Scalable Parallel-in-Time with Multigrid: Theory, Applications, and Recent Developments

Prof. Jacob B. Schroder
University of New Mexico, Dept. of Mathematics and Statistics
University of Wuppertal, Dept. of Mathematics

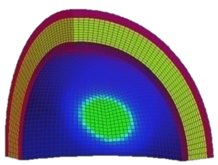
Go20 Conference on Scientific Computing and Software
Gozo, Malta
May 22, 2025

Outline

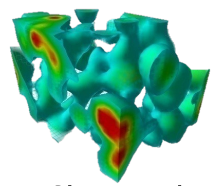
- Introduction to parallel-in-time with multigrid
  - Motivation
  - Multigrid reduction in time (MGRIT) overview
- Description of the XBraid parallel-in-time code
- Application areas
  - Focus: application of XBraid to deep learning
- Conclusions and future work

Multigrid is well suited for extreme scale

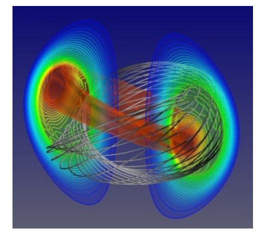
- For many applications, the fastest and most scalable solvers are already multigrid methods



Elasticity / Plasticity



Quantum Chromodynamics



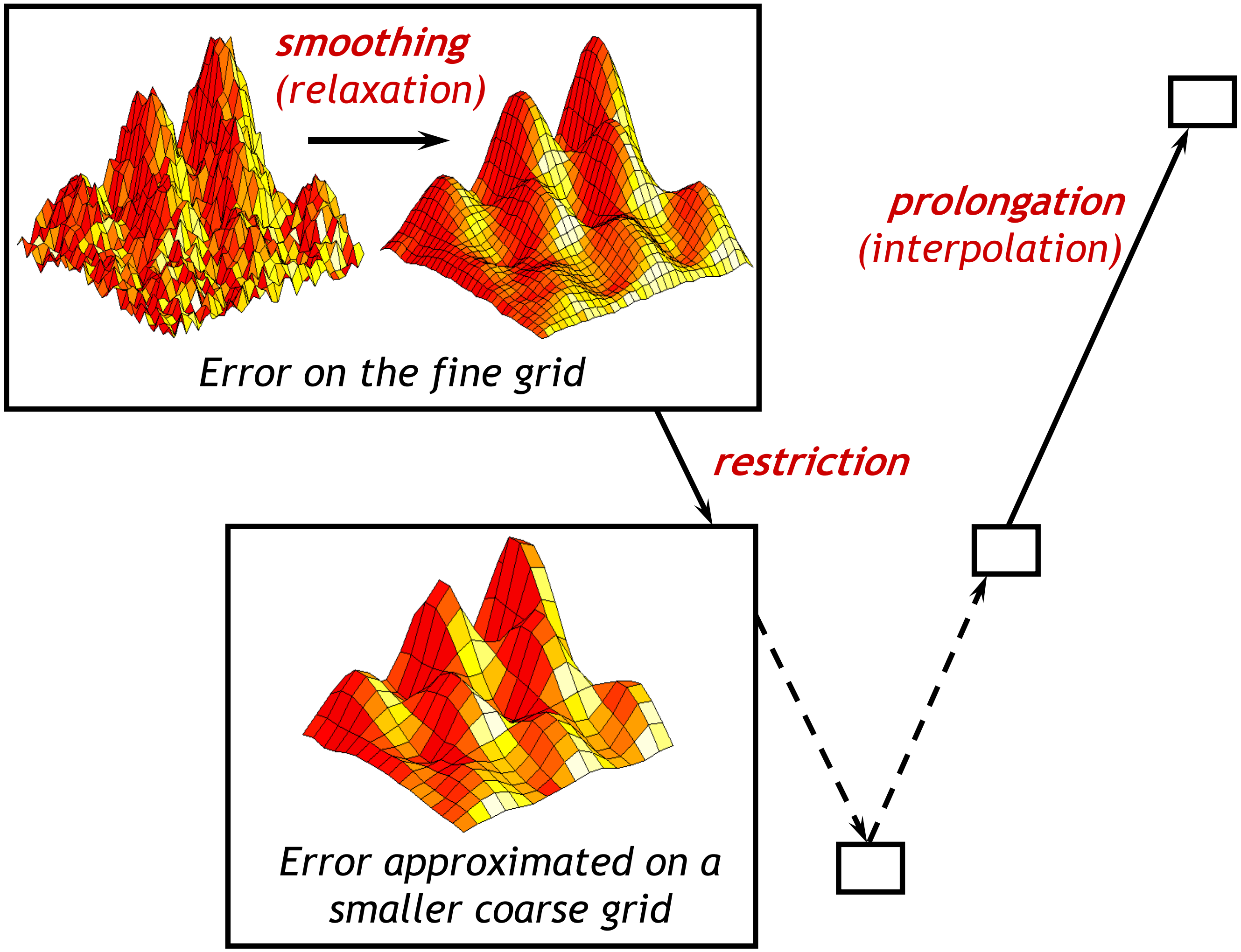
Magnetic Fusion Energy

- Extreme scale solvers need to:
  - Exhibit extreme levels of parallelism (e.g., billion or more cores)
    Spatial multigrid has already scaled to over 1 million cores
  - Minimize data movement
    Multigrid is $O(N)$ optimal
  - Exploit machine heterogeneity
    If the user's problem can exploit heterogeneity, then so can multigrid
  - Be resilient to faults
    Multigrid has already shown good resilience (iterative and multilevel helps)

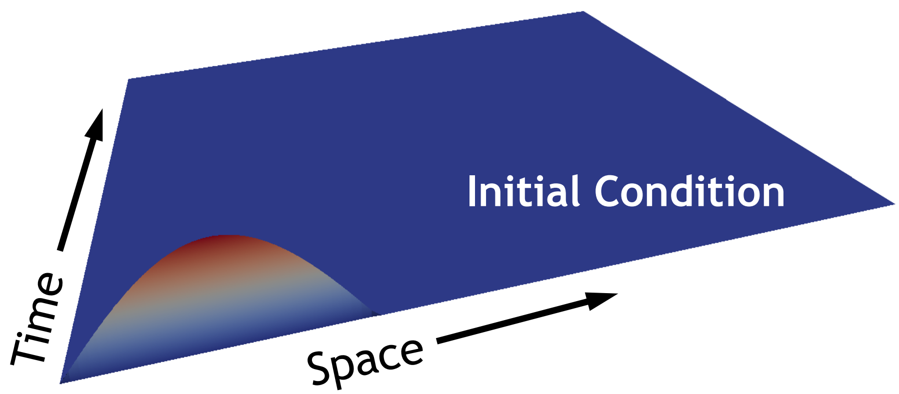
Parallel-in-time approach: Leverage spatial multigrid research

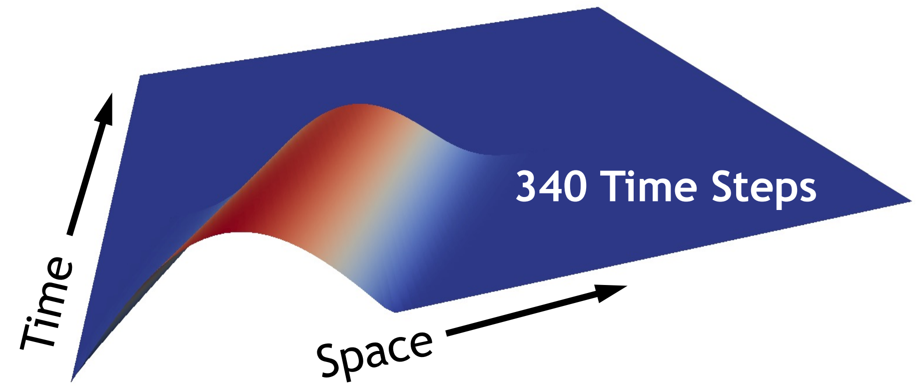
Solve:

