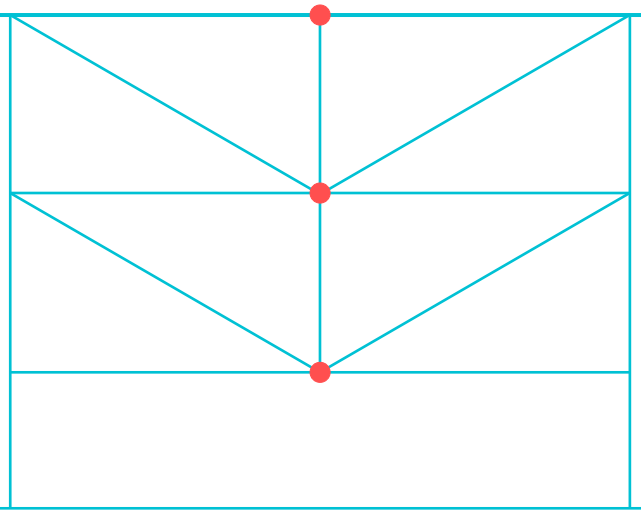
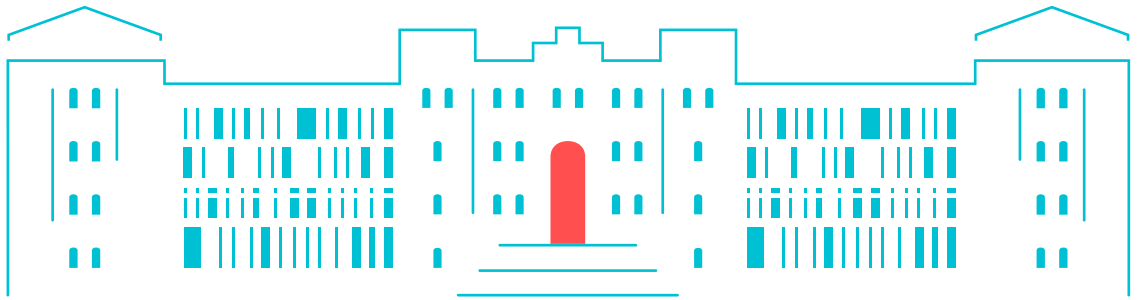
ML-enhanced Numerics: From Super-resolution to Neural Operators for Parallel-in-Time Methods

Sebastian Götschel

TUHH
Hamburg
University of
Technology





Go20 Conference, Malta, 2025

Acknowlegements

Thomas Baumann (JSC), Gayatri Čaklović (KIT), Abdul Qadir Ibrahim, **Thibaut Lunet**, Daniel Ruprecht, Martin Schreiber (U Grenoble Alpes), Robert Speck (JSC)

ExaOcean

Philip Freese, Fabricio Lapolli (MPI-M), Yen-Sen Lu (JSC), Max Witte (DKRZ) Lars Hoffmann (JSC), Christopher Kadow (DKRZ), Peter Korn (MPI-M)

SPONSORED BY THE

Federal Ministry of Education and Research

This project received funding from the German Federal Ministry of Education and Research (BMBF) under grant 16ME0679K.

NeuralPinT

Chelsea John, Andreas Herten, Stefan Kesselheim (all JSC)

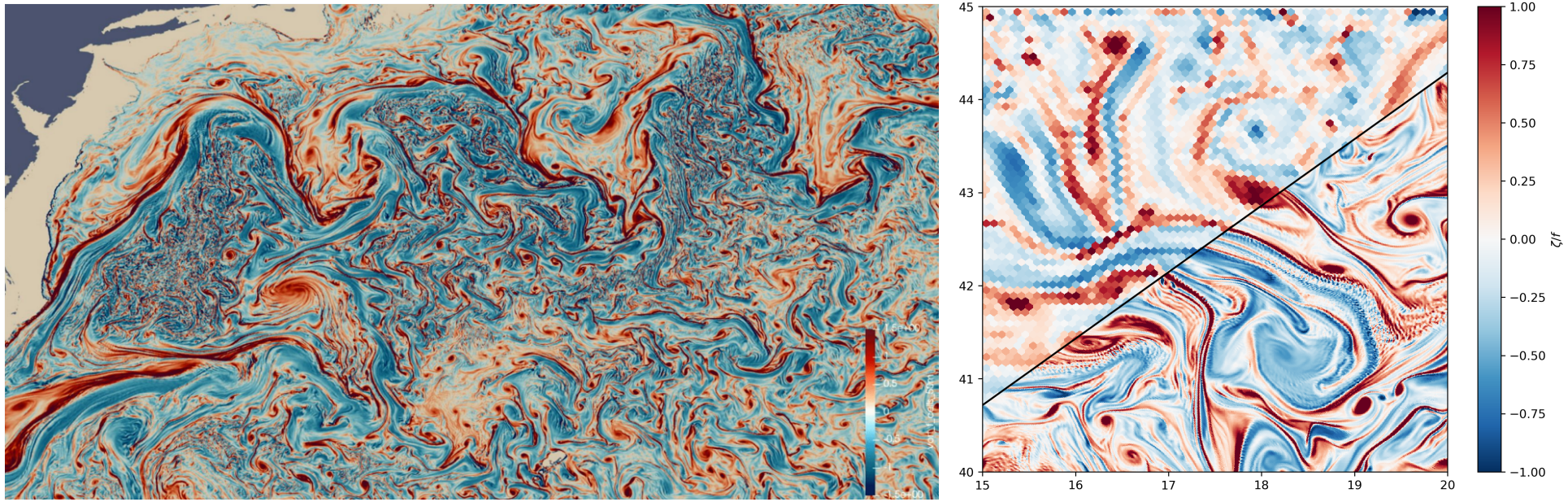
Funded by the European Union

This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 101118139. The JU receives support from the European Union's Horizon Europe Programme.

EuroHPC Joint Undertaking

Sub-mesoscale eddies

TUHH



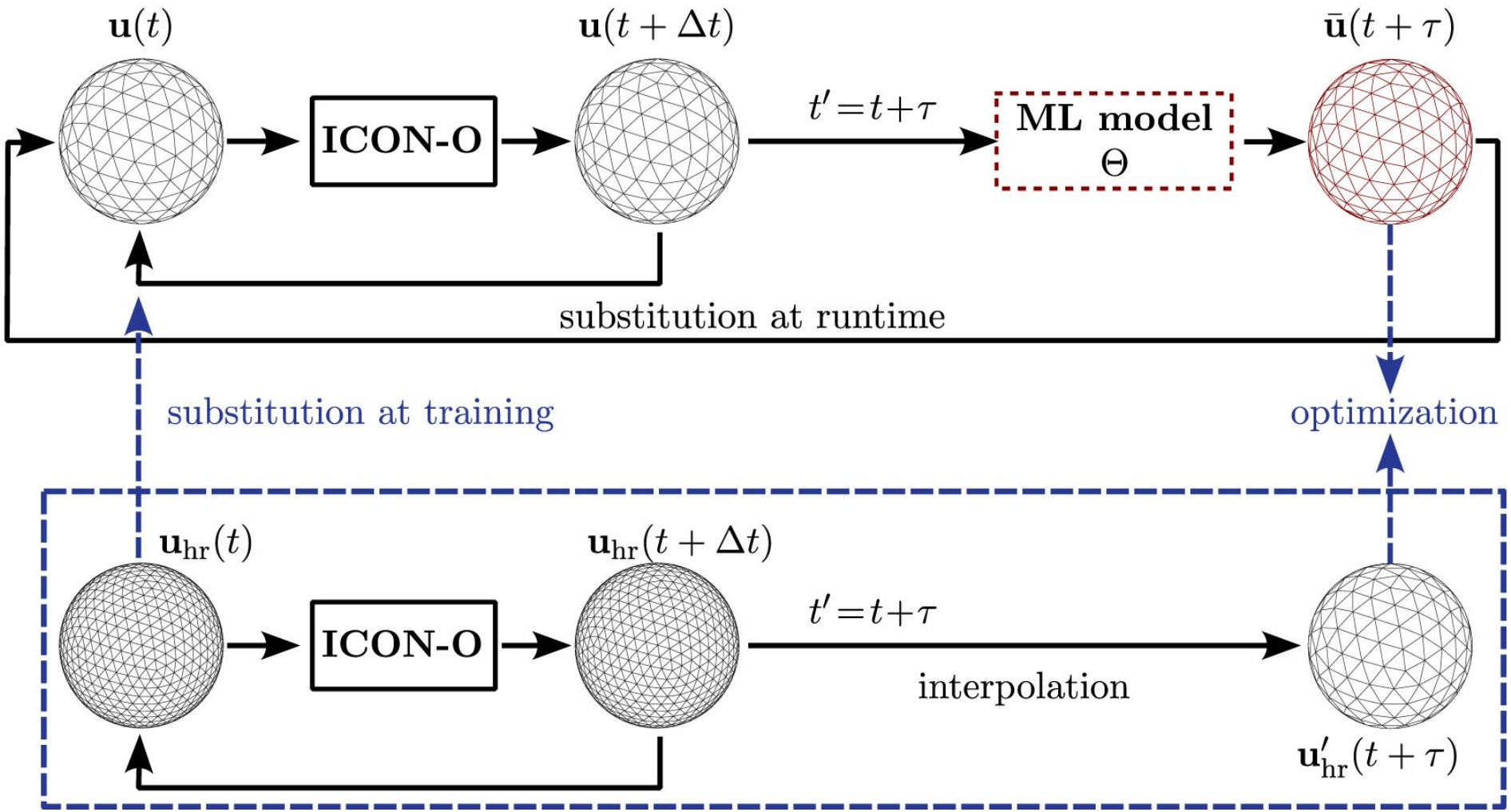
Vorticity in an ICON-O simulation of sub-mesoscale eddies in the North Atlantic (courtesy P. Korn, F. Lapolli, MPI-M)

- resolving sub-mesoscale eddies requires local resolution of $\approx$ 600m
- throughput 64 days per 24h runtime: runs over climatological timescales impractical

➤ solve on coarser grid and correct, add time parallelism

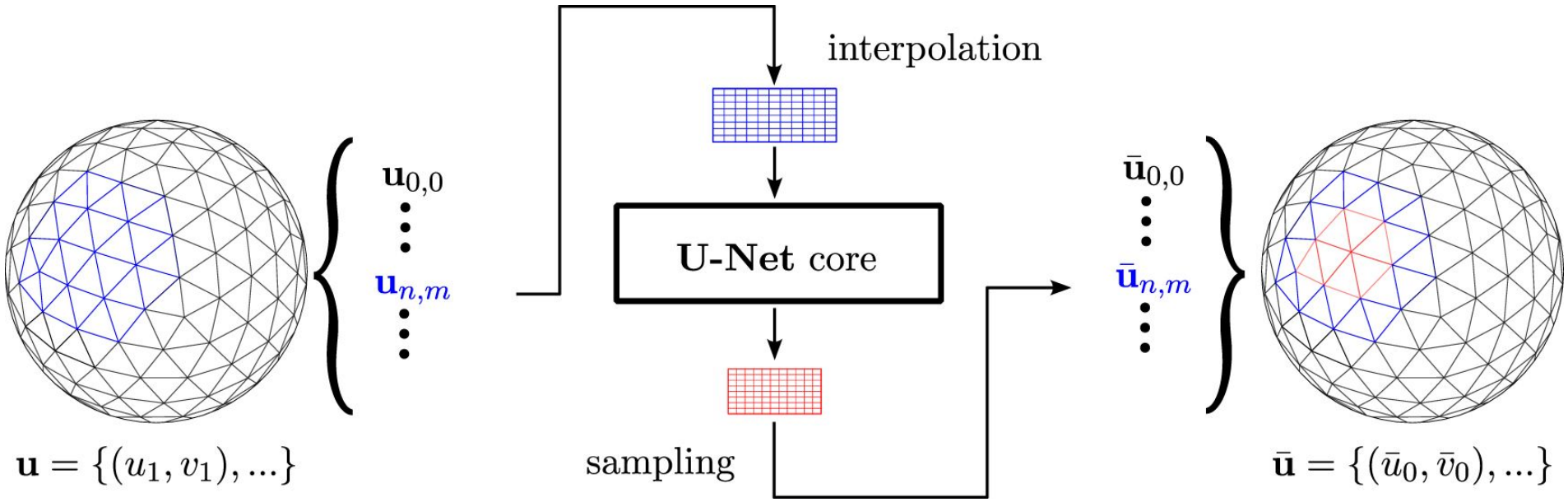
Dynamic deep-learning based superresolution

