MACHINE LEARNING FOR PARALLEL-IN-TIME METHODS?

Sebastian Götschel

joint work with Judith Angel, Abdul Qadir Ibrahim & Daniel Ruprecht

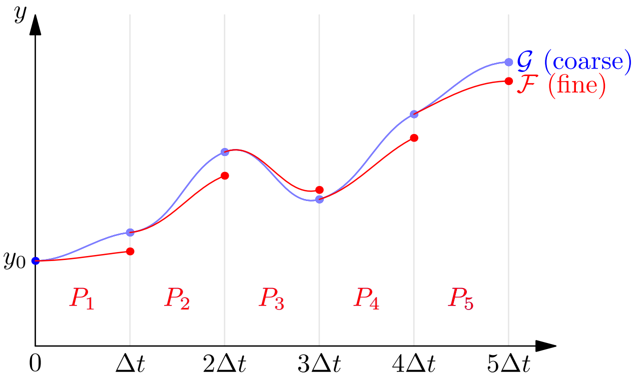
Chair Computational Mathematics
Institute of Mathematics (E-10)
Hamburg University of Technology

PinT Workshop 2022, Marseille

July 11, 2022

Parareal [Lions/Maday/Turinici '01]

- IVP $y_t = f(t,y),\ y(0) = y_0$
- Compute $y(t_j)$ on a time grid $0 = t_0 < t_1 < \cdots < t_N = T$

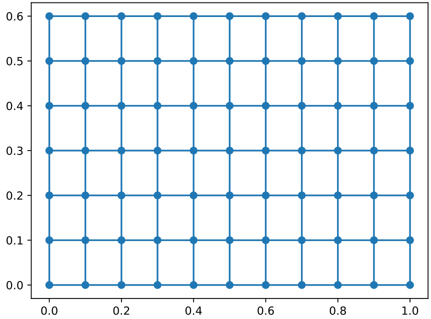


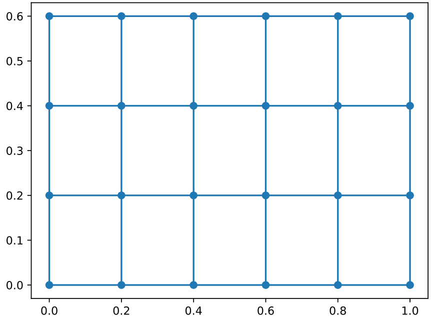
$$Y_{j+1}^{[k+1]} = \mathcal{F}(Y_j^{[k]}) + \mathcal{G}(Y_j^{[k+1]}) - \mathcal{G}(Y_j^{[k]})$$

speedup:

$$S(N_p) \le \min\left(\frac{N_p}{N_{\mathrm{it}}}, \frac{\text{runtime fine}}{\text{runtime coarse}}\right)$$

Spatial coarsening [Fischer/Hecht/Maday '05]





A fine mesh for $\mathcal{F}$ and a coarse mesh for $\mathcal{G}$.

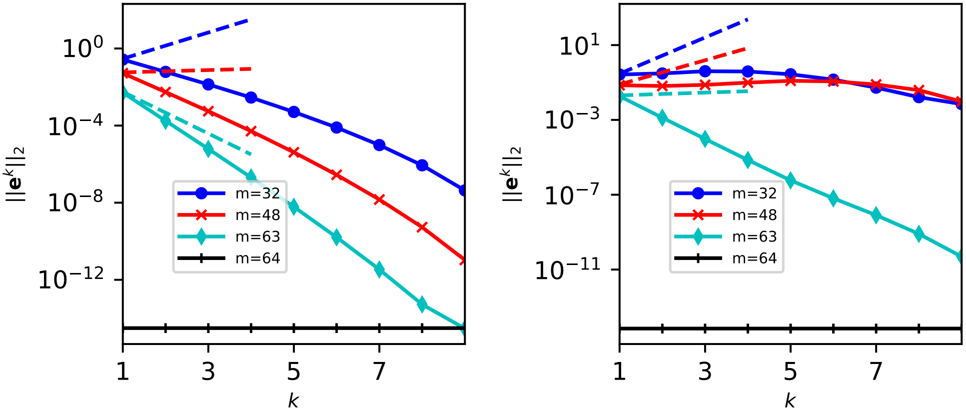
Parareal becomes

$$Y_{n+1}^{[k+1]} = \mathbf{I}\mathcal{G}(\mathbf{R}\,Yn^{[k+1]}) + \mathcal{F}(Y_n^{[k]}) - \mathbf{I}\mathcal{G}(\mathbf{R}\,Y_n^{[k]})$$

with $\mathbf{I}$ = interpolation and $\mathbf{R}$ = restriction.

