Physics-Informed Neural Network in Groundwater Inverse Modeling

Quan Guo

Advisor: Dr. Jian Luo

Water Resource Engineering
Civil Engineering

Georgia Tech

Outline

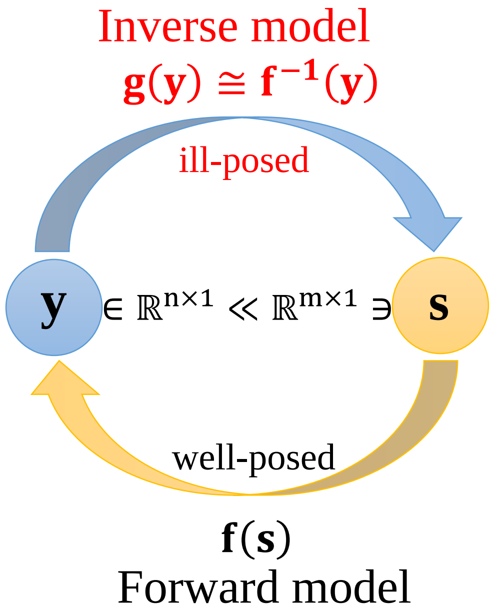
- Groundwater Inverse Modeling
- Physics-Informed Neural Network (PINN)
- Groundwater PINN
- Future Work

Groundwater Inverse Modeling

General model for measurable system

$$\mathbf{y} = \mathbf{f}(\mathbf{s}) + \boldsymbol{\epsilon}$$

$\mathbf{y}$ – Available measurements, $\in \mathbb{R}^{n\times 1}$
$\mathbf{f}$ – A deterministic model (such as governing PDE in physical system)
$\mathbf{s}$ – Interested variables or parameters in forward model, $\in \mathbb{R}^{m\times 1}$
$\boldsymbol{\epsilon}$ – Noise term caused by measurement error, limitation of exactness



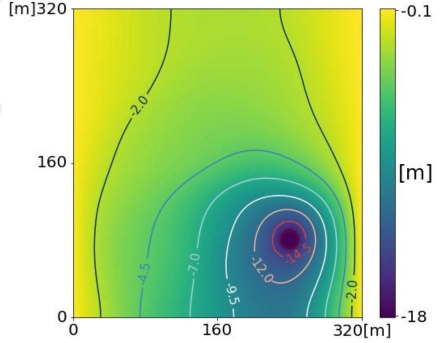
To solve inverse problem, we usually need implement **iterative method** and introduce **regularization** based on background knowledge.

Groundwater inverse modeling

Groundwater physical system

y – hydraulic heads (*h*)

- Point measurements – fixated locations
- Solved by finite element method with hydrological and hydraulic parameters well known
- Hundreds of measurements



Inverse model

$\mathbf{s} \cong \mathbf{f}^{-1}(\mathbf{y})$

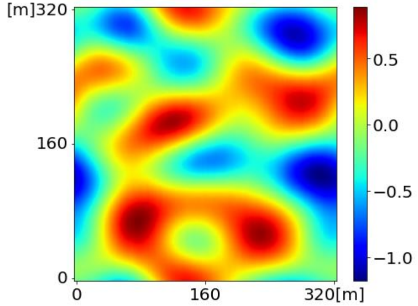
ill-posed

$\mathbf{y} \in \mathbb{R}^{n\times 1} \ll \mathbb{R}^{m\times 1} \ni \mathbf{s}$

well-posed

$\mathbf{y} = \mathbf{f}(\mathbf{s})$

Forward model



s – hydraulic transmissivity (*T*) its natural logarithm (*lnT*)

- Large scale – field sites
- Large-dimensional – 10^6 unknowns
- Dozens of measurements (sparse)
- Spatially-correlated – geostatistics

Georgia Tech

Groundwater Inverse Modeling

Groundwater physical system

$$S_s \frac{\partial h}{\partial t} = -\nabla \cdot \mathbf{q} + Q \qquad \text{Mass conservation}$$
$$\mathbf{q} = T \nabla h \qquad \text{Darcy's Law}$$

S_s – specific storage; h – hydraulic head; T – hydraulic transmissivity; $\mathbf{q}$ – flux; Q – source/sink

Conditions of a pumping event, :

1. Pumping at one location – (x_p, y_p)
2. Constant pumping rate – Q_p

Simplifications:

1. Steady state – $\frac{\partial h}{\partial t}$
2. 2D domain – (x, y)
3. Isotropic and confined aquifer

For non-pumping grid: $\nabla \cdot [T(x_e, y_e) \nabla h(x_e, y_e)] = 0, \quad (x_e, y_e) \in \Omega$

For pumping grid: $\nabla \cdot [T(x_p, y_p) \nabla h(x_p, y_p)] = Q_p, \quad (x_p, y_p) \in \Omega$

Neumann boundary: $\mathbf{n} \cdot \nabla h(x_N, y_N) = q_N, \quad (x_N, y_N) \in \Gamma_N$

Dirichlet boundary: $h(x_D, y_D) = h_D, \quad (x_D, y_D) \in \Gamma_D$

$$h = \mathrm{FEM}(T; \Gamma_N, \Gamma_D)$$

Georgia Tech

Groundwater Inverse Modeling

Geostatistical Approach – Bayesian Inference

Likelihood of unknown variable $\mathbf{s}$ given data $\mathbf{y}$

$$\boxed{\mathrm{p}(\mathbf{s}|\mathbf{y})} = \frac{\boxed{\mathrm{p}(\mathbf{y}|\mathbf{s})}\boxed{\mathrm{p}(\mathbf{s})}}{\int \mathrm{p}(\mathbf{y}|\mathbf{s})\mathrm{p}(\mathbf{s})\mathrm{d}\mathbf{s}}$$

Prior distribution of unknown variable $\mathbf{s}$

Posterior distribution of unknown variable $\mathbf{s}$

Georgia Tech

Groundwater Inverse Modeling

Geostatistical Approach – Bayesian Inference

$$p(\mathbf{y}|\mathbf{s}) \propto \exp\left(-\frac{1}{2}(\mathbf{y}-\mathbf{f}(\mathbf{s}))^{\mathrm{T}}\mathbf{R}^{-1}(\mathbf{y}-\mathbf{f}(\mathbf{s}))\right)$$

Likelihood of unknown variable $\mathbf{s}$ given data $\mathbf{y}$

$$\boxed{p(\mathbf{s}|\mathbf{y})} = \frac{\boxed{p(\mathbf{y}|\mathbf{s})}\boxed{p(\mathbf{s})}}{\int p(\mathbf{y}|\mathbf{s})p(\mathbf{s})d\mathbf{s}}$$

Prior distribution of unknown variable $\mathbf{s}$

$$p(\mathbf{s}) \propto \exp\left(-\frac{1}{2}(\mathbf{s}-\boldsymbol{\mu}_{\mathbf{s}})^{\mathrm{T}}\mathbf{C}^{-1}(\mathbf{s}-\boldsymbol{\mu}_{\mathbf{s}})\right)$$

Posterior distribution of unknown variable $\mathbf{s}$

Covariance matrix of error

Prior mean of variable $\mathbf{s}$

$$p(\mathbf{s}|\mathbf{y}) \propto \exp\left(-\frac{1}{2}(\mathbf{y}-\mathbf{f}(\mathbf{s}))^{\mathrm{T}}\mathbf{R}^{-1}(\mathbf{y}-\mathbf{f}(\mathbf{s})) - \frac{1}{2}(\mathbf{s}-\boldsymbol{\mu}_{\mathbf{s}})^{\mathrm{T}}\mathbf{C}^{-1}(\mathbf{s}-\boldsymbol{\mu}_{\mathbf{s}})\right)$$

Covariance matrix of variable $\mathbf{s}$

Georgia Tech

Groundwater Inverse Modeling

Geostatistical Approach – Bayesian Inference

Covariance matrix of error

Prior mean of variable $\mathbf{s}$

$$\ell(\mathbf{s}) = \frac{1}{2}\left(\mathbf{y} - \mathbf{f}(\mathbf{s})\right)^{\mathrm{T}} \mathbf{R}^{-1}\left(\mathbf{y} - \mathbf{f}(\mathbf{s})\right) + \frac{1}{2}(\mathbf{s} - \mathbf{X}\boldsymbol{\beta})^{\mathrm{T}} \mathbf{C}^{-1}(\mathbf{s} - \mathbf{X}\boldsymbol{\beta}) \qquad \mathbf{s} \sim \mathrm{N}(\mathbf{X}\boldsymbol{\beta}, \mathbf{C})$$