TUHH



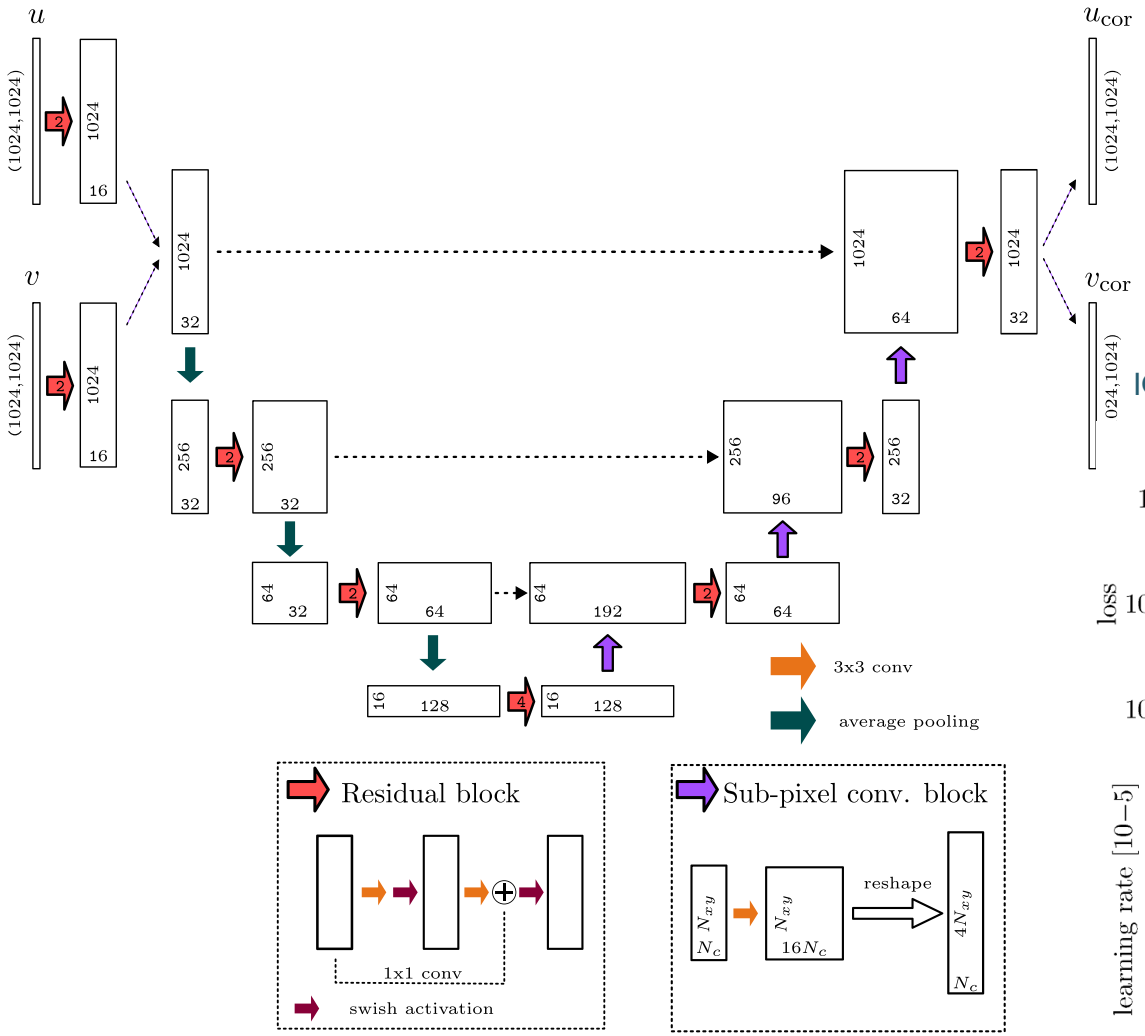
Dynamic deep-learning based superresolution

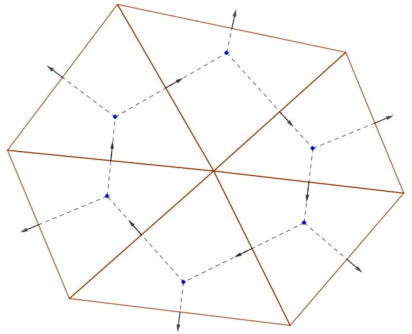
TUHH



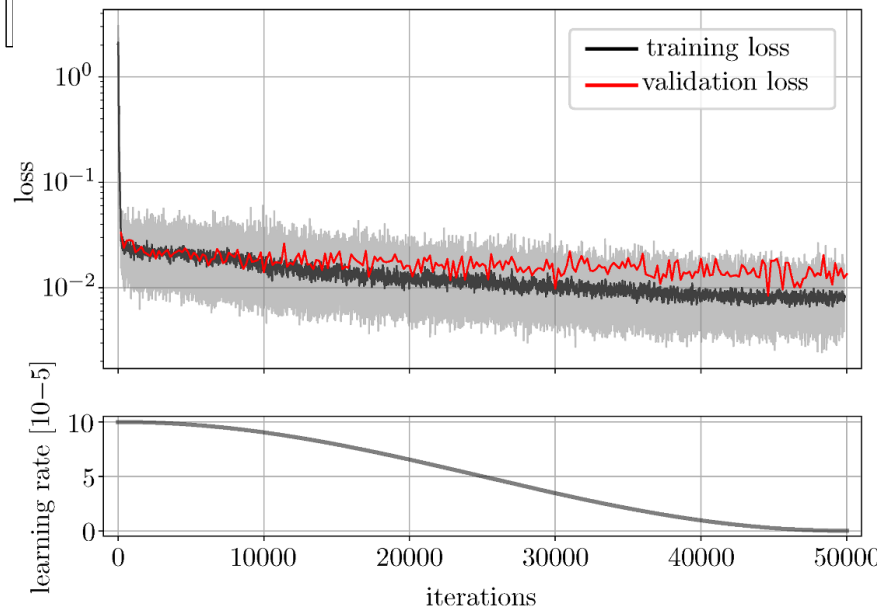
Dynamic deep-learning based superresolution

TUHH





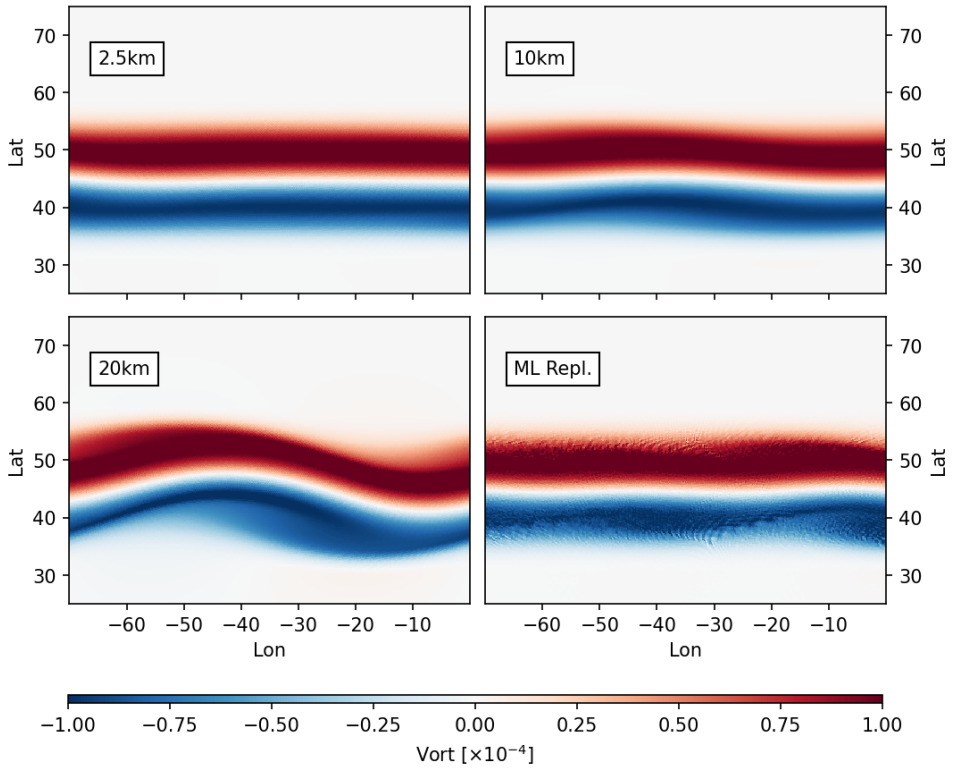
ICON-O grid cell: height in cells, velocity on edges



Dynamic deep-learning based superresolution

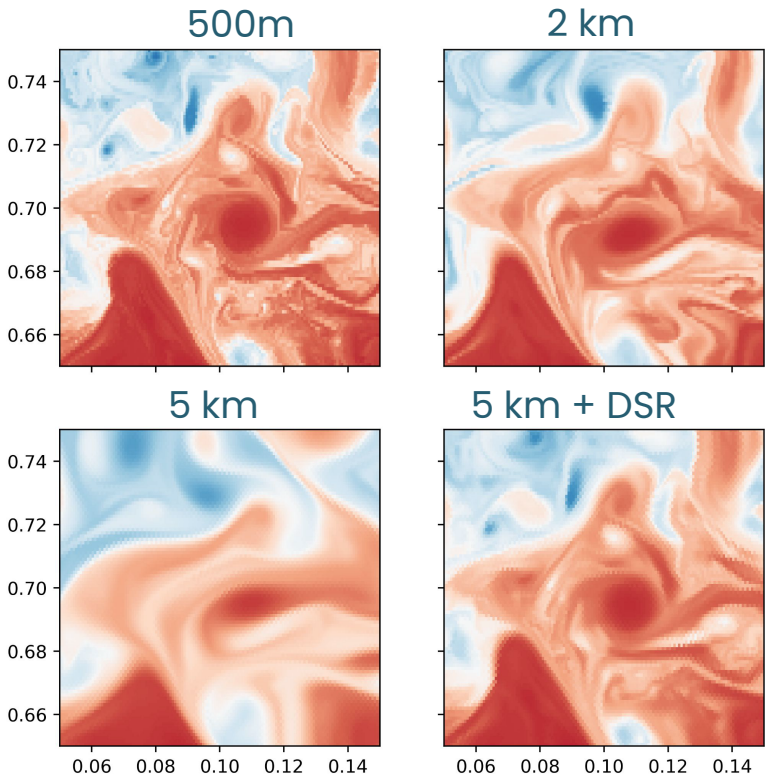
TUHH

Barotropic instability (SWE, 2D)



- DSR corrects onset of instability on coarse grid

Baroclinic instability (3D)



temperature after 20 days, correction every 6h

(courtesy M. Witte, DKRZ)