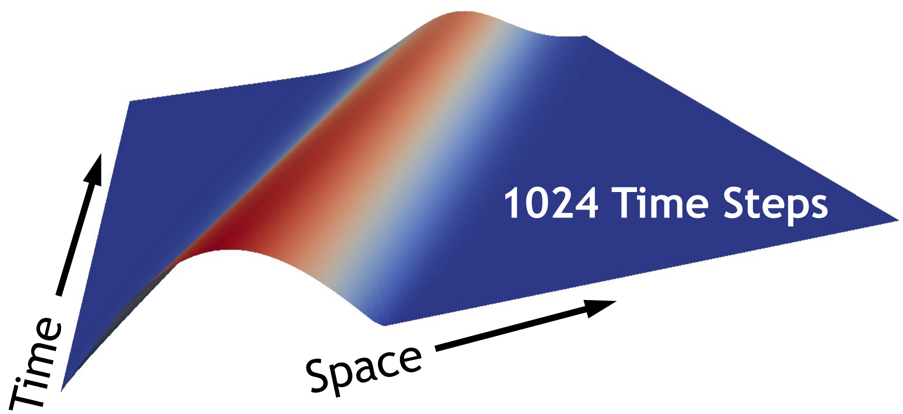
$A(u) = b$



Time stepping is sequential





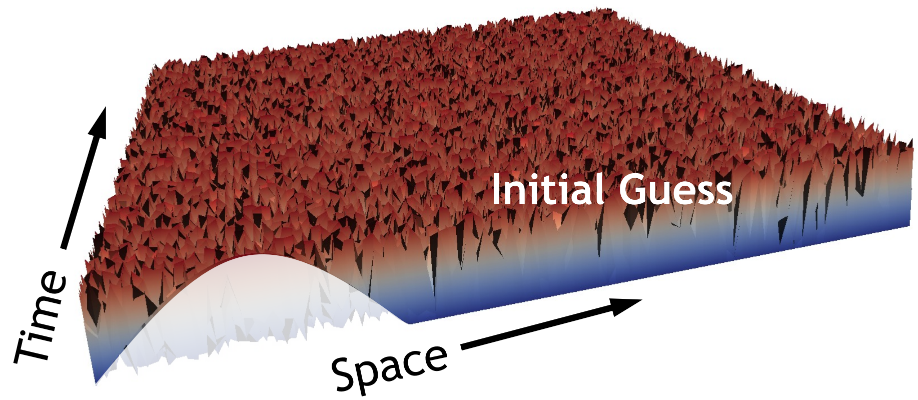


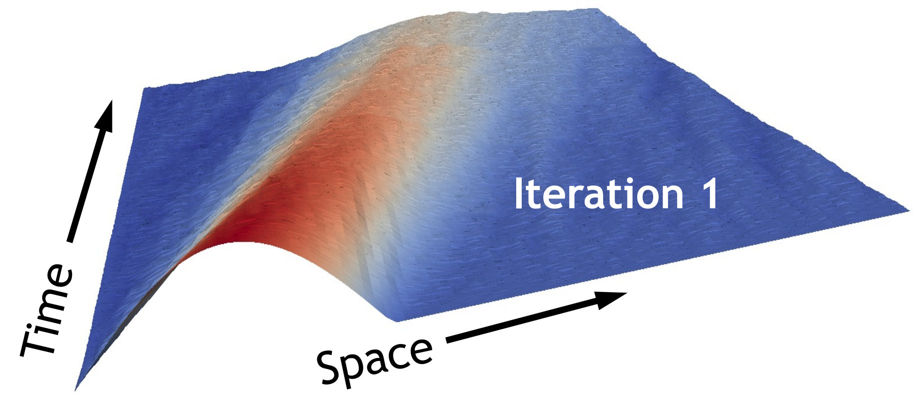
- Advection with 1st-order upwinding

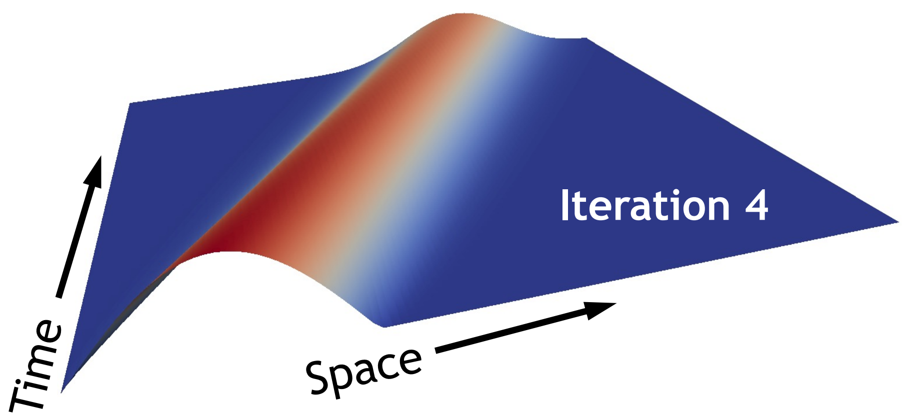
$$u_t = -cu_x$$

- Initial condition sine hump →propagates serially

Multigrid-in-time converges to serial solution in parallel







- Advection with 1st-order upwinding

$$u_t = -cu_x$$

- Initial condition sine hump
- Multilevel → fast data propagation
- Highly parallel, large speedups[1,2]
- Algorithmic research needed!

1. Gander, *50 Years of Time Parallel Time Integration*. Springer, 2015.
2. Ong, Schroder, *Applications of Time Parallelization*. CVS, Springer, 2020.

Technical approach

- Consider the **general** one-step method

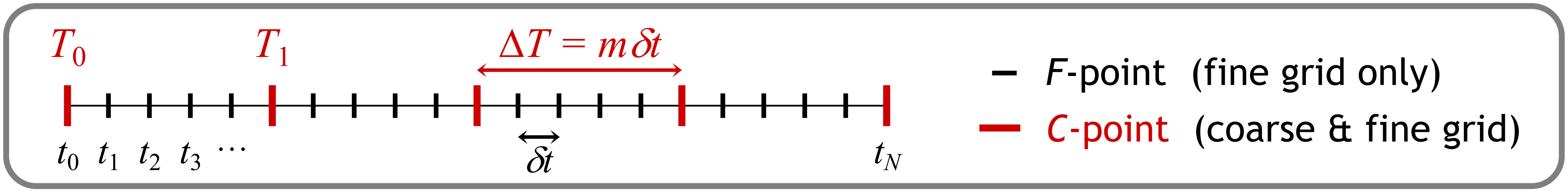
$$\boldsymbol{u}_i = \Phi_i(\boldsymbol{u}_{i-1}) + \boldsymbol{g}_i, \quad i = 1, 2, ..., N$$

- In the linear setting *(for simplicity)*, time marching $\equiv$ forward solve
  - This is an *O(N)* direct method, **but sequential**

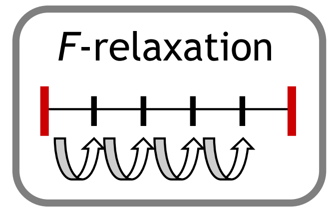
$$A\mathbf{u} \equiv \begin{pmatrix} I & & & \\ -\Phi & I & & \\ & \ddots & \ddots & \\ & & -\Phi & I \end{pmatrix} \begin{pmatrix} \boldsymbol{u}_0 \\ \boldsymbol{u}_1 \\ \vdots \\ \boldsymbol{u}_N \end{pmatrix} = \begin{pmatrix} \boldsymbol{g}_0 \\ \boldsymbol{g}_1 \\ \vdots \\ \boldsymbol{g}_N \end{pmatrix} \equiv \mathbf{g}$$

- Instead solve this system **iteratively** with a multigrid method
  - Extend multigrid reduction (MGR, 1979) to the time dimension
  - *O(N)*, highly parallel

Multigrid reduction in time (MGRIT)[1]



- Relaxation alternates between F- and C-points
  - F-relaxation is integration over coarse intervals (block Jacobi)



- Coarse system is a time rediscretization with fewer time-points
  - Main research question: approximate impractical Φ^m with Φ_Δ

$$A_\Delta = \begin{pmatrix} I & & & \\ -\Phi^m & I & & \\ & \ddots & \ddots & \\ & & -\Phi^m & I \end{pmatrix} \quad \Rightarrow \quad B_\Delta = \begin{pmatrix} I & & & \\ -\Phi_\Delta & I & & \\ & \ddots & \ddots & \\ & & -\Phi_\Delta & I \end{pmatrix}$$

  - Non-intrusive: time discretization unchanged, user only provides Φ

1. Falgout, Friedhoff, Kolev, MacLachlan, Schroder, *Parallel Time Integration with Multigrid*, SISC, 2014.

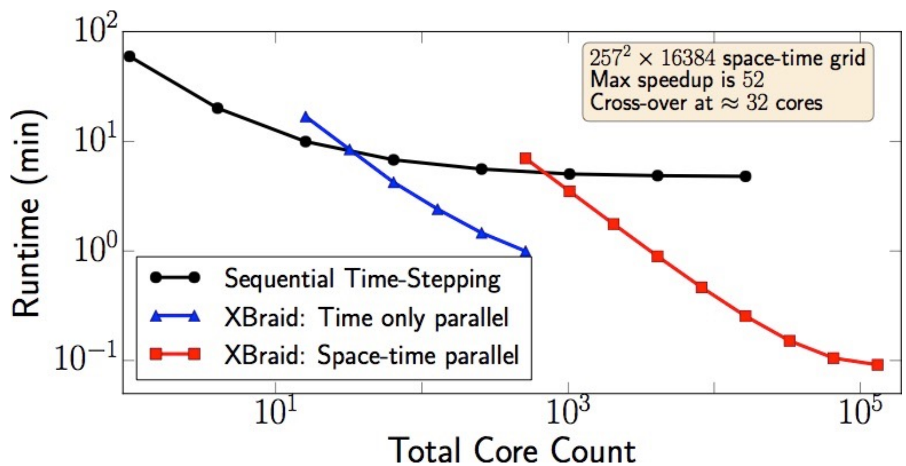
Broad summary of MGRIT

- Expose **concurrency** in the time dimension with multigrid
- **Non-intrusive**, with unchanged fine-grid problem
- Converges to **same solution** as sequential marching

$$\begin{pmatrix} I & & & \\ -\Phi & I & & \\ & \ddots & \ddots & \\ & & -\Phi & I \end{pmatrix}$$

- Optimal for variety of parabolic problems[1,2]
- Extends to **nonlinear** problems with FAS formulation
- In simple two-level setting, MGRIT $\equiv$ Parareal
- Sharp two-level[2] and multi-level[3] convergence theory

- Large speedups available, but in a new way
  - Time stepping is already O(N)
  - Useful only beyond a crossover
  - More time steps → more speedup potential



1. Falgout, Friedhoff, Kolev, MacLachlan, Schroder, *Parallel Time Integration with Multigrid*, SISC, 2014.
2. Dobrev, Kolev, Petersson, Schroder, *Two-level Convergence Theory for MGRIT*, SISC, 2017.
3. Hessenthaler, Southworth, Nordsletten, Rohrle, Falgout, Schroder, *Multilevel convergence analysis of MGRIT*, SISC, 2020.

