Karlsruher Institut für Technologie

Seminar work:
Machine Learning Methods for Solving Differential Equations

Winter 2021/22

Advisor: Vadim Arzamasov

Dmitrii Seletkov

IPD Böhm - Lehrstuhl für Systeme der Informationsverwaltung

Agenda

Motivation

Foundations

Theory-guided Approaches
- Continuous Approaches
- Discrete Approaches

Neural Operators

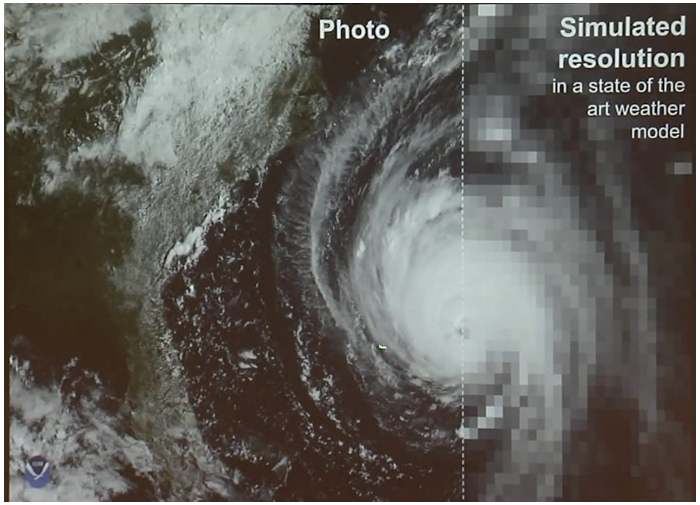
High-dimensional Approaches

Conclusion

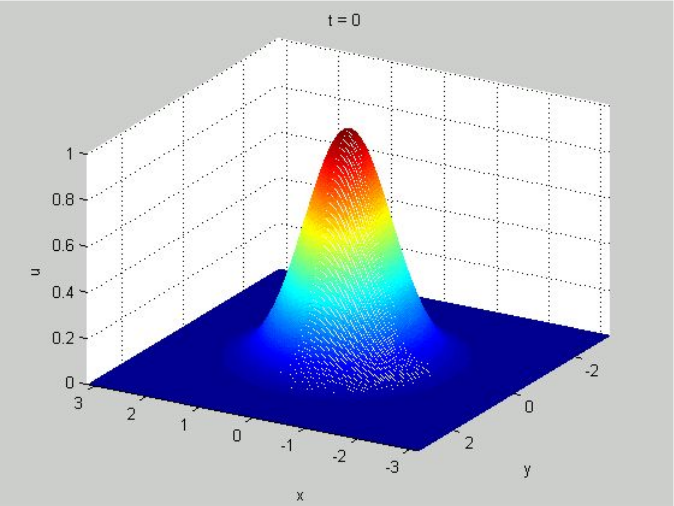
Motivation

Motivation

- Differential equations:
  - govern our physical world
  - ubiquitous modelling tool: physics, finance, chemistry, etc.
- Conventional numerical solvers:
  - infeasible computational cost
  - applied math: ad hoc “effective equations”
- Machine Learning: ad hoc data solutions



Simulation of Weather vs. Real Life [Bre19]



Simulation of Viscous Fluid [Burg]

Foundations

Foundations (1)

- Partial Differential Equation (PDE): partial derivatives wrt. > 1 independent variables
- Boundary Conditions (BC) and Initial Conditions (IC): space and time

- Linear vs. Non-linear vs. Quasi-linear
- Second-order PDEs with $B^2 - 4AC$:
  - < 0 elliptic (e.g. Poisson)
  - = 0 parabolic (e.g. Navier-Stokes)
  - > 0 hyperbolic (e.g. Burger's)

$$A\frac{\partial^2 u}{\partial x^2} + B\frac{\partial^2 u}{\partial x \partial y} + C\frac{\partial^2 u}{\partial y^2} + D = 0$$

$$a(x,y)\frac{\partial u}{\partial x} + b(x,y)\frac{\partial u}{\partial y} + c(x,y)\, u = 0$$

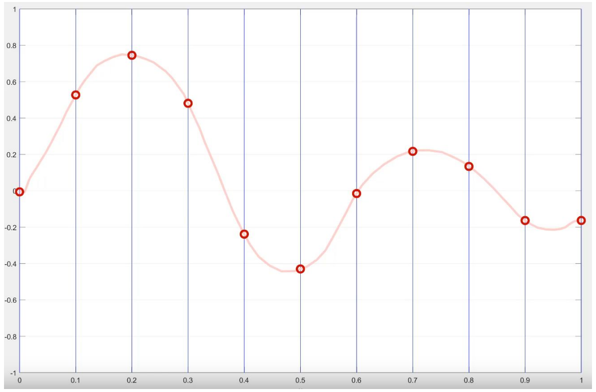
$$a(x,y,u)\frac{\partial u}{\partial x} + b(x,y,u)\frac{\partial u}{\partial y} + c(x,y,u)\, u = 0$$

$$a(x,y,u,u_x,u_y)\frac{\partial u}{\partial x} + b(x,y,u,u_x,u_y)\frac{\partial u}{\partial y} + c(x,y,u)\, u = 0$$

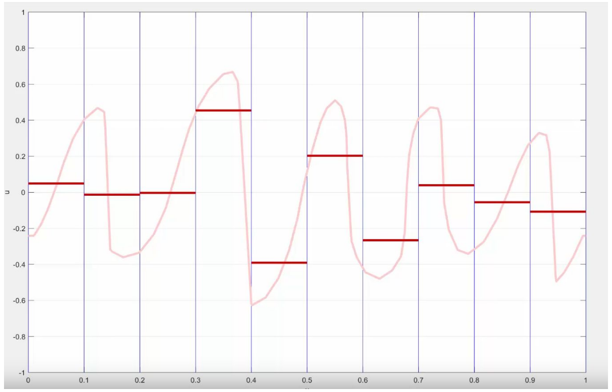
Linear vs. Quasi-linear vs. Fully Non-linear PDEs