Witte, Lapolli, Freese, G., Ruprecht, Korn, Kadow: Dynamic deep learning based super-resolution for the shallow water equations. Machine Learning: Science and Technology 6, 015060. https://dx.doi.org/10.1088/2632-2153/ada19f (2025)

(Physics-informed/Fourier) Neural Operators [Li et al. '21/'23]

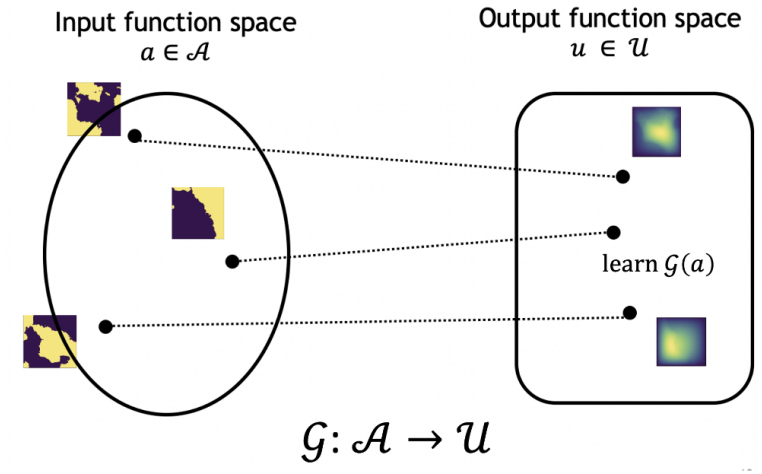


Figure: [Li et al. '21/'23]

- trainable without data (PDE residual as loss)
- fast to evaluate once trained
- leverage heterogeneous architectures

➤ use as fast propagators in Parallel-in-Time (PinT) methods

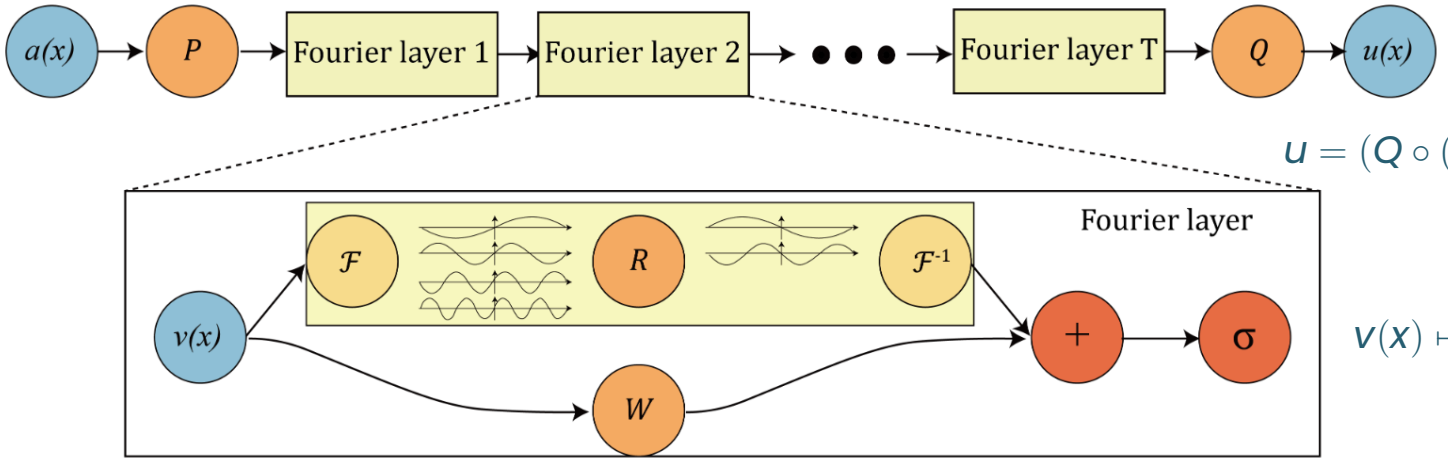


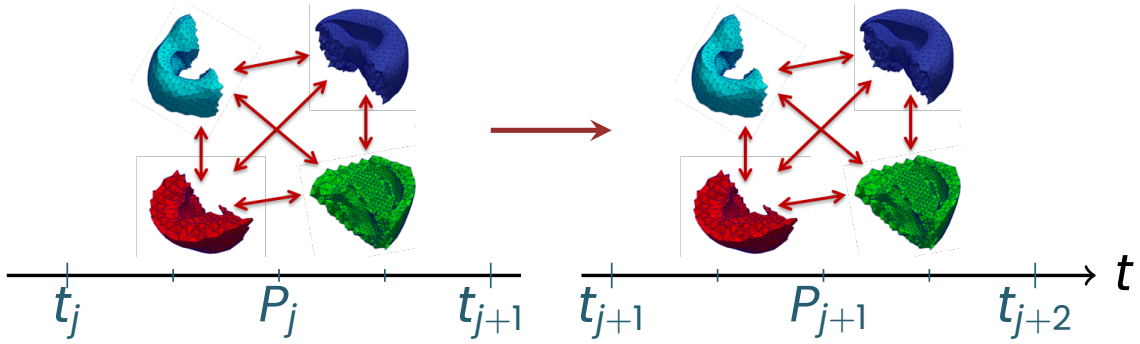
Figure: [Li et al. '21]

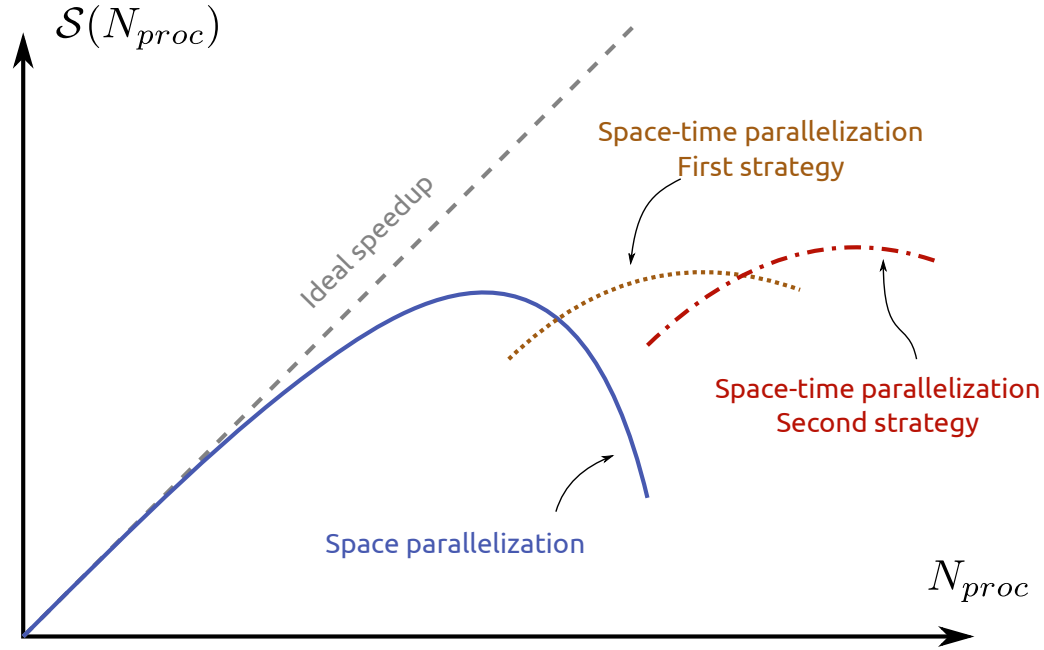
$$u = (Q \circ (K_\ell \circ \sigma_\ell \circ \cdots \circ \sigma_1 \circ K_0) \circ P)(a)$$

K an integral operator:

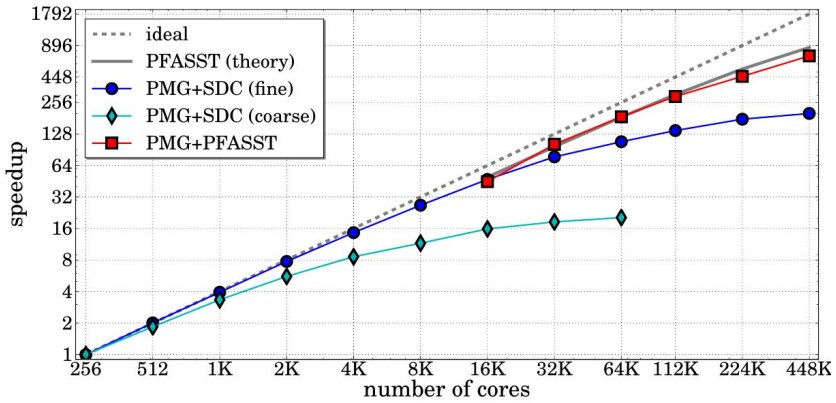
$$v(x) \mapsto \int \kappa(x, y) v(y)\, \mathrm{d}y + W v(v)$$

What PinT promises





[scaling sketch courtesy of Thibaut Lunet]

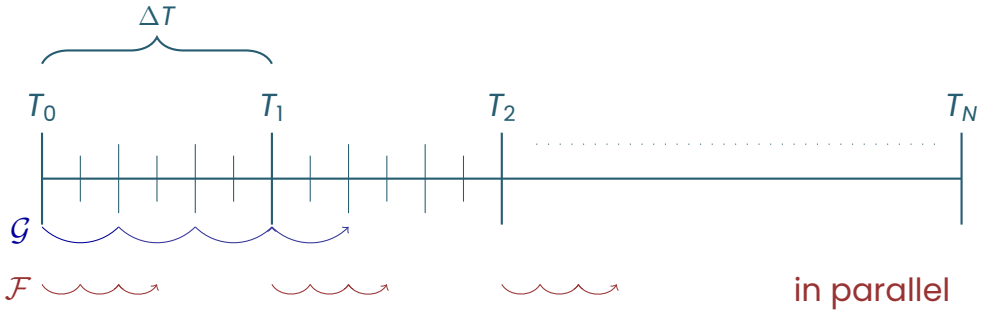


* D. Ruprecht, R. Speck, M. Emmett, M. Bolten, and R. Krause, "*Poster: Extreme-scale space-time parallelism*," in Proceedings of the 2013 Conference on High Performance Computing Networking, Storage and Analysis Companion, ser. SC '13 Companion, Denver, Colorado, USA, 2013.

