MINISTÉRIO DA CIÊNCIA E TECNOLOGIA
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

Implementation of the ecRad Radiation Module Using Physics Informed Machine Learning

Doctorate Proposal Exam - Eduardo Furlan Miranda

Supervisors: Dr. Roberto Pinto Souto & Dr. Stephan Stephany

APPLIED COMPUTING POST-GRADUATE PROGRAM (CAP/INPE)

INTRODUCTION
Physics-Informed Machine Learning (PIML)

- Class of methods that integrate ML algorithms with physical constraints and mathematical models

- Applicable to complex problems

- Physics-Informed Neural Network (PINN) was the “initial” class of PIML methods for problems with PDEs - Partial Differential Equations

- PINN data-driven models are derived using law of physics described by general nonlinear PDEs and constraints (ICs and BCs)

INTRODUCTION
Why PIML ?

- Recent approach proposed for identifying and solving dynamical systems

- The literature shows promising speedups when porting a standard module of a numerical meteorological model to PIML

- DNN and GPU can be used to achieve performance

- Availability of frameworks for execution on GPU, such as Nvidia Modulus, Uber Horovod and others

INTRODUCTION
Challenges of using PIML approaches

- Lack of reproducibility in related articles
- **Incomplete/lacking availability of real-world PIML implementations - code, documentation and/or datasets**
- Porting parts/modules of standard numerical code to PIML
- PIML execution using new frameworks like Nvidia Modulus

INTRODUCTION
PIML - some approaches and resources

- Recent PIML approaches for dynamical system modeling and control:
  - Physics-informed learning for system identification
  - Physics-informed learning for control
  - Analysis and verification of PIML models

- Some PIML classes of methods:
  - **Physics-Informed Neural Network (PINN)**
  - Physics-constrained ML
  - Physics-guided ML
  - Physics-encoded ML
  - Deep Operator Networks (DeepONets)

INTRODUCTION
Physics-Informed Neural Network (PINN)

- **PINN class of methods is PIML !!!**

- PINN direct problem (solution of PDEs)

- PINN inverse problem (PDE parameter discovery)

- Suitable for sparse, limited, incomplete, noisy data, irregular domains and complex non-linear patterns

- **Former toy problem** of this thesis already developed (PINN application for the 1D Burgers' Equation, parameter discovery & solution)

INTRODUCTION
PINN - some approaches and resources

- **PINN architecture**
  - MLP: the most common
  - Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Auto-Encoder (AE), etc.
- **PINN variations**
  - Variational hp-VPINN, conservative PINN (CPINN), and physically constrained DNNs (PCNN), etc.
- **PINN frameworks and implementations**
  - Deep Ritz Method (DRM), Deep Ritz Method (DRM), Deep Galerkin Method (DGM), etc.

INTRODUCTION
PINN - some approaches and resources

- Current research explores architectures, activation functions, loss functions, and gradient optimization
- The PINN mainstream is still the PDE direct problem, but the number of works to solve inverse problems has increased
- Soft BCs: PINN model the PDE with unknown IC/BC
- Hard BCs: impose known IC/BC via a customized DNN architecture

INTRODUCTION
Objectives of this thesis - I

- Propose and implement performance improvements using PIML **only in the gas-optical scheme** of the ecRad radiation module of a weather/climate model

- Current trend in Meteorology and Climate models

- The ecRad radiation module is used in an operational model of the European Centre for Medium-Range Weather Forecasts (ECMWF)

- Any radiation module is processing demanding and thus is not executed every timestep or grid point

INTRODUCTION
Objectives of this thesis - II

- Target part of the ecRad radiation module is the **gas-optical scheme** (most processing demanding)
- Former approach proposed by Ukkonen et al. (2024), **already reproduced using a DNN-based stand-alone gas-optical scheme, part of an offline ecRad implementation, which is the current toy problem**
- **GOAL: a improved PIML-based gas-optical scheme with good accuracy and less processing demanding**
- Incremental approach: DNN, PINN, other PIML (eventually even proposing a new PIML approach)

INTRODUCTION
Related state-of-the art work

- **Speedup of 3** by refactoring the solver and replacing the gas optics module with a PINN (Ukkonen et al., 2024, 2023, 2020)

- **Speedup of 7** in the ECMWF Long-wave Radiative Transfer model (Chevallier et al., 2020)

- **Speedup of up to 10^5** of the Longwave Radiation parameterization for the NCAR CAM and NSIPP GCM (Krasnopolsky et al., 2006)

CURRENT TOY PROBLEM
Non-PIML approach reproduced

- DNN-based approach proposed (Ukkonen & Hogan 2024, 2023, 2020, and others), for the ecRad gas-optics scheme
- **In this work,** executed and analyzed using local PC and the LNCC Santos Dumont supercomputer
- Comparison of ecRad module using gas-optics original F90 scheme and new DNN-based scheme
- TensorFlow/Python is used only for DNN training

CURRENT TOY PROBLEM
ecRad gas-optics scheme

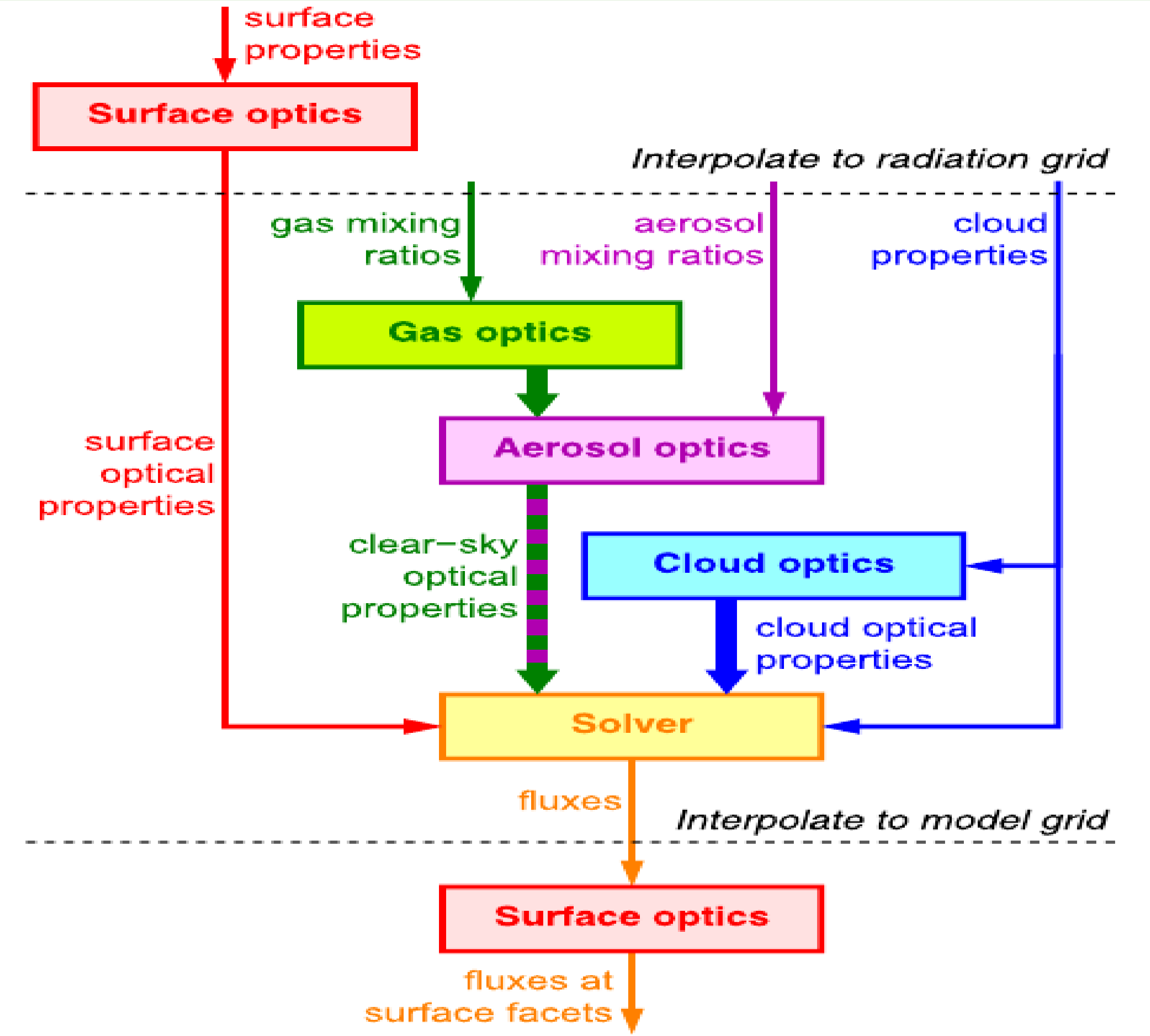
- Atmospheric radiation is the most influential scheme in numerical climate and weather models, but is processing demanding…