Foundations (2)

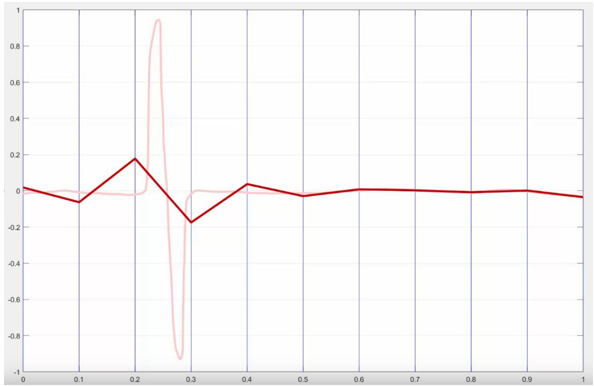
- Traditional Numerical Methods:
  - Finite Difference: values of the function of grid points
  - Finite Volume: averages of the function between grid points
  - Finite Element: best approximation within basis functions



Finite difference [Wang17]



Finite volume [Wang17]

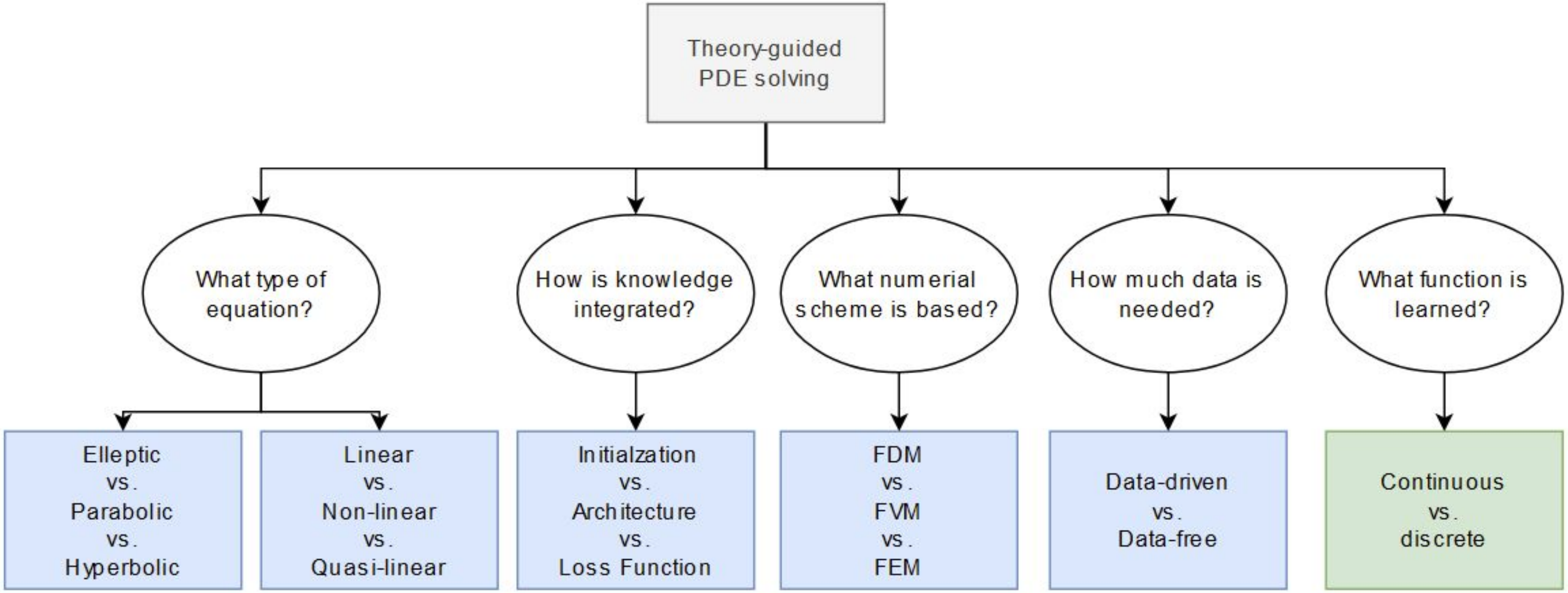


Finite element [Wang17]

Theory-guided Approaches

Theory-guided Approaches: Roadmap

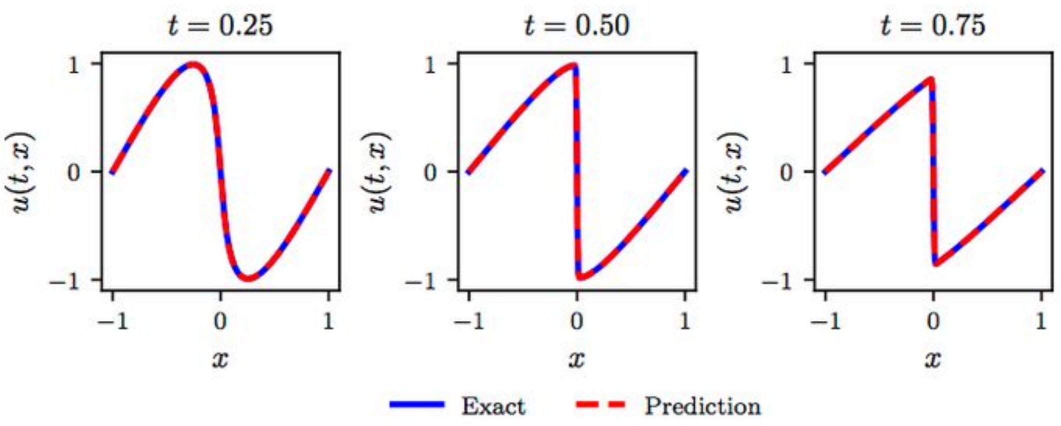
- Theory-guided: include underlying physics of the process
- Also formulated as *physics-informed* neural networks (PINNs)
- Research field in many areas
- Classification is versatile