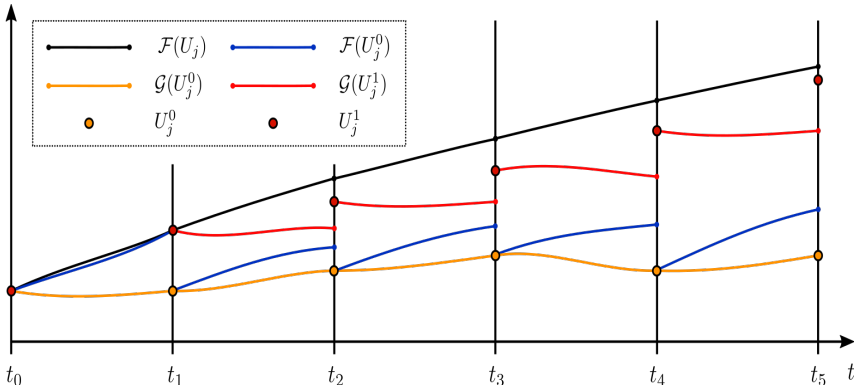
Parareal [Lions et al. '01]

TUHH

Solve IVPs $y_t = f(y)$



in parallel



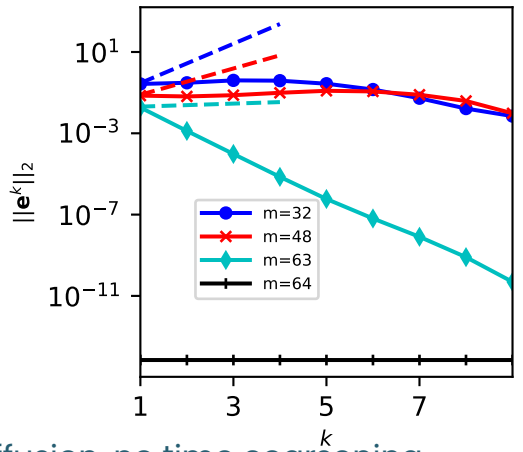
sketch courtesy of Kamran Pentland

In k'th Parareal iteration:

$$Y_n^k = \mathcal{G}(Y_{n-1}^k) + \mathcal{F}(Y_{n-1}^{k-1}) - \mathcal{G}(Y_{n-1}^{k-1}), \quad n = 1, \ldots, N$$

speedup: $S(N_p) \leq \min\left(\frac{N_p}{N_{\text{it}}}, \frac{\text{runtime fine}}{\text{runtime coarse}}\right)$

Aim : Make $\mathcal{G}$ cheaper while keeping K constant



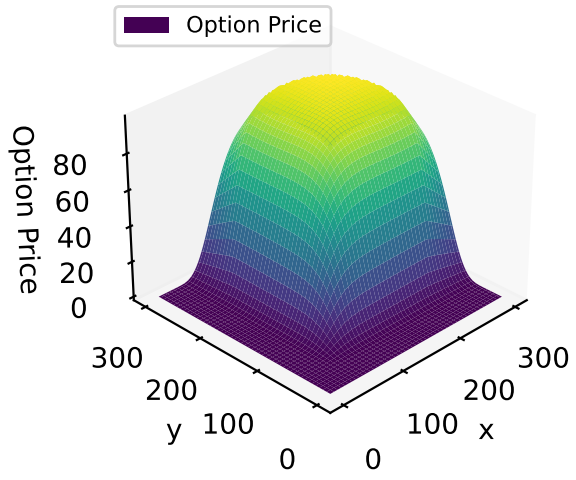
1d adv., no num. diffusion, no time coarsening

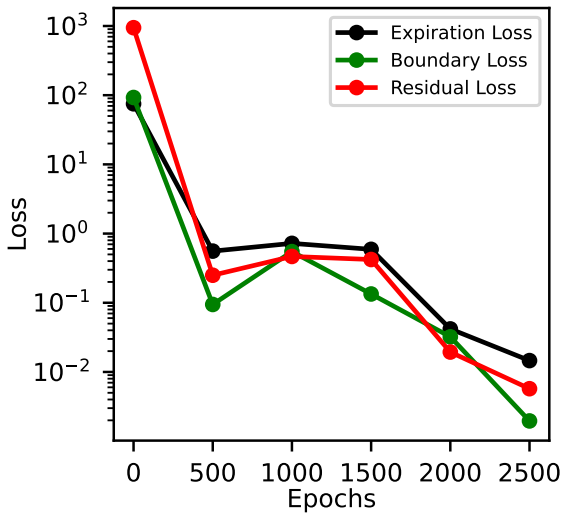
Test problem: Black-Scholes equation

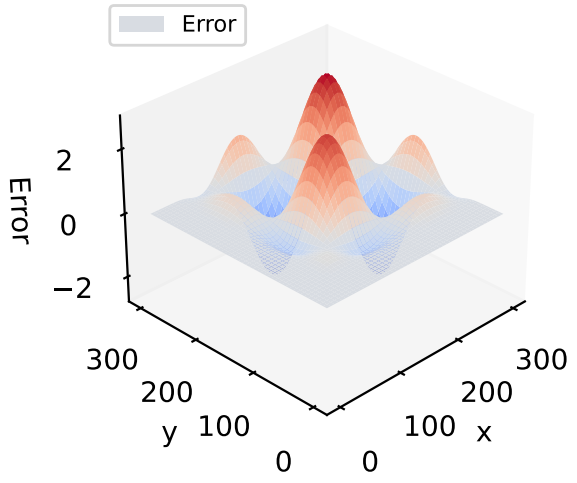
TUHH

$$\frac{\partial u}{\partial t} + \frac{1}{2}\sigma_1^2 x^2 \frac{\partial^2 u}{\partial x^2} + \frac{1}{2}\sigma_2^2 y^2 \frac{\partial^2 u}{\partial y^2}$$
$$+ \rho\sigma_1\sigma_2 xy \frac{\partial^2 u}{\partial x \partial y} + rx\frac{\partial u}{\partial x} + ry\frac{\partial u}{\partial y} - ru = 0$$

+ Dirichlet boundary conditions, terminal condition







Neural operators for Parareal

TUHH

Train ML model to predict $y(t_n) \mapsto y(t_{n+1})$

1. feedforward neural network (fully connected) PINN-P
2. Fourier neural operator PINO

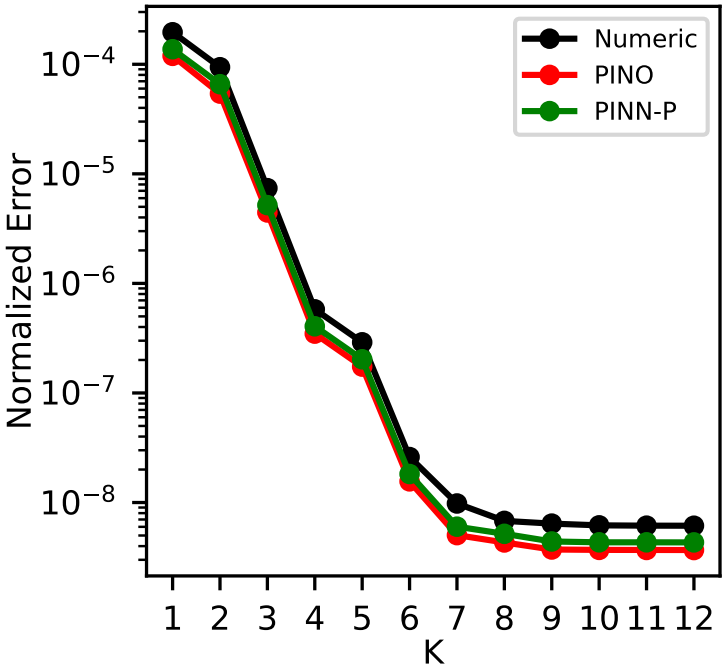
Loss function:

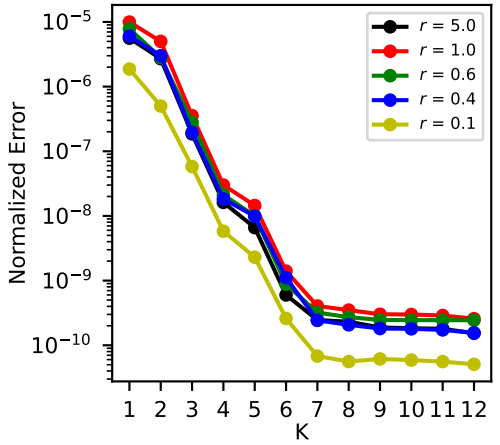
- PDE residual $f(V)$: $\text{MSE}_\text{f} = \frac{1}{N_f} \sum_{i=1}^{N_f} \|f(\tilde{V}(t_i, S_i))\|^2$
- boundary loss term: $\text{MSE}_\text{b} = \frac{1}{N_b} \sum_{i=1}^{N_b} \left\|\tilde{V}(t_i, S_i) - V(t_i, S_i)\right\|^2$

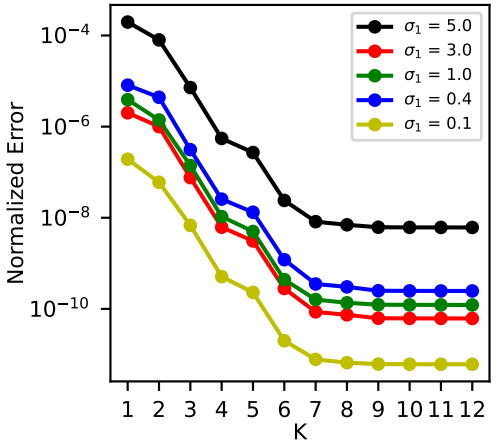
  (better use hard constraints)
- loss at expiration: $\text{MSE}_\text{exp} = \frac{1}{N_\text{exp}} \sum_{i=1}^{N_\text{exp}} \left\|\tilde{V}(T, S_i) - \max(S_i - K, 0)\right\|^2$
- total loss: $\text{MSE}_\text{total} = \text{MSE}_f + \text{MSE}_\text{exp} + \text{MSE}_b$ (no data loss!)