- GPROF showed that the gas-optical scheme is the most costly of the radiation module

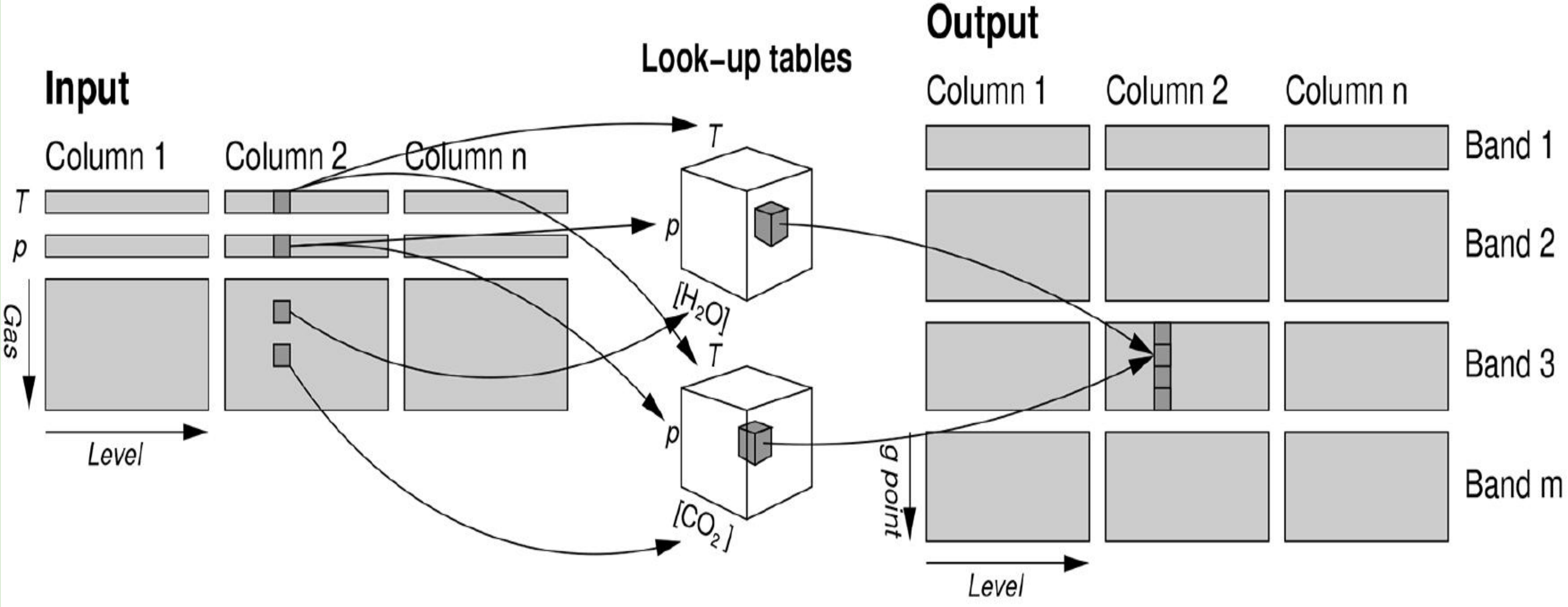
- DNN-based version of ecRad gas-optical scheme replacing the standard F90 RRTMGP implementation, improved processing performance

- Training of the DNN using TensorFlow/Python and porting of the resulting model to F90 ecRad

