Supplemental Materials

Exp. S & N Data

Metrics For Optimization

Collection of Values Measuring:

"Feasibility"
The likelihood of these conditions existing in **ANY** experimental scenarios *(within reason)*.
Better for **Generalization**.

OR

"Relevance"
The likelihood of these conditions existing in the **SPECIFIC** experimental scenarios we are studying. (e.g., a specific machine)

Should Be Formulated As Follows:

$$0 > M_B P_{Thresh} = f(N) - P_{Thresh}$$

- Normalizes the metric and prevents any distinction between samples considered 'Feasible'.
- Extracts specific characteristics of the density evolution.
- Defined threshold that acts as a boundary of 'Feasibility' classes.

Some Potential Metrics

1) Maximum (Global) Density
$f(N) = \max(N)$ | $P_{Thresh} \sim N_G$

2) Maximum Edge Density
$f(N) = \max(N(r_{max}))$ | arb. P_{Thresh}

3) Minimum Core Density
$f(N) = \min(N(r=0))$ | arb. P_{Thresh}

4) Maximum Inversion
$f(N) = \max(N(r=r_c)/N(r=0))$ | arb.

Used In

Experimental Data

Structure of Input Data

1) Density Evolution → $N(r, t)$
2) Ionization Source → $N(r, t)$

In the future, this pipeline can be adapted to more **Geometries**, **Dimensions**, **Measurements**, and **Models**.

Model Assumptions

We assume that there exists some known mapping between inputs:

$$F_1(I_1, P) = F_2(I_2)$$

*Where **P** are unknown parameters.*
*This can be a **differential mapping**.*

Informs

Informs

Representative Forms & Parameter Space

Defining Rep. Forms

We reduce the dimensionality of the profiles through assigning a **Representative Form** for each such that they are constructed as follows:

$$P_i = f_i(p_{(i,1)}, \dots, p_{(i,n)})$$

*Where P_i are the profiles and $p_{(i,n)}$ are **nI Free Parameters** which encode all of the information needed to construct P_i.*
Notice that this is stipulating that all relevant profiles can be represented in this form

Constructing Param. Space

We define our **Parameter Space** by setting boundaries on each **Free Parameter** which was defined. We must also must set which ranges should be on a **logarithmic** vs. **linear scale**:

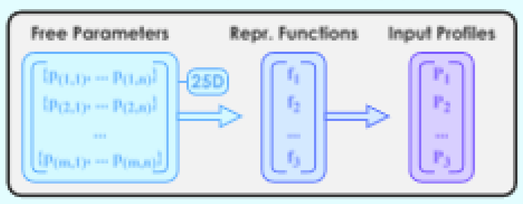
$$\min_{(i,n)} < P_{(i,n)} < \max_{(i,n)}$$
$$\text{Scale}_{(i,n)} = \text{Log} \,/\, \text{Linear}$$

*The **Scale** assignment, while not theoretically necessary, allows the **Parameter Space** to better reflect the **Probability Space**.*

Used In

Profile Constructor

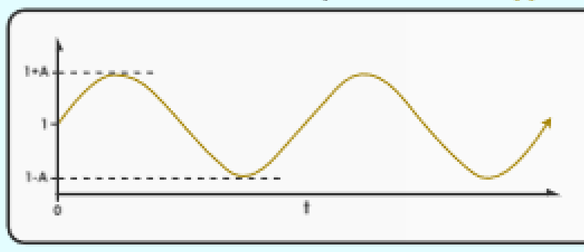
Module Overview & Structure



Profile Types

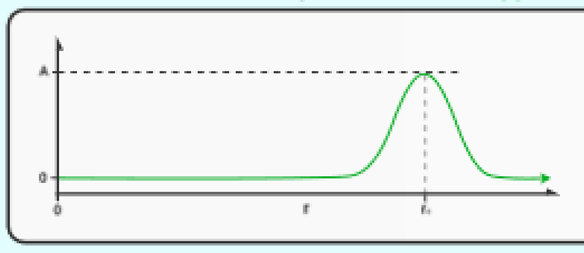
Input Profiles
T. Source - 1D Time - (ST)
S. Source - 1D Space - (SR)
Diffusion - 1D Space - (D)
Convection - 1D Space - (V)
Init. Density - 1D Space - (N0)
Note that S is given by $S(x,t) = SR(r)ST(t)$

Temporal Source: ST(t)



