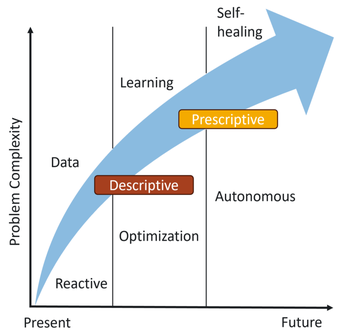
Physics-Informed Neural Networks for Distribution System State Estimation

D. Glover, R. Sadnan, A. Reiman

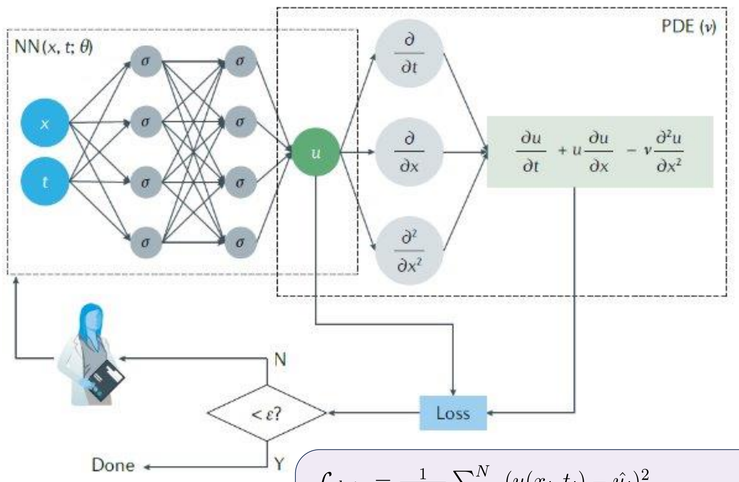
1. Motivation for Deep Learning in DSSE

- Modern distribution networks growing due to aggressive DER integration.
- Challenges introduced to utilities conducting system health analysis due to various uncertainties – weather forecast models, load dynamics, topology shifts.
- WAM infrastructure, microPMUs, smart meters providing lots of data!
- Deep Learning (DL) emerging to solve complex problems in power systems, but…
  I. Neural Networks often viewed as 'black-box' solutions, loss of interpretability.
  II. Learning models may lack feasible solution space.
  III. Must respect the underlying physical equations governing power system.



2. PINN Fundamentals

Algorithm 1: The PINN algorithm.
Construct a neural network (NN) $u(x,t;\theta)$ with θ the set of trainable weights w and biases b, and σ denotes a nonlinear activation function. Specify the measurement data $\{x_i, t_i, u_i\}$ for u and the residual points $\{x_j, t_j\}$ for the PDE. Specify the loss $\mathcal{L}$ in Eq. (3) by summing the weighted losses of the data and PDE. Train the NN to find the best parameters θ^* by minimizing the loss $\mathcal{L}$.



Loss Functions

$$\mathcal{L}_{data} = \frac{1}{N_{data}} \sum_{i=1}^{N} (u(x_i, t_i) - \hat{u}_i)^2$$

$$\mathcal{L}_{pde} = \frac{1}{N_{pde}} \sum_{j=1}^{N} \left(\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} - v\frac{\partial^2 u}{\partial x^2}\right)^2 | (x_j, t_j)$$

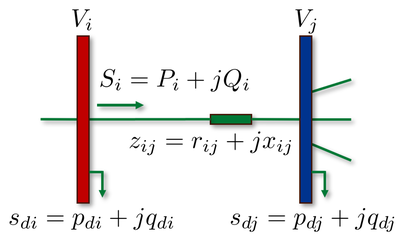
$$\mathcal{L}_{PINN} = w_{data}\mathcal{L}_{data} + w_{pde}\mathcal{L}_{pde} \qquad (3)$$

Method & Drawbacks

- Surrogate model (DNN) and PDE residual for a PINN[1].
- Automatic differentiation on output for partial derivatives.
- Total loss function combines MSE metrics for both networks.
- Optimized via gradient descent on loss terms for parameter adjustments.
- Requires significant hyperparameter tuning and weight balancing to train.
- Additional constraints introduce more loss function terms: multi-objective.
- No feasible solution space guarantee.

3. Linearized Distribution Power Flow

- Traditional *Branch Flow* Model
- Neglect nonlinear terms for improved linear approximations
- Derive Linear Distribution Flow Model – **LEARN THIS!!**
- Can we train a PINN to approximate power flow while *learning the LinDistFlow model*?