CURRENT TOY PROBLEM
Esquematic of ecRad module - 5 main schemes

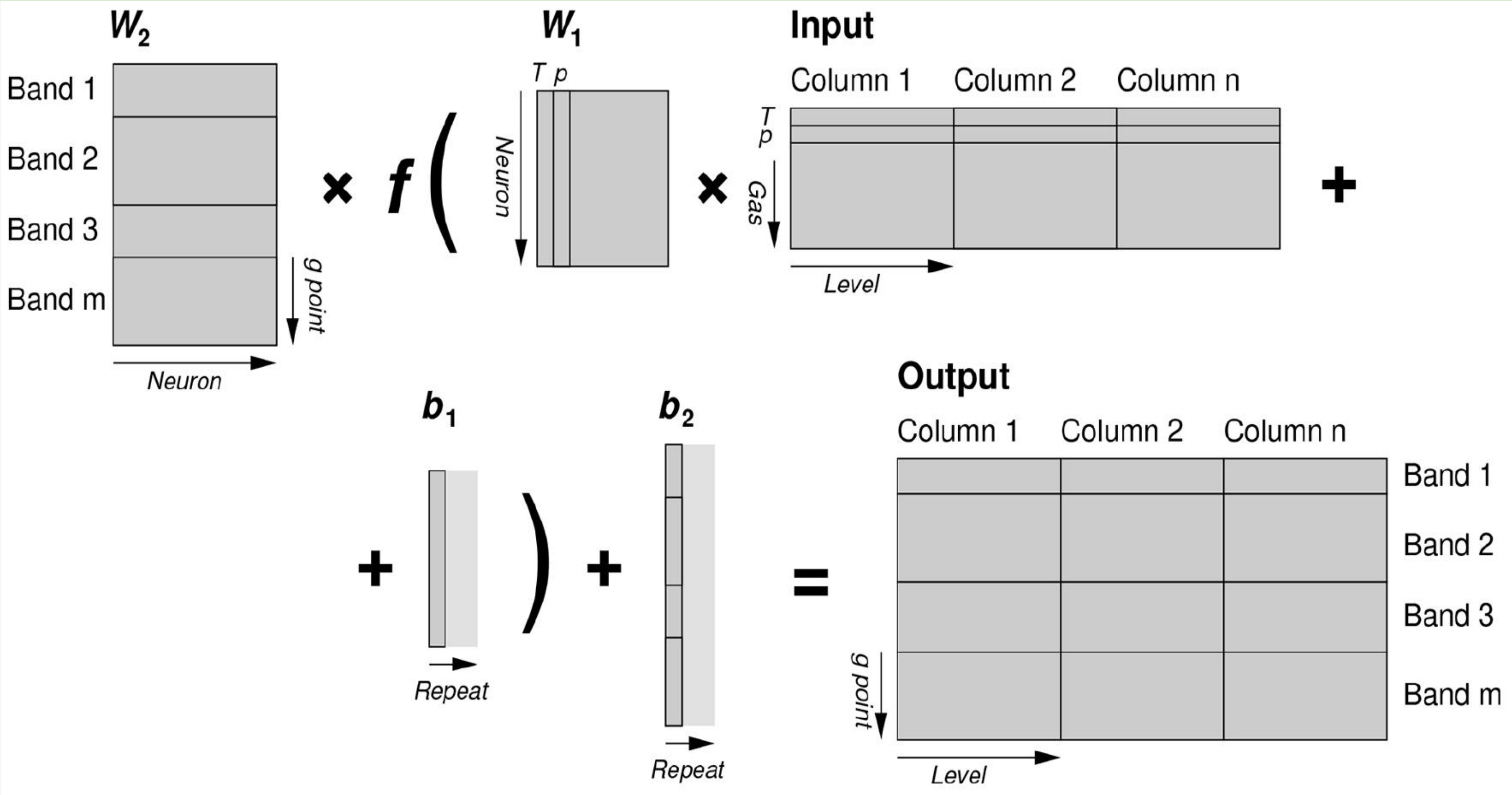


CURRENT TOY PROBLEM
RRTMGP numerical gas optics scheme



T - temperature, p - pressure, Gas - relative abundance, Band - LW or SW, gpoints - correlated k-distribution method, Level - atmospheric layer, Look-up table - table kernels

CURRENT TOY PROBLEM
DNN-based gas-optics scheme



CURRENT TOY PROBLEM
DNN gas-optics scheme

- **MLP architecture**

Predicted	Input	2 Layers	Output
LW absorption	18	58-58	256
LW emission	18	16-16	256
SW absorption	7	48-48	224
SW scattering	7	16-16	224

CURRENT TOY PROBLEM
DNN loss function

$$loss = \alpha \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 + (1-\alpha) \sum_{\substack{i=1 \\ i \text{ odd}}}^{N} ((y_{i+1} - y_i) - (\hat{y}_{i+1} - \hat{y}_i))^2$$

- α is a coefficient representing a trade-off between heating rate and radiative forcing errors
- y is the target vector
- ŷ is the DNN output vector
- N is the size of dataset
- The loss function uses the Mean Squared Error (MSE)
- The training dataset is derived from a variety of sources

FORMER TOY PROBLEM
PINN applied to the 1D Burgers' Equation

- Comparing accuracy of PINN and a numerical method (SINDy) for PDE parameter discovery of the 1D Burgers' Equation
- Evaluating PINN performance for different CP sizes and hyperparameters of this toy problem
- PINN-based neural network training generated a model then used for parameter discovery & solution
- Visual evaluation of the PINN solution using the exact solution as reference

FORMER TOY PROBLEM
1D Burgers' Equation

- **Velocity field u of a fluid (dimension x and time t)**

$$u_t + \lambda_1 u u_x - \lambda_2 u_{xx} = 0 \qquad x \in [-1, 1],\ t \in [0, 1],$$

IC: $$u(0, x) = -\sin(\pi x),$$

BC: $$u(t, -1) = u(t, 1) = 0$$

- **Unknown parameters:** coefficients of the differential operators: $\lambda_1 = 1.0$, $\lambda_2 = \nu = 0.01/\pi$ or $0.1/\pi$ (**high viscosity and low viscosity,** respectively, the latter causes discontinuities)

FORMER TOY PROBLEM
PINN applied to the 1D Burgers' Equation

- MLP architecture: 2-neuron input layer, 1-to-8 hidden layers, hyperbolic tangent activation function, 10-to-30 neurons per hidden layer and a single-neuron output layer

- The loss function uses the Mean Squared Error (MSE), being minimized by the Generalized Limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) optimization algorithm

- Each iteration is one epoch (singe batch per epoch)

FORMER TOY PROBLEM
PINN training input data (set of CPs)

- **CPs - Collocation Points (exact/true points)**

- Problem sizes (1D grid points x timesteps)
  - 128x64, 256x100, 256x128, 512x256
  - Exact solution: Gaussian Quadrature Method (GQM)

- No division of PINN input data into training, validation and test sets (training -> resulting model)
  - Random sample of 2,000 CPs from given dataset
  - PDE parameters discovery (from training)
  - PDE predicted solution (no further training)

FORMER TOY PROBLEM
PINN loss function

- Two-term MSE (iterations k and k-1)

$$MSE^k = MSE_u^k + MSE_f^{k-1}$$

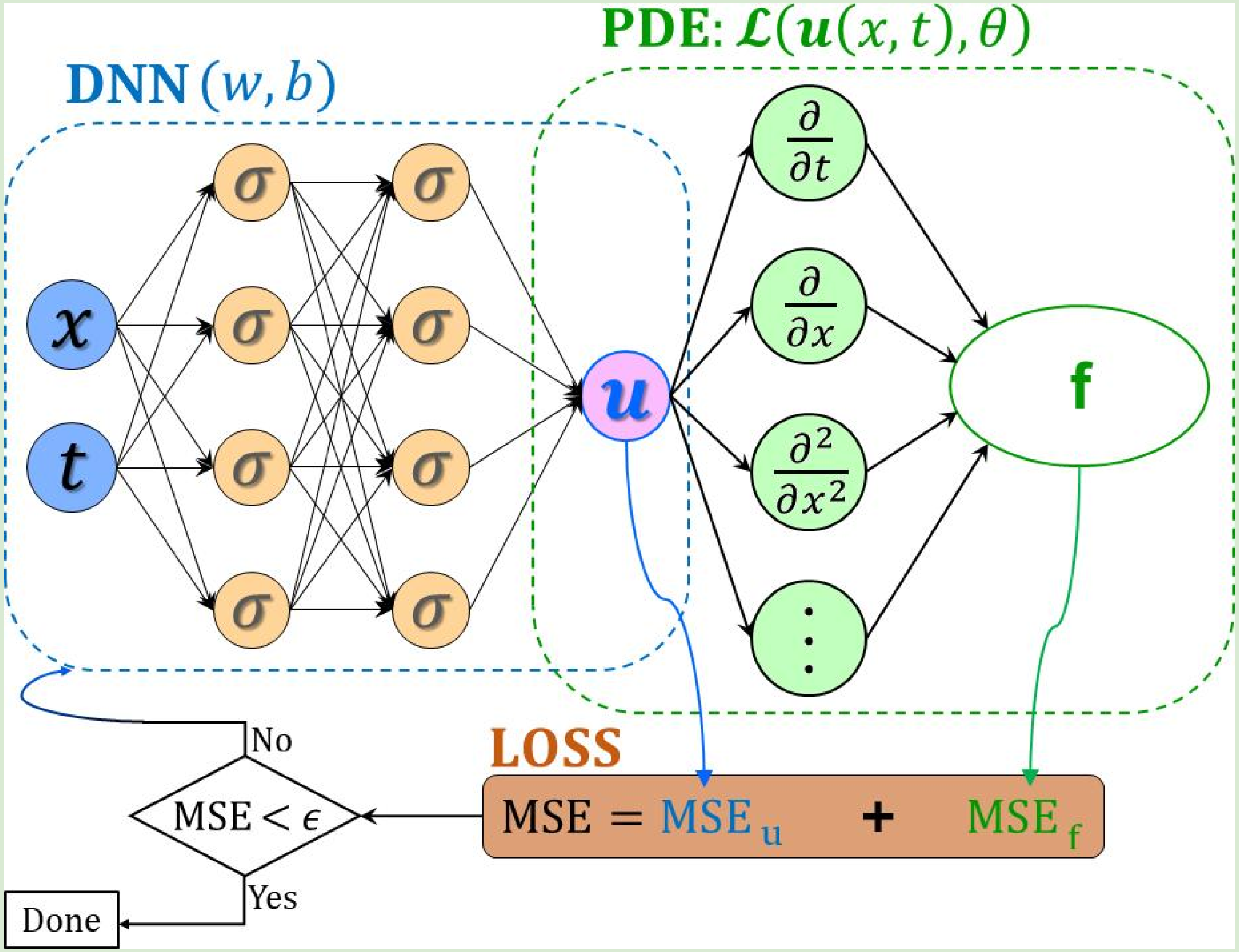
- MSE_u - PINN solution matching of CPs points at iteration k:

$$MSE_u^k = \frac{1}{N}\sum_{i=1}^{N} |u(t_u^i, x_u^i) - u^i|^2$$

- MSE_f - residual of the known PDE for the set of CPs predicted by the PINN at iteration (k-1)

$$MSE_f^{k-1} = \frac{1}{N}\sum_{i=1}^{N} |f(t_u^i, x_u^i)|^2$$

FORMER TOY PROBLEM
PINN loss function



FORMER TOY PROBLEM
Numerical SINDy parameter discovery

- Sparse Identification of Nonlinear Dynamical Systems (SINDy) method (Brunton et al., 2016)

- Uses sparse regression to create a linear combination of basis functions to capture the dynamic behavior of the considered physical system

- Iterative optimization using an objective function

- Applications: linear and nonlinear oscillators, chaotic systems, fluid dynamics, and others

FORMER TOY PROBLEM
Numerical SINDy parameter discovery

- Temporal evolution of x(t) is modeled using the nonlinear function

$$\frac{d}{dt}x(t) = f(x(t))$$

- The vector $x(t)=[x_1(t), x_2(t), \ldots x_n(t)]^\top$ represents the state of the physical system at time t

FORMER TOY PROBLEM
Numerical SINDy parameter discovery

- The problem solved by SINDy is

$$\dot{X} = \Theta\left(X\right)\Xi$$

- "Θ" : matrix f(x(t)) of basis **functions** applied to the input data X, i.e. Θ(X)
- "Ξ" : matrix of **coefficients** that indicates which terms in Θ(X) are significant
  - Reconstructs the governing equations of the dynamical system
- Along the iterations, these **coefficients** are optimized until achieving convergence

FORMER TOY PROBLEM
Numerical SINDy parameter discovery

Includes 4 different Sparse Regression Optimizers

- Sequentially Threshold Least Squares (STLSQ)
- Orthogonal Least Squares of Forward Regression (FROLS)
- Sparse Relaxed Regularized Regression (SR3)
- Sparse Stepwise Regression (SSR)

http://pysindy.readthedocs.io

FORMER AND CURRENT TOY PROBLEMS
Computing environment

- PC local machine 1: CPU 6-core Intel i7 9750h, 8 GB RAM, GPU Nvidia GTX 1050, 3 GB VRAM
- PC local machine 2: CPU 2-core Intel i7 7500U, 16 GB RAM, GPU Nvidia Geforce 940MX, 4 GB VRAM
- LNCC SDumont single Bull Sequana X1120, CPU 2x 24-core Intel Xeon Gold 6252 Skylake, 384 GB RAM, GPU 4x Nvidia Volta V100, 32 GB VRAM
- Python 3.7, TensorFlow 1.15 and 2.16, PySINDy 1.7.5

RESULTS - CURRENT TOY PROBLEM RRTMGP numerical gas-optics (GPROF)

(local PC machine)

% time	cumulative seconds	self seconds	calls	self ms/call	total ms/call	routine
17.42	0.27	0.27	12	22.50	34.06	CloudsSW
12.90	0.47	0.20	12	16.67	49.17	GasOptics
12.90	0.67	0.20	12	16.67	30.10	CloudsLW
9.68	0.82	0.15	11	13.64	13.64	Aerosol
7.74	0.94	0.12	4817	0.02	0.02	TransSW

routine	name
CloudsSW	___radiation_tripleclouds_sw_MOD_solver_tripleclouds_sw
GasOptics	___radiation_ifs_rrtm_MOD_gas_optics
CloudsLW	___radiation_tripleclouds_lw_MOD_solver_tripleclouds_lw
Aerosol	___radiation_aerosol_optics_MOD_add_aerosol_optics
TransSW	___radiation_two_stream_MOD_calc_ref_trans_sw

RESULTS - CURRENT TOY PROBLEM DNN-based gas-optics scheme (GPROF)

(local PC machine)

% time	cumulative seconds	self seconds	calls	self ms/call	total ms/call	routine
16.67	0.25	0.25	12	20.83	50.83	GasOptics
16.67	0.50	0.25	12	20.83	40.00	CloudsSW
14.00	0.71	0.21	832	0.25	0.25	TransSW
9.33	0.85	0.14	12	11.67	11.67	Aerosol
6.00	0.94	0.09	12	7.50	20.00	CloudsLW

- **CloudsLW time dropped from 200 to 90 ms (PC)**
- **Same test could not be repeated in the LNCC Santos Dumont supercomputer**

RESULTS - FORMER TOY PROBLEM PINN x SINDy models - elapsed times

(local PC machine)

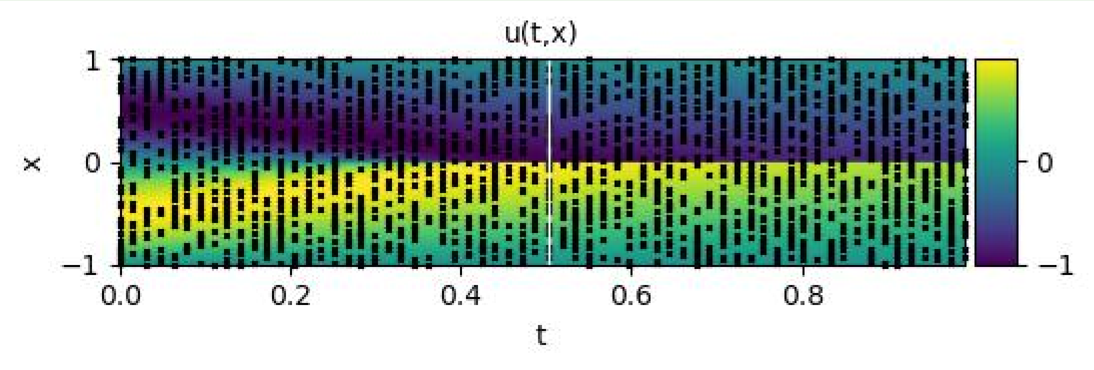
0.01/π (low viscosity)