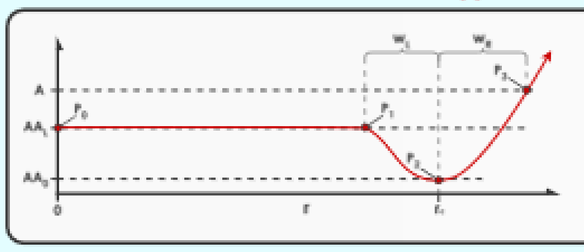
Repr. Function:
$ST(t) = A\sin(2\pi ft + \theta) + 1$
Free Params:
$P_{ST} = \{A, f, \theta\}$

Spatial Source: SR(r)



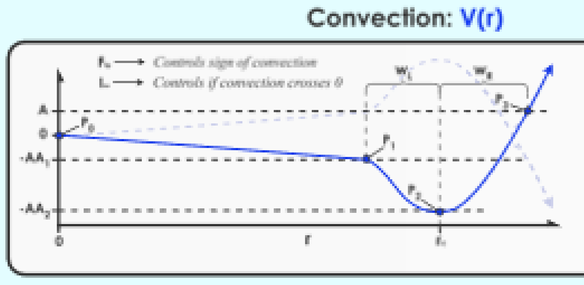
Repr. Function:
$SR(r) = Ae^{-(r - r_c)^2/\sigma^2}$
Free Params:
$P_{SR} = \{A, r_c, \sigma\}$

Diffusion: D(r)



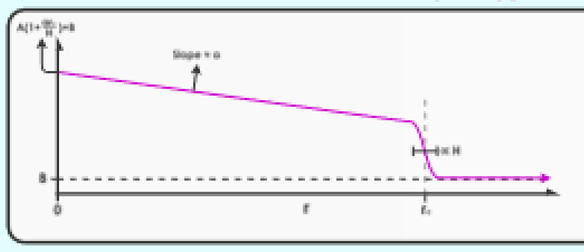
Repr. Function:
$D(r) = PCHip(P_0, P_1, P_2, P_3)$
Free Params:
$P_D = \{A, A_1, A_2, r_c, w_1, w_2\}$

Convection: V(r)



Repr. Function:
$V(r) = PCHip(P_0, P_1, P_2, P_3)$
Free Params:
$P_V = \{A, A_1, A_2, r_c, w_1, w_2, P_n, L_n\}$

Initial Density: N0(r)



Repr. Function:
$N0(r) = A\,\text{Mtanh}(\frac{r_c - r}{H}, \alpha) + B$
Free Params:
$P_{N0} = \{A, r_c, H, \alpha, B\}$

Samples

Profiles

Mapping Optimization Loop

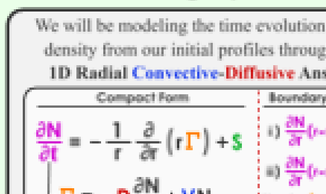
Bayesian Optimization

In **Bayesian Optimization**, we seek to minimize a given **Objective Function** within a given **Parameter Space**:

$$y \leftarrow f_{obj}(\vec{Z}) \mid \vec{Z} \in \vec{P}$$

To do so, we construct a **Surrogate Model** of the **Probability Space**:

$$P(y \mid \vec{Z})$$



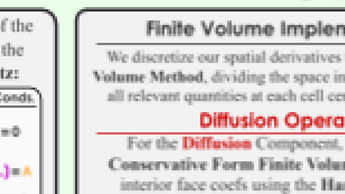
Tree-Struct. Parzen Est. (TPE)

TPE flips the normal surrogate relation and stores two surrogate models:

"Good Model": $P(\vec{Z} \mid y \le y_*) \rightarrow P(\vec{Z} \mid \checkmark)$
"Bad Model": $P(\vec{Z} \mid y > y_*) \rightarrow P(\vec{Z} \mid \times)$

Samples are then selected by finding the most **"Promising"** out of a random set:

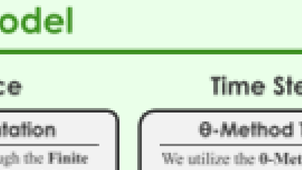
$$\text{"Promise"} \propto P(\vec{Z} \mid \checkmark)/P(\vec{Z} \mid \times)$$



Kernel Density Est. (KDE)

Constructing surrogate models requires us to estimate the represented probability, typically through **Kernel Smoothing:**

$$P(\vec{Z} \mid \checkmark) = \sum K(\vec{Z} \mid \checkmark)$$



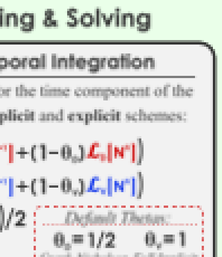
With multiple inputs, **KDE** can include **Covariance *H*** between parameters, which models **Cross-Parameter Relationships**.