Covariance matrix of variable $\mathbf{s}$

$$\mathbf{s} \in \mathbb{R}^{m\times 1} \quad \mathbf{s} = \mathbf{X}\boxed{\boldsymbol{\beta}} + \mathbf{C}\overline{\mathbf{H}}^{\mathrm{T}}\boxed{\boldsymbol{\xi}}$$

Intermediate Variable $\in \mathbb{R}^{n\times 1}$

Drift Coefficient $\in \mathbb{R}^{p\times 1}$

Georgia Tech

Groundwater Inverse Modeling

Geostatistical Approach – MAP Solution

$$\frac{\partial \ell(\mathbf{s})}{\partial \boldsymbol{\xi}} = 0 \qquad \frac{\partial \ell(\mathbf{s})}{\partial \boldsymbol{\beta}} = 0 \quad \xrightarrow{\text{yields}} \quad \overset{(n+p)\times(n+p)}{\boxed{\begin{bmatrix} \overline{\mathbf{H}}\mathbf{C}\overline{\mathbf{H}}^{\mathrm{T}} + \mathbf{R} & \overline{\mathbf{H}}\mathbf{X} \\ (\overline{\mathbf{H}}\mathbf{X})^{\mathrm{T}} & 0 \end{bmatrix}}} \begin{bmatrix} \hat{\boldsymbol{\xi}} \\ \hat{\boldsymbol{\beta}} \end{bmatrix} = \begin{bmatrix} \mathbf{y} - \mathbf{f}(\mathbf{s}_{\mathrm{i}-1}) + \overline{\mathbf{H}}\mathbf{s}_{\mathrm{i}-1} \\ 0 \end{bmatrix}$$

$$\mathbf{s}_{\mathrm{i}} = \mathbf{X}\hat{\boldsymbol{\beta}} + \mathbf{C}\overline{\mathbf{H}}^{\mathrm{T}}\hat{\boldsymbol{\xi}}$$

Computaional complexity of $\overline{\mathbf{H}}\mathbf{C}\overline{\mathbf{H}}^{\mathrm{T}} + \mathbf{R}$ is $O(mn^2)$, which is not scalable to n, i.e., dimension of s.

Georgia Tech

Groundwater Inverse Modeling

Geostatistical Approach – Bottleneck

- Large-dimensional inverse problem: up to millions of unknown variables
  - Storage of matrices: covariance matrices: $O(n^2)$
  - Matrix computation: $O(mn^2)$
  - Determination of Jacobian matrices: $O(mn)$
- Non-Gaussian posterior due to the complexity and non-linearity of $\mathbf{f}(\mathbf{s})$
- Large number of iterative forward model runs for nonlinear inverse problems
- High computational cost of each forward model run on large-dimensional parameters fields

A Recent USGS Case:
Inversion of a 21×21×52 hydraulic conductivity field given 4,000 transient drawdown measurements took 18 days with the help of massive parallelization on the USGS high-performance computing facilities, and may require 140 days on desktop computers *[Tiedeman and Barrash, 2019]*.

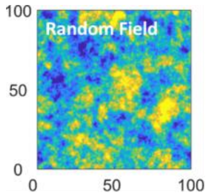
Georgia Tech

Groundwater Inverse Modeling

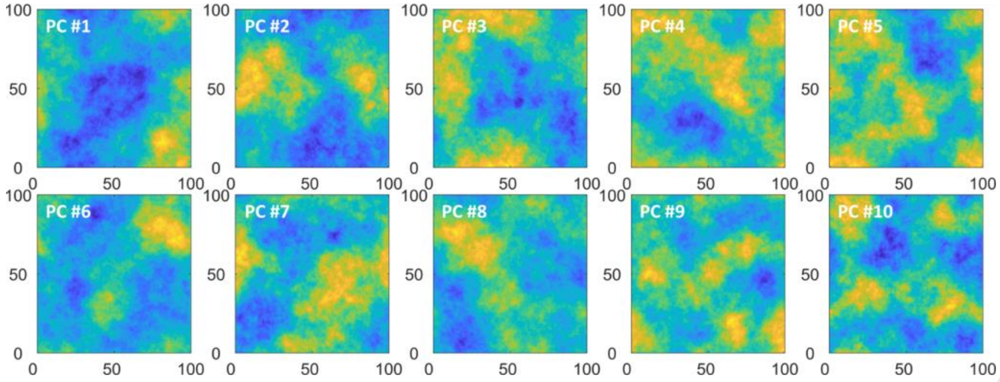
Geostatistical Approach – Improvements

- New framework and computational approaches for geostatistical approach (GA):
  - Apply principal component analysis (PCA) to reduce problem dimension.
  - Reformulate the geostatistical approach onto principal component coefficients (RGA).

[Zhao and Luo, 2020]



- $\mathbf{C} = \mathbf{V}\mathbf{D}\mathbf{V}^{\mathrm{T}} = [\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_{\mathrm{m}}] \begin{bmatrix} D_1 & 0 & 0 & 0 \\ 0 & D_2 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & D_m \end{bmatrix} \begin{bmatrix} \mathbf{v}_1{}^{\mathrm{T}} \\ \mathbf{v}_2{}^{\mathrm{T}} \\ \vdots \\ \mathbf{v}_m{}^{\mathrm{T}} \end{bmatrix}$



Groundwater Inverse Modeling

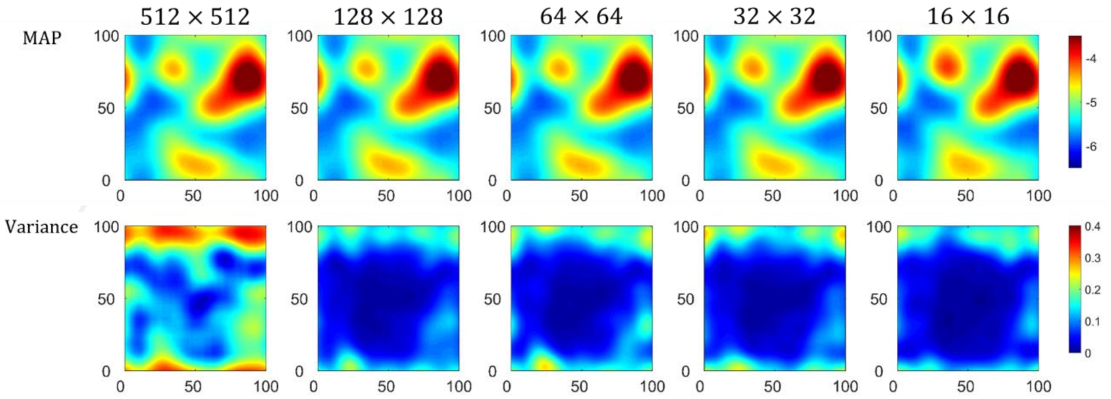
Reformulated Geostatistical Approach – Achievements