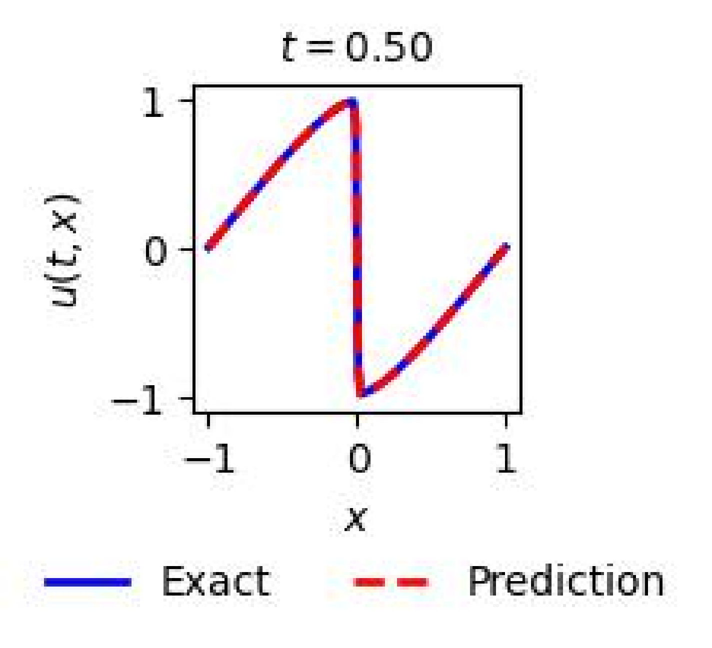
Model	Elapsed [s]
128x64	
PINN Train	47.567
PINN Predict	0.371
STLSQ	0.031
FROLS	0.140
SR3	0.054
SSR	0.068
256x128	
PINN Train	53.033
PINN Predict	0.732
STLSQ	0.049
FROLS	0.071
SR3	0.098
SSR	0.086
512x256	
PINN Train	52.633
PINN Predict	3.067
STLSQ	0.105
FROLS	0.625
SR3	0.118
SSR	0.181

0.1/π (high viscosity)

Model	Elapsed [s]
128x64	
PINN Train	22.633
PINN Predict	0.361
STLSQ	0.013
FROLS	0.060
SR3	0.015
SSR	0.043
256x128	
PINN Train	36.100
PINN Predict	0.995
STLSQ	0.033
FROLS	0.074
SR3	0.015
SSR	0.060
512x256	
PINN Train	23.133
PINN Predict	3.020
STLSQ	0.086
FROLS	0.588
SR3	0.095
SSR	0.262

RESULTS - FORMER TOY PROBLEM
PINN visual assessment, 128x64, low viscosity





- **PINN performed the simulation accurately capturing the non-linear behavior, but SINDy would require a finer discretization...**

RESULTS - FORMER TOY PROBLEM
Parameter discovery results - low viscosity

Correct PDE: 0.003183 u_xx - 1.0 uu_x (viscosity = 0.01/pi)

Model	Discovered equation and parameters
	128x64 problem size
PINN	0.0033735 u_xx - 0.99912 uu_x
STLSQ	0.06420 u + 0.00505 u_xx - 1.06304 uu_x + + 0.00469 uuu_xx - 0.00001 uu_xxx
FROLS	-0.418 u
SR3	0.064 u + 0.005 u_xx - 1.063 uu_x + 0.005 uuu_xx
SSR	0.064 u + 0.005 u_xx - 1.063 uu_x + 0.005 uuu_xx
	256x128 problem size
PINN	0.0031779 u_xx - 0.99942 uu_x
STLSQ	0.00395 u_xx - 1.00869 uu_x + 0.00126 uuu_xx
FROLS	0.015 u + 0.004 u_xx - 1.003 uu_x
SR3	0.004 u_xx - 1.009 uu_x + 0.001 uuu_xx
SSR	0.011 u + 0.004 u_xx - 1.011 uu_x + 0.001 uuu_xx
	512x256 problem size
PINN	0.0031403 u_xx - 0.99850 uu_x
STLSQ	0.00339 u_xx - 1.00534 uu_x + 0.00041 uuu_xx
FROLS	0.006 u + 0.004 u_xx - 1.006 uu_x
SR3	0.003 u_xx - 1.005 uu_x
SSR	0.006 u + 0.003 u_xx - 1.007 uu_x

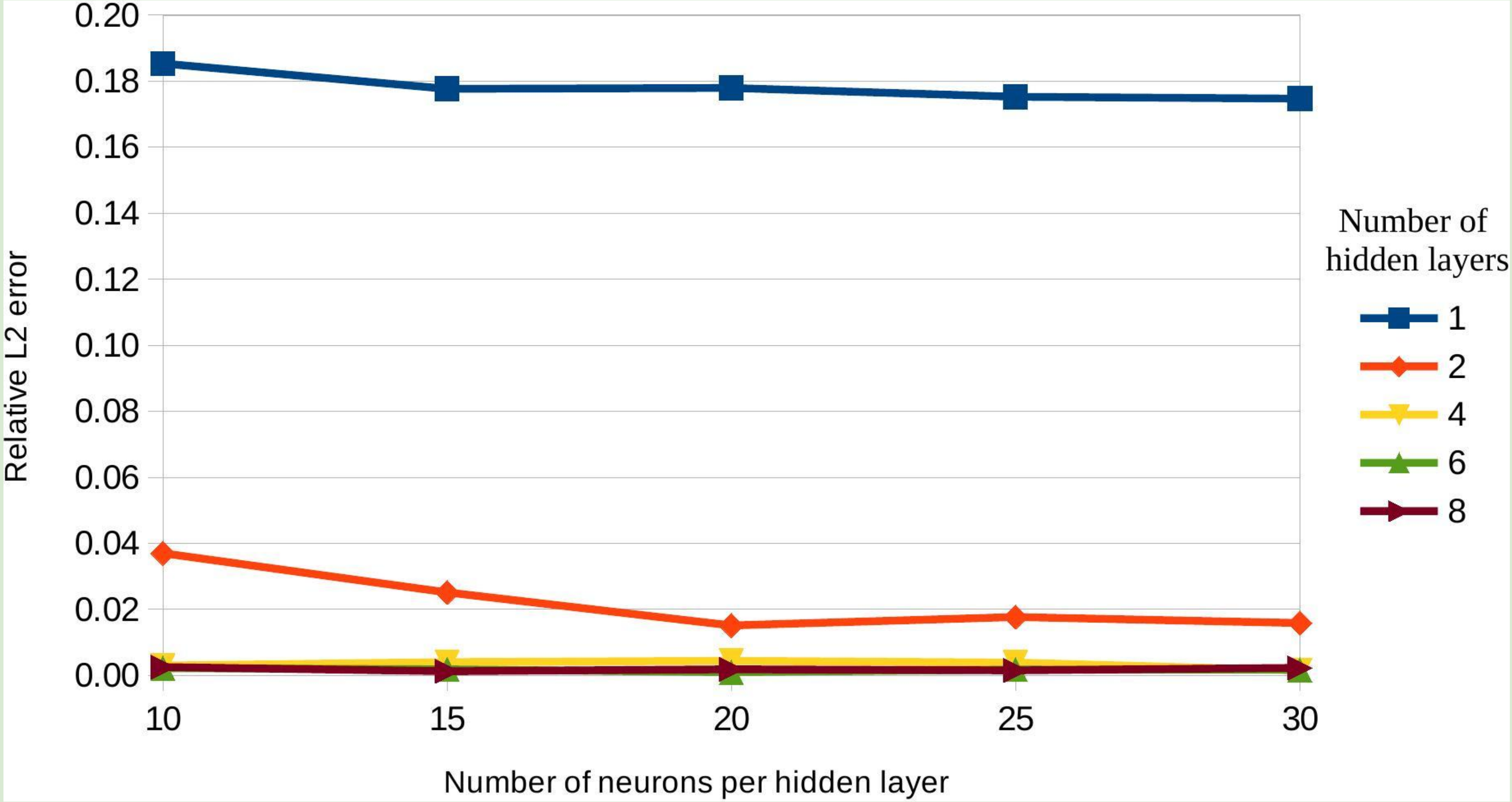
RESULTS - FORMER TOY PROBLEM
L2 error & Training time x Hidden Layers