Preliminaries:

i = node (parent)

j = node (child)

(i,j) = edge (line)(i,j)

$3\phi = \rho_i, \phi_i \in \{a,b,c\}$

$S_{ij}^{pq} = P_{ij}^{pq} + Q_{ij}^{pq}, \quad pq \in \phi_{ij}$

$z_{ij}^{pq} = r_{ij}^{pq} + jx_{ij}^{pq}, \quad pq \in \phi_{ij}$

$$P_{ij,t}^{\rho\rho} = \sum_{k:j\to k} P_{jk,t}^{\rho\rho} + p_{Load,j,t} - p_{DER,j,t} \qquad \forall \rho \in \{a,b,c\} \qquad (1)$$

$$Q_{ij,t}^{\rho\rho} = \sum_{k:j\to k} Q_{jk,t}^{\rho\rho} + q_{Load,j,t} - q_{DER,j,t} \qquad \forall \rho \in \{a,b,c\} \qquad (2)$$

$$v_{i,t}^{\rho} - v_{j,t}^{\rho} = \sum_{q\in\phi_j} 2\mathbb{R}[S_{ij,t}^{pq}(z_{ij}^{pq})^*] \qquad \forall p,q \in \{a,b,c\} \qquad (3)$$

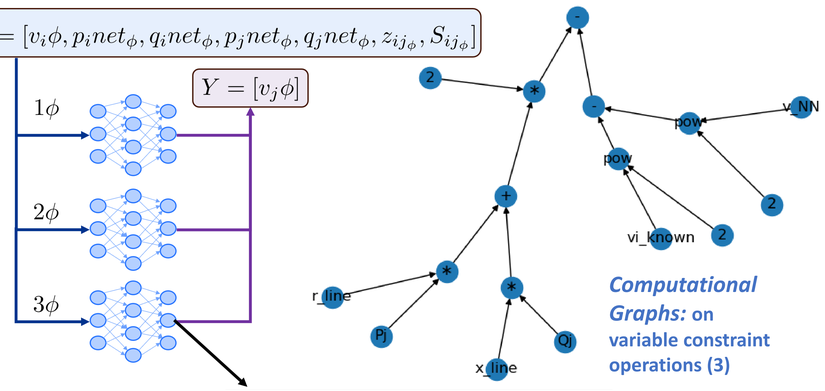
$$v_{min}^2 \le v_{i,t}^{\rho} \le v_{max}^2 \qquad (4)$$

4. NeuroMANCER PINN Design & Simulation Study

- Open-source DP SciML library[2]
- Parametric constrained opt tasks
- PyTorch framework (wrapper)
- Build torch.NN structure
- Create Neuro-map from X:Y
- Define vars, obj, constraints
- Design problem, optimizer
- Train/Test Model

- IEEE 123bus system w/ 55 DERs
- Variable load dynamics
- QSTS, 5760 steps, 15 min step
- Conduct centralized load flow
- Select random lines, get bus pair info
- Collect LinDistFlow and DSS voltages
- Captures all physical state information in power system from measurements

$$X = [v_i\phi, p_i net_\phi, q_i net_\phi, p_j net_\phi, q_j net_\phi, z_{ij_\phi}, S_{ij_\phi}]$$



NM allows for hard constraints (4) on NN activation functions!

Computational Graphs: **on variable constraint operations (3)**

```
MLP_bounds(insize=11, outsize=1, bias=True,
           linear_map=slim.maps['linear'],
           nonlin=nn.GELU, hsizes=[32, 32],
           min=0.9, max=1.1)
```

***Variables*:** **Symbolic Handlers**

```
# vars from Dataset
x = variable('x')
r_line = x[:, 6]
x_line = x[:, 7]
Pj = x[:, 4]
Qj = x[:, 5]
```

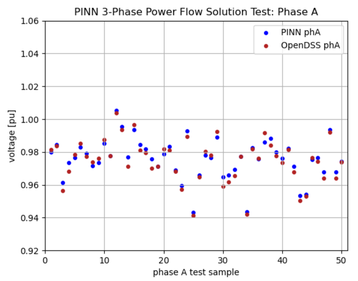
***Objectives*: inst variables or metric (torch.mean)**

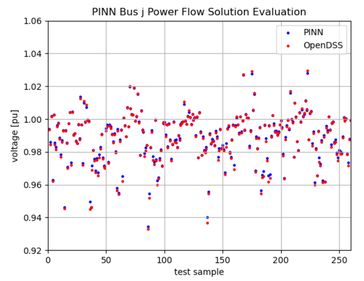
```
def get_objective():
    f_scale = 100  # speed up convergence
    mse_loss = f_scale * (v - v_target)**2
    obj = mse_loss.minimize(weight=1.0, name='obj')
    return obj
```