- cheaper coarse propagator

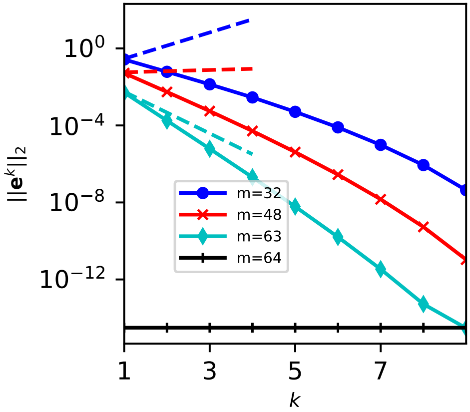
But: often slow convergence with spatial coarsening

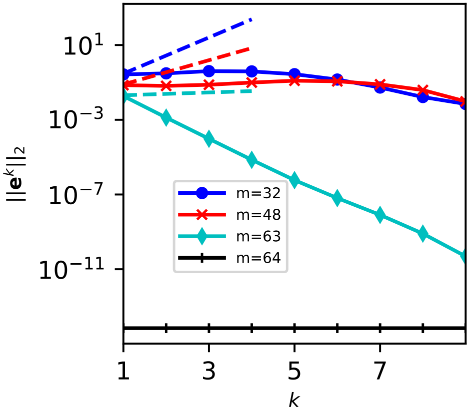


Linear advection: convergence of Parareal with (left) and without (right) numerical diffusion.

[Angel/G./Ruprecht arXiv:2111.10228]

But: often slow convergence with spatial coarsening





Linear advection: convergence of Parareal with (left) and without (right) numerical diffusion.

[Angel/G./Ruprecht arXiv:2111.10228]

Can we use machine learning to improve convergence?

Supervised learning

Given: pairs $(x, y) \in \mathcal{X} \times \mathcal{Y}$ from unknown joint probability distribution $\mathcal{D}$

Goal: determine prediction function $h : \mathcal{X} \to \mathcal{Y}$ such that $h(x)$ is a good predictor of true output $y(x)$

- minimize generalization error: $\min_{h \in \mathcal{H}} \mathbb{E}_{(x,y)\sim\mathcal{D}} [\ell(h(x), y)]$

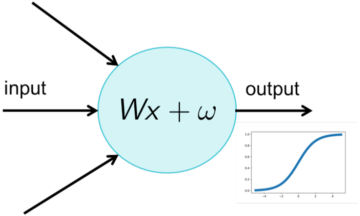
prediction function h from some fixed class $\mathcal{H}$

expectation wrt. distribution $\mathcal{D}$

loss of the model at data point (x, y), e.g., $\ell(h(x), y) = (y - h(x))^2$

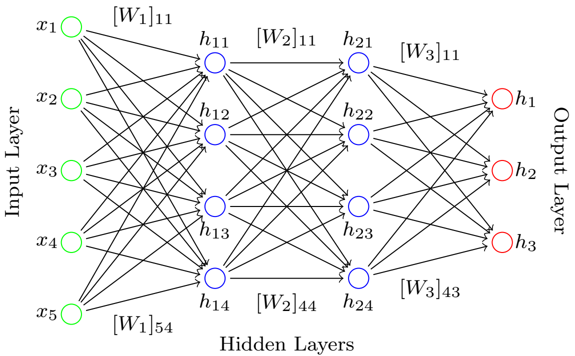
$\mathcal{D}$ unknown, but samples $(x_i, y_i), i = 1, \ldots, N$ available; h parameterized by weights w