Possible Classification of Theory-guided Approaches for Solving PDEs

Theory-guided: Continuous [Rai19]

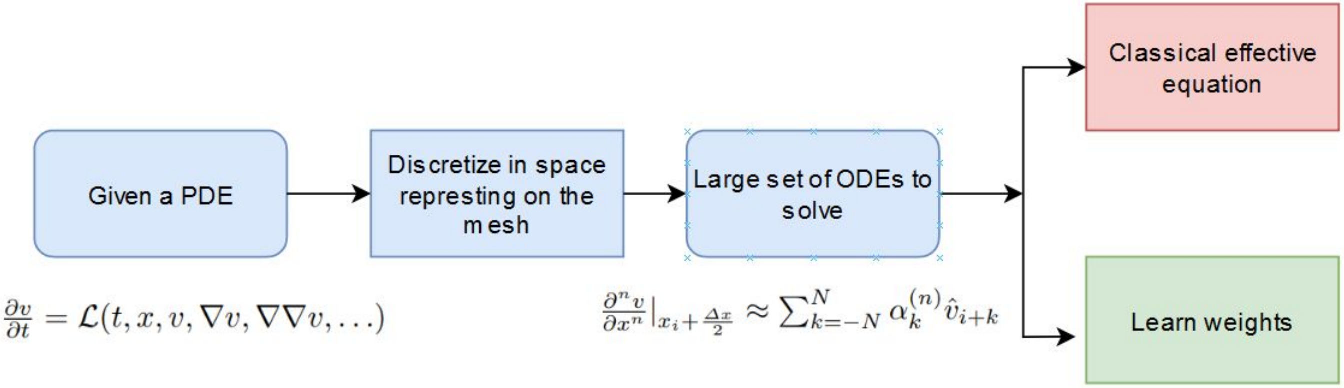
- Approximate continuous function based on point-wise AD
- FC architecture
- Baseline for many problems
- Loss function incorporates knowledge of PDE:
  - PDE residuals and data mismatch
- Limitations:
  - high training cost
  - hard to formulate IC and BC for PDEs with more than 2-D



Results of Continuous PINNs for 1-D Burgers PDE [Rai19]

Theory-guided: Discrete

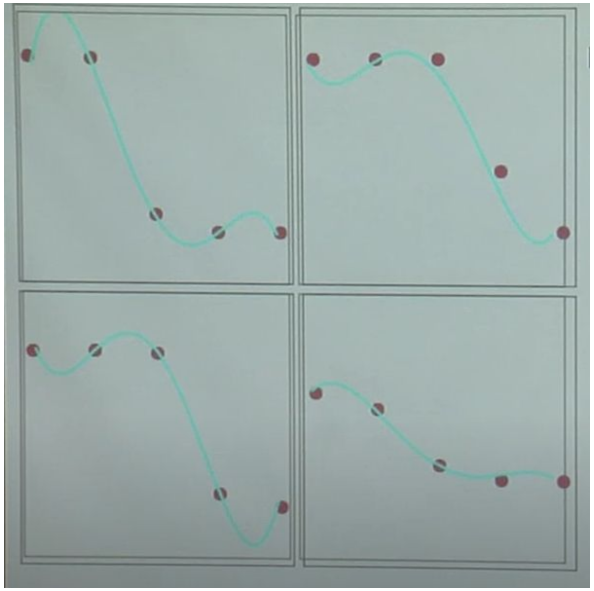
- Learn directly spatiotemporal solutions
  - end-to-end
  - derivatives of physics-informed loss based on discretization scheme
- CNN architecture
- Better efficiency and scalability
- Many approaches:
  - Data-driven Discretization: CNN [Bar19, Zhu20, Koch21]
  - DiscretizationNet: generative encoder-decoder CNN [Ran20]
  - Physics-informed graph neural Galerkin networks [Gao21]



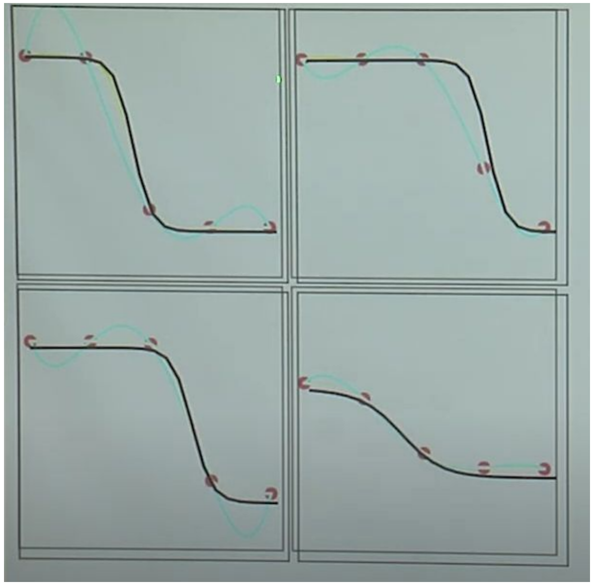
Pipeline of Solving PDE by Discretization

Data-driven Discretization

- **Problem**
  - Deriving numerical *effective equations*: ad hoc and difficult
- **Proposal**
  - Solutions of high-dim. PDE are low dim. manifold [Titi90]
  - Parameterize the manifold
  - ML approximates the manifolds -> *data-driven discretization*