(SDumont Sequana X1129)

Hidden layers	Number of neurons per hidden layer				
	10	15	20	25	30
	Relative L2 Error (%)				
1	18.54	17.77	17.80	17.53	17.47
2	3.70	2.52	1.52	1.77	1.59
4	0.30	0.41	0.44	0.39	0.16
6	0.22	0.19	0.10	0.18	0.17
8	0.26	0.13	0.19	0.16	0.23
	Training - processing time (seconds)				
1	4.2	5.4	5.0	21.7	9.4
2	35.9	51.8	39.3	55.5	70.7
4	51.2	43.1	33.4	40.9	47.4
6	59.7	40.3	42.5	35.2	38.6
8	58.7	60.0	58.6	54.4	84.5

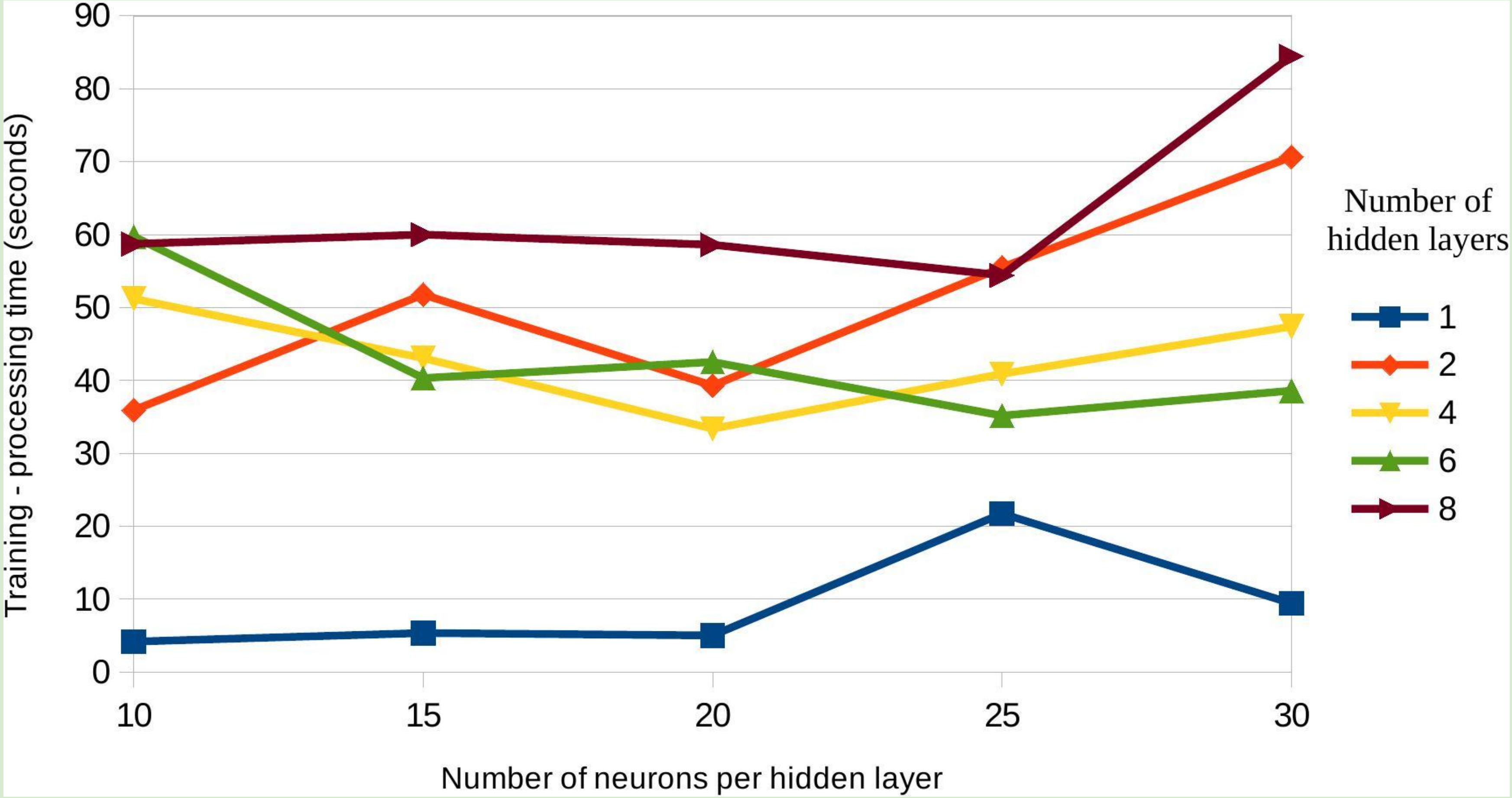
RESULTS - FORMER TOY PROBLEM
Relative L2 error (%) - low viscosity

(SDumont Sequana X1129)