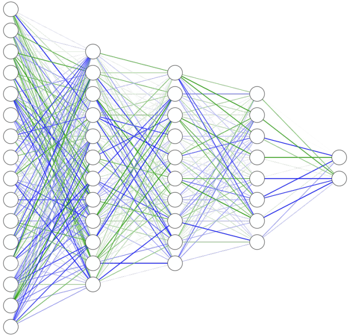
- empirical risk minimization: $\min_{w} \frac{1}{N} \sum_{i=1}^{N} \ell(h(w; x_i), y_i)$



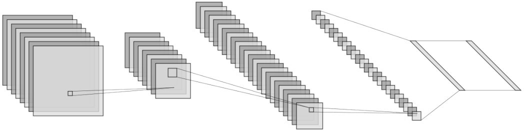
Neuron

$$h(w;x) = a_l(W_l \dots (a_2(W_2(a_1(W_1 x + \omega_1)) + \omega_2)) \dots)$$

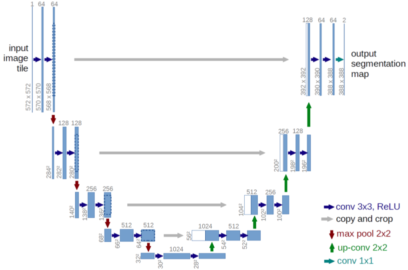




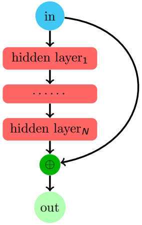
fully connected



convolutional NN



UNet [Ronneburger et al. '15]

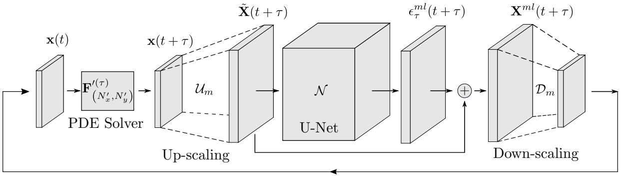


ResNet
[He et al. '16]

Machine learning for PDEs (without aiming for completeness)

Hamburg University of Technology

1. Learning multigrid operators, e.g., [Katrutsa et al. '19; Tomasi/Krause '21; ...]
2. Superresolution, e.g., [Kochkov et al. '21; Pathak et al. '21; ...]: coarse simulation enhanced using ML to populate the finer scales



related: learned correction for Parareal [Nguyen/Tsai '21]

3. Training neural networks as solvers, e.g., [Raissi et al. '19; Chen et al. '21; Li et al. '21; Stender et al. '22; ...], also for Parareal, e.g., [Agboh et al. '20;...]

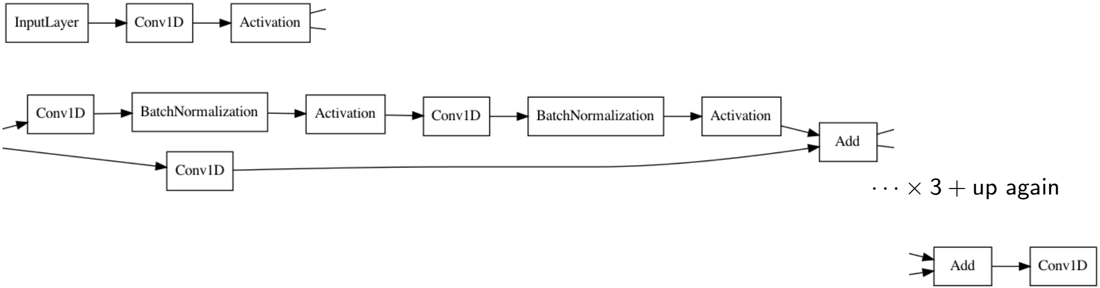
more machine learning

Approach 1: Superresolution

Idea: augment coarse solution by learned correction $\mathcal{G}(Y) = I\tilde{\mathcal{G}}(RY) + \Delta Y$, $\quad \Delta Y = NN(I\tilde{\mathcal{G}}(RY))$