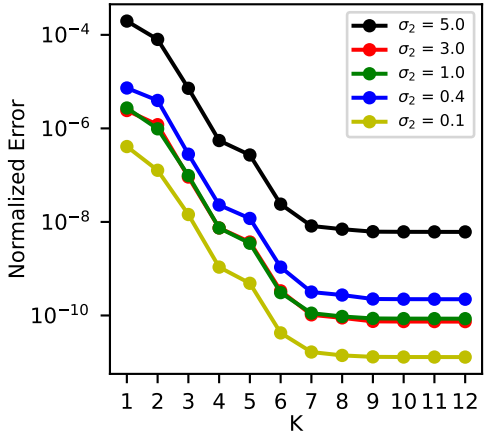
Parareal performance

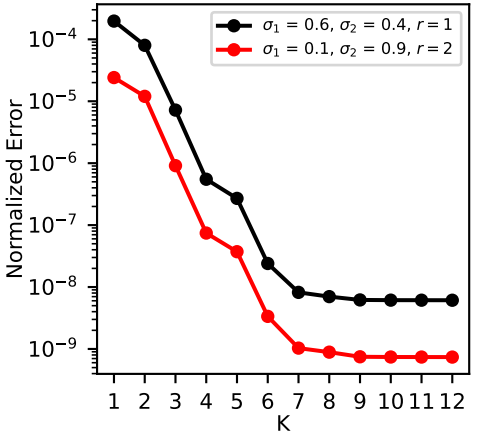
TUHH



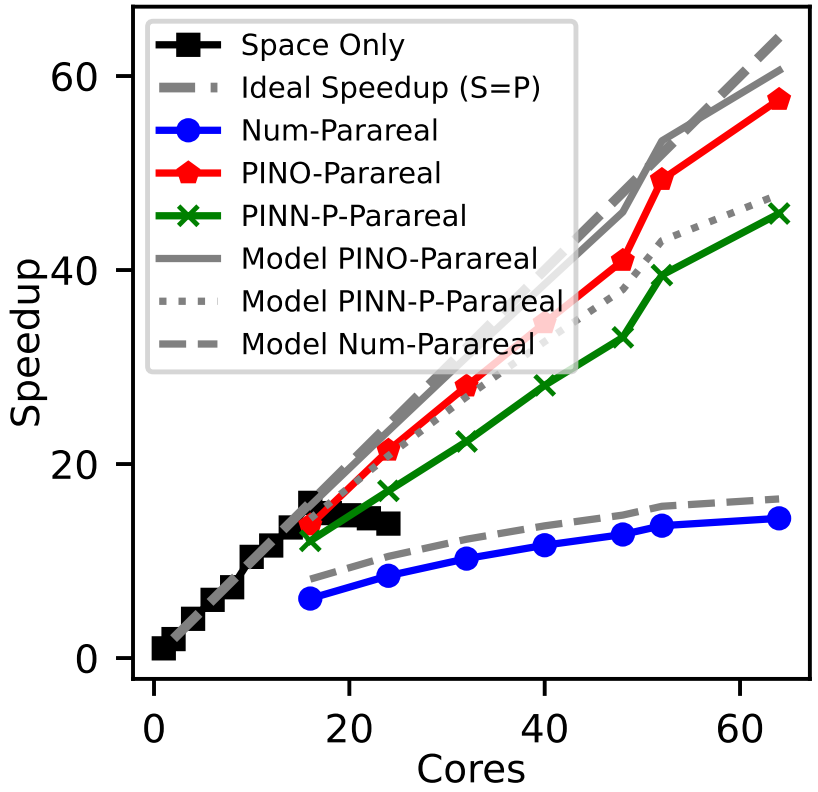


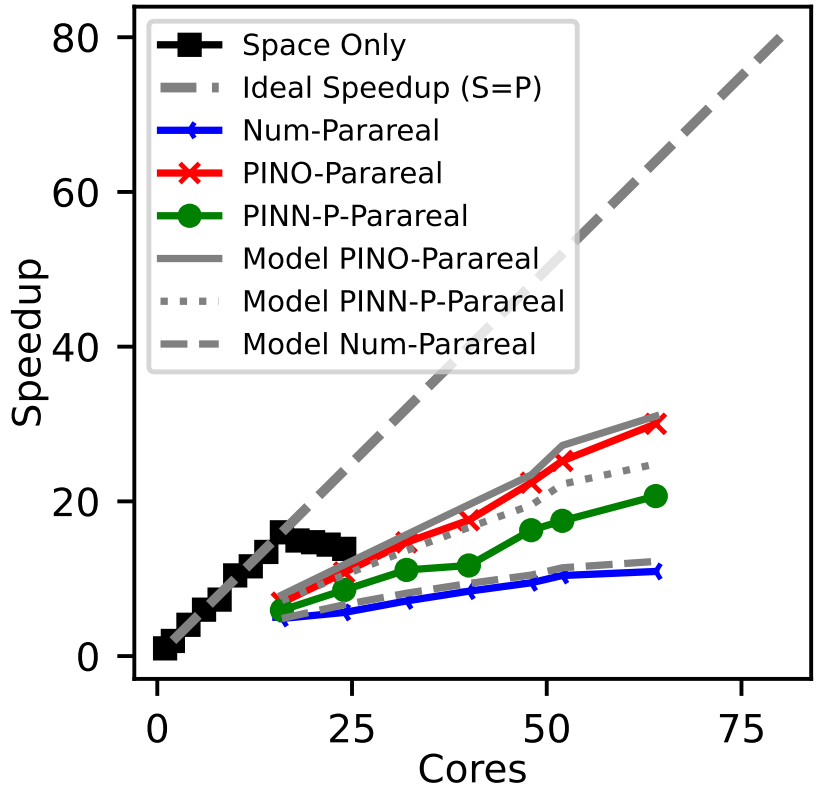






Parareal performance





- Ibrahim, G., Ruprecht: Parareal with a physics-informed neural network as coarse propagator, Parallel Processing. Euro-Par 2023. Lecture Notes in Computer Science, vol 14100. Springer (2023) https://doi.org/10.1007/978-3-031-39698-4_44
- Ibrahim, G., Ruprecht: Space-time parallel scaling of Parareal with a Fourier Neural Operator as coarse propagator, arXiv:2404.02521 (accepted at PASC 2025)

Rayleigh-Bénard Convection

Convective motion in fluids due to temperature gradient

$$\frac{\partial U}{\partial t} + U \cdot \nabla U = -\frac{1}{\rho}\nabla p + \mu \nabla^2 U + g e_z T$$

$$\frac{\partial T}{\partial t} + U \cdot \nabla T = \alpha \nabla^2 T$$

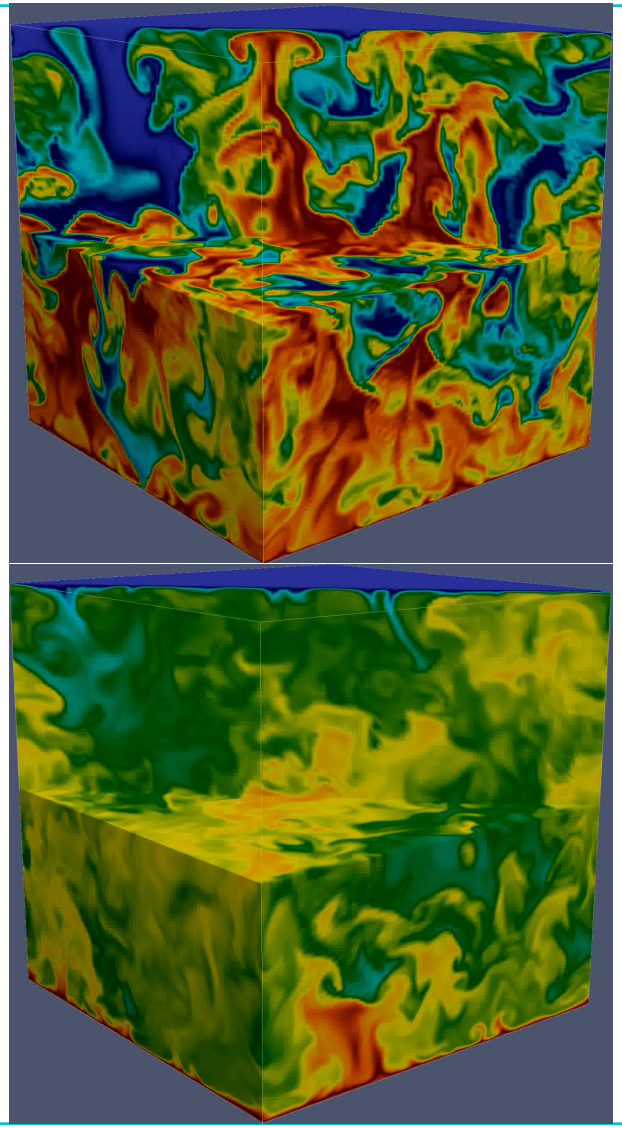
$$\operatorname{div}(U) = 0$$

$$\int_\Omega p \, dx = 0$$

$$\mathrm{Ra} = \frac{\rho \beta \Delta T L^3 g}{\mu \alpha}$$

Parareal: moderate speedup, performance degrades for high Ra

[Clarke et al. '20]



Spectral Deferred Corrections (SDC) [Dutt et al. '00]

Fully implicit RKM/Collocation: dense matrix in Butcher tableau

$$\begin{array}{c|c} c & Q \\ \hline & b^T \end{array} \qquad k_i = f\left(t_n + c_i h, y_n + h\sum_{j=1}^{s} q_{ij} k_j\right) \qquad y_{n+1} = y_n + h\sum_{i=1}^{s} b_i k_i$$

- expensive to solve $Y - \Delta t Q F(Y) = Y_0$

➤ use preconditioned fixed-point iteration

- find suitable preconditioner (lower triangular)
- iterate

$$Y^{k+1} - \Delta t Q_\Delta F(Y^{k+1}) = Y_0 + \Delta t (Q - Q_\Delta) F(Y^k)$$

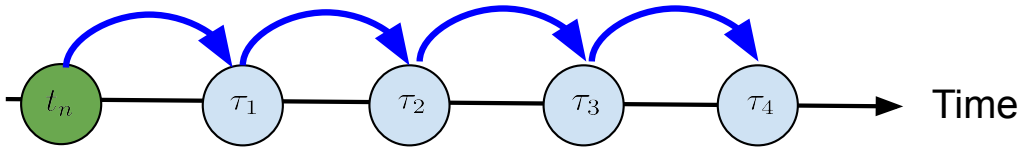
- preconditioner inversion by forward substitution ("sweep")

Preconditioners, e.g., based on Euler

$$Q_\Delta^{\text{EE}} = \begin{bmatrix} 0 & 0 & \dots & 0 & 0 \\ \Delta\tau_2 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \Delta\tau_2 & \Delta\tau_3 & \dots & \Delta\tau_M & 0 \end{bmatrix} \quad Q_\Delta^{\text{IE}} = \begin{bmatrix} \Delta\tau_1 & 0 & \dots & 0 \\ \Delta\tau_1 & \Delta\tau_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \Delta\tau_1 & \Delta\tau_2 & \dots & \Delta\tau_M \end{bmatrix}$$

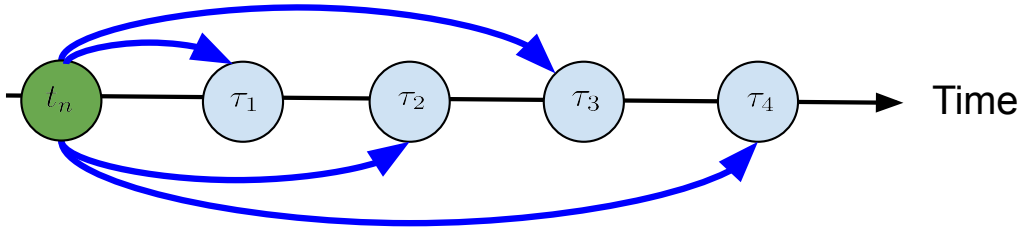
or St. Martin's LU trick [Weiser '14]

$$Q_\Delta^{\text{LU}} = U^T \text{ for } Q^T = LU$$



$$y_m^{k+1} = y_n + \Delta t \sum_{j=1}^{M} q_{m,j} f(y_j^k) + \Delta t \sum_{j=1}^{m} \alpha_j \left(f(\tau_j, y_j^{k+1}) - f(\tau_j, y_j^k) \right)$$

Choose diagonal Q_Δ



$$y_m^{k+1} = y_n + \Delta t \sum_{j=1}^{M} q_{m,j} f(y_j^k) + \Delta t \alpha_m \left(f(\tau_m, y_m^{k+1}) - f(\tau_m, y_m^k) \right)$$

Simple choices, e.g.,

$$Q_\Delta^{\text{Qpar}} = \text{diag}(q_{ii}), \qquad Q_\Delta^{\text{IEpar}} = \text{diag}(\tau_1, \ldots, \tau_m)$$

but: slow convergence [van der Houwen/ Sommeijer 1991, Speck '18]

Alternative: optimize coefficients (e.g., [Speck '18, pySDC])

Parallel SDC – optimize coefficients

TUHH

Dahlquist's test equation:

$$y' = \lambda y,\ y(0) = 1 \quad \lambda \in \mathbb{C}, t \in [0,1]$$

SDC sweep:

$$(I - \Delta t Q_\Delta) y^{k+1} = \Delta t \lambda (Q - Q_\Delta) y^k + y_0$$

error propagation matrix:

$$e^{k+1} = K(z) e^k, \quad K(z) = z(I - zQ_\Delta)^{-1}(Q - Q_\Delta)$$

with $z = \Delta t \lambda$, e^k error to collocation solution

➤ minimize spectral radius $\rho(K)$

non-stiff limit: $K_{\text{NS}} = \lim_{|z| \to 0} \frac{K(z)}{z} = Q - Q_\Delta$

stiff limit: $K_{\text{S}} = \lim_{|z| \to \infty} K(z) = I - Q_\Delta^{-1} Q$

optimization problems difficult to solve (already for moderate M)