Outline

- Introduction to parallel-in-time with multigrid
  - Motivation
  - Multigrid reduction in time (MGRIT) overview

- Description of the XBraid parallel-in-time code

- Application areas
  - Focus: application of XBraid to deep learning

- Conclusions and future work

XBraid: Open source, non-intrusive, and flexible

- User also writes several wrapper routines:
  - *Step*, *Init*, *Clone*, *Sum*, *SpatialNorm*, *Access*, *BufPack*, *BufUnpack*
  - *Coarsen*, *Refine* (optional, for spatial coarsening)

- Example: *Step*(***u, status***)
  - Advance vector *u* from time *tstart* to *tstop*

- Code stores only *C*-points to minimize storage
  - Memory multiplier per processor:
    ~*O(log N)* with time coarsening, *O(1)* with space-time coarsening

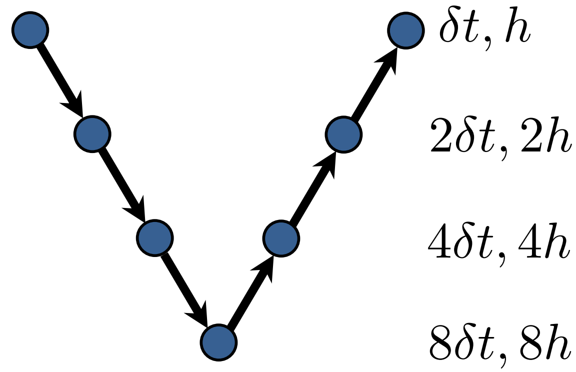
- Processes time-intervals to overlap communication and computation

- XBraid supports C, C++, Fortran, and Python interfaces

- XBraid has been integrated with various DOE codes
  (SunDials, GridDyn, Tycho2, and TorchBraid)
  https://github.com/XBraid/xbraid

XBraid Features

- Space-time coarsening speeds up computations and greatly reduces memory use[1]
  - User writes optional *Coarsen()* & *Refine()*
  - Works well for implicit parabolic



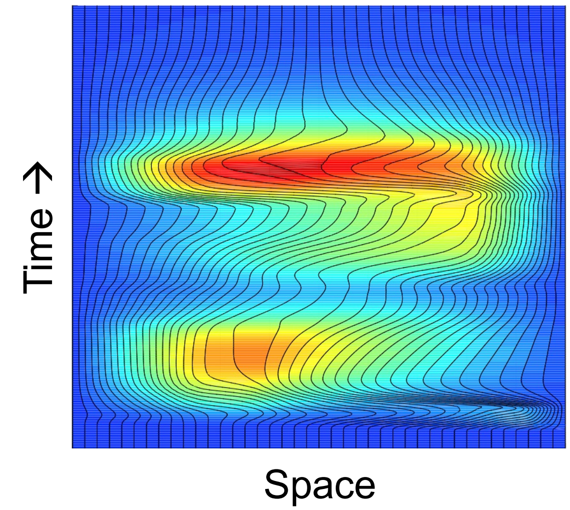
- Multi-step methods[2]
  - Convert to one-step method by grouping unknowns

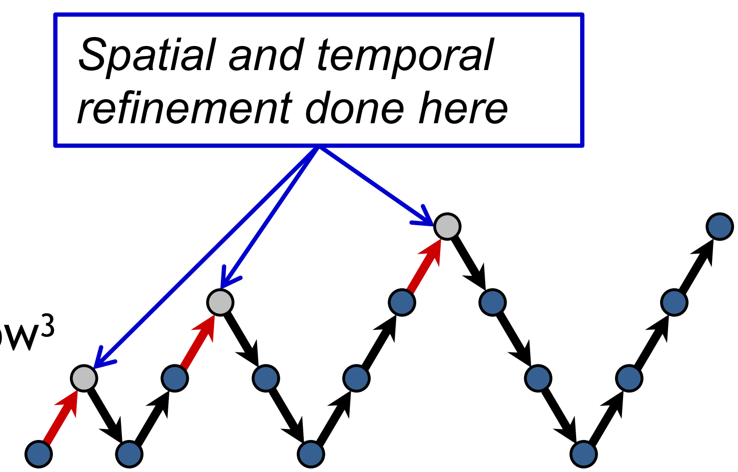
- Free Richardson-based extrapolation error estimator / correction[3]
  - Based on coarse time grid

1. Falgout, Manteuffel, O'Neill, S., *MGRIT for Nonlinear Parabolic Problems*, SISC, 2017.
2. Falgout, Lecouvez, Woodward, *A parallel-in-time algorithm for variable step multistep methods*, J. Comp. Sci., 2019.
3. Falgout, Manteuffel, O'Neill, Schroder, *Multigrid reduction in time with Richardson extrapolation*, ETNA, 2021.

XBraid Features: Adaptivity in time and space

- Moving spatial mesh[1]
  - 1D diffusion with time dependent source
  - Unsteady flow around moving cylinder
- Temporal refinement via Full Multigrid (FMG)
  - DAE power grid simulations in GridDyn[2] (52x speedup)
- Temporal and spatial refinement via FMG
  - 2D heat equation with FOSLS (6x speedup)
  - 13x speedup with space-time AMR for Couette Flow[3]
- Temporal load balancing under development





1. Falgout, Manteuffel, Schroder, Southworth, *Parallel-in-Time for Moving Meshes*, 2016. LLNL-TR-681918.
2. Schroder, Lecouvez, Falgout, Woodward, Top, *Parallel-in-Time Solution of Power Systems with Scheduled Events*, PES IEEE, 2018.
3. Christopher, Gao, Guzik, Falgout, Schroder, *Space-Time Parallel Alg. with Adaptive Mesh Refinement for CFD*, CVS Springer, 2020.

Outline

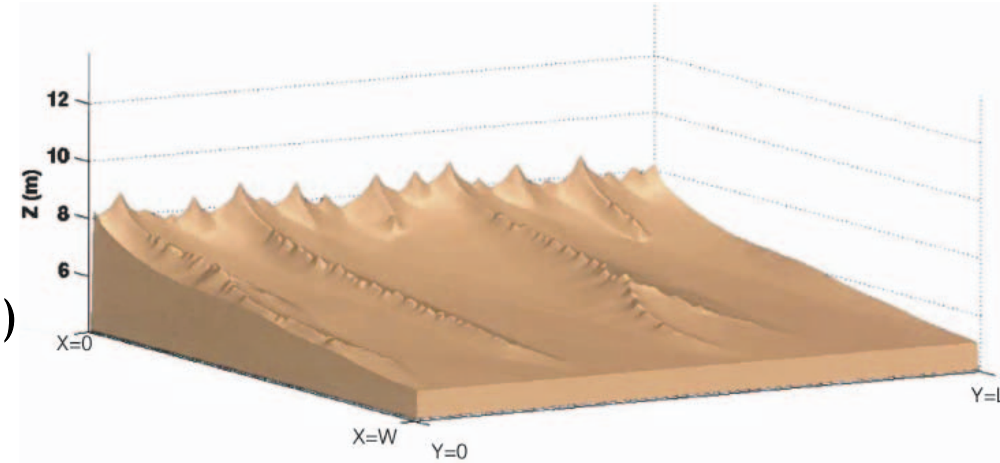
- Introduction to parallel-in-time with multigrid
  - Motivation
  - Multigrid reduction in time (MGRIT) overview
- Description of the XBraid parallel-in-time code
- Application areas
  - Focus: application of XBraid to deep learning
- Conclusions and future work

Experiments coupling our code XBraid with various application research codes

- Navier-Stokes (compressible and incompressible), Shallow Water
  - Strand2D, CarT3D, Cyclops, Chord
- Heat equation (including moving mesh example)
  - MFEM, hypre
- Elasticity (e.g., cardiac modeling)
  - CHeart
- Nonlinear diffusion, the *p*-Laplacian
  - MFEM
- Power-grid simulations
  - GridDyn+SunDials
- Explicit time-stepping coupled with space-time coarsening
  - Advection, Burger's Equation
  - MFEM
- Optimization (XBraid-adjoint), Machine Learning
  - CoDiPack, PyTorch (TorchBraid)

The p-Laplacian: nonlinear diffusion

- Solve $u_t = \nabla \cdot \left(|\nabla u|^{p-2} \nabla u \right)$
- 2D linear finite elements
  - 16K x 20K space-time problem
  - Backward Euler (Newton's method)
- Parallel results[1]
  - Crossover at ~40 processors in time
  - Speedup of 18x at 130K cores



Surface Erosion[2]

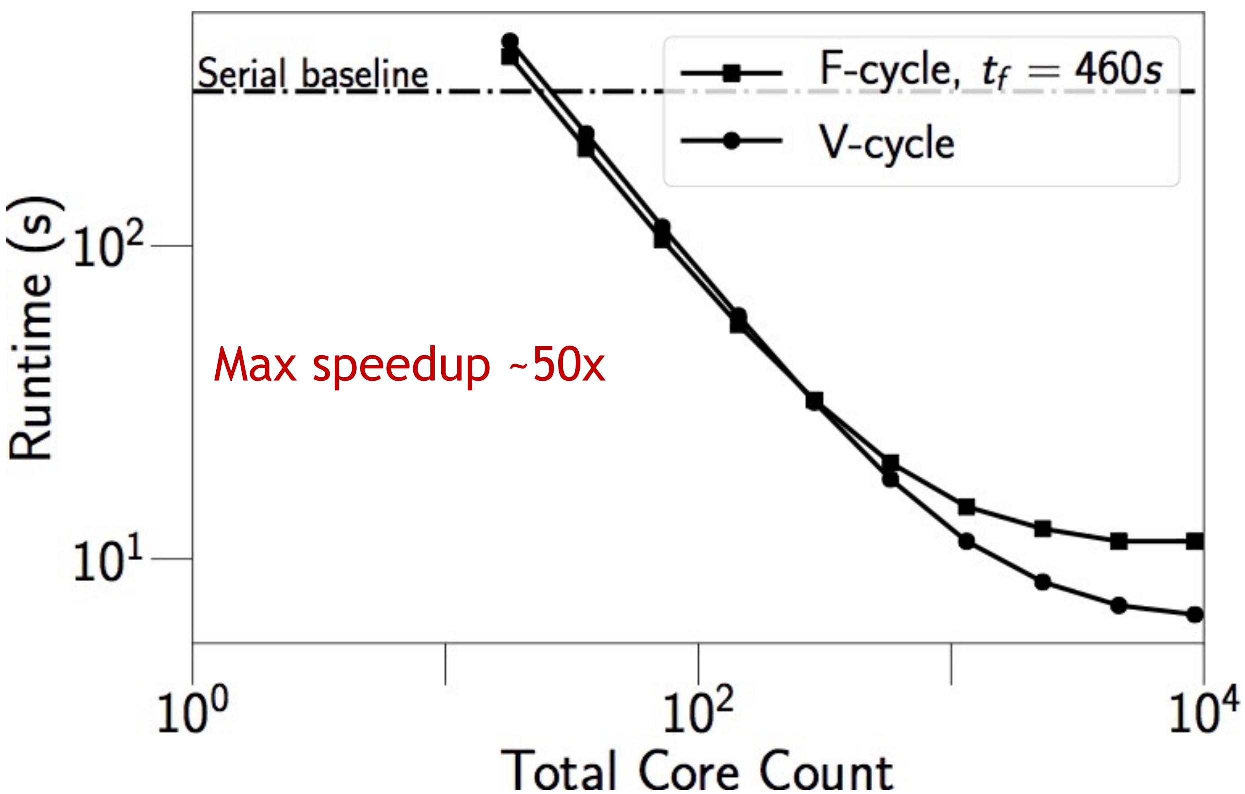
- Important parameters for performance
  - Full storage and space-time coarsening
  - Adjusting the Newton tolerance for the early iterations