Data: pairs $\mathcal{F}(Y_i), I\tilde{\mathcal{G}}(RY_i)$

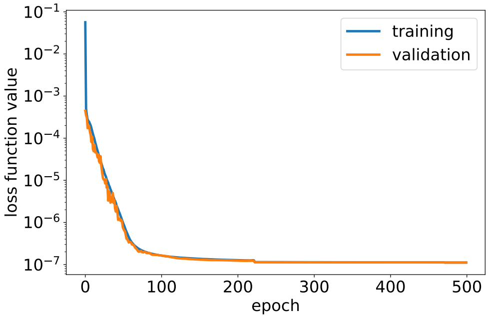
Output: $\Delta Y \approx \mathcal{F}(Y_i) - I\tilde{\mathcal{G}}(RY_i)$



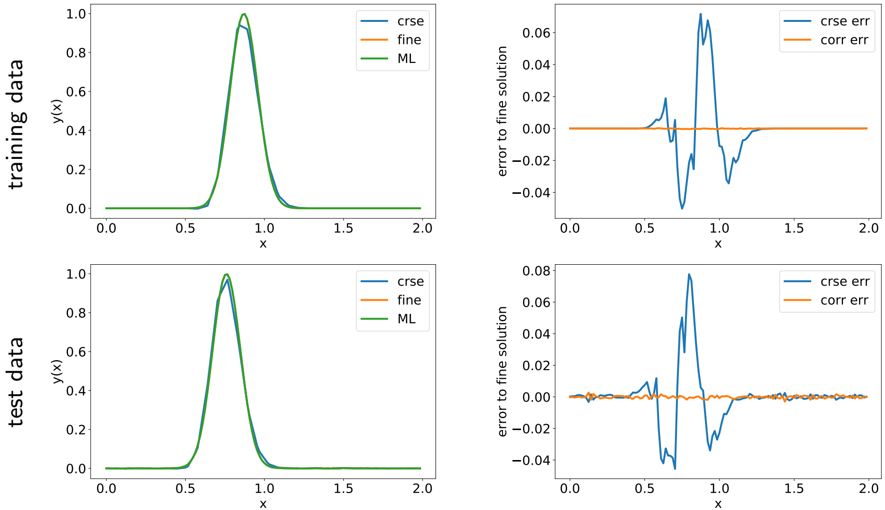
- specific to type of equation, e.g., $y_t - vy_x = 0$, type of coarse/fine propagator
- somewhat flexible wrt. spatial resolution (convolutions)

Example: linear advection $y_t - vy_x = 0$

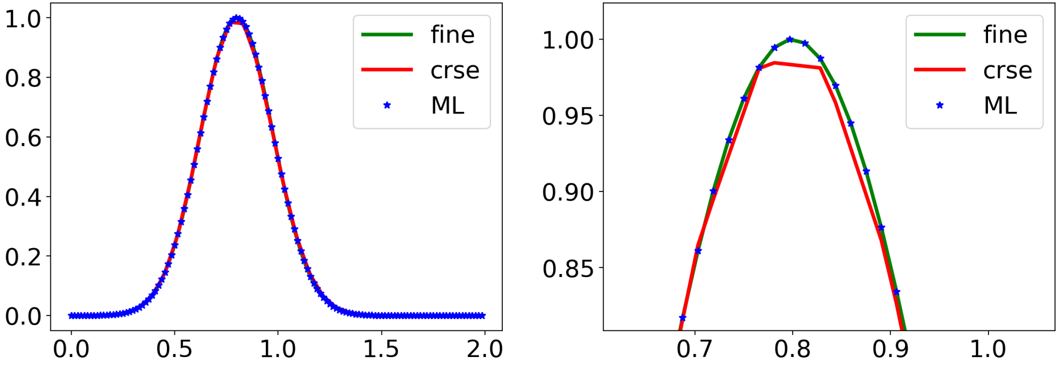
- solver: pyParareal `https://github.com/Parallel-in-Time/pyParareal`
- discretization: centered FD+trapezoidal rule, $N_x = 128/32$, $N_p = 10$, $N_t = 10/5$
- data generation from various initial conditions $y(x,t) = \exp(-\frac{x-x_0-vt}{\sigma^2})$ (varying t, x_0, v, σ)
- training: Tensorflow/Keras, ℓ_2 loss (ℓ_∞ similar), Adam optimizer