Pareto Archive

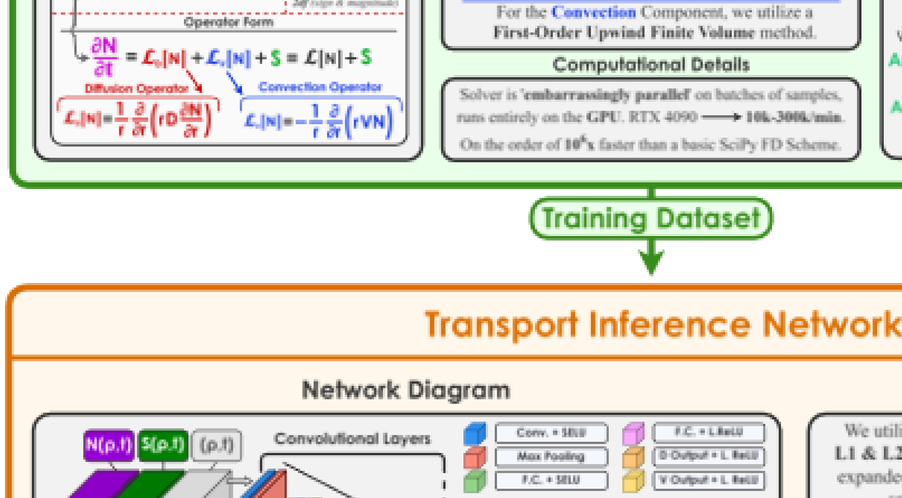
With multiple inputs, the **Objective Function** no longer outputs a scalar:

$$\vec{y} \leftarrow F_{obj}(\vec{Z}) \mid \vec{Z} \in \vec{P}$$

To keep track of the best samples, we implement a **Pareto Archive** to store:



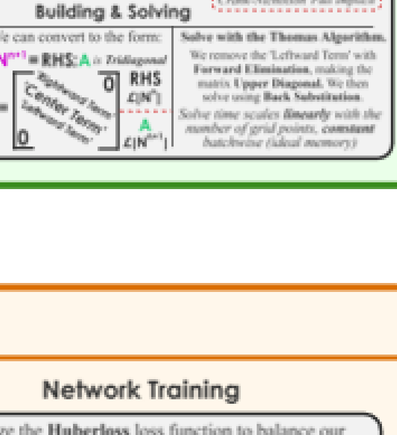
Optimization Pipeline



Analysis of Mapping

T-SNE High Dim. Projection

The **T-SNE** algorithm projects high-dimensional data down to viewable dimensions (2 or 3) through a **Density Weighted Distance** metric for similarity.



*This allows us to compare the **Similarity** of distributions of high-dimensional data and gives insight into the **Underlying Structure**.*
*Note the T-SNE is **Relative** to the **Bulk Distribution**!*

N Evolutions

Numerical Model

Modeling Equation

We will be modeling the time evolution of the density from our initial profiles through the **1D Radial Convective-Diffusive Ansatz:**

Compact Form

$$\frac{\partial N}{\partial t} = -\frac{1}{r}\frac{\partial}{\partial r}(r\Gamma) + S$$
$$\Gamma = -D\frac{\partial N}{\partial r} + VN$$

Boundary Conds.
i) $\frac{\partial N}{\partial r}(r=0) = 0$
ii) $\frac{\partial N}{\partial r}(r=r_{max}) = A$
Note that A is treated as a free parameter, adding ΔdV (sign & magnitude)

Operator Form

$$\frac{\partial N}{\partial t} = \mathcal{L}_D[N] + \mathcal{L}_v[N] + S = \mathcal{L}[N] + S$$

Diffusion Operator: $\mathcal{L}_D[N] = \frac{1}{r}\frac{\partial}{\partial r}\left(rD\frac{\partial N}{\partial r}\right)$
Convection Operator: $\mathcal{L}_v[N] = -\frac{1}{r}\frac{\partial}{\partial r}(rVN)$

Discretizing Space

Finite Volume Implementation

We discretize our spatial derivatives through the **Finite Volume Method**, dividing the space into cells and defining all relevant quantities at each cell center and boundary.

Diffusion Operator

For the **Diffusion** Component, we utilize the **Conservative Form Finite Volume** method, with interior face coefs using the **Harmonic Mean.**

Convection Operator

For the **Convection** Component, we utilize a **First-Order Upwind Finite Volume** method.

Computational Details

Solver is **'embarrassingly parallel'** on batches of samples, runs entirely on the **GPU**. RTX 4090 ⟶ **10k-300k/min**.
On the order of 10^6**x** faster than a basic SciPy FD Scheme.