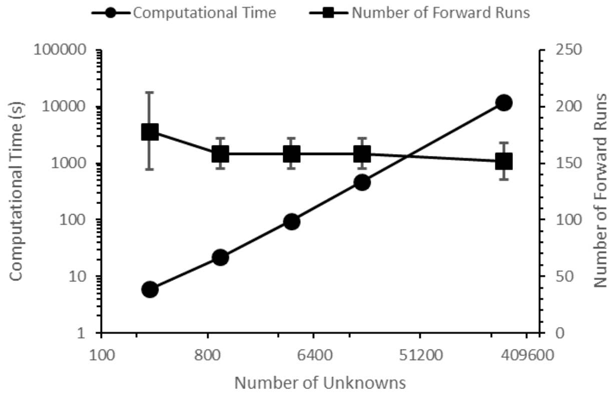
$$\mathbf{s} = \mathbf{X}\boldsymbol{\beta} + \mathbf{V}_k\mathbf{a}$$

$$\frac{\partial \ell(\mathbf{s})}{\partial \mathbf{a}} = 0 \qquad \frac{\partial \ell(\mathbf{s})}{\partial \boldsymbol{\beta}} = 0 \quad \xrightarrow{\text{yields}} \quad \underbrace{\boxed{\begin{bmatrix} \overline{\mathbf{H}}_{\mathbf{a}}^{\mathrm{T}}\mathbf{R}^{-1}\overline{\mathbf{H}}_{\mathbf{a}} + \mathbf{I} & \overline{\mathbf{H}}_{\mathbf{a}}^{\mathrm{T}}\mathbf{R}^{-1}\overline{\mathbf{H}}_{\boldsymbol{\beta}} \\ \overline{\mathbf{H}}_{\boldsymbol{\beta}}^{\mathrm{T}}\mathbf{R}^{-1}\overline{\mathbf{H}}_{\mathbf{a}} & \overline{\mathbf{H}}_{\boldsymbol{\beta}}^{\mathrm{T}}\mathbf{R}^{-1}\overline{\mathbf{H}}_{\boldsymbol{\beta}} \end{bmatrix}}}_{(k+p)\times(k+p)} \begin{bmatrix} \hat{\mathbf{a}} \\ \hat{\boldsymbol{\beta}} \end{bmatrix} = \begin{bmatrix} \overline{\mathbf{H}}_{\mathbf{a}}^{\mathrm{T}}\mathbf{R}^{-1}\left(\mathbf{y} - \mathbf{f}(\bar{\mathbf{a}}, \bar{\boldsymbol{\beta}}) + \overline{\mathbf{H}}_{\mathbf{a}}\bar{\mathbf{a}} + \overline{\mathbf{H}}_{\boldsymbol{\beta}}\bar{\boldsymbol{\beta}}\right) \\ \overline{\mathbf{H}}_{\boldsymbol{\beta}}^{\mathrm{T}}\mathbf{R}^{-1}\left(\mathbf{y} - \mathbf{f}(\bar{\mathbf{a}}, \bar{\boldsymbol{\beta}}) + \overline{\mathbf{H}}_{\mathbf{a}}\bar{\mathbf{a}} + \overline{\mathbf{H}}_{\boldsymbol{\beta}}\bar{\boldsymbol{\beta}}\right) \end{bmatrix}$$

As k is usually a small number, computational complexity of $\overline{\mathbf{H}}_{\mathbf{a}}\mathbf{C}\overline{\mathbf{H}}_{\mathbf{a}}^{\mathrm{T}} + \mathrm{I}$ is $O(n)$.

This RGA algorithm is much more scalable!





More Efficient & Scalable Model?

Matrix-wise to Pointwise Estimator

Hope to find a pointwise, continuous function to approximate the spatial variables:

$$F(x, y) \cong h(x, y)$$
$$G(x, y) \cong T(x, y)$$

What type of function should it be?

Ploynomial? Exponential? or more complicated form?

In 2019, Physics-Informed Neural Network (PINN) was borned.

[Raissi., et al, 2019]

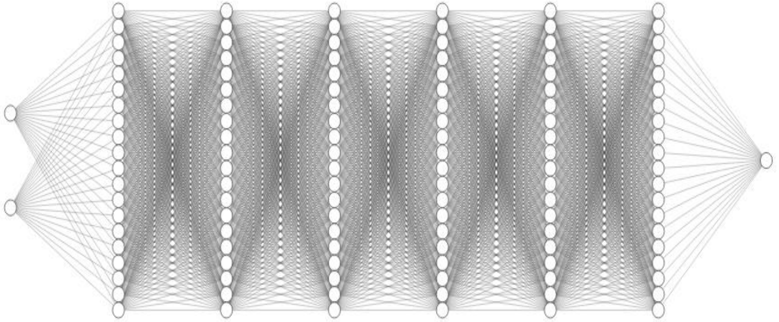
Physics-Informed Neural Network

Deep Neural Network

Neural network is a (recurrent) nonlinear regression model with learnable coefficients.

$$\boldsymbol{h}_1 = \sigma_0(\boldsymbol{W}_0\boldsymbol{x} + \boldsymbol{b}_0)$$
$$\boldsymbol{h}_2 = \sigma_1(\boldsymbol{W}\boldsymbol{h}_1 + \boldsymbol{b}_1)$$
$$\dots\dots$$
$$\boldsymbol{h}_n = \sigma_{n-1}(\boldsymbol{W}_{n-1}\boldsymbol{h}_{n-1} + \boldsymbol{b}_{n-1})$$
$$\boldsymbol{y} = \sigma_n(\boldsymbol{W}_n\boldsymbol{h}_n + \boldsymbol{b}_n)$$
$$\boldsymbol{y} = f_n(f_{n-1}(\dots f_1(f_0(\boldsymbol{x}))))$$

$\boldsymbol{x}$ is input variable vector;
$\boldsymbol{y}$ is predicted output vector;
$\boldsymbol{h}_i$ is hidden feature vector;
$\sigma_i(\cdot)$ is chosen nonlinear map;
$\boldsymbol{W}_i$ is learnable weight matrix;
$\boldsymbol{b}_i$ is learnable bias vector.



Neural network can become a universal function approximator as its depth is sufficiently large, e.g., deep neural network (DNN).
[Goodfellow, I., 2016]

DNN usually has a lot of coefficients, the number can be up to billions.

Georgia Tech

Physics-Informed Neural Network

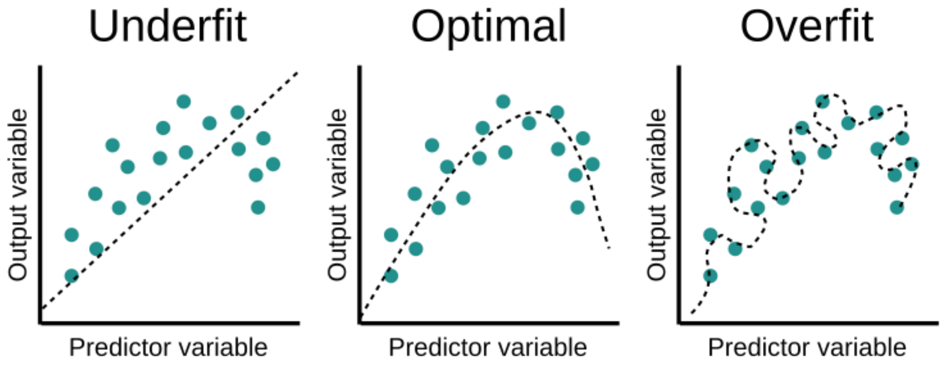
DNN training

Training DNN means learn the value of $\boldsymbol{W}_i$and $\boldsymbol{b}_i$ to make predictions closet to data. This data fitting process is not much different from regression.

Loss function: mean squared error $l = \frac{1}{m}\sum_{j=1..m}\left\|\widehat{\boldsymbol{y}}^j - \boldsymbol{y}^j\right\|_2^2$

Optimizer: Newton method $\boldsymbol{\theta}_i^{k+1} = \boldsymbol{\theta}_i^k + \eta\frac{\partial \boldsymbol{l}}{\partial \boldsymbol{\theta}_i^k}$; $\boldsymbol{\theta}_i = \{\boldsymbol{W}_i;\ \boldsymbol{b}_i\}$

To obtain a robust DNN model, it requires fast computation and large amount of data. Otherwise, it is easily overfitting since there are too many coefficients in the model.



Physics-Informed Neural Network

Physical Constraints

In physical systems, the measurements are very sparse, which cannot afford the learning of DNN.

To make DNN applicable to such predictive tasks, we must use physical constraints coming from:

Governing equations
Boundary conditions
Expert knowledge

$$\mathcal{L}(y, x) = 0$$

$$\frac{\partial y}{\partial x}(x \in \Omega_N); y(x \in \Omega_D)$$

$$y \geq y_{min}$$

PDE related!

Can DNN learn from these PDE value?

Physics-Informed Neural Network

Physical Constraints

In physical systems, the measurements are very sparse, which cannot afford the learning of DNN.

To make DNN applicable to such predictive tasks, we must use physical constraints coming from:

Governing equations
Boundary conditions
Expert knowledge

$$\mathcal{L}(y, x) = 0$$

$$\frac{\partial y}{\partial x}(x \in \Omega_N);\, y(x \in \Omega_D)$$

$$y \geq y_{min}$$

PDE related!

Can DNN learn from these PDE value?