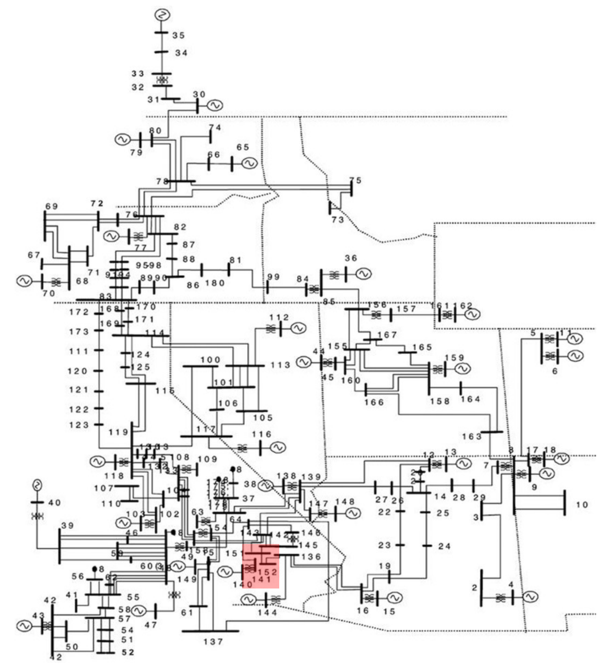
➔ In general, parallel-in-time works well for parabolic problems

1. Falgout, Manteuffel, O'Neill, S., *MGRIT for Nonlinear Parabolic Problems*, SISC, 2017.
2. Image courtesy of Birnir, Rowlett. *Mathematical Models for Erosion and the optimal Transportation of Sediment*. Int. J. Nonlinear Sci. Numer. Simul. 2013

Powergrid (DAE)

- Discontinuous square pulse applied to bus 141 every second[1]
  - Must handle discontinuities (events) for real-world relevance
  - Explore scalability w.r.t. number of discontinuities, 460s simulation has 460 events
  - Adaptively refine in time around discontinuities for improved accuracy





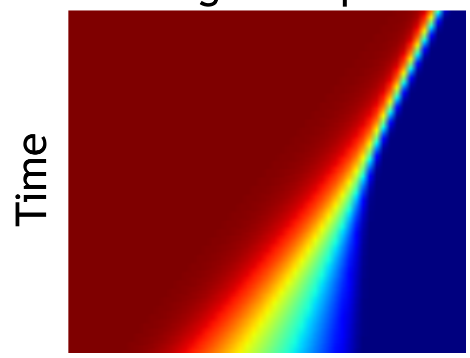
WECC System: 179 buses and 793 unknowns

1. Schroder, Lecouvez, Falgout, Woodward, Top, *Parallel-in-Time Solution of Power Systems with Scheduled Events*, PES IEEE, 2018.

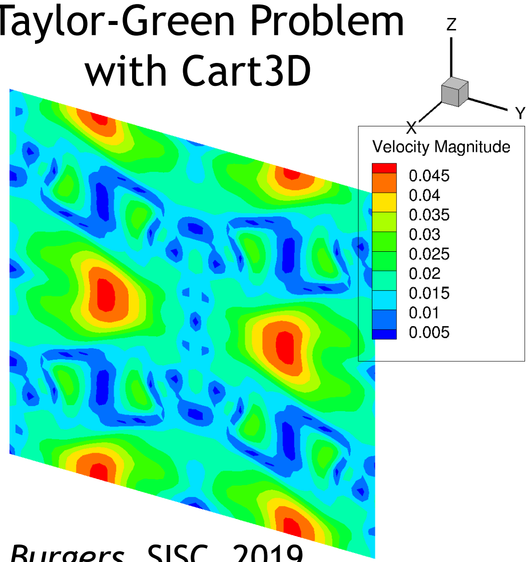
Hyperbolic problems and fluid dynamics

- Pure hyperbolic problems need significant artificial dissipation for **standard** MGRIT, parareal and other methods to work well
  - Expensive cycling (F-cycles) and expensive relaxation (FCF / FCFCF) also needed[1]
- For **modest** Reynolds numbers, success is more easily attained for CFD
- Navier-Stokes in 2D and 3D[2,3,4]
  - Multiple codes: Strand2D, Cart3D, CHeart, Chord
  - Compressible and incompressible

1D Burgers' Equation



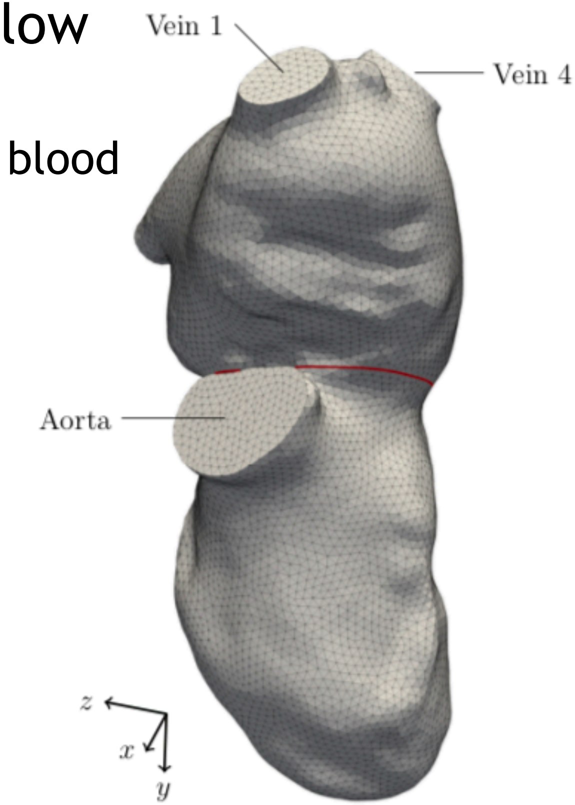
Taylor-Green Problem with Cart3D



1. De Sterck, Howse, Schroder, et al., *Parallel-in-Time MG with Adaptive Coarsening for Inviscid Burgers*, SISC, 2019.
2. Falgout, Katz, Kolev, Schroder, Wissink, Yang, *Parallel Time Integration with MGRIT for Compressible Fluid Dyn.*, 2014.
3. Christopher, Gao, Guzik, Falgout, Schroder, *Space-Time Parallel Alg. with Adaptive Mesh Refinement for CFD*, CVS Springer, 2020.
4. Christopher, Gao, Guzik, Falgout, Schroder, *Fully Parallelized Space-Time Adaptive Meshes for the Compressible Navier-Stokes Equations Using MGRIT, CVS Springer, 2020.*

Periodic fluid-structure interaction (FSI)

- Goal: speedup biomedical simulations, e.g., blood flow
  - Example problem: Periodic nonlinear flow in left ventricle
  - Equations: elasticity for solid deformations, Navier-Stokes for blood
- Periodicity allows for greater MGRIT efficiency[1]
  - MGRIT simulates only one periodic time interval
  - Standard method simulates many intervals until steady state
  - Modest Reynolds number of ~3000
- 20 processors in time → 5x speedup
- Leveraged multilevel convergence theory[2] to inform algorithm development

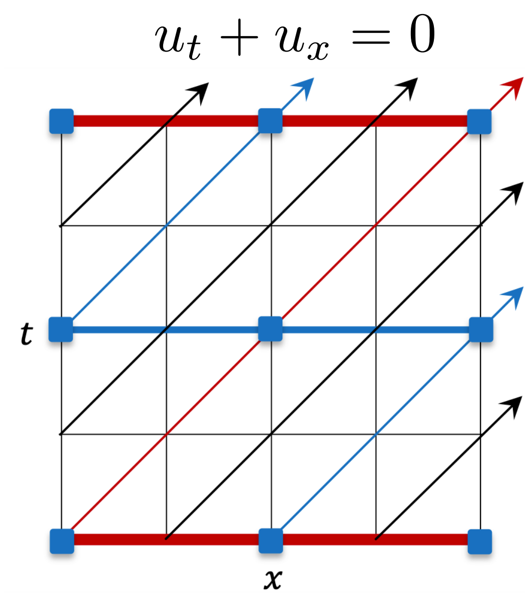


Left ventricle

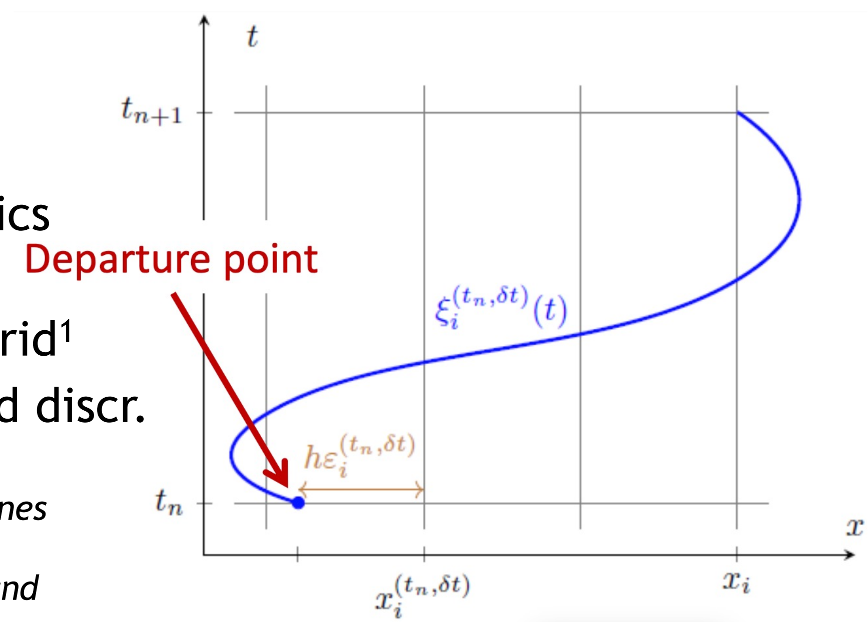
1. Hessenthaler, Falgout, Schroder, Nordsletten, Roehrle, *Time-Periodic Steady-State Solution of Fluid-Structure Interaction and Cardiac Flow Problems through MGRIT*. Comput. Meth. Appl. Mech. Eng., 2021.
2. Hessenthaler, Southworth, Nordsletten, Rohrle, Falgout, Schroder, *Multilevel convergence analysis of MGRIT*, SISC, 2020.