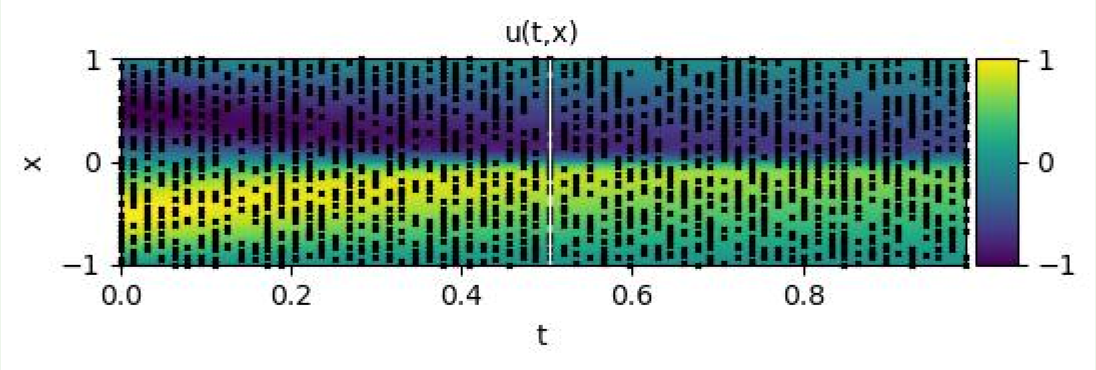


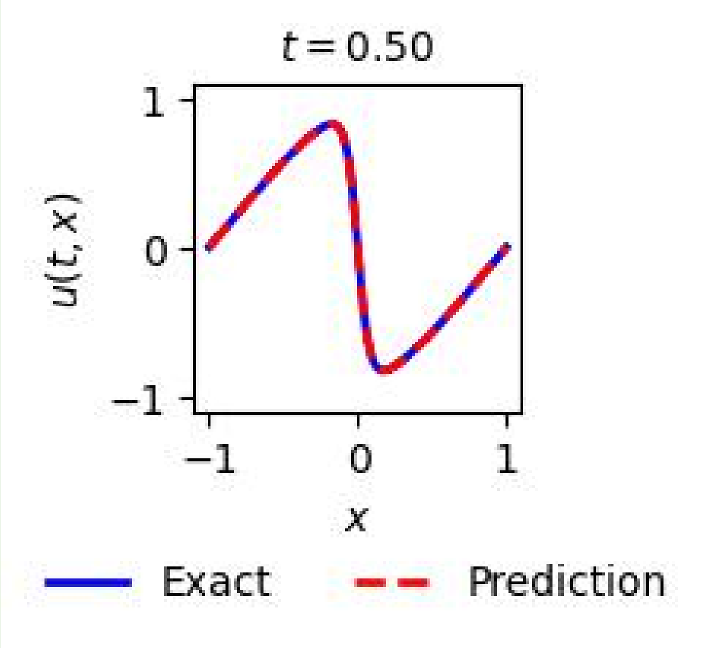
RESULTS - FORMER TOY PROBLEM
Processing times - low viscosity

(SDumont Sequana X1129)



RESULTS - FORMER TOY PROBLEM
PINN visual assessment, 128x64, high viscosity





- **Both PINN and SINDy performed the simulation accurately**

RESULTS - FORMER TOY PROBLEM
Parameter discovery results, high viscosity

Correct PDE: 0.03183 u_xx - 1.0 uu_x (viscosity = 0.1/pi)

Model	Discovered equation and parameters
	128x64 problem size
PINN	0.0318582 u_xx - 0.99928 uu_x
STLSQ	0.03222 u_xx - 1.00012 uu_x
FROLS	0.032 u_xx - 1.000 uu_x
SR3	0.032 u_xx - 1.000 uu_x
SSR	0.003 u + 0.032 u_xx - 1.002 uu_x
	256x128 problem size
PINN	0.0318372 u_xx - 0.99924 uu_x
STLSQ	0.03193 u_xx - 1.00002 uu_x
FROLS	0.032 u_xx - 1.000 uu_x
SR3	0.032 u_xx - 1.000 uu_x
SSR	0.001 u + 0.032 u_xx - 1.000 uu_x
	512x256 problem size
PINN	0.0318292 u_xx - 0.99936 uu_x
STLSQ	0.03186 u_xx - 1.00001 uu_x
FROLS	0.032 u_xx - 1.000 uu_x
SR3	0.032 u_xx - 1.000 uu_x
SSR	0.032 u_xx - 1.000 uu_x

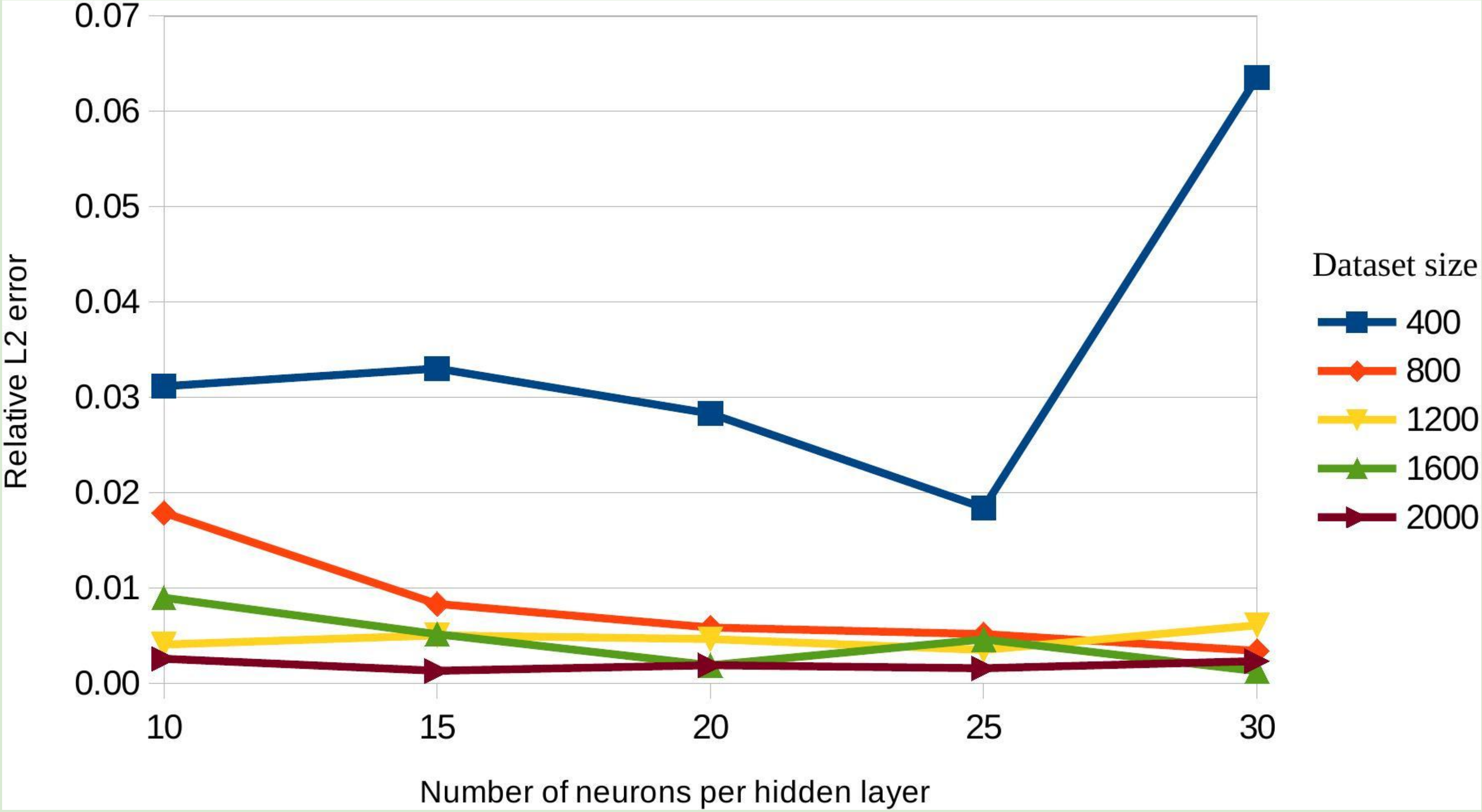
RESULTS - FORMER TOY PROBLEM L2 error and training time x CP size