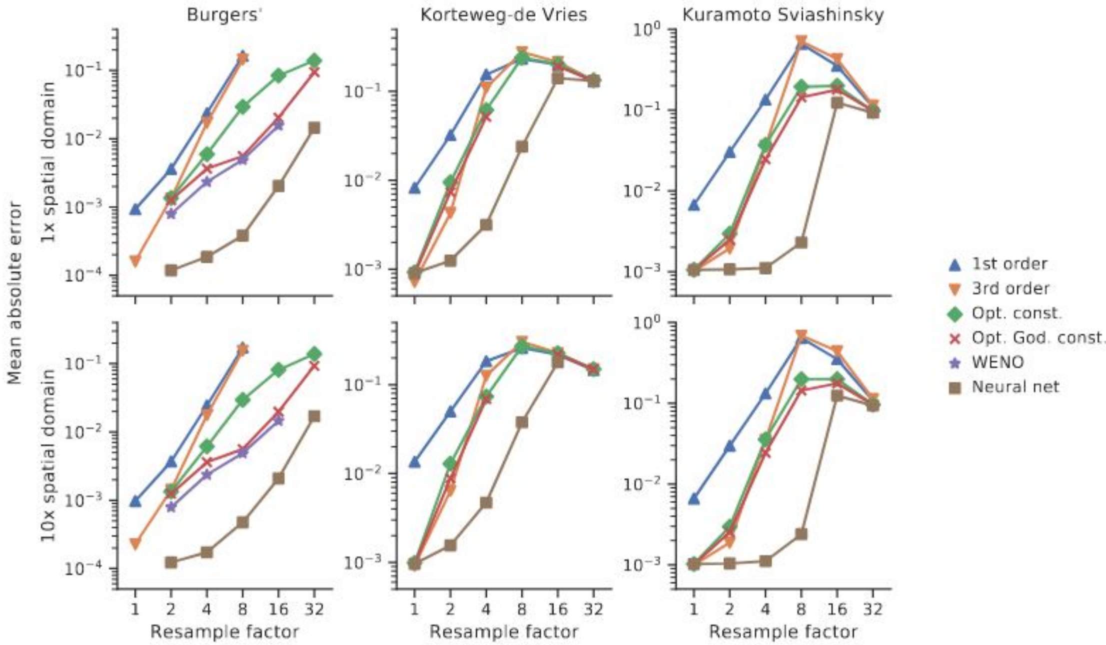
Poly Interpolation [Bre19]



Neural Net Prediction [Bre19]

Results and Limitation

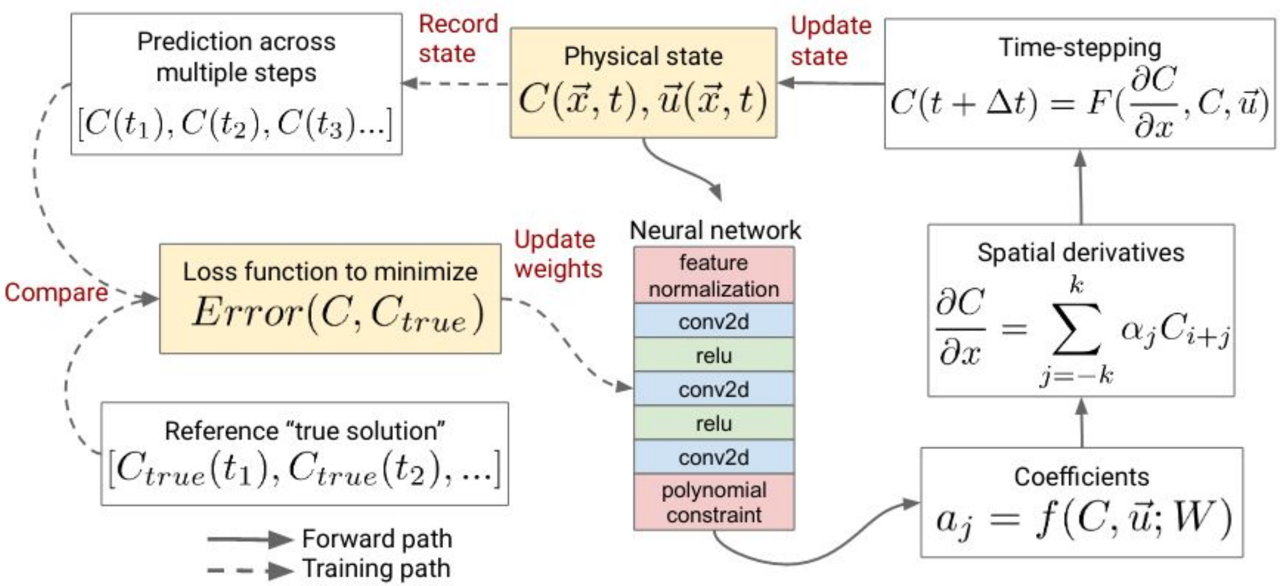
- **Results:**
  - Evaluated on Burgers, Korteweg-de Vries, Kuramoto Sviashinsky
  - 4-8x coarser than standard numerical finite methods
- **Limitations:**
  - Only one-dimensional problems
  - Speed



Results of Data-Driven Discretization [Bar19]

Data-driven Discretization for 2D

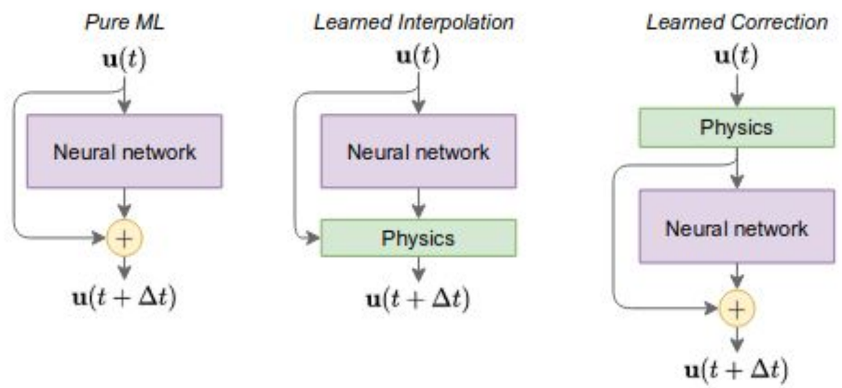
- Following work
- The main modification: MAE for multiple timesteps
- Integrate FD coefficients into FV Solver
- Physical Constraints before and after NN



Data-driven Discretization Framework for 2D [Zhu20]

