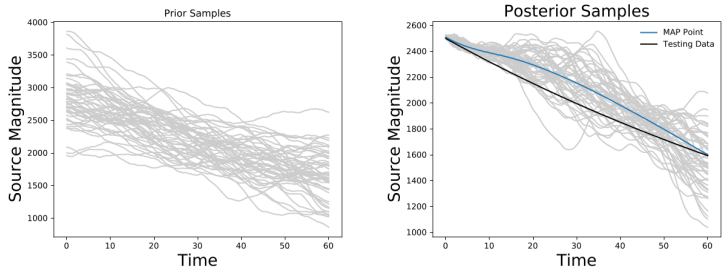

Figure 6: Prior (left) and posterior (right) distribution over $\mathbf{z}$, predicted using the observations illustrated in Figure 4 and the data shown in Figure 2.

- The observations for the prediction of $\mathbf{z}$ are contaminated with multiplicate Gaussian noise
- For the posterior: the black curve is the testing data (which was never used in the training), the blue curve is the MAP point, and the grey curves are samples from the posterior distribution.

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.
- The effect of increasing/decreasing data and noise on the source prediction fidelity must be explored.
- There are a vast array of options for all stages of constructing the surrogate operator A. We are working on an optimization suite to automate the architecture, loss, and training hyperparameter optization of the trained A.
- Proceed to study 3D simulation datasets, including E3SM simulations of the Mt. Pinatubo eruption.