Training results



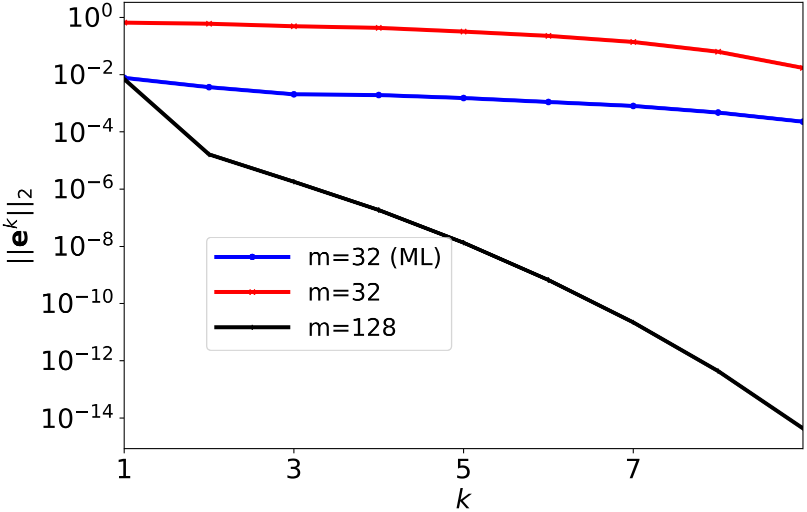
Application during Parareal run



Coarse, fine, and corrected coarse solution at $t = \Delta t$.

$\|\text{fine} - \text{coarse}\| = 1.6 \cdot 10^{-2}$, $\|\text{fine} - \text{ML}\| = 9.2 \cdot 10^{-5}$

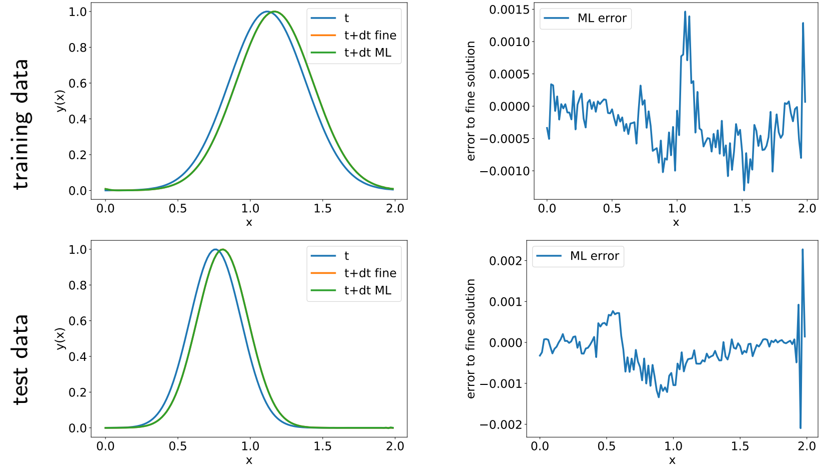
However...



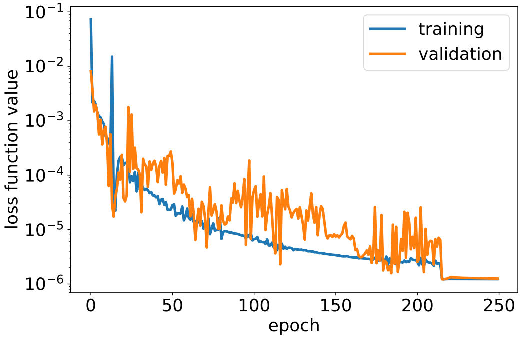
10^0
10^{-2}
10^{-4}
10^{-6}
10^{-8}
10^{-10}
10^{-12}
10^{-14}
$||\mathbf{e}^k||_2$
m=32 (ML)
m=32
m=128
1
3
5
7
k

Approach 2: Learn coarse solver

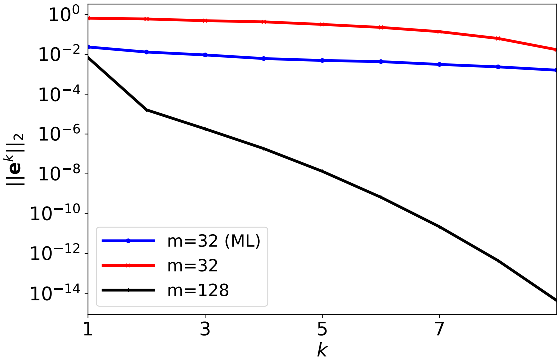
Idea: train mapping $Y_n \mapsto Y_{n+1}$ to replace coarse solver (using UNet as before)



Well, that doesn't help



Training progress



Parareal convergence.