Pure hyperbolic: important recent progress

- Key insight: coarsen only in time, not space[2,3]
  → Do not coarsen away characteristics!
  - Coarse-grid time-stepping Φ_Δ should be semi-Lagrangian-like and follow characteristics



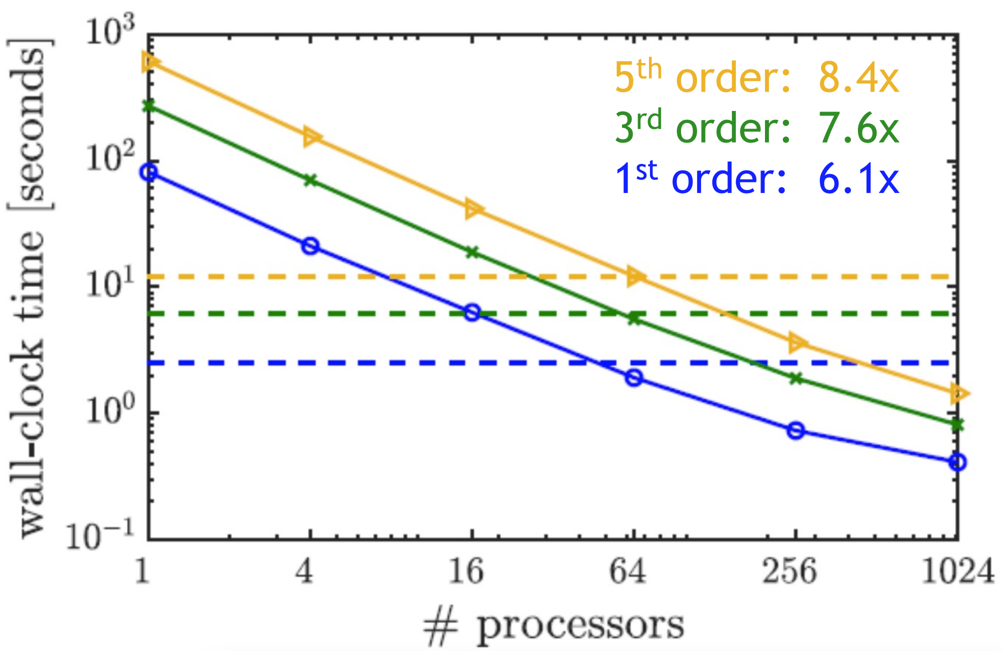
- Semi-Lagrangian Φ_Δ good for coarse-grid
  - Automatically adjust numerical domain of dependence in response to large coarse-grid time-step sizes
  - Integrate backwards along characteristics and interpolate
  - Paired with method-of-lines on finest-grid[1]
  - Precise matching of coarse and fine-grid discr.



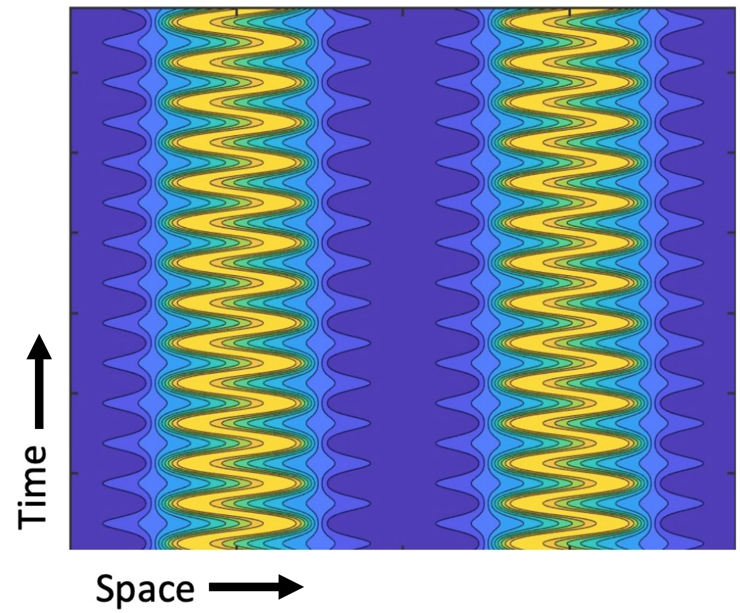
1. De Sterck, Falgout, Krzysik, Schroder, *Efficient MGRIT for Method-of-Lines Discretizations of Linear Advection*, J. Sci. Comput., 2023.
2. De Sterck, Falgout, Friedhoff, Krzysik, MacLachlan, *Optimizing MGRIT and Parareal coarse-grid operators for linear advection*, NLAA, 2021.
3. De Sterck, Falgout, Krzysik, *Fast MGRIT for Advection via Modified Semi-Lagrangian Coarse-Grid Operators*, SISC, 2023.

Pure hyperbolic: important recent progress

- Performance on linear advection excellent
  - Coarse-grid propagator designed to match the fine-grid propagator (not PDE)
  - If fine-grid time discretization is order p, then coarse-grid matches fine-grid discretization to order $p+1$
  - Effective for higher-order problems



$$u_t + \cos(2\pi t)\cos(2\pi x)u_x = 0$$



Recent work extends this to 1D hyperbolic systems

Parallel-in-Time for Chaotic Problems

- Chaotic problems are exceedingly sensitive to perturbations
  → How do we construct a sufficiently accurate coarse-grid?
  → Improve the coarse-grid equation (FAS) for nonlinear multigrid[1]

- Classic FAS is a constant correction τ about current coarse solution v

$$B_\Delta(v_{new}) = f + \tau$$

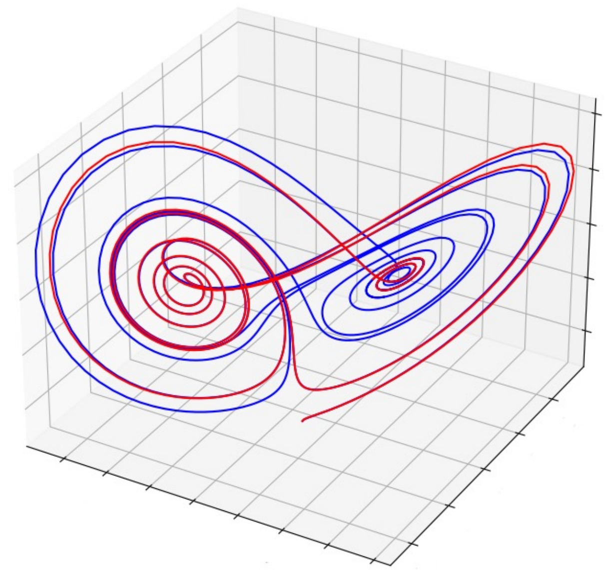
$$\tau_i = \Phi^m(v_i) - \Phi_\Delta(v_i)$$

- Delta-correction[2,3] $\mathbf{D}$ adds a linear correction

$$(B_\Delta + \mathbf{D})(v_{new}) = f + \tau + \mathbf{D}v$$

$$\mathbf{D}_i := (D_u\Phi^m - D_u\Phi_\Delta)(v_i)$$

→ Use low-rank approximation to $\mathbf{D}$ in practice



Lorenz System (butterfly)

1. Brandt, *Multi-level adaptive solutions to boundary-value problems*, Math. Comput., 1977
2. Yavneh, Dardyk, *A multilevel nonlinear method*, SISC, 2006.
3. Vargas, Falgout, Günther, Schroder, *Multigrid Reduction in Time for Chaotic Dynamical Systems*, SISC, 2024.

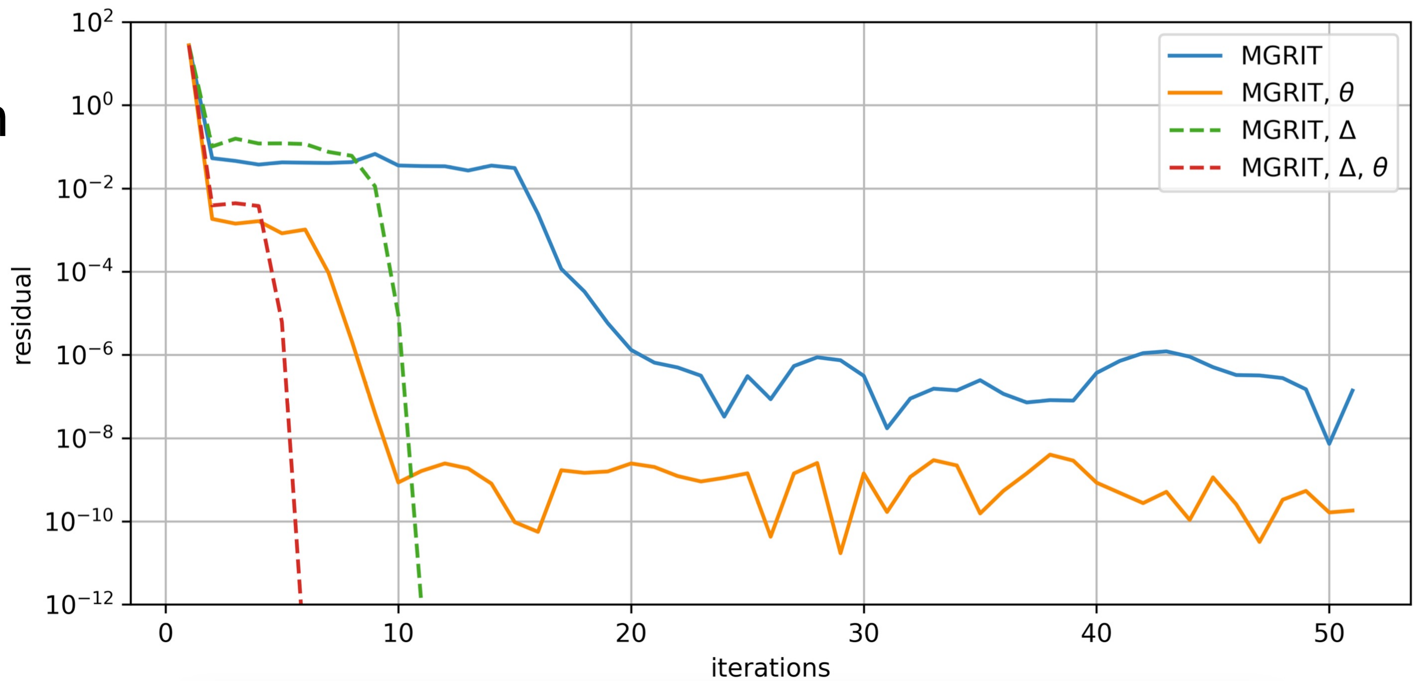
Results for Lorenz equation

- Further improve Φ_Δ by blending explicit and implicit methods
  - Blend so that coarse-grid matches fine-grid to higher-order
  - Coarse-grid propagator: Choose θ to match behavior of fine-grid