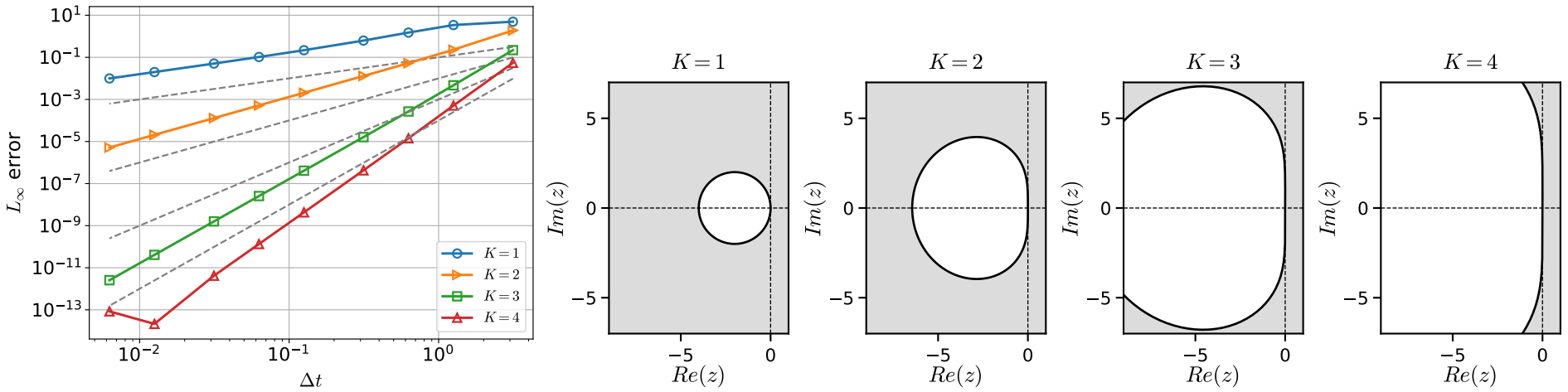
Optimized parallel SDC – non-stiff limit

TUHH

Aim for nilpotency of $K(z)$: $\rho(K_{\text{NS}}) = 0$ for

$$Q_\Delta^{\text{MIN-SR-NS}} = \text{diag}\left(\frac{\tau_1}{M}, \ldots, \frac{\tau_M}{M}\right)$$

➤ each SDC iteration with $Q_\Delta^{\text{MIN-SR-NS}}$ preconditioning removes the respective lowest order term in the error



convergence and stability MIN-SR-NS (4 Radau-II nodes), gray: unstable

Optimized parallel SDC – stiff limit

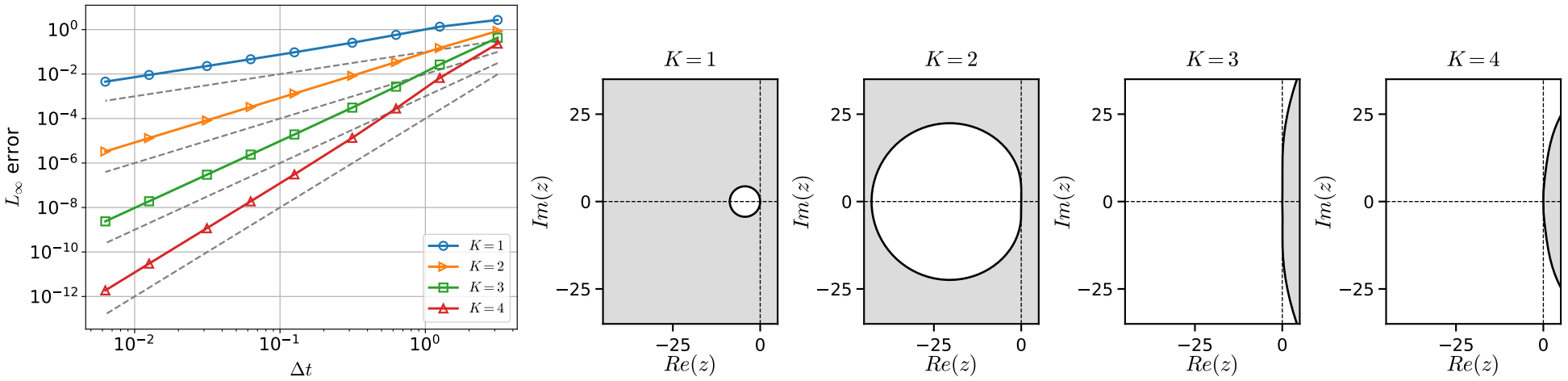
TUHH

$Q_\Delta^{\text{MIN-SR-S}}$ obtained by numerically minimizing

$$\left|\det\left[(1-t)I + tQ_\Delta^{-1}Q\right] - 1\right|, \quad \forall t \in \{\tau_1, \ldots, \tau_M\}$$

obtains a local minimum of $\rho(K_S)$

- but: no guarantee for existence or uniqueness of solutions
- numerical experiments: increasingly ordered coefficients in Q_Δ lead to better stability



convergence and stability MIN-SR-S (4 Radau-II nodes), gray: unstable

Optimized parallel SDC – stiff limit

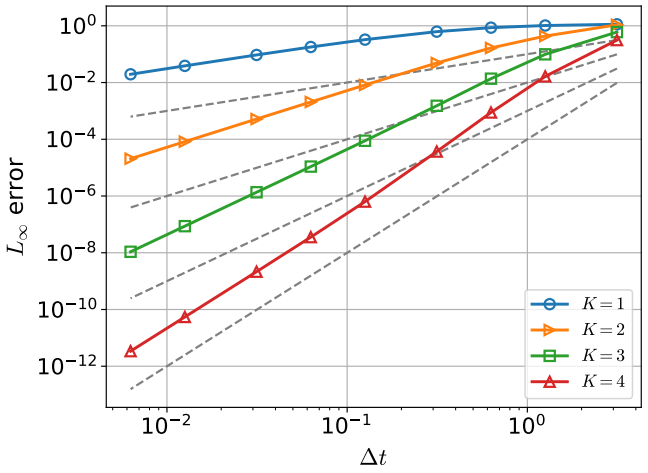
TUHH

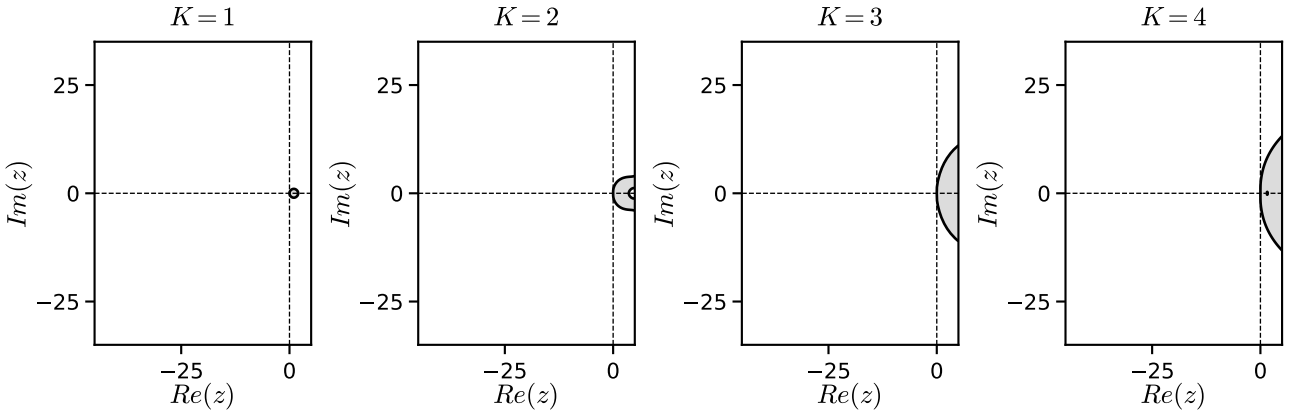
Alternative: iteration-dependent preconditoner $Q_\Delta^{(k)}$
error propagation matrix $K^{(k)}(z) = z(I - zQ_\Delta^{(k)})^{-1}(Q - Q_\Delta^{(k)})$

$$\lim_{|z|\to\infty} e^k = K_S^{(k)} \cdots K_S^{(1)} e^0 \text{ with } K_S^{(k)} = \lim_{|z|\to\infty} K^{(k)}(z) = I - (Q_\Delta^k)^{-1} Q$$

Then:

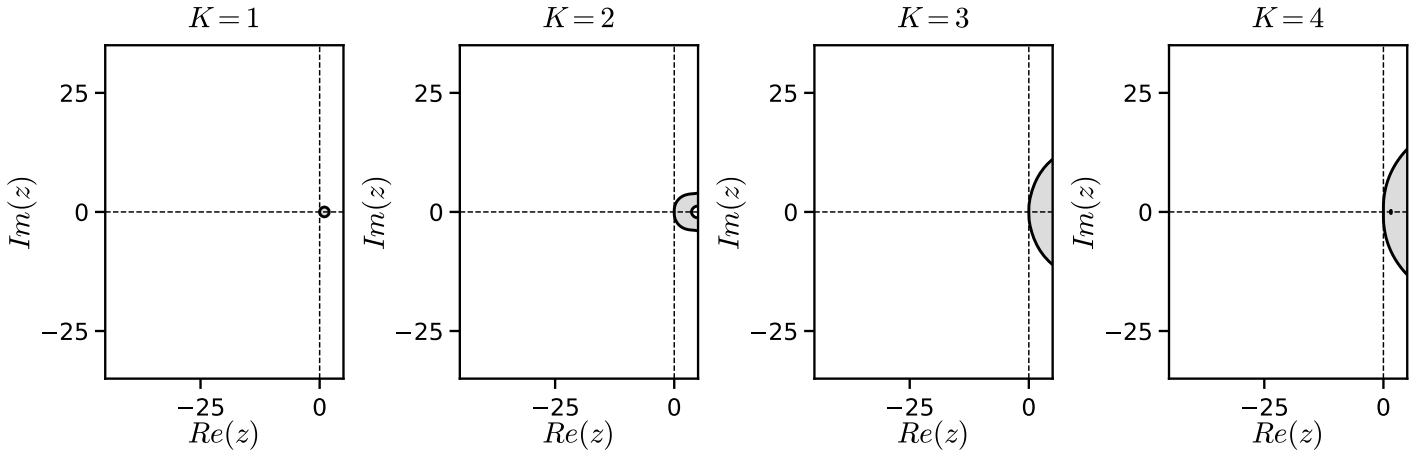
$$Q_\Delta^{\text{MIN-SR-FLEX},(k)} = \text{diag}(\frac{\tau_1}{k}, \ldots, \frac{\tau_M}{k}) \text{ has } K_S^{(M)} \cdot \cdots \cdot K_S^{(1)} = 0$$