Outlook: Learn coarse solver by PINN

Idea: train mapping $Y_n \mapsto Y_{n+1}$ to replace coarse solver
Physics-informed neural network: take PDE into account

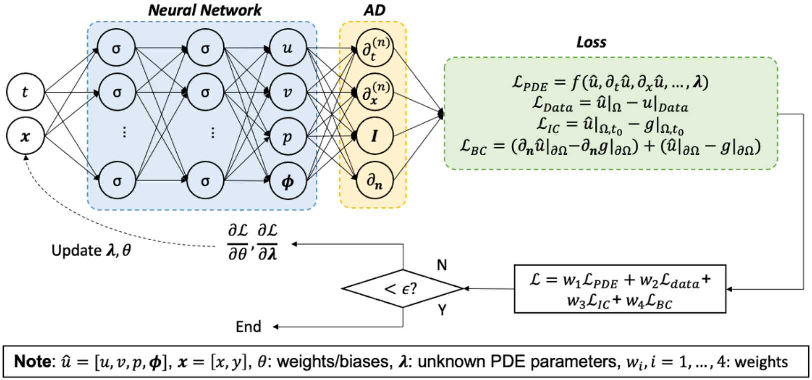
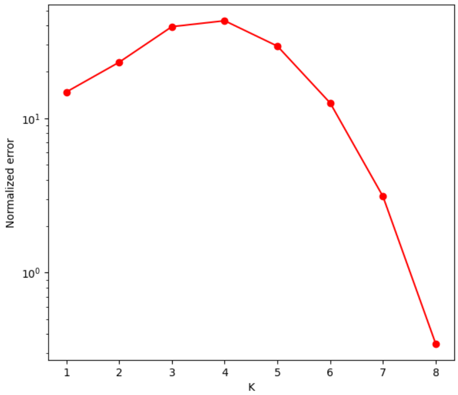
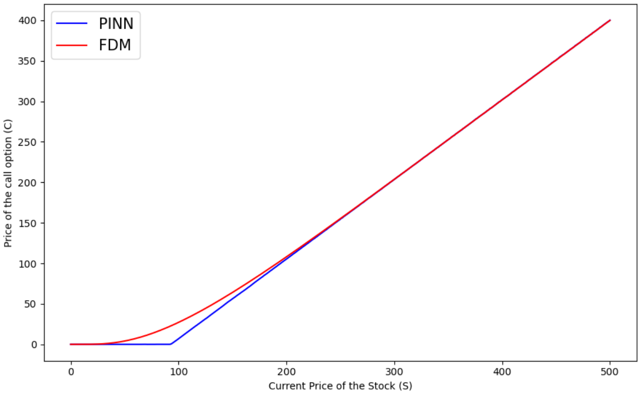


Figure: [Cai et al., arXiv:2105.09506]

First results





Left: Parareal convergence for Black-Scholes with FD-discretization; Right: PINN improves accuracy of solver.

Summary

Hamburg University of Technology

ML to speed up Parareal convergence?

- several potential approaches: superresolution, training coarse solver
- expensive training, but reasonably good generalization
- reduced error, no improved convergence
- standard loss functions seem insufficient
- vast amount of hyperparameters for ML, better understanding required

Summary

Hamburg University of Technology

ML to speed up Parareal convergence?

- several potential approaches: superresolution, training coarse solver
- expensive training, but reasonably good generalization
- reduced error, no improved convergence
- standard loss functions seem insufficient
- vast amount of hyperparameters for ML, better understanding required

Thank you!

sebastian.goetschel@tuhh.de