$$u' = f(u)$$

$$u_{i+1} = u_i + \theta\, hf(u_i) + (1-\theta)\, hf(u_{i+1})$$

- Results for Lorenz equation
  - 2-level
  - FCF-relaxation

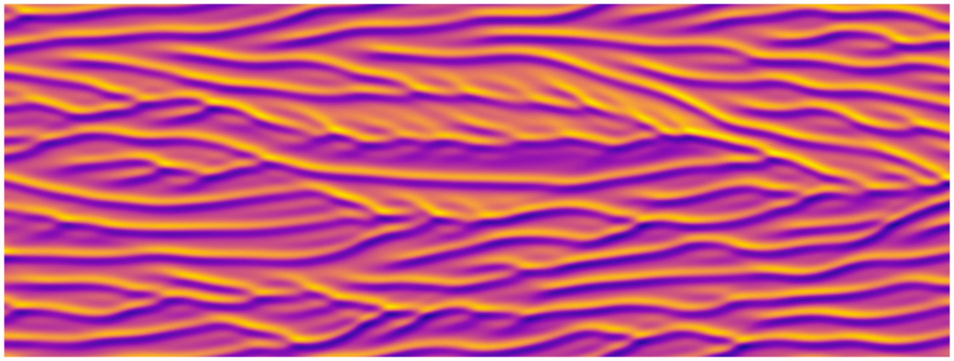


Results for Kuramoto-Sivashinsky (KS) equation

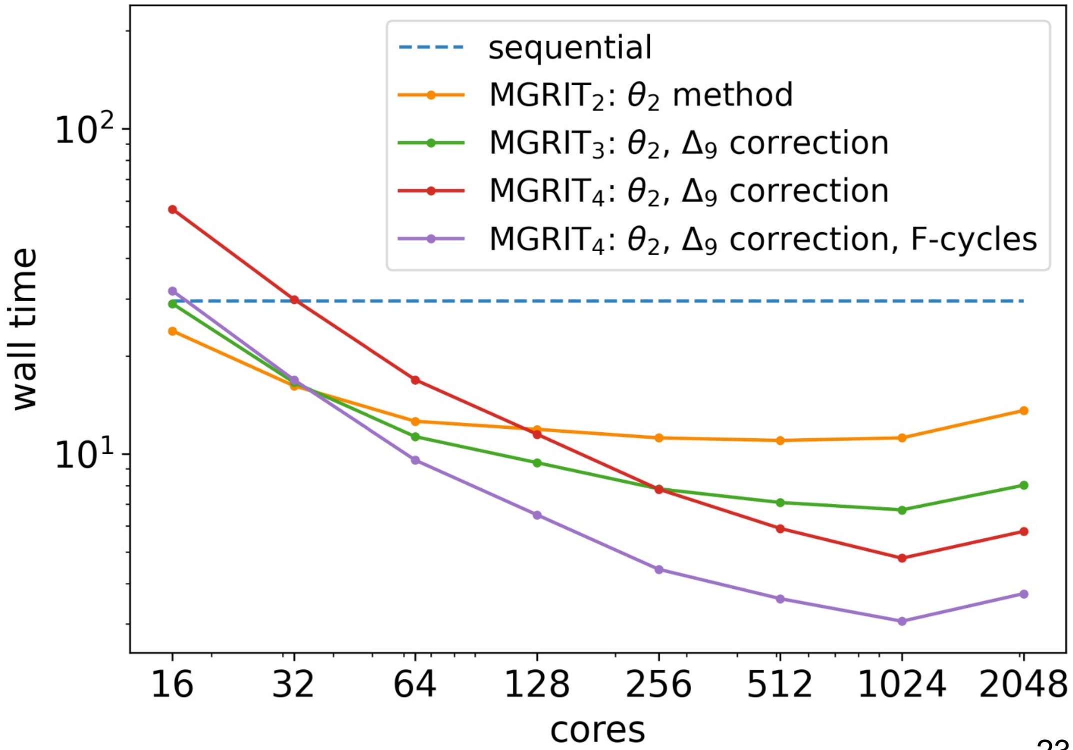
- KS equation

$$u_t = -u_{xx} - u_{xxxx} - u\, u_x$$

$$u(t,0) = u(t,L)$$



- Use low-rank Delta correction
  - Finitely many chaotic Lyapunov exponents
- Use 2nd order blended coarse-grid time-stepper (θ method)
- Speed-up ~10x
  - Over 10 Lyapunov time for true chaotic dynamics



Outline

- Introduction to parallel-in-time with multigrid
  - Motivation
  - Multigrid reduction in time (MGRIT) overview

- Description of the XBraid parallel-in-time code

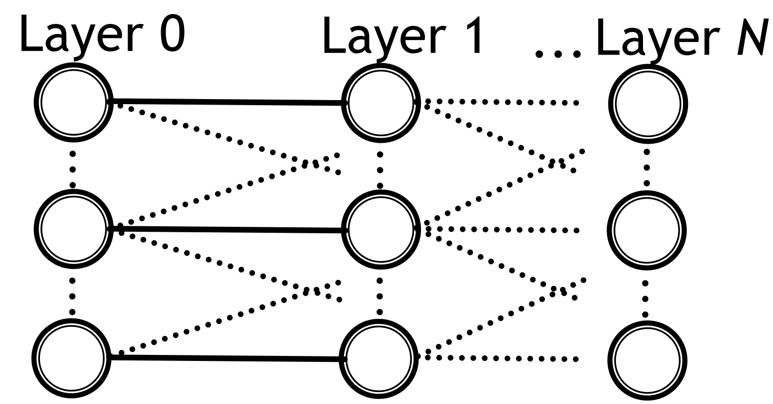
- Application areas
  - **Focus: application of XBraid to deep learning**

- Conclusions and future work

MGRIT for Deep Neural Networks (DNNs)

- DNNs are routinely used for many tasks
- However, training cost/times can be prohibitive (days, weeks, months...)
  - Due to the *many* forwards and backwards passes through the network

→ **Goal:** parallelize, speed up training



- Feed-forward network
  - Training pair: (y_{data}, c_{data})
  - W_n, b_n, y_n : Layer n weights, biases, state
  - **ResNet Propagation** (forward problem):

$$y_0 = y_{data}$$

$$y_{n+1} = y_n + F(W_n y_n + b_n) \quad \forall\, n = 0, \ldots, N-1$$

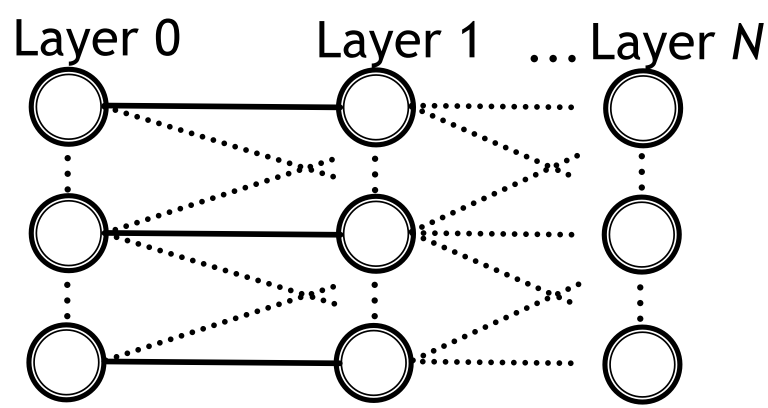
  - Learning problem:

$$\min_{W_n, b_n} \text{Loss}(y_N, c_{data}) \quad \text{subject to above forward problem}$$

MGRIT for Deep Neural Networks (DNNs)

- DNNs are routinely used for many tasks
- However, training cost/times can be prohibitive (days, weeks, months...)
  - Due to the *many* forwards and backwards passes through the network

→ **Goal:** parallelize, speed up training



- Feed-forward network
  - Training pair: (y_{data}, c_{data})
  - W_n, b_n, y_n : Layer n weights, biases, state
  - **ResNet Propagation** (forward problem):

$$y_0 = y_{data}$$

$$y_{n+1} = y_n + F(W_n y_n + b_n) \quad \forall n = 0, \ldots, N-1$$

Insert time-step parameter !

MGRIT for DNNs

- Some popular deep networks (e.g., ResNets[1,2], Transformers[3,4], GRU[5]) have an equivalence to time-stepping
- Take ResNet architecture and augment with time-step parameter h

$$y_0 = y_{data}$$
$$y_{n+1} = y_n + hF(W_n y_n + b_n) \quad \forall\, n = 0, \ldots, N-1$$
$$\Leftrightarrow$$
$$y(0) = y_{data}$$
$$\frac{dy(t)}{dt} = F(W(t)y(t) + b(t)), \quad \forall t \in (0, T)$$

ResNet $\equiv$ Forward Euler discretization
Backprop $\equiv$ Discrete adjoint

- Training problem becomes

$$\min_{W(t),b(t)} \text{Loss}(y(T), c_{data}) \quad \text{subject to above ODE}$$

1. Haber, Ruthotto. *Stable Architectures for Deep Neural Networks*. Inverse Probl., 2017.
2. Weinan, *A Proposal on Machine Learning via Dynamical Systems*, Comm. Math. Stat., 2017.
3. *Understanding and improving transformer from a multiparticle dynamic system point of view*, Y. Lu et al., 2019. arXiv:1906.02762
4. *Stateful ODE-Nets using basis function expansions, A. Queiruga et al.*, Advances in Neural Information Processing Systems, 2021.
5. *Parallel Training of GRU Networks with a Multi-Grid Solver for Long Sequences*, E. Moon and E. C. Cyr, ICLR 2022.