Yes!

Georgia Tech

Physics-Informed Neural Network

Physical Constraints

In physical systems, the measurements are very sparse, which cannot afford the learning of DNN.

To make DNN applicable to such predictive tasks, we must use physical constraints coming from:

Governing equations
Boundary conditions
Expert knowledge

$$\mathcal{L}(y, x) = 0$$

$$\frac{\partial y}{\partial x}(x \in \Omega_N); y(x \in \Omega_D)$$

$$y \geq y_{min}$$

PDE related!

Can DNN learn from these PDE value?

Yes!

How?

Physics-Informed Neural Network

Physical Constraints

In physical systems, the measurements are very sparse, which cannot afford the learning of DNN.

To make DNN applicable to such predictive tasks, we must use physical constraints coming from:

Governing equations
Boundary conditions
Expert knowledge

$$\mathcal{L}(y, x) = 0$$

$$\frac{\partial y}{\partial x}(x \in \Omega_N); y(x \in \Omega_D)$$

$$y \geq y_{min}$$

PDE related!

Can DNN learn from these PDE value? Yes!

How? Use automatic differentiation

Physics-Informed Neural Network

Backpropagation

To learn coefficients minimizing the loss, we need to compute the gradients:

$$\boldsymbol{\theta}_i^{k+1} = \boldsymbol{\theta}_i^k + \eta \frac{\partial \boldsymbol{l}}{\partial \boldsymbol{\theta}_i^k}$$ (Update coefficients at *i*-th layer in DNN)

Gradient w.r.t. $\boldsymbol{\theta}_i$ can be backpropagated from loss function:

$$l = g(\boldsymbol{y}, \hat{\boldsymbol{y}})$$

Considering:

$$\boldsymbol{y} = f_n(f_{n-1}(\dots f_1(f_0(\boldsymbol{x}))) = F(\boldsymbol{x}; \boldsymbol{\theta})$$

Apply chain rule on gradient computation:

$$\begin{aligned}\frac{\partial l}{\partial \boldsymbol{\theta_i}} &= \frac{\partial l}{\partial \boldsymbol{y}} \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{\theta_i}} \\ &= \frac{\partial l}{\partial \boldsymbol{y}} \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{h}_n} \frac{\partial \boldsymbol{h}_n}{\partial \boldsymbol{h}_{n-1}} \dots \frac{\partial \boldsymbol{h}_i}{\partial \boldsymbol{\theta_i}} \\ &= g' f_n' f_{n-1}' \dots f_i' \\ &= g' F_{\boldsymbol{\theta}_i}(\boldsymbol{x}; \boldsymbol{\theta})\end{aligned}$$

Automatic Differentiation (AD)

Physics-Informed Neural Network

Approximation of partial derivatives

Leverage AD from output to input variables:

$$\boldsymbol{y} = f_n(f_{n-1}(\dots f_1(f_0(\boldsymbol{x})))) = F(\boldsymbol{x};\boldsymbol{\theta})$$

$$\begin{aligned}\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}} &= \frac{\partial F(\boldsymbol{x};\boldsymbol{\theta})}{\partial \boldsymbol{x}} \\ &= \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{h}_n}\frac{\partial \boldsymbol{h}_n}{\partial \boldsymbol{h}_{n-1}}\dots\frac{\partial \boldsymbol{h}_1}{\partial \boldsymbol{x}} \Leftarrow \text{chain rule} \\ &= f_n' f_{n-1}' \cdots f_1' \\ &= F_x(\boldsymbol{x};\boldsymbol{\theta})\end{aligned}$$

$F_x(\boldsymbol{x};\boldsymbol{\theta})$ can be used to approximate the first order partial derivitives.

For second order derivitives, we simply do it again: $\frac{\partial^2 \boldsymbol{y}}{\partial \boldsymbol{x}^2} = \frac{\partial F_x(\boldsymbol{x};\boldsymbol{\theta})}{\partial \boldsymbol{x}} = F_{xx}(\boldsymbol{x};\boldsymbol{\theta})$

F, F_x and F_{xx} share the same set of coefficients $\boldsymbol{\theta}$

Georgia Tech

Groundwater PINN

Forward Model PINN

Design a neural network NN with spatial coordinates (x, y) as input and water heads under pumping test (h) as output

Physical constraints

$$\nabla \cdot [T(x_e, y_e)\nabla h(x_e, y_e)] = 0, \qquad (x_e, y_e) \in \Omega$$

$$\nabla \cdot \left[T(x_p, y_p)\nabla h(x_p, y_p)\right] = Q_p, \qquad (x_p, y_p) \in \Omega$$

$$\mathbf{n} \cdot \nabla h(x_N, y_N) = q_N, \qquad (x_N, y_N) \in \Gamma_N$$

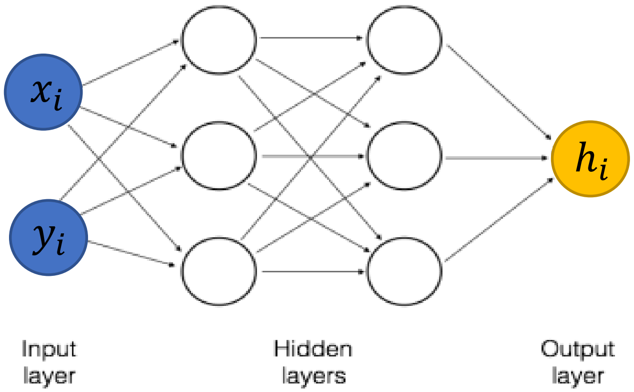
$$h(x_D, y_D) = h_D, \qquad (x_D, y_D) \in \Gamma_D$$

$$\nabla \cdot [T(x_e, y_e)\nabla NN(x_e, y_e)] = 0, \qquad (x_e, y_e) \in \Omega$$

$$\nabla \cdot \left[T(x_p, y_p)\nabla NN(x_p, y_p)\right] = Q_p, \qquad (x_p, y_p) \in \Omega$$

$$\mathbf{n} \cdot \nabla NN(x_N, y_N) = q_N, \qquad (x_N, y_N) \in \Gamma_N$$

$$NN(x_D, y_D) = h_D, \qquad (x_D, y_D) \in \Gamma_D$$



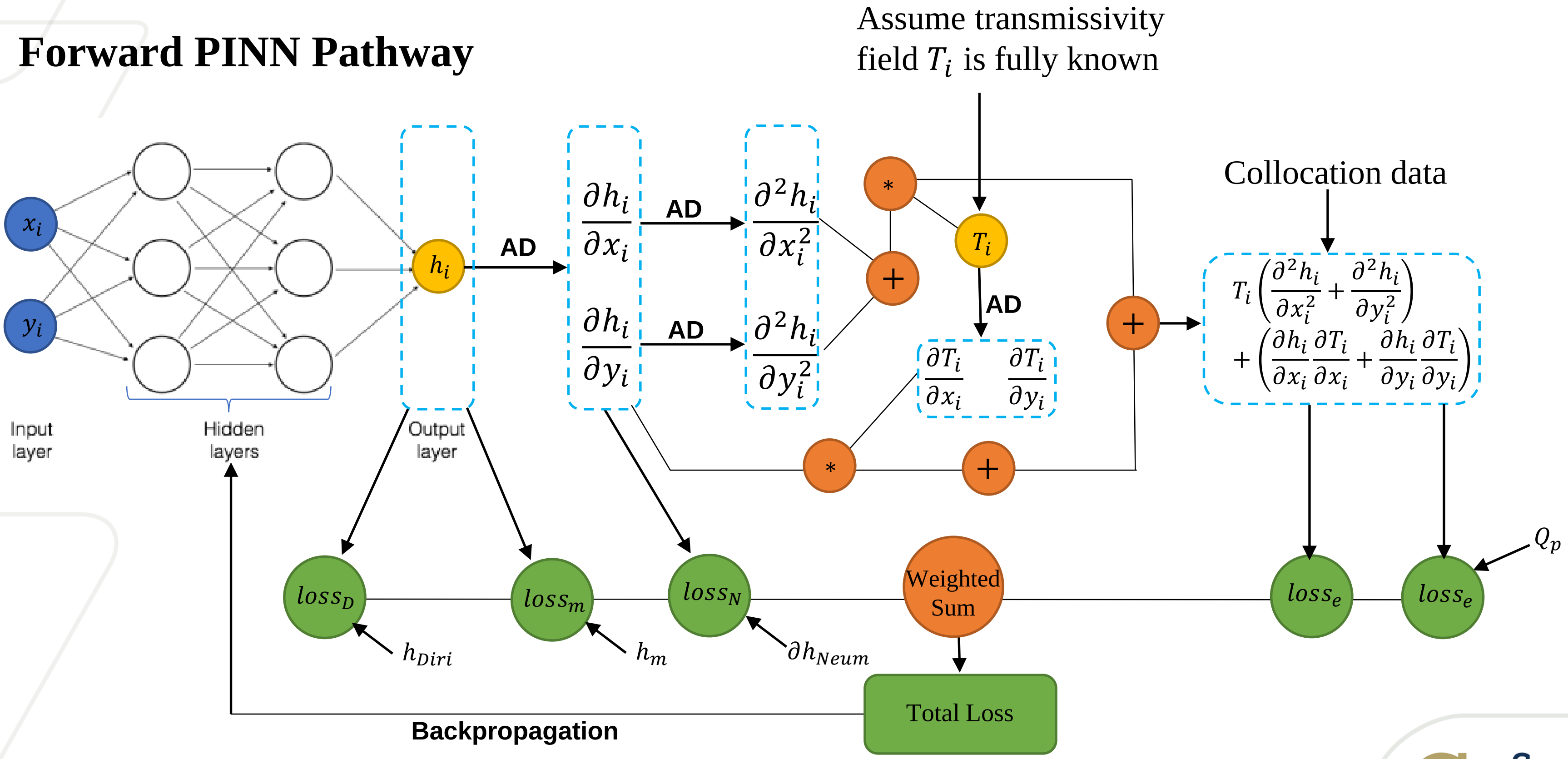
Besides, we have some monitored water heads:

$$NN(x_m, y_m) = h_m, \qquad (x_m, y_m) \in \Omega$$

Data Match

Groundwater PINN

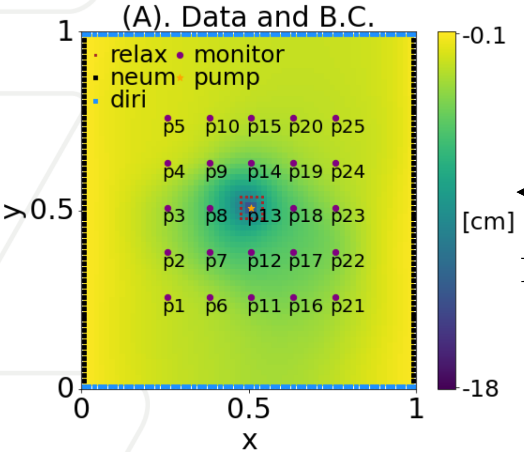
Forward PINN Pathway



Groundwater PINN

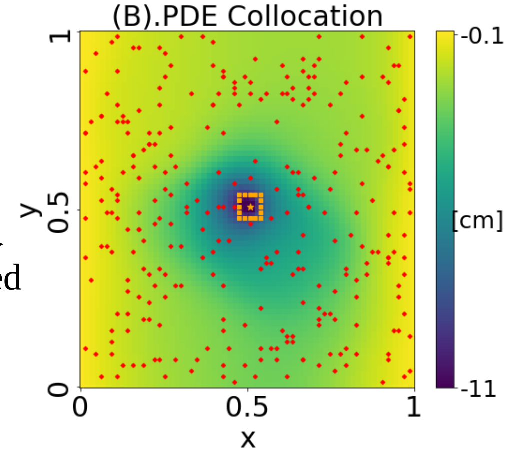
Train Forward PINN

NN is a pointwise, continuous function $F(x, y; \boldsymbol{\theta})$. To train it, we need collect points with interests from the map.



Determined Data

Randomly Selected



Type of points	Number
Pumping (x_p, y_p)	1
Neumann (x_N, y_N)	64 ×2
Dirichlet (x_D, y_D)	64 ×2
Monitored (x_m, y_m)	24
PDE (x_e, y_e)	300

$$\nabla \cdot [T(x_p, y_p)\nabla NN(x_p, y_p)] = Q_p, \quad (x_p, y_p) \in \Omega$$

$$\mathbf{n} \cdot \nabla NN(x_N, y_N) = q_N, \quad (x_N, y_N) \in \Gamma_N$$

$$NN(x_D, y_D) = h_D, \quad (x_D, y_D) \in \Gamma_D$$

$$NN(x_m, y_m) = h_m, \quad (x_m, y_m) \in \Omega$$

$$\nabla \cdot [T(x_e, y_e)\nabla NN(x_e, y_e)] = 0, (x_e, y_e) \in \Omega$$

Not every point is used for PDE constraint!
In each training iteration, we use 300 points as PDE batch for (x_e, y_e).

Georgia Tech

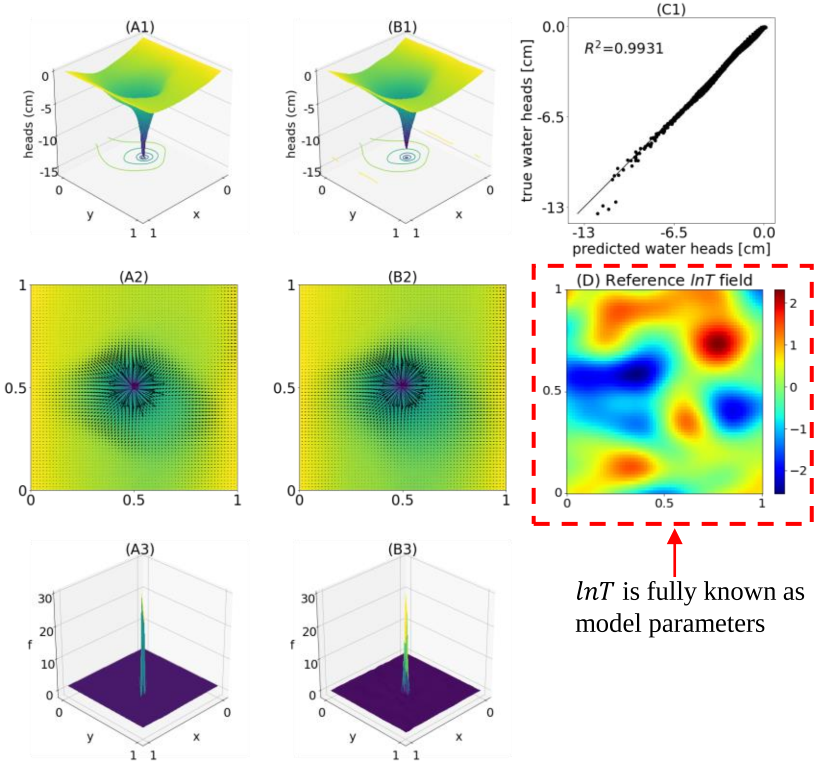
Groundwater PINN

Forward PINN Experiment

Table 1. Geostatistical and hydrogeological parameters for hydraulic tomography experiments

Parameter	Units	Value
Domain size, $L_x \times L_y$	$m \times m$	160 x 160
Grid spacing, $\Delta x \times \Delta y$	$m \times m$	2.5 x 2.5
Spatial resolution, $n_x \times n_y$		64 x 64
Aquifer thickness, b	m	1
Mean log hydraulic conductivity, $E[lnT]$	m/hr	0.0
Correlation length, $\lambda_x \times \lambda_y$	$m \times m$	24 x 20
Variance, σ^2_{lnK}		1.0
Natural gradient, J	m/m	0.0
Pumping rate, Q_p	m^3/hr	3.6
Top & Bottom boundary conditions	Impermeable ($\frac{\partial h}{\partial y} = 0$)	
Left & Right boundary conditions	Constant ($h = 0$)	

Relative residual: $\epsilon_{NN} = \frac{\|NN(x,y) - h(x,y)\|_2^2}{\|h(x,y)\|_2^2} < 5\%.$