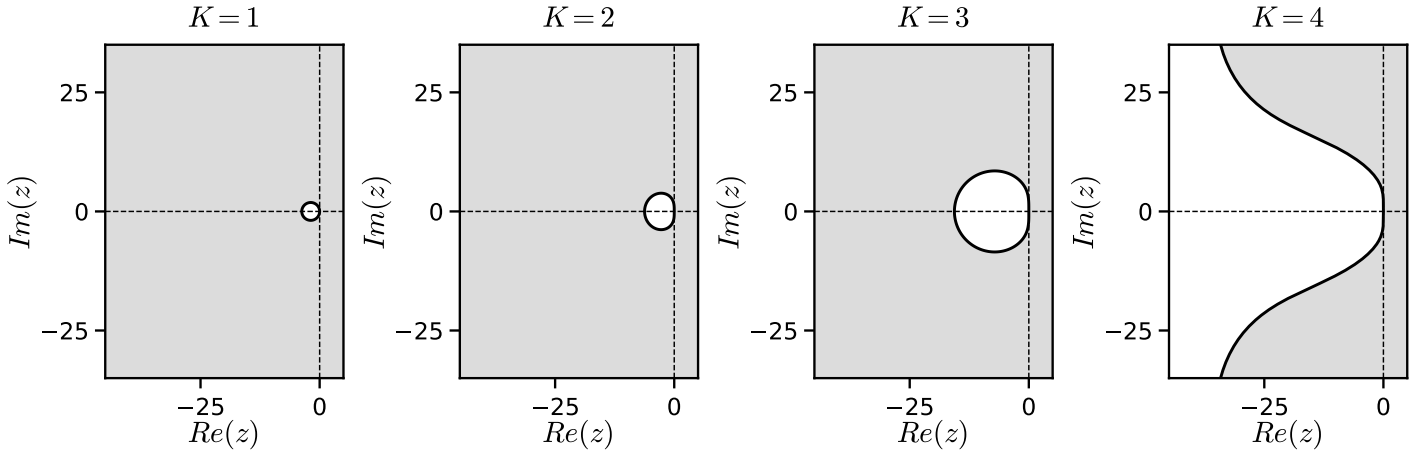


convergence and stability MIN-SR-FLEX (4 Radau-II nodes), gray: unstable

Optimized parallel SDC – stability

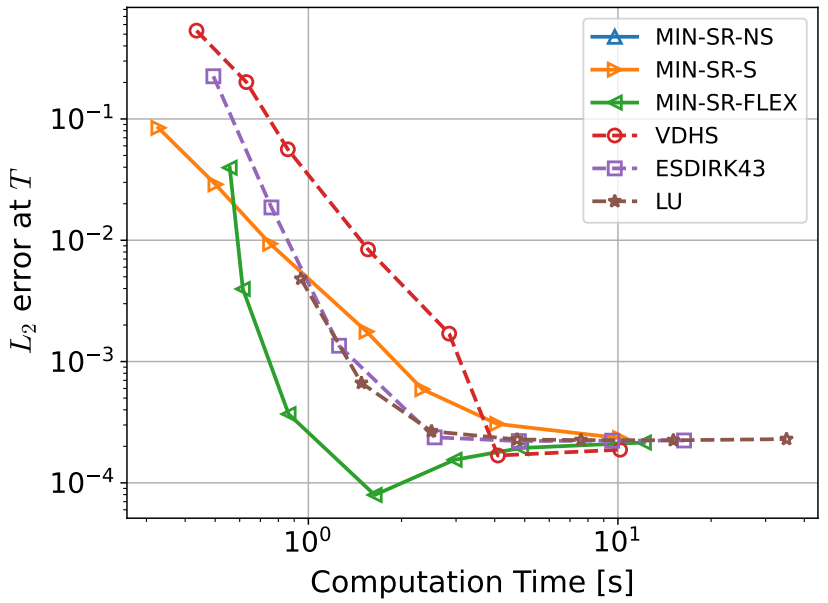


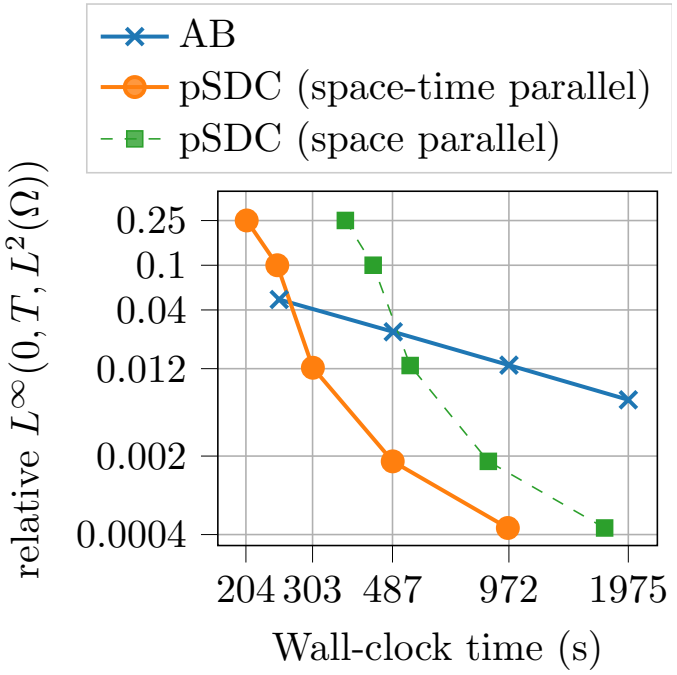
MIN-SR-FLEX (4 Radau-II nodes), similar to LU preconditioner, gray: unstable



[van der Houwen/ Sommeijer 1991] by minimizing $\rho(I - Q_\Delta^{-1}Q)$, gray: unstable

Parallel SDC: Performance





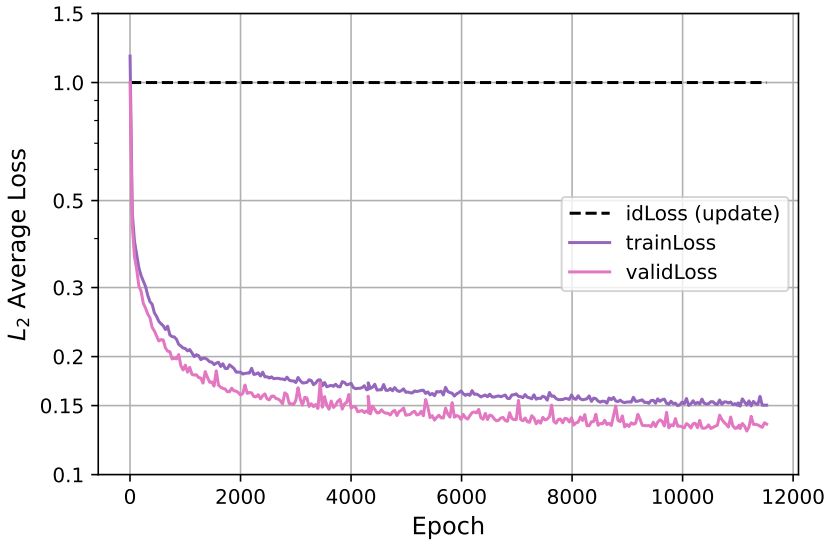
Left: Allen-Cahn equation (pySDC); Right: SWE in ICON-O with MIN-SR-FLEX, 48 cores (4 time, 12 space/48 space)

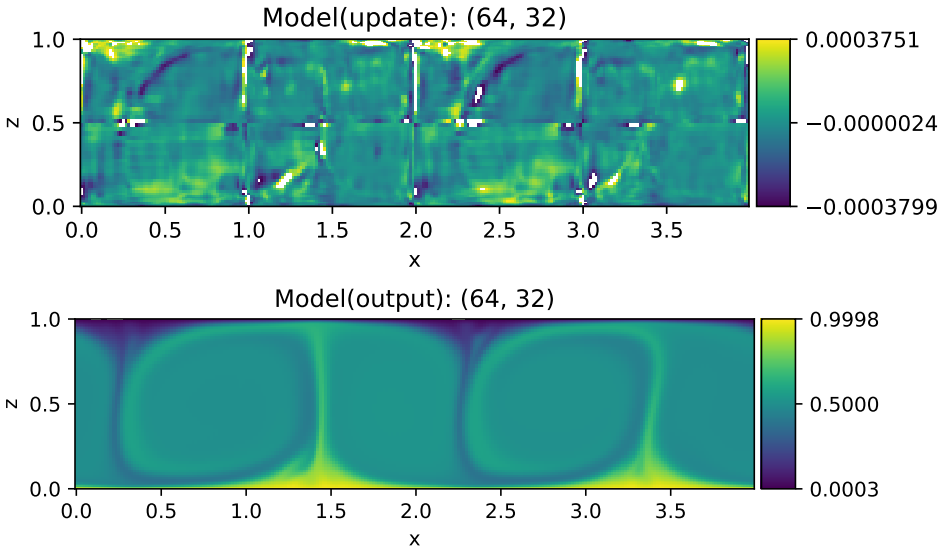
- Čaklović, Lunet, G., Ruprecht: Improving Efficiency of Parallel Across the Method Spectral Deferred Corrections, SIAM J. Sci. Comput. 47(1), 2025
- Freese, G., Lunet, Ruprecht, Schreiber: Parallel performance of shared memory parallel spectral deferred corrections, arXiv 2403.20135, 2024 (under revision in IJHPCA)

FNO for RBC

TUHH

FNO to predict $y(t_n) \mapsto \Delta y = y(t_{n+1}) - y(t_n)$





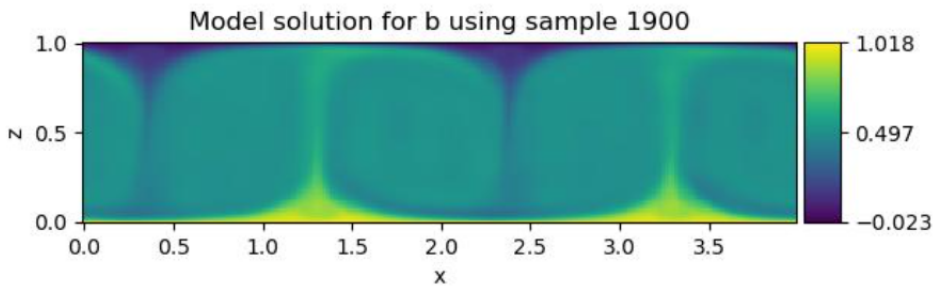
Variable	Δ**t** $= 1e^{-3}$	Δ**t** $= 1e^{-2}$	Δ**t** $= 1e^{-1}$	Δ**t** $= 1e^{0}$
velocity$_x$	2.4e-05	1.8e-03	1.9e-02	1.6e-01
velocity$_z$	3.2e-05	1.8e-03	1.9e-02	1.4e-01
buoyancy	1.8e-05	9.5e-04	1.0e-02	7.5e-02
pressure	8.5e-06	6.5e-04	7.1e-03	6.0e-02
average	**2.1e-05**	1.3e-03	1.4e-02	1.1e-01

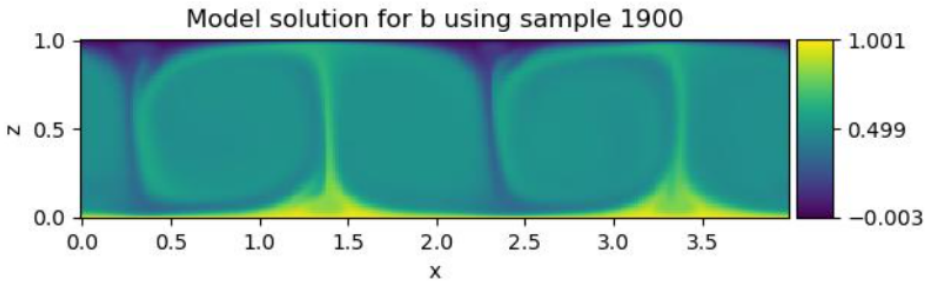
PINO to improve parallel SDC?