MGRIT for DNNs

- Transformers are powerful (e.g., ChatGPT), where attention mechanisms capture long-range dependencies within sequences

- Transformers can also be extended to the layer-parallel setting
  - Ignoring pre- and post-processing
  - Transformers consist of encoder and decoder layers of the following form:

$$\mathbf{X}^{[n+1]} = \mathbf{X}^{[n]} + \mathbf{F}_{\text{enc}}(t_n, \mathbf{X}^{[n]}) \qquad \mathbf{X}^{[n+1]} = \mathbf{X}^{[n]} + \mathbf{F}_{\text{dec}}(t_n, \mathbf{X}^{[n]})$$

$$\mathbf{F}_{\text{enc}}(t_n, x) := \varphi_1(x) + \varphi_2\left(x + \varphi_1(x)\right) \qquad \mathbf{F}_{\text{dec}}(t_n, x) := \varphi_1(x) + \varphi_3\left(x + \varphi_1(x)\right) + \varphi_2\left(x + \varphi_1(x) + \varphi_3\left(x + \varphi_1(x)\right)\right)$$

  - Where $\mathbf{X}^{[n]}$ is the network state at layer n, and

$$\varphi_1 := \mathrm{SA} \circ \mathrm{LN}, \varphi_2 := \mathrm{MLP} \circ \mathrm{LN}, \varphi_3 := \mathrm{CA} \circ \mathrm{LN}$$

for self-attention (SA), cross-attention (CA), and layer-norm (LN)

Related ODE Transformer works:

1. *Understanding and improving transformer from a multiparticle dynamic system point of view*, Y. Lu et al., 2019. https://www.arxiv.org/abs/1906.02762
2. *Stateful ODE-Nets using basis function expansions, A. Queiruga et al.,* Advances in Neural Information Processing Systems, 2021.

MGRIT for DNNs

- Transformers are powerful (e.g., ChatGPT), where attention mechanisms capture long-range dependencies within sequences

- Transformers can also be extended to the layer-parallel setting
  - Ignoring pre- and post-processing
  - Transformers consist of encoder and decoder layers of the following form:

$$\mathbf{X}^{[n+1]} = \mathbf{X}^{[n]} + \mathbf{F}_{\text{enc}}(t_n, \mathbf{X}^{[n]}) \qquad \mathbf{X}^{[n+1]} = \mathbf{X}^{[n]} + \mathbf{F}_{\text{dec}}(t_n, \mathbf{X}^{[n]})$$

$$\mathbf{F}_{\text{enc}}(t_n, x) := \varphi_1(x) + \varphi_2(x + \varphi_1(x)) \qquad \mathbf{F}_{\text{dec}}(t_n, x) := \varphi_1(x) + \varphi_3(x + \varphi_1(x)) + (x + \varphi_1(x)))$$

Insert time-step parameter[2] !   Insert time-step parameter[2] !

  - Where $\mathbf{X}^{[n]}$ is the network state at layer n, and

$$\varphi_1 := \text{SA} \circ \text{LN}, \varphi_2 := \text{MLP} \circ \text{LN}, \varphi_3 := \text{CA} \circ \text{LN}$$

for self-attention (SA), cross-attention (CA), and layer-norm (LN)

Related ODE Transformer works:

1. *Understanding and improving transformer from a multiparticle dynamic system point of view*, Y. Lu et al., 2019. https://www.arxiv.org/abs/1906.02762
2. *Stateful ODE-Nets using basis function expansions, A. Queiruga et al.,* Advances in Neural Information Processing Systems, 2021.

MGRIT for DNNs

- Transformers are powerful (e.g., ChatGPT), where attention mechanisms capture long-range dependencies within sequences

- Transformers can also be extended to the layer-parallel setting
  - Ignoring pre- and post-processing
  - Transformers consist of encoder and decoder layers of the following form:

$$\mathbf{X}^{[n+1]} = \mathbf{X}^{[n]} + \mathbf{F}_{\text{enc}}(t_n, \mathbf{X}^{[n]}) \qquad \mathbf{X}^{[n+1]} = \mathbf{X}^{[n]} + \mathbf{F}_{\text{dec}}(t_n, \mathbf{X}^{[n]})$$

$$\mathbf{F}_{\text{enc}}(t_n, x) := \varphi_1(x) + \varphi_2\left(x + \varphi_1(x)\right) \qquad \mathbf{F}_{\text{dec}}(t_n, x) := \varphi_1(x) + \varphi_3\left(x + \varphi_1(x)\right) + \left(x + \varphi_1(x)\right))$$

Insert time-step parameter[2] ! Insert time-step parameter[2] !

  - Where $\mathbf{X}^{[n]}$ is the network state at layer n, and

$$\varphi_1 := \mathrm{SA} \circ \mathrm{LN}, \varphi_2 := \mathrm{MLP} \circ \mathrm{LN}, \varphi_3 := \mathrm{CA} \circ \mathrm{LN}$$

    for self-attention (SA), cross-attention (CA), and layer-norm (LN)

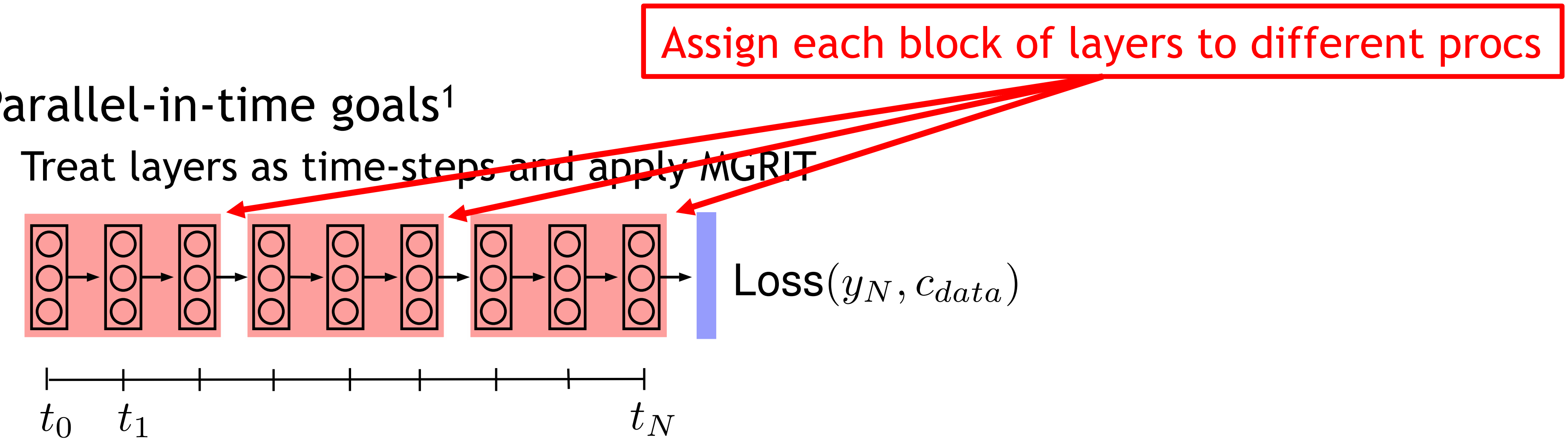
- Learning problem remains unchanged

Parallel-in-time and ODE-like neural networks

- Network propagation is equivalent to a forward Euler discretization, and backpropagation is equivalent to discrete adjoint!

  → *Remember:* Φ is layer-step in a DNN
  → *Use equivalence to apply XBraid to forward and backward problems*

- Parallel-in-time goals[1]
  - Treat layers as time-steps and apply MGRIT



  - Good strong and weak scaling with respect to number of network layers
    → Train a network with 5 layers with same wall-clock time as 1000 layers
  - Solve the same training problem (no shortcuts) as the sequential training version
  - Provide novel layer-parallelism (decoupled layer computations in parallel)

1. Günther, Ruthotto, Schroder, Cyr, Gauger. *Parallel-in-Layer Optimization for Training of Deep ResNets.* SIMODS, 2020.

Code: Layer-parallelism with multigrid

- **TorchBraid: XBraid (MGRIT code in MPI/C) and PyTorch**

  https://github.com/Multilevel-NN/torchbraid

  `$ pip install path/to/torchbraid`

- Python, C/MPI coupling with Cython (a little messy, but also impressive)

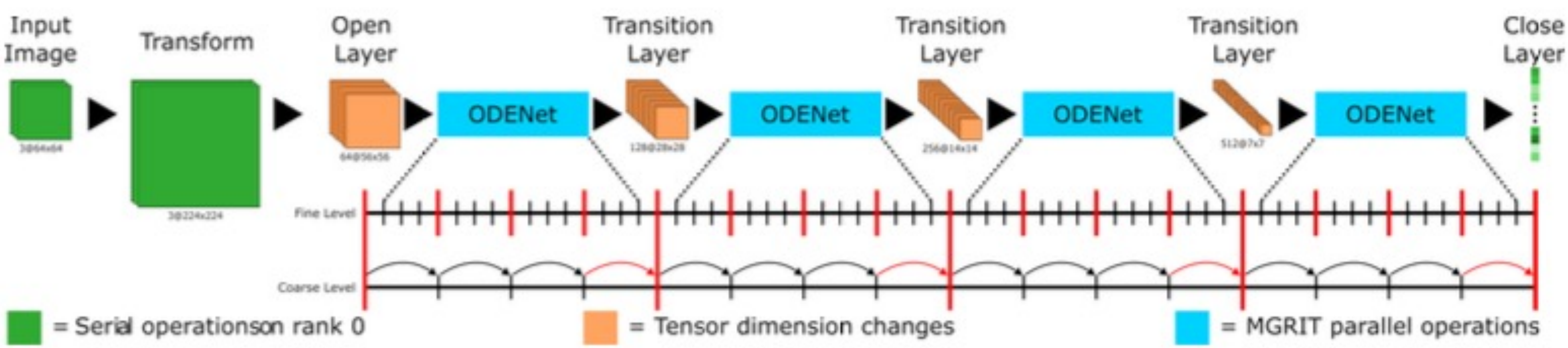
- Parallel Dataset and Dataloader functions inherited from PyTorch
  - Only load on root processor