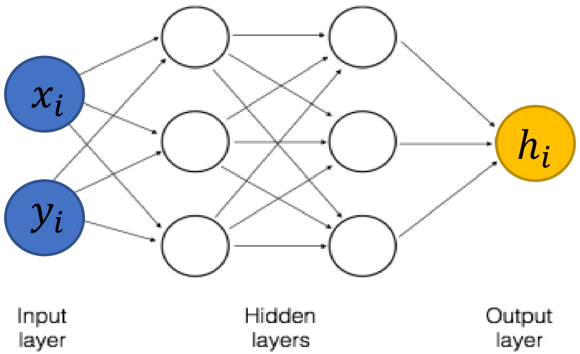


lnT is fully known as model parameters

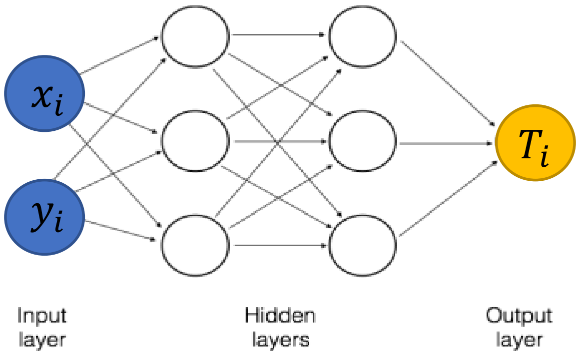
Figure 1. Comparison of NN model with numerical simulation for a pumping test. Column A is numerical simulation results, and column B is NN results. (A1) numerical simulation of hydraulic head distribution, (A2) the gradient field, (A3) numerical evaluated PDE residual; (B1) NN of hydraulic head distribution, (B2) approximated gradient field, (B3) approximated PDE residual; (C1) reference water heads vs. predicted water heads. (D) lnT field.

Groundwater PINN

Inverse Model PINN



NN



TNN

$$\nabla \cdot [TNN(x_e, y_e)\nabla NN(x_e, y_e)] = 0, \qquad (x_e, y_e) \in \Omega$$

$$\nabla \cdot \left[TNN(x_p, y_p)\nabla NN(x_p, y_p)\right] = Q_p, \qquad (x_p, y_p) \in \Omega$$

$$\mathbf{n} \cdot \nabla NN(x_N, y_N) = q_N, \qquad (x_N, y_N) \in \Gamma_N$$

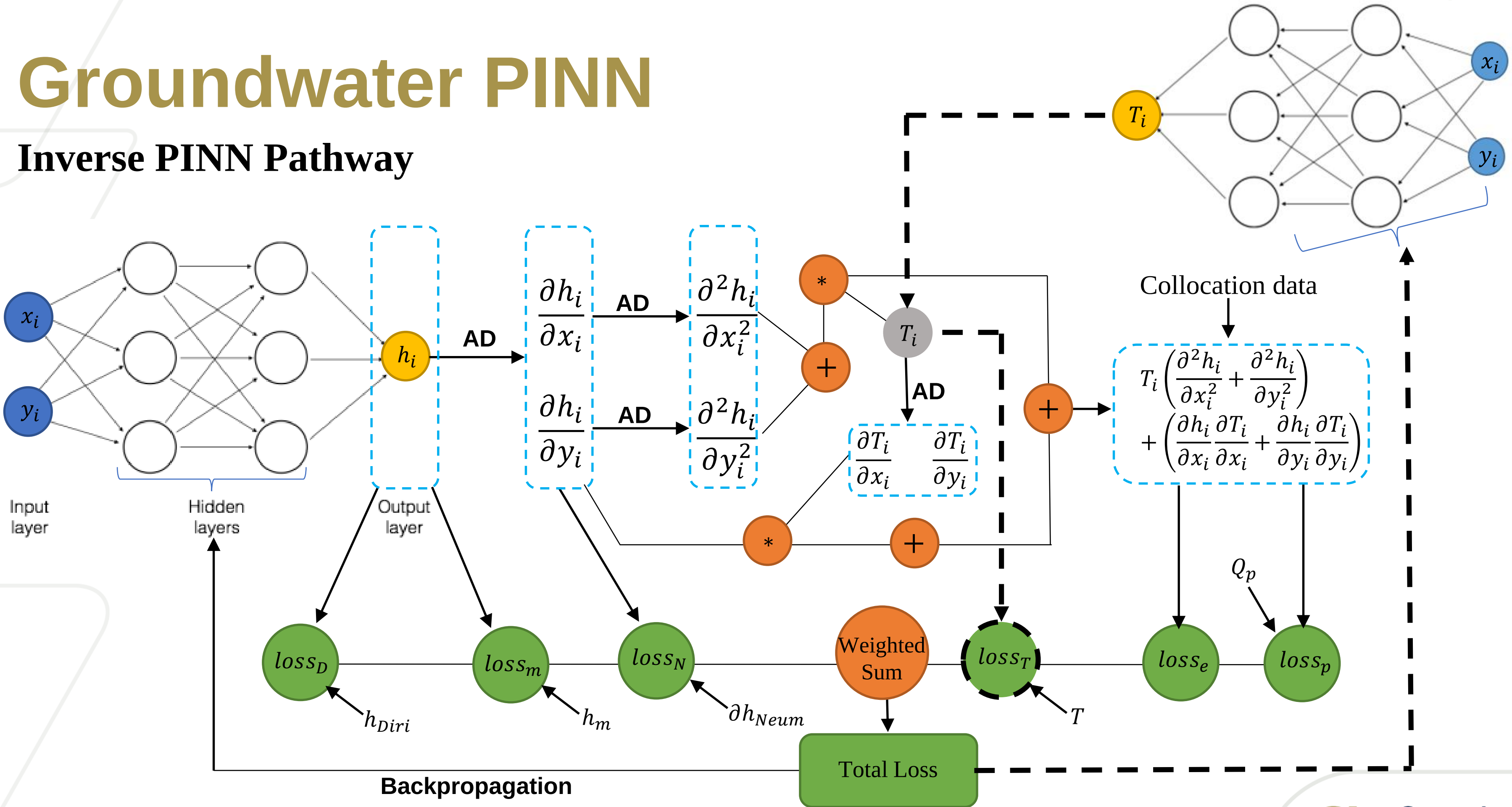
$$NN(x_D, y_D) = h_D, \qquad (x_D, y_D) \in \Gamma_D$$

Monitored water heads: $NN(x_m, y_m) = h_m, \quad (x_m, y_m) \in \Omega$

Measurements of transmissivity: $TNN(x_T, y_T) = T(x_T, y_T)$

Groundwater PINN

Inverse PINN Pathway

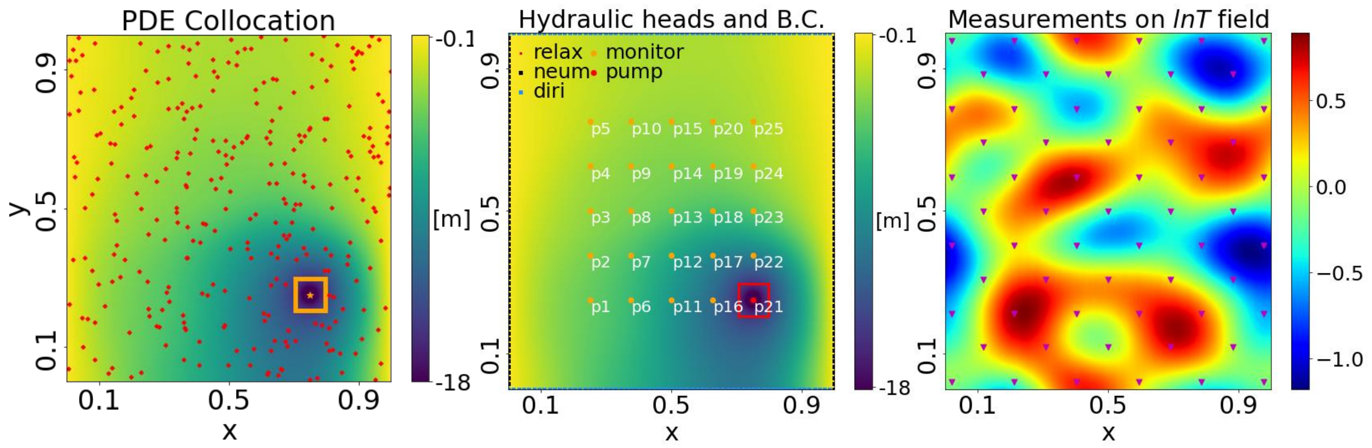


Georgia Tech

Groundwater PINN

Train Inverse PINN

Beside the data for NN, we also need data (direct measurements of transmissivity) for continuous function TNN