Time Stepping & Solving

θ-Method Temporal Integration

We utilize the **θ-Method** for the time component of the PDE, which combines **implicit** and **explicit** schemes:

$$N^{n+1} = N^n + \Delta t\left(\theta_v \mathcal{L}_D[N^{n+1}] + (1-\theta_v)\mathcal{L}_v[N^n]\right)$$
$$+ \Delta t\left(\theta_v \mathcal{L}_v[N^{n+1}] + (1-\theta_v)\mathcal{L}_v[N^n]\right)$$
$$+ \Delta t\left(S^{n+1} + S^n\right)/2$$

Default Theta: $\theta_D = 1/2$ *Crank-Nicholson* $\theta_v = 1$ *Full Implicit*

Matrix Form Building & Solving

We can convert to the form: $AN^{n+1} = RHS$; A *is Tridiagonal*

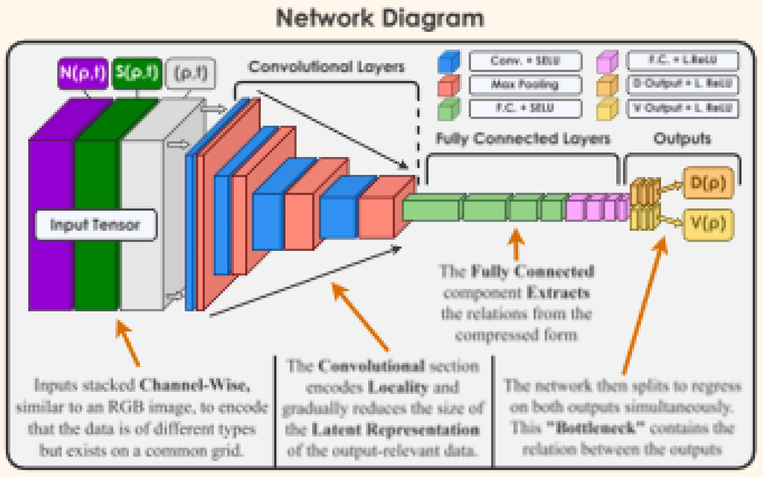
$$A = \begin{bmatrix} \text{Diagonal Term} & \text{Upward Term} & 0 \\ \text{Downward Term} & \text{Center Term} & \text{Upward Term} \\ 0 & \text{Downward Term} & \text{Diagonal Term} \end{bmatrix} \quad RHS = \begin{bmatrix} 0 \\ \mathcal{L}[N^n] \\ A\mathcal{L}[N^{n+1}] \end{bmatrix}$$

Solve with the Thomas Algorithm.
We remove the 'Leftward Term' with **Forward Elimination**, making the matrix **Upper Diagonal.** We then solve using **Back Substitution**.
*Solve time scales **linearly** with the number of grid points, **constant** batchwise (ideal memory)*

Training Dataset

Transport Inference Network

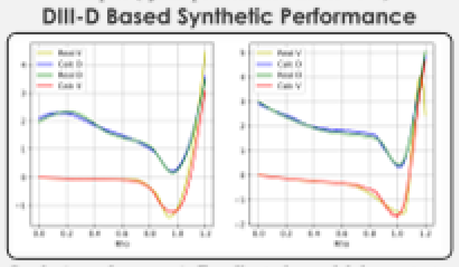
Network Diagram



Network Training

We utilize the **Huberloss** loss function to balance our **L1 & L2 Error Penalties**. Training is done on either an expanded set of 100k samples or a smaller set with 10k samples (synth. performance differs ~5%)

DIII-D Based Synthetic Performance



*Synthetic performance is **Excellent**; the model demonstrates the ability to **Invert** the relations of our numerical model. Model training on the expanded set takes ~1-2 hours on 4090.*

Inferred D & V

Evaluating Inferred Exp. Transport.

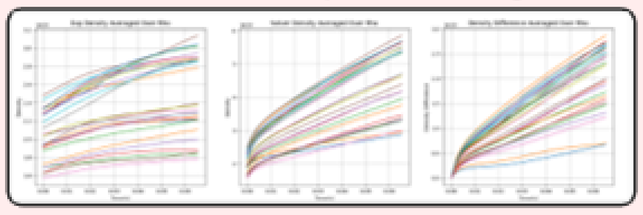
Quantifying Experimental Performance

There is no reliable source of experimental transport parameter data at large scales, we need a **different approach** than traditional direct error metrics.

Inferred D & V ⟶ Density Solver ⟶ Solved N(r,t)

*We **compare** with experiment as the evolution **encodes information** about transport & fueling.* → Experimental N(r, t)

Performance on Unoptimized Training Data



Exp. S & N Data