ML acceleration for Navier-Stokes Equation

- Following work
- **Problem:** Lack of generalization for the ML approaches
- **Proposal:**
  - Based on existing DNS and LES numerical methods
  - Replace error-prone parts: closure and discretization
    - Learned Interpolation (LI), instead of Polynomial Interpolation
    - Learned Correction (LC): model a residual correction to the discretization
- **Results:**
  - 50-80x Speed-Up
  - 8-10x coarser resolution by remaining accuracy

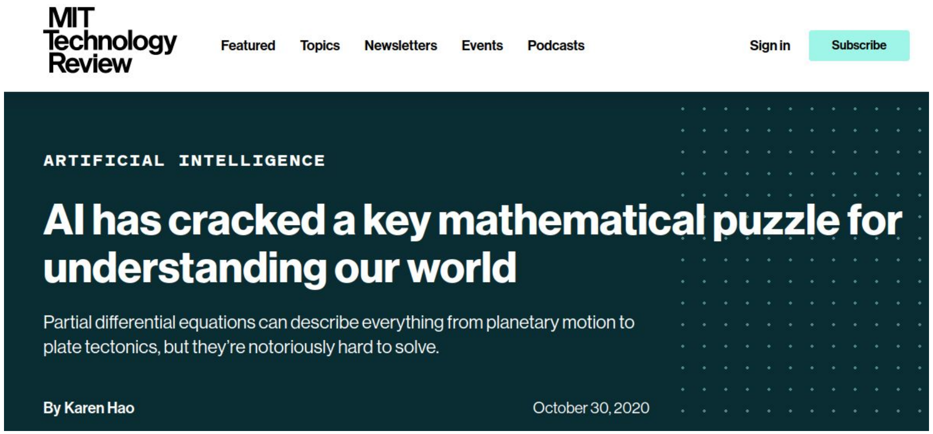


Difference between Pure ML, LI, LC [Koch21]

Neural Operators

Neural Operators

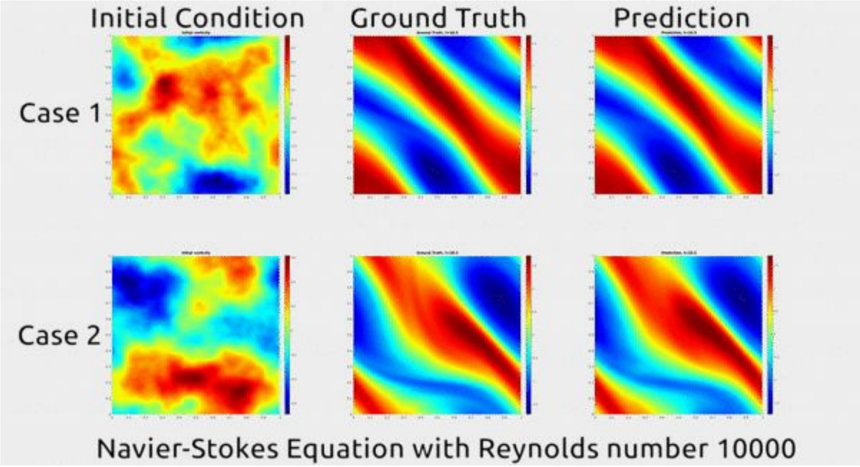
- Universal approximation theorem [Chen20]:
  - NN can approximate arbitrary continuous function
  - NN can approximate any non-linear continuous functional or operator
- Difference: mapping between Euclidean spaces vs. function spaces
- Advantages:
  - Mesh-independent
  - No restrictions with IC and BC
- Fundamental work: DeepONet [Lu19]
- Recent work: Fourier Neural Operator [Li20]



Headline of a Technology Review Article about Fourie Neural Operator [Hao20]

Fourier Neural Operator (FNO): Motivation

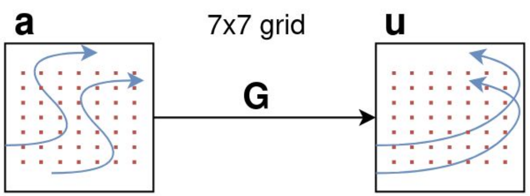
- **Problem**
  - Traditional Solvers: trade-off discretization vs. computation cost
  - Classical data-driven ML methods: faster, but solve one instance of PDE tied in one discretization (mesh-dependent)
- **Proposal**
  - Zero-shot super resolution (mesh-independent)
  - Generalization: no BC dependency

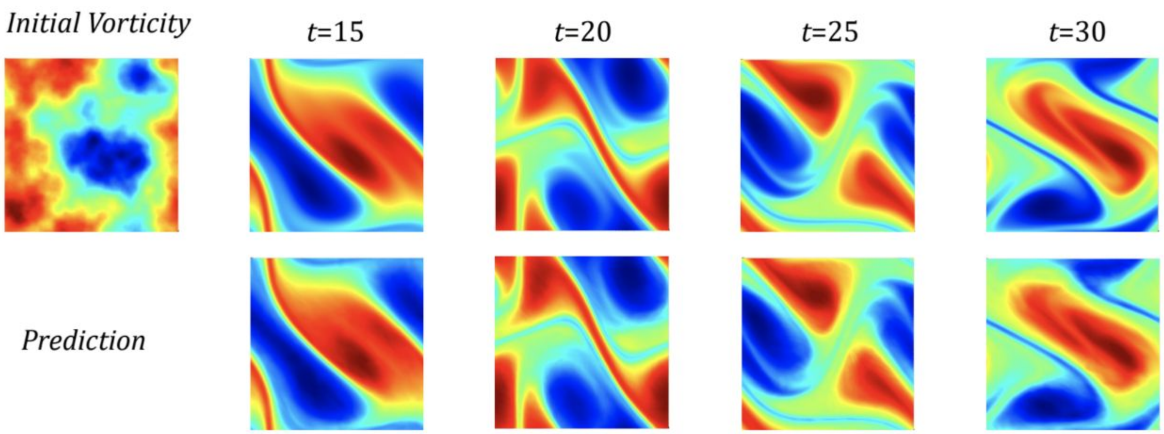


Demonstration of Prediction of FNO Method [Li20]