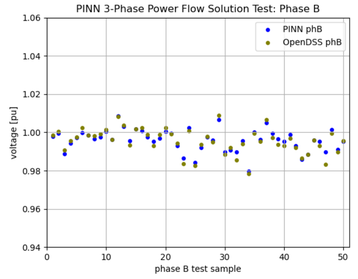
***Constraints*: variables (==, <, >, <=, >=) Returns dict {value, penalty, loss}**

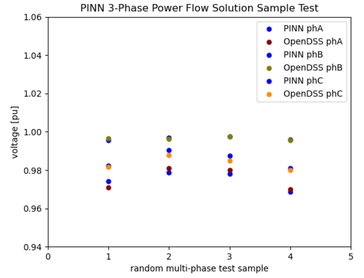
```
# power flow constraints
p_lhs = Pj
p_rhs = Pi - pj_net
p_con = (p_lhs == p_rhs)
p_con.name = 'pflow_con'
```

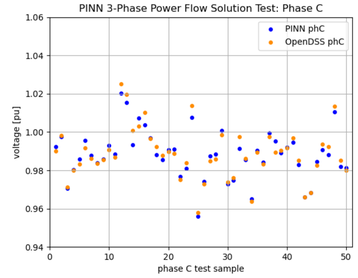
5. PINN Evaluation OpenDSS

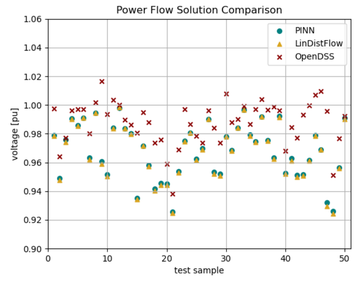












Number Samples	5500
Batch	64
Mean Loss	0.0105
Train Penalty Loss	0.0111
Objective Loss	6.5021e-06
Constraint:	**Violation:** {a, b, c}
LDF Eq 1 Real PFlow	{0.0, 0.0, 0.0}
LDF Eq 2 Reactive PFlow	{0.0, 0.0, 0.0}
LDF Eq 3 Voltage Drop	{1.394e-03, 7.581e-04, 1.814e-03}
V bus2 limit >= 0.95 p.u.	{0.0, 0.0, 0.0}
V bus2 limit <= 1.05 p.u.	{0.0, 0.0, 0.0}

Number Samples	260
Batch	12
Mean Loss	0.0103
Test Penalty Loss	0.008
Objective Loss	3.2732e-06
Constraint:	**Violation:** {a, b, c}
LDF Eq 1 Real PFlow	{0.0, 0.0, 0.0}
LDF Eq 2 Reactive PFlow	{0.0, 0.0, 0.0}
LDF Eq 3 Voltage Drop	{4.404e-03, 2.469e-03, 7.367e-03}
V bus2 limit >= 0.95 p.u.	{0.0, 0.0, 0.0}
V bus2 limit <= 1.05 p.u.	{0.0, 0.0, 0.0}

6. Future Work Direction

- Train DSSE PINN on larger distribution feeders e.g. IEEE 8900 bus.
- Use the constrained PINN framework to solve more rigorous constrained optimization problems: DistOPF, etc.
- Use PINNs to determine unreported PV systems for the utility[3].

7. References

1. Karniadakis, George Em, Ioannis G. Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. "Physics-informed machine learning." Nature Reviews Physics 3, no. 6 (2021): 422-440.
2. Tuor, A., Drgona, J., Skomski, E., Koch, J., Chen, Z., Dernbach, S., ... & Vrabie, D. (2021). NeuroMANCER: Neural modules with adaptive nonlinear constraints and efficient regularizations. *URL https://github. com/pnnl/neuromancer*.
3. R. Sadnan, R. Mayur, D. Glover, A. Reiman, J. Follum. "Parameter Estimation of Unreported PVs using Distribution System State Estimation Algorithm." (in preparation)

For additional information, contact:
Staff Name | (509) 371-6791 | daniel.glover@pnnl.gov | daniel.glover1@wsu.edu

U.S. DEPARTMENT OF ENERGY | BATTELLE
PNNL is operated by Battelle for the U.S. Department of Energy
WASHINGTON STATE UNIVERSITY
8/23/2024
Pacific Northwest NATIONAL LABORATORY
www.pnnl.gov