- Compatible with data parallelism

- GPU-to-GPU direct communication key for performance
  - Requires CUDA-capable MPI, which you can test with
    `$ make tests -direct -gpu`

Results: Tiny ImageNet

- Extend TorchBraid ODENets to use max pooling and batch normalization
- Maintain max pooling layers as C-points on coarse levels
- Layer-parallel batch normalization
  - During inference, batch norm is identical to standard
  - During training, batch norm running averages are only updated during final fine-level evaluation at a layer

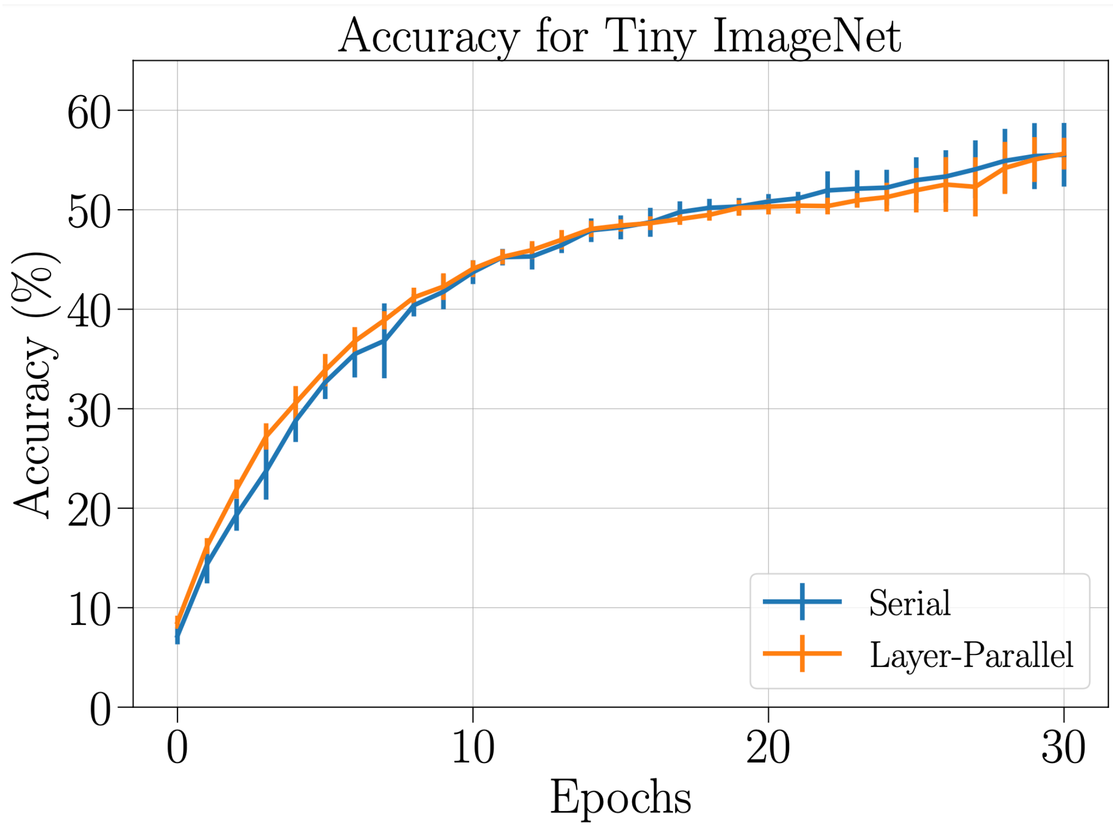


Results: Tiny ImageNet

- Train on PLEIADES cluster at University of Wuppertal
  - 5 compute nodes, each containing 8 A100 GPUs
  - Utilize GPUs for training and compare to serial time on GPU

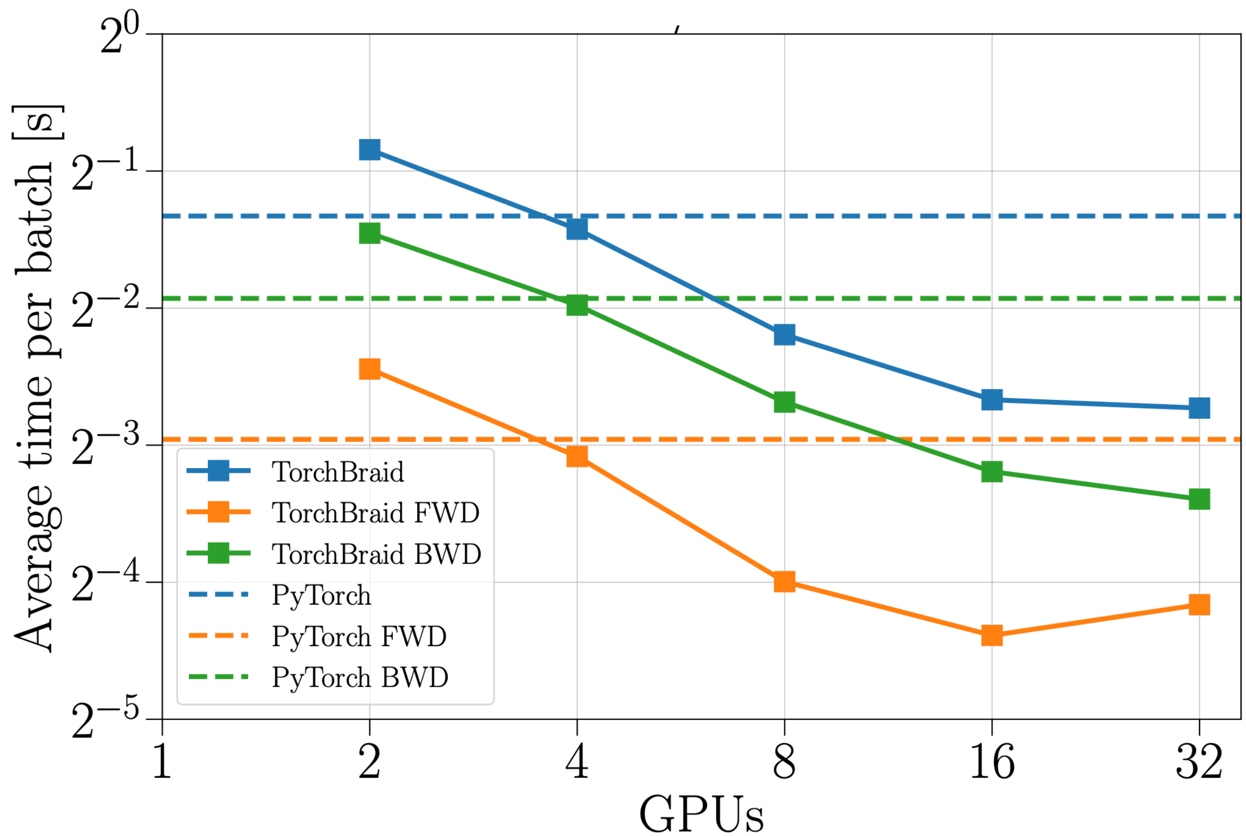
- Training setup
  - 256 total layers
  - 64 layers per ODENet block
  - 30 epochs and batch size of 50
  - 16 training runs
    - Vertical bars: std. dev

- Accuracy still good, but potential bias in layer-parallel gradient is current research topic



Results: Tiny ImageNet

- Strong scaling on Glinda
  - One A100 per node
  - Compare forward, backward, and overall times



- Crossover point: 4 GPUs

- 2.5x speedup at 16 GPUs

- Major coding effort! But, still work to be done on efficiency.
  - Transformer training is ongoing work

Nearly 50 years of research exists, but has only scratched the surface

- Earliest work goes back to 1964 by Nievergelt
  - Led to multiple shooting methods, Keller (1968)
- Space-time multigrid methods for parabolic problems
  - Hackbusch (1984); Horton (1992); Horton and Vandewalle (1995)
  - The latter is one of the first optimal & fully parallelizable methods to date
- Parareal was introduced by Lions, Maday, and Turincini in 2001
  - Probably the most widely studied method
  - Gander and Vandewalle (2007) show that parareal is two-level FAS multigrid
- Discretization specific work includes
  - Minion, Williams (2008, 2010) – PFASST, spectral deferred correction / parareal
  - De Sterck, Manteuffel, McCormick, Olson (2004, 2006) – FOSLS
- Research on these methods is ramping up!
  - Ruprecht, Speck, Schaufele, Götschel, Gander, De Sterck, …
    not an exhaustive list

Summary and conclusions

- Sequential time integration bottleneck is real
  - Parallel in time is needed for future architectures
  - This is a major paradigm shift
- We apply multigrid reduction to the time dimension
  - Multigrid is ideal for extreme scale (optimal, resilient, ...)
  - Result is a flexible and non-intrusive approach
  - Demonstrated speedups for a variety of problems
- There is much future work to be done!
  - More problem types, more complicated discretizations
  - Performance improvements, adaptive meshing
  - Enabling novel parallelism in machine learning
  - ...

Selected references

Parallel-in-Time

1. Falgout, Friedhoff, Kolev, MacLachlan, Schroder, *Parallel Time Integration with Multigrid*, SIAM J. Sci. Comput. (SISC), 2014.
2. Dobrev, Kolev, Petersson, Schroder, *Two-level Convergence Theory for MGRIT*, SIAM J. Sci. Comput. (SISC), 2017.
3. Guenther, Ruthotto, Schroder, Cyr, Gauger, *Layer-parallel training of deep residual neural networks*. SIAM J. Math. Data Sci. (SIMODS), 2020.
4. Sugiyama, Schroder, Southworth, Friedhoff, *Weighted Relaxation for Multigrid Reduction in Time*. Numer. Lin. Alg. Appl. Submitted, June 2021.
5. Ong, Schroder, *Applications of Time Parallelization*. CVS, Springer, 2020. *Review paper.*

Software

1. XBraid
   https://github.com/XBraid/xbraid
   Co-developed with Lawrence Livermore National Lab

2. TorchBraid
   https://github.com/Multilevel-NN/torchbraid
   Co-developed with Sandia National Lab