FNO: Formal Problem Setting

- Navier-Stokes PDE in 2D:
- Goal: find $G_\theta : \mathcal{A} \to \mathcal{U}, \quad \theta \in \Theta$
  - where A and U are function spaces and Theta is a model
  - Interchangeable: Data-points and functions f: x,y,t -> value
  - Why not just data points?
    - Data points: Sample on particular resolution
    - Functions: once learnt, any resolution possible

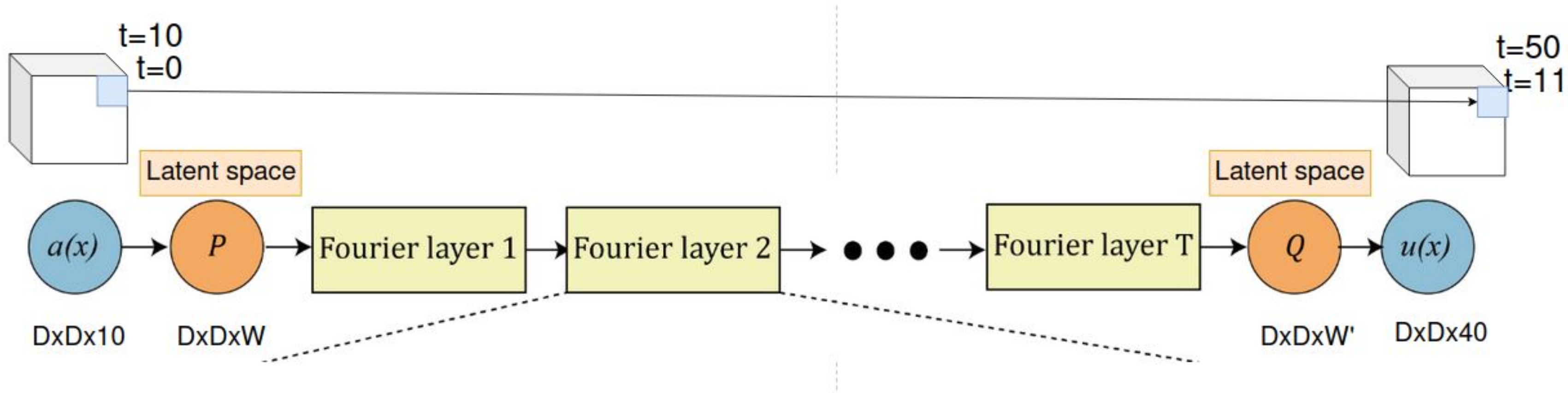




Evolving of the Vorticity in Time Predicted by FNO [Li20]

FNO: Method (1)

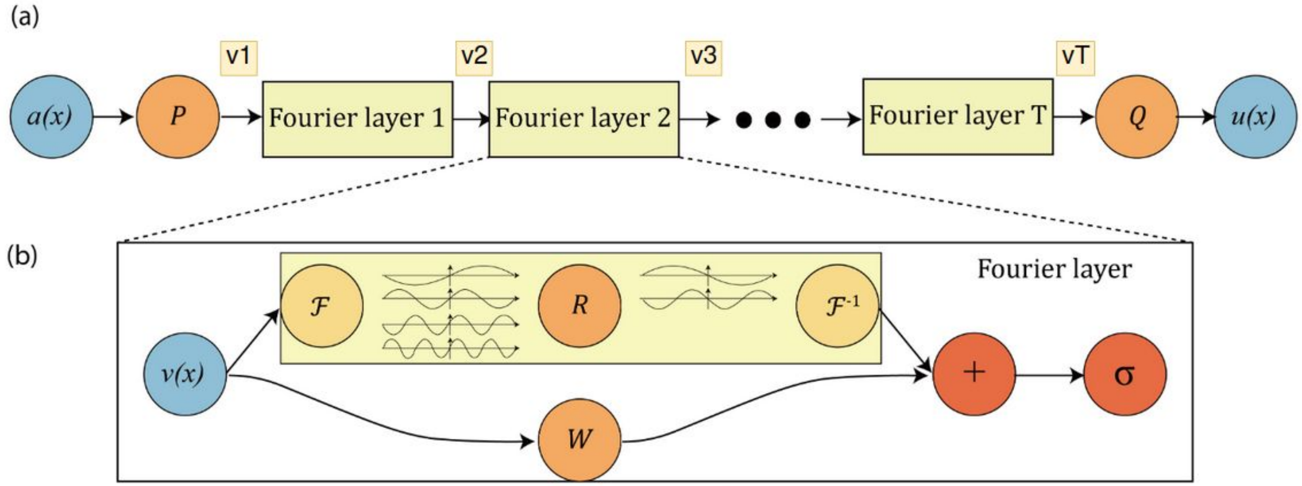
- Two main steps in the pipeline
  - Up- and Down-projection P and Q
    - Work pointwise (analogy: channels)
    - At each pixel want to know the value after evolving in time
    - Conv1x1
  - Fourier Layers with Fourier Neural Operators (FNO)



Pipeline of FNO method [Li20]

FNO: Method (2)