Type of points	Number	Type of points	Number
Pumping (x_p, y_p)	1	Direct (x_T, y_T)	61
Neumann (x_N, y_N)	64×2	Monitored (x_m, y_m)	24
Dirichlet (x_D, y_D)	64×2	PDE (x_e, y_e)	300

Resolution: 1024×1024

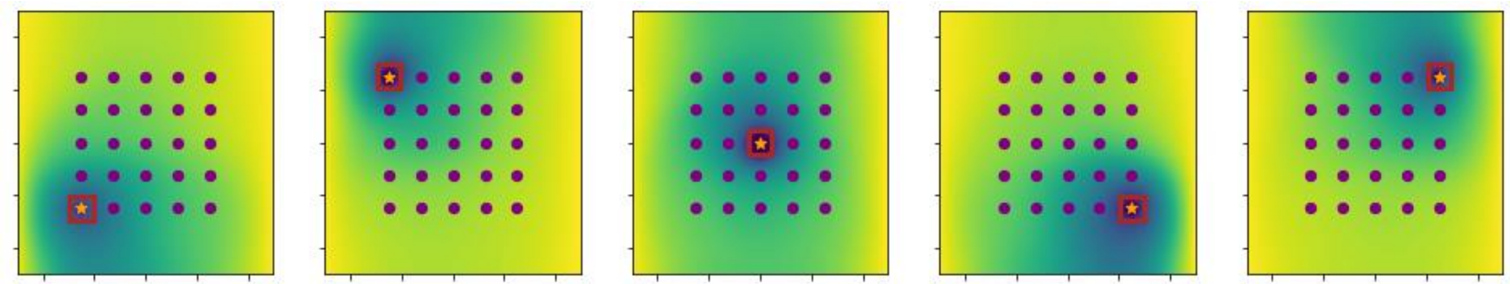
Total data: 642

Groundwater PINN

Hydraulic Tomography

Pumping tests are conducted at different locations $\Rightarrow$

For non-pumping grid:	$\nabla \cdot [T(x_e, y_e)\nabla h(x_e, y_e)] = 0, \quad (x_e, y_e) \in \Omega$
For pumping grid:	$\nabla \cdot [T(x_p, y_p)\nabla h(x_p, y_p)] = Q_p, \quad (x_p, y_p) \in \Omega$
Neumann boundary:	$\mathbf{n} \cdot \nabla h(x_N, y_N) = q_N, \quad (x_N, y_N) \in \Gamma_N$
Dirichlet boundary:	$h(x_D, y_D) = h_D, \quad (x_D, y_D) \in \Gamma_D$



P1 P5 P13 P21 P25

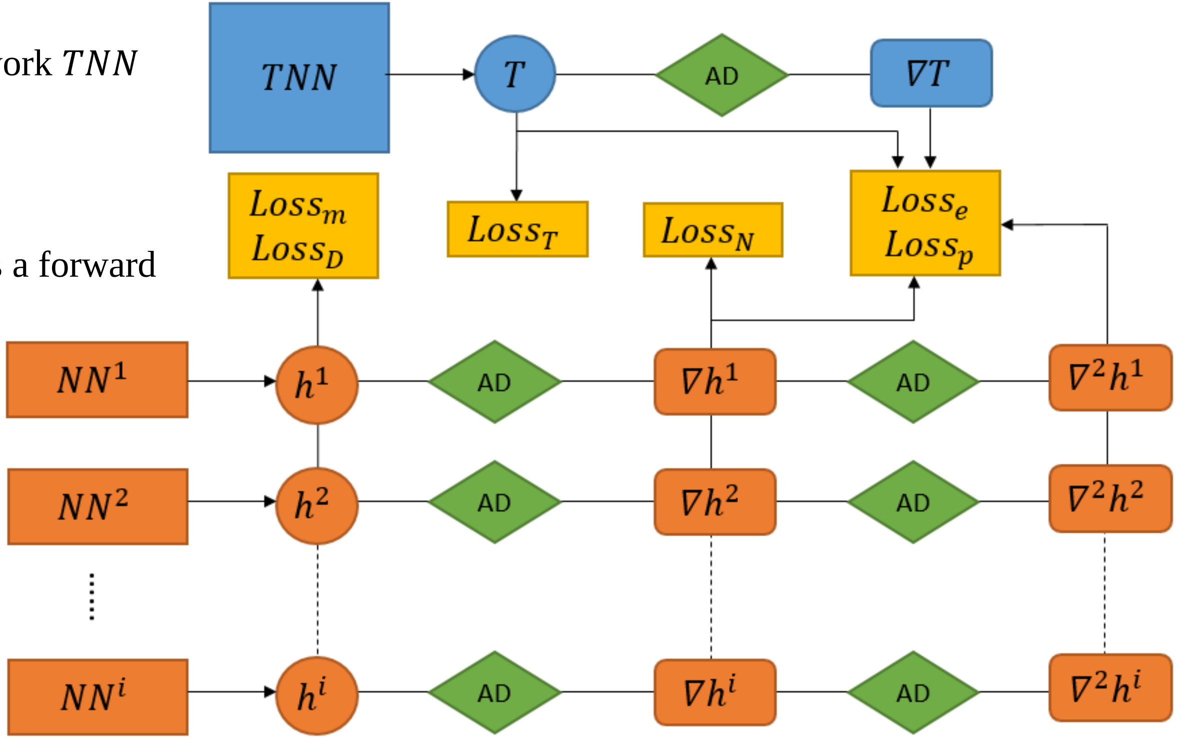
Georgia Tech

Groundwater PINN

Hydraulic Tomography-PINN

Only one inverse network TNN

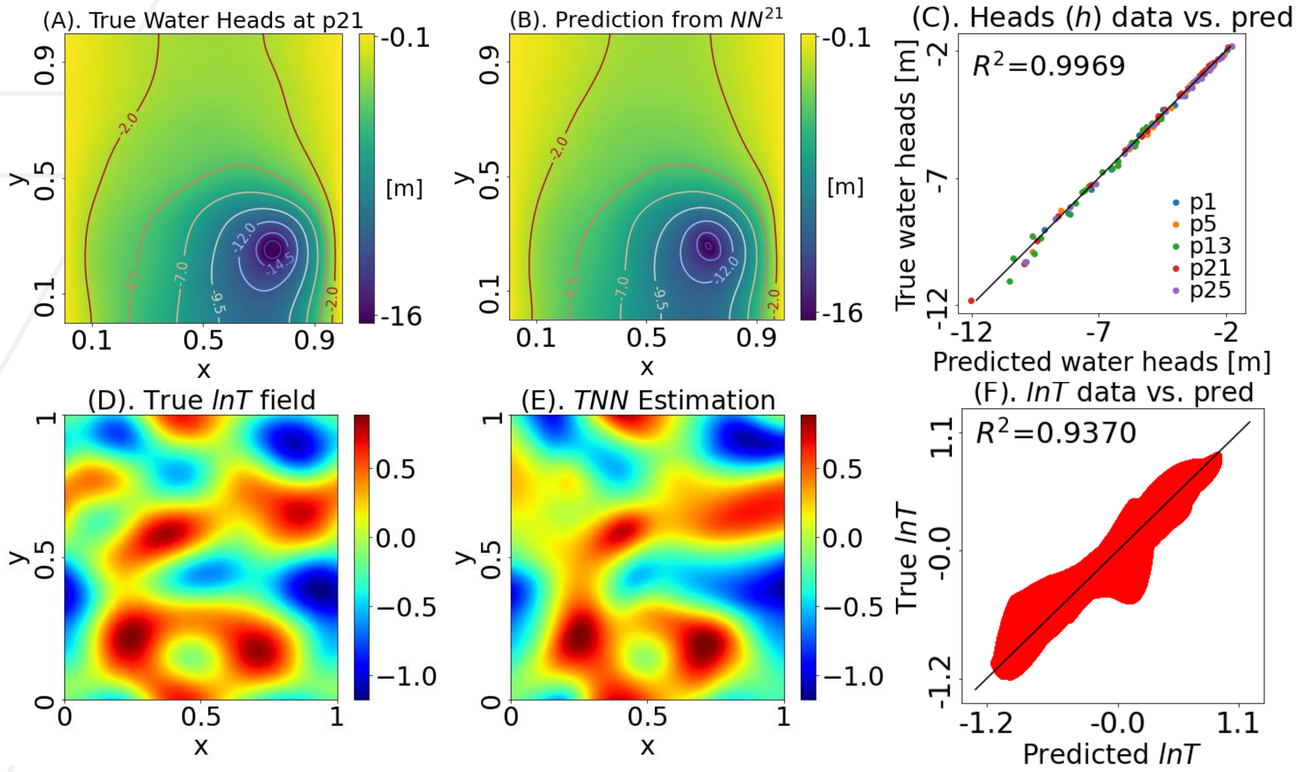
Each pumping test has a forward network



Groundwater PINN

$$\varepsilon(x,y) = \frac{|TNN(x,y) - T(x,y)|}{T^{max} - T^{min}}, (x,y) \in \Omega$$