(SDumont Sequana X1129)

Dataset size	Number of neurons per hidden layer				
	10	15	20	25	30
Relative L2 Error (%)					
400	3.12	3.30	2.83	1.84	6.36
800	1.79	0.83	0.59	0.52	0.34
1200	0.41	0.50	0.46	0.35	0.61
1600	0.90	0.51	0.19	0.46	0.13
2000	0.26	0.13	0.19	0.16	0.23
Training - processing time (seconds)					
400	57.9	82.3	83.3	59.8	58.6
800	79.7	53.5	63.2	45.0	63.0
1200	63.7	52.2	43.8	42.1	56.8
1600	59.9	27.5	45.3	46.5	56.4
2000	58.7	60.0	58.6	54.4	84.5

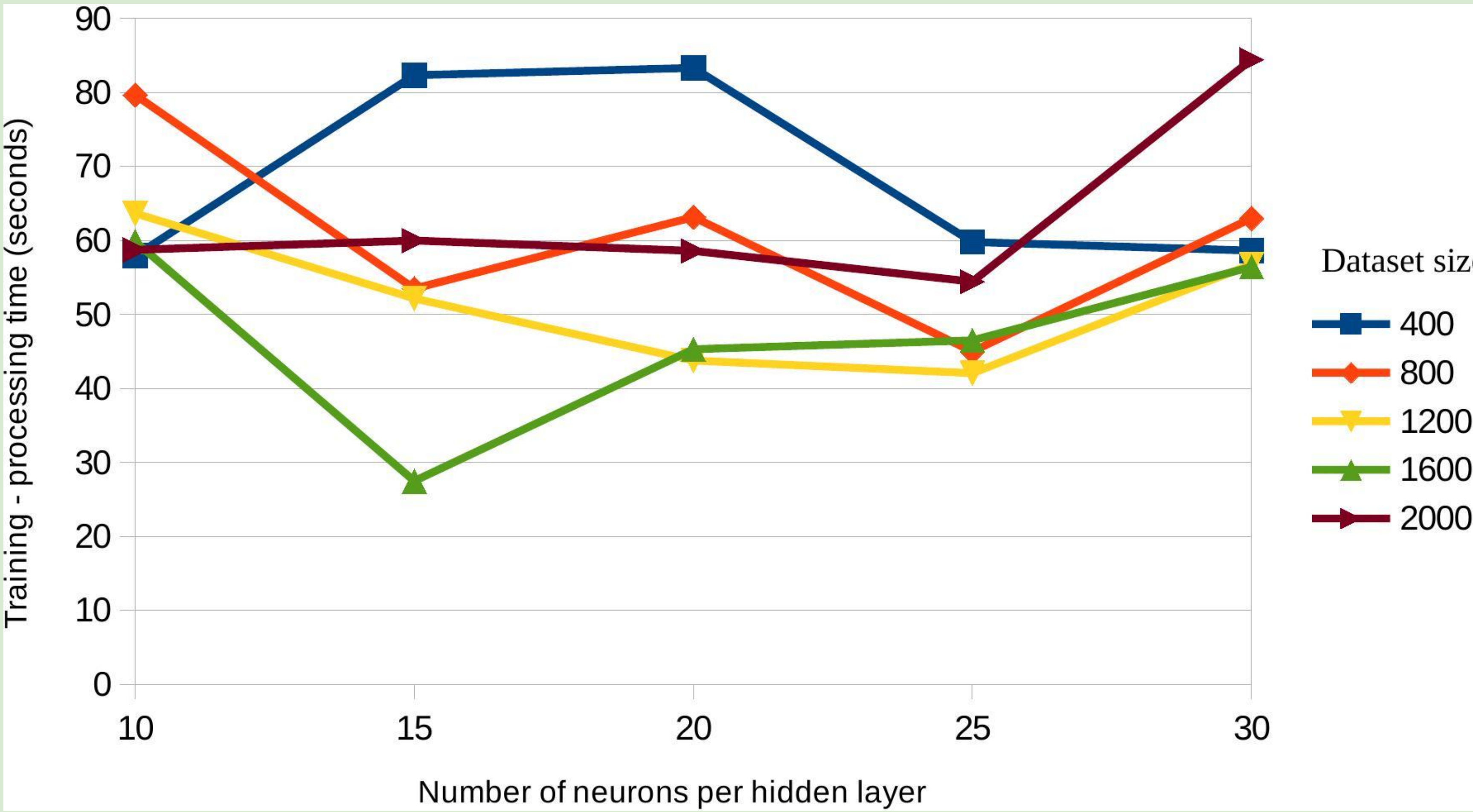
RESULTS - FORMER TOY PROBLEM
Relative L2 error (%) - high viscosity

(SDumont Sequana X1129)



RESULTS - FORMER TOY PROBLEM
Processing times - high viscosity

(SDumont Sequana X1129)



RESULTS - FORMER TOY PROBLEM
Prediction time [s] - high viscosity

(SDumont Sequana X1129)

Number of hidden layers	Number of neurons per hidden layer		
	10	20	30
1	0.647	0.636	0.675
4	0.704	0.724	0.705
8	1.092	0.867	0.789

(Complements the previous graphs)

WORK PLAN
Proposed tasks/steps of the thesis

- Bibliographic research
- PIML-based radiation module implementation
- Optimization of the former DNN-based module
- PINN-based implementation
- Further PIML approaches implementation
- Further case studies, PIML, PINN
- Article submission
- Thesis writing

WORK PLAN
Schedule

Tasks	2024	2025			2026	
	9-12	1-4	5-8	9-12	1-4	5-8
Bibliographic research	■	■	■	■	■	
PIML-based radiation module implementation	■	■				
Optimization of the former DNN-based module		■	■			
PINN-based implementation			■	■		
Further PIML approaches implementation				■	■	
Further case studies, PIML, PINN				■	■	
Article submission		■	■	■	■	■
Thesis writing	■	■	■	■	■	■

CONCLUSION
Final considerations

- PIML is an innovative and promising approach that combines data-driven and physics-based strategies
- Propose to replace RRTMGP gas-optics numerical scheme of the ecRad radiation module with an implementation using PIML
- Incremental approach: DNN, PINN, other PIML
- Current toy problem already reproduced DNN-based implementation of the gas-optics scheme

CONCLUSION
Post-thesis future works

- Explore new PIML-based approaches for 2D/3D inverse and direct problems in other applications
- Real-world problems with limited-size or noisy datasets
- Use of GPU frameworks like Modulus and Horovod
- The resulting PIML-based gas-optical scheme can employed in the microphysis of the MONAN global model (INPE & other institutions)

Thanks!

Source code: http://github.com/efurlanm/pd1b24/

Eduardo Furlan Miranda. Applied Computing Post-graduation Program (CAP/INPE). <efurlanm@gmail.com>.

Stephan Stephany. Coordination of Applied Research and Technological Development (COPDT/INPE). <stephan.stephany@inpe.br>.

Roberto Pinto Souto. National Laboratory for Scientific Computing (LNCC). <rpsouto@lncc.br>.