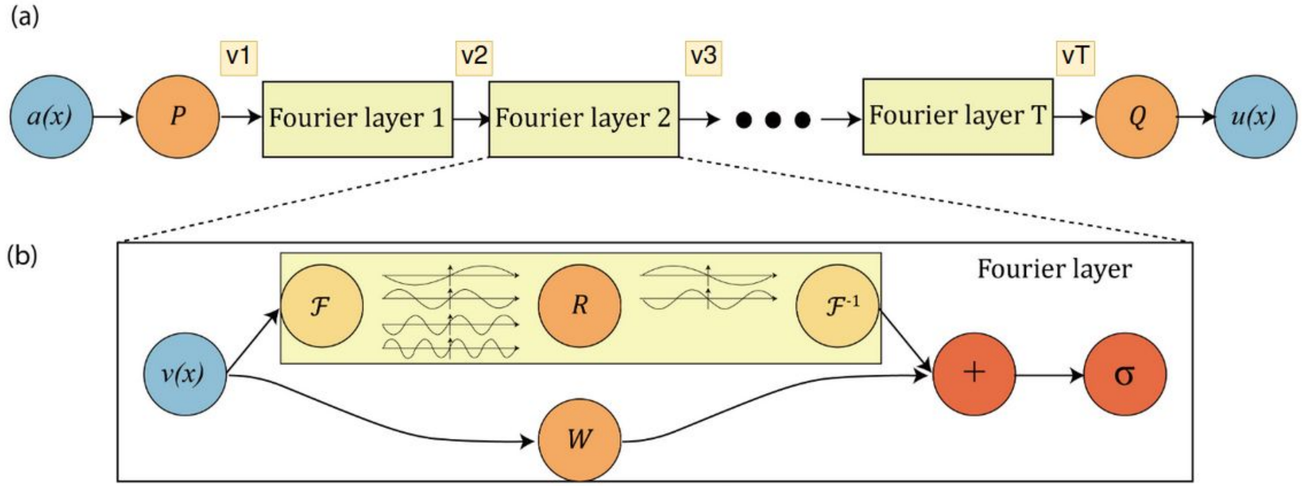
- Fourier Layer:
  - Goal: Iterative update
  - Kernel Integral operator:
    - parameterized by NN
    - Convolve the space
  - Impose restrictions to get FNO
- FNO:
  - Transform into Fourier Space and back
  - In Fourier Space: Convolving is just Multiplication with R



Main Structure of Fourier Layer [Li20]

FNO: Method (3)

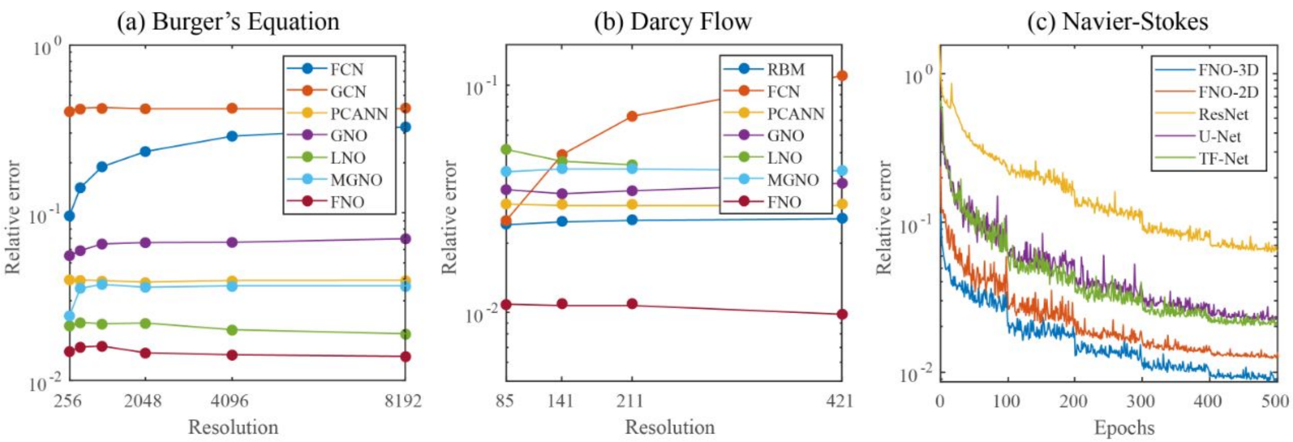
- Fourier Neural Operator
  - F: Transform into Fourier Space
  - Cut away top Fourier modes
    - Intuition: Regularization and Generalization
  - R: multiplication equivalent to Convolving
  - F-1: Transform back to Input Space
- Main advantages:
  - Independency to discretization
  - These types of problems profit from Fourier Space



Main Structure of Fourier Layer [Li20]

Results

- SOTA Performance for all investigated PDEs
- Zero-shot super-resolution (discretization-invariant)
- 3x faster by superior accuracy
- Turbulent flow problems
- Future works:
  - Other types of problems
  - Application in computer vision

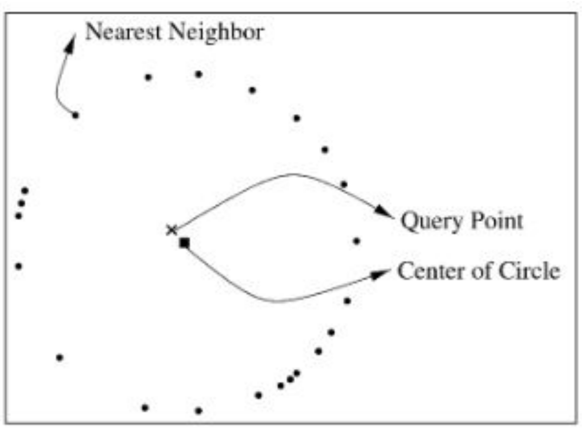


Comparison of Results of FNO with Other Approaches [Li20]

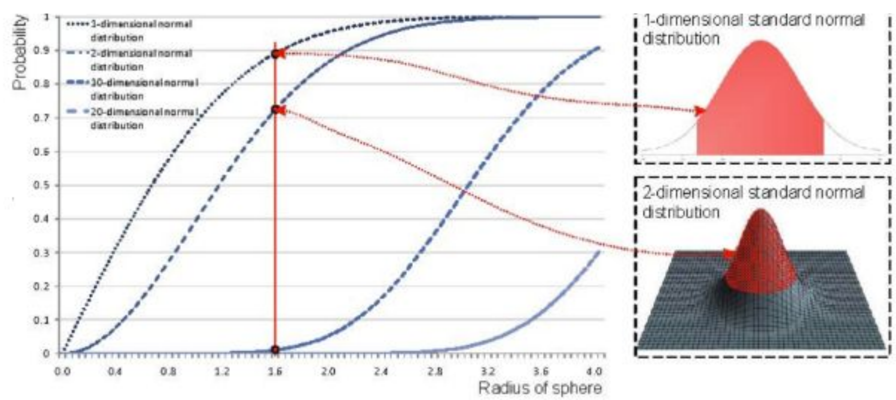
High-dimensional Approaches

High-dimensional Approaches

- Some spheres require solving PDEs with > 100
- The curse of dimensionality:
  - concentration of norms and distances
  - instability of neighborhood
  - changing behavior of distribution of data
- Numerical Solvers: exploding number of grid points
- Pure ML Solvers: exp. growth of the complexity of non-linear models
- Solution: combination of Numerical and ML Solvers