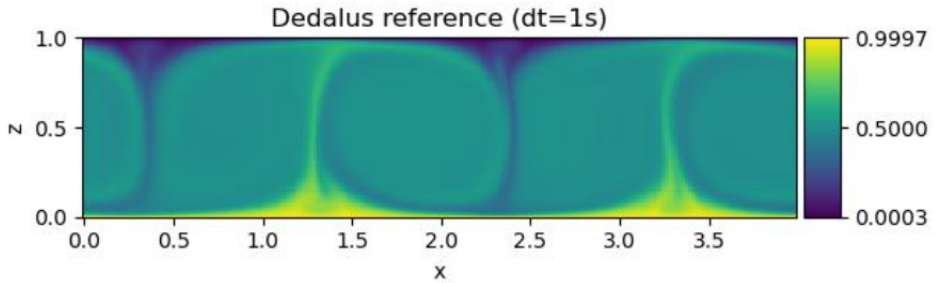
TUHH

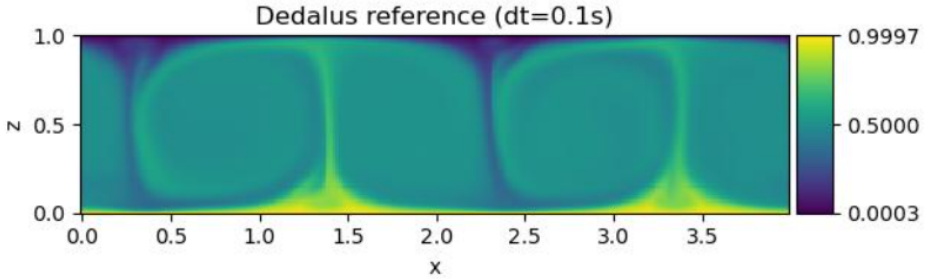
Use a PINO to build an initial guess for the next time-step in SDC:

- potentially very cheap, can be run on GPU
- hopefully more accurate than a simple identity operator (i.e copy)





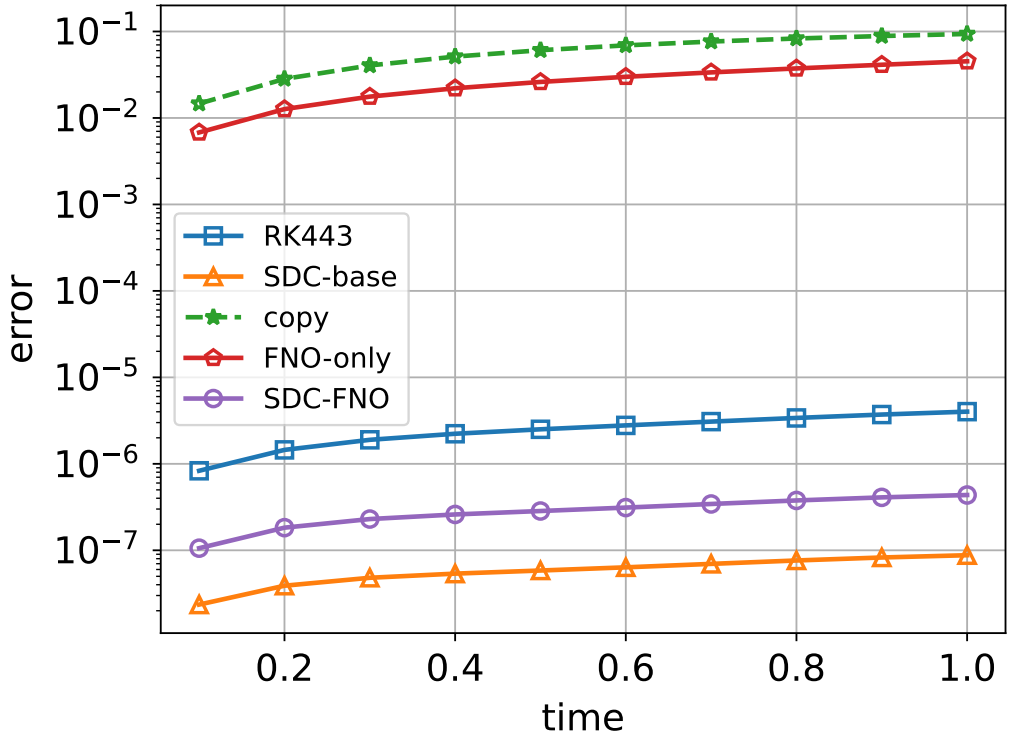


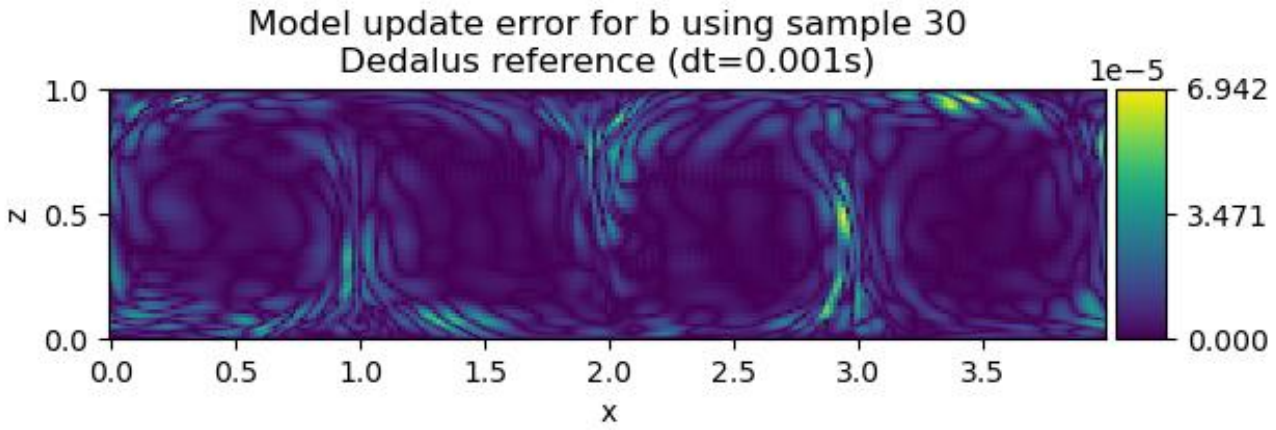


PINO to improve parallel SDC?

TUHH

- $\Omega = [0, 4] \times [0, 1], N_x = 256, N_z = 64$
- $T = [0, 1], \Delta t = 0.01$ (stability limit), 4 nodes pSDC (MIN-SR-FLEX)
- pSDC + `dedalus` (pseudo-spectral solver)

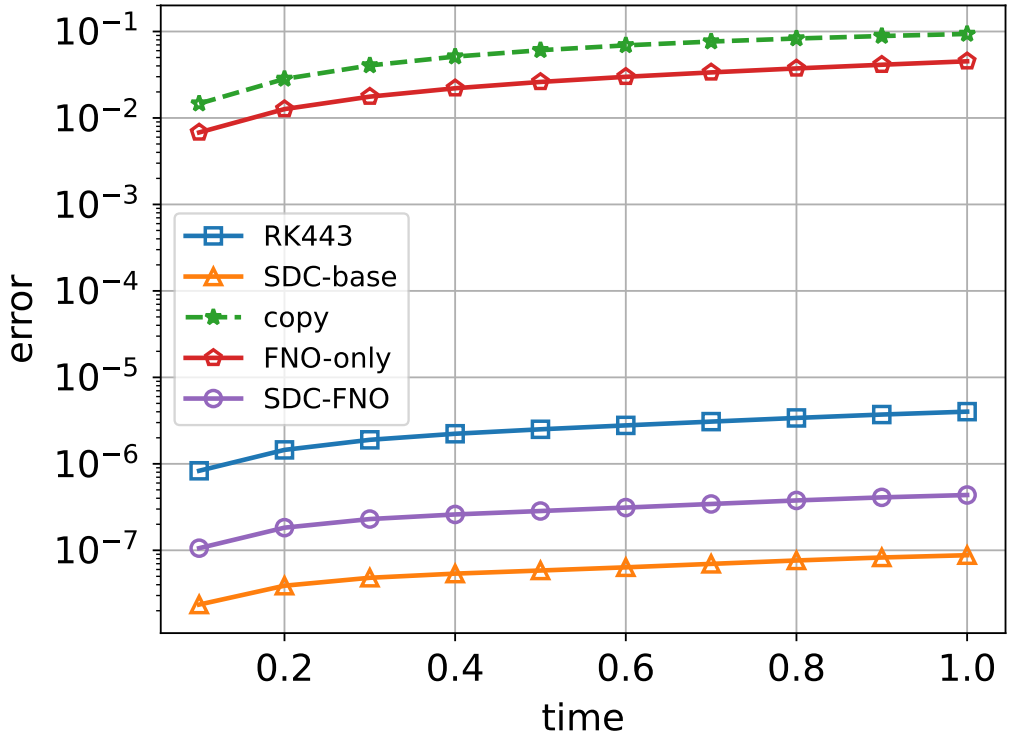


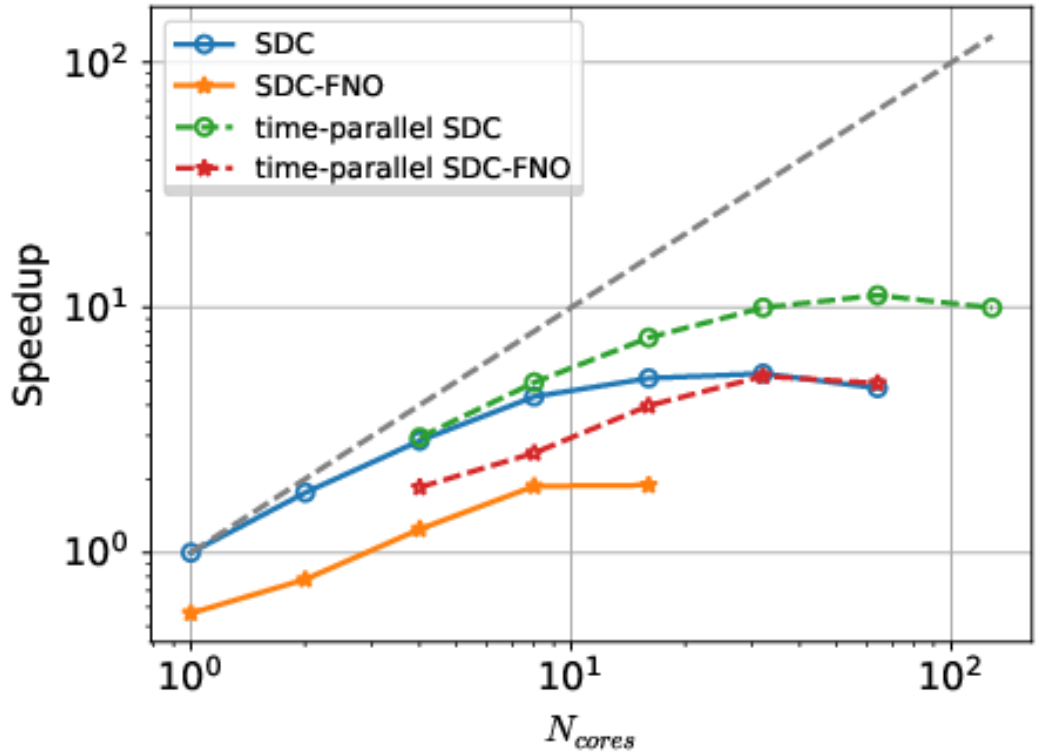


PINO to improve parallel SDC?

TUHH

- $\Omega = [0, 4] \times [0, 1], N_x = 256, N_z = 64$
- $T = [0, 1], \Delta t = 0.01$ (stability limit), 4 nodes pSDC (MIN-SR-FLEX)
- pSDC + `dedalus` (pseudo-spectral solver)





Conclusion

TUHH

- Dynamic deep learning-based superresolution can improve challenging simulations
- Neural operators can make for good coarse propagators in Parareal
- ML models allow leveraging GPU architectures for enhanced inference performance
- …but numerical methods don't always benefit from ML
- Parallel SDC can give good speedups for challenging problems

Thanks!

sebastian.goetschel@tuhh.de