Hydraulic Tomography-PINN



Forward Performance

Relative residual ϵ_{NN^i} at

P1: 6.00%,

P5: 9.37%,

P13: 6.57%,

P21: 7.13%,

P25: 8.40%

Inverse Performance

Relative residual

ϵ_{TNN}=9.09%

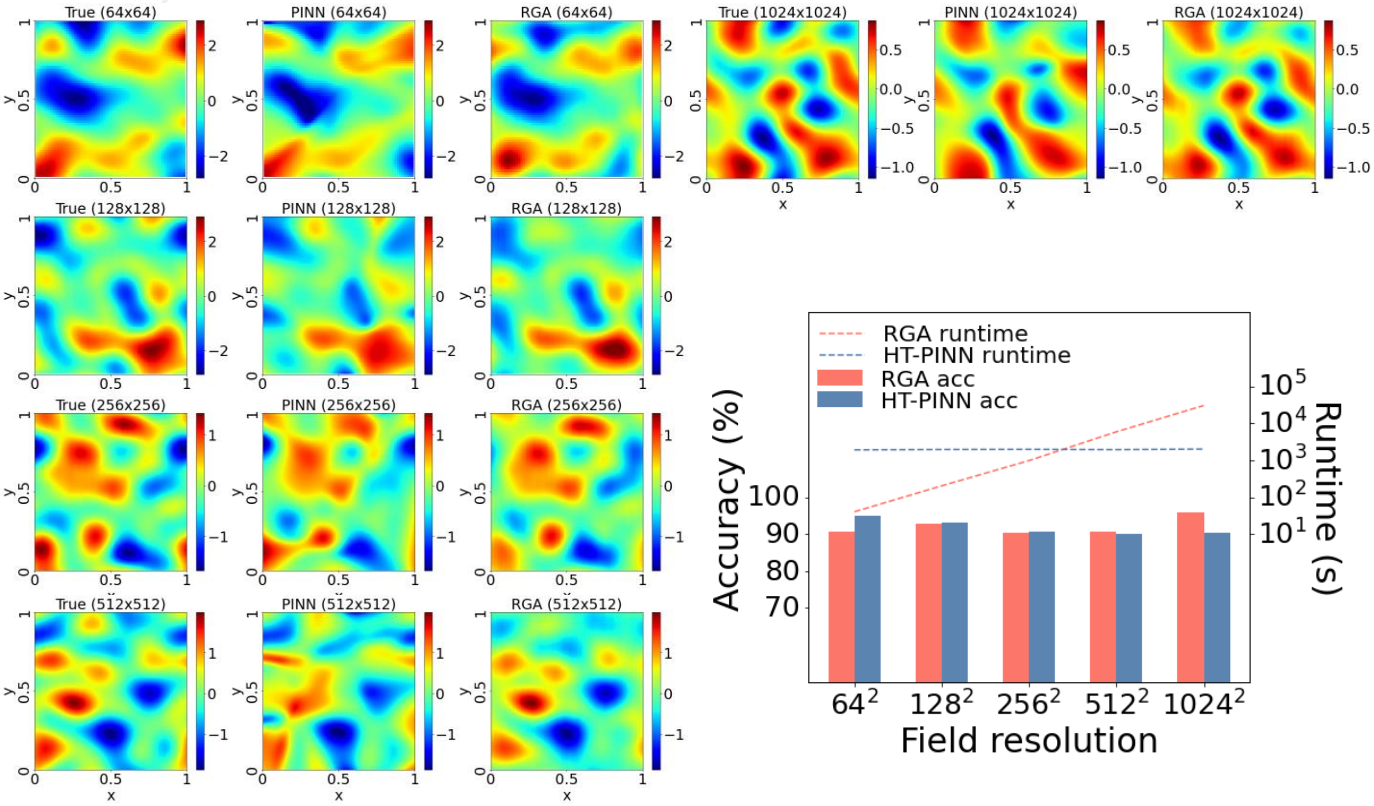
Accuracy

ε =96.85%

Georgia Tech

Groundwater PINN

Scalability of HT-PINN



Model	RGA	HT-PINN
Accuracy	> 90%	> 90%
N_h	24×5	24×5
N_{lnT}	0	61
Covariance	Yes	No
Scalability	Linear	Constant

Groundwater PINN

Model Discussion

	HT-PINN	GA Inverse model
Type	Lagre regression model	Optimize Bayesian posteriori
Regularization	Physical constraints (PDE)	Geostatistical assumption (covariance)
Pros	Easy to get convergence Scalable (pointwise computation)	Not data demanding Theory-guided (robust and interpretive)
Cons	Demands direct measurements Data fitting and lack interpretation	Need iteration Matrix-wise computation

Future work

Model Extension

1. Modify current HT-PINN to 3D (x, y, z), trainsient model (x, y, z, t).
2. Extend PINN to other type of groundwater inverse problem such as: tracer concentration test
3. Add geostatistical constraints to HT-PINN, hopefully, the data demands can be reduced
4. Upgrade DNN to convolution neural network (CNN), enhance model efficiency and generality

Q & A

Many Thanks!

Appreciate any questions

Reference

Goodfellow, I., Bengio, Y., & Courville, Aaron. (2016), Deep Learning, MIT Press.

Guo, Q. and Luo, J. (2021), High-dimensional inverse modeling of hydraulic tomography by physics informed neural network (HT-PINN) with batch training technique. [under review]

Kitanidis, P., & Lee, J. (2014), Principal Component Geostatistical Approach for large-dimensional inverse problems, Water Resources Research, 50, 5428 - 5443.

Kitanidis, P. K. (1995), Quasi-Linear Geostatistical Theory for Inversing, Water Resources Research, 31(10), 2411-2419.

Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2019), Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, *Journal of Computational Physics*, *378*, 686-707

Tartakovsky, A. M., Marrero, C. O., Perdikaris, P., Tartakovsky, G. D., & Barajas-Solano, D. (2020), Physics-Informed Deep Neural Networks for Learning Parameters and Constitutive Relationships in Subsurface Flow Problems, *Water Resources Research*, *56*(5), e2019WR026731, doi: https://doi.org/10.1029/2019WR026731.

Tiedeman, C. R., & Barrash, W. (2019). Hydraulic tomography: 3D hydraulic conductivity, fracture network, and connectivity in mudstone. *Groundwater*, doi: https://doi.org/10.1111/gwat.12915

Wang, N., Chang, H., & Zhang, D. (2021a), Deep-Learning-Based Inverse Modeling Approaches: A Subsurface Flow Example, *Journal of Geophysical Research: Solid Earth*, *126*(2), e2020JB020549.

Yeh, T., & Liu, S. (2000), Hydraulic tomography: Development of a new aquifer test method, *Water Resources Research*, *36*, 2095-2105.

Zhao, Y., & Luo, J. (2020), Reformulation of Bayesian Geostatistical Approach on Principal Components, *Water Resources Research*, *56*.

Zhao, Y., & Luo, J. (2021a), A Quasi-Newton Reformulated Geostatistical Approach on Reduced Dimensions for Large-Dimensional Inverse Problems, *Water Resources Research*, *57*(1), e2020WR028399.

Zhao, Y., & Luo, J. (2021b), Bayesian inverse modeling of large-scale spatial fields on iteratively corrected principal components, *Advances in Water Resources*, *151*, 103913.

Georgia Tech