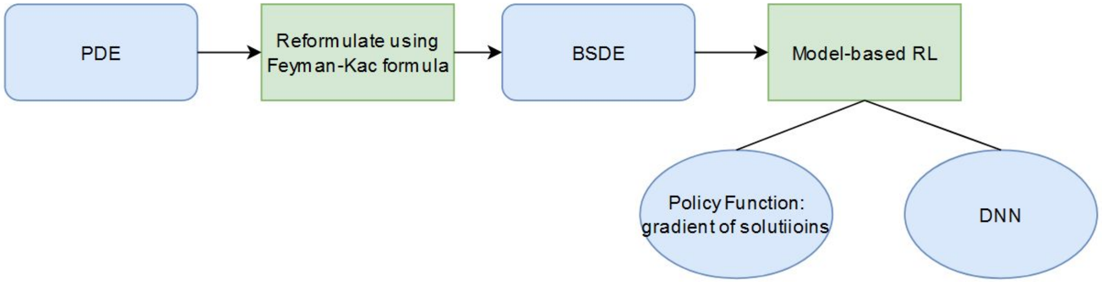
Instability of Neighborhood [Bey99]



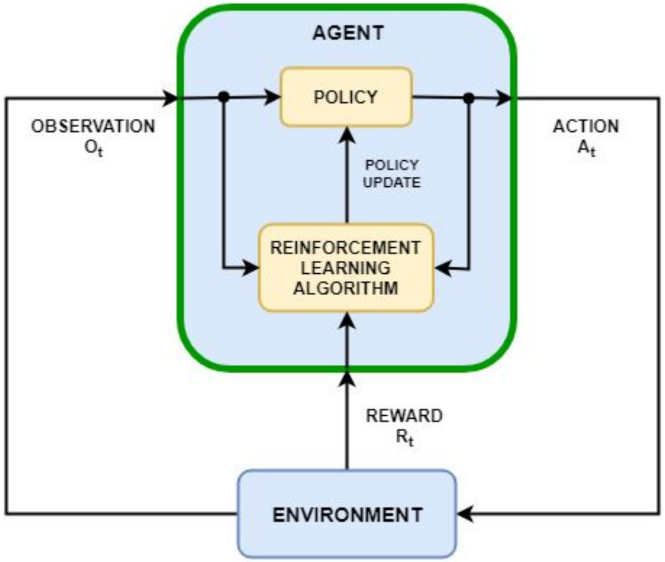
Normal Distributions in High-dim. space [DS2]

High-dimensional Approaches: BSDE [Han18]

- Method:
  - Reformulate PDE to BSDE - Backward Stochastic Differential Equations
- Results and Limitation:
  - 100-dim non-linear parabolic PDEs
  - less computational cost, better accuracy
  - Reformulation is ad hoc



Pipeline of BSDE Approach



Reinforcement Learning Pipeline [MAT]

High-dimensional Approaches: DLM [Sir18]

- Deep Galerkin Method:
  - Gelerkin Method: Numerical method looks for linear combination of basis functions to find the solution for a underlying PDE.
  - Replace basis functions and their linear combination with FC-layers
- Results:
  - 200-dim quasi-linear PDEs
  - No restrictions to parabolic
  - Mesh-free

Conclusion

Conclusion

- PDEs are omnipresent
- Main paradigms:
  - Theory-guided
  - Neural Operators
  - High-dimensional
- Combination of ML and Numerical Methods
  - Reduction of computational cost
  - Improvement of accuracy
- Desirable goal: Generalization
- Diversity of applied ML techniques

References

[DS2] Lecture Data Science 2, KIT, summer 2021.

[Bey99] Kevin Beyer et al. “When Is ”Nearest Neighbor” Meaningful?” In: ICDT. Vol. 1540. Lecture Notes in Computer Science. Springer, 1999, pp. 217–235.

[Han18] Jiequn Han, Arnulf Jentzen, and Weinan E. “Solving high-dimensional partial differential equations using deep learning”. In: Proceedings of the National Academy of Sciences 115.34 (Aug. 2018).

[Sir18] Justin Sirignano and Konstantinos Spiliopoulos. “DGM: A deep learning algorithm for solving partial differential equations”. In: Journal of Computational Physics 375 (Dec. 2018), pp. 1339–1364.

[Mou20] Ali Mousavi, “Off-Policy Estimation for Infinite-Horizon Reinforcement Learning”, Google AI Blog.

[Lu19] Lu Lu, Pengzhan Jin, and George Em Karniadakis. “DeepONet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators”. In: CoRR abs/1910.03193 (2019).

References (2)

[Chen95] Tianping Chen and Hong Chen. “Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its applications to dynamic systems”. In: Neural Networks, IEEE Transactions on (Aug. 1995), pp. 911–917.

[Li20] Zongyi Li et al. Fourier Neural Operator for Parametric Partial Differential Equations. 2021. arXiv: 2010.08895 [cs.LG].

[Gao21] Han Gao, Matthew J. Zahr, and Jian-Xun Wang. Physics-informed graph neural Galerkin networks: A unified framework for solving PDE-governed forward and inverse problems. 2021.

[Bar19] Yohai Bar-Sinai et al. “Learning data-driven discretizations for partial differential equations”. In: Proceedings of the National Academy of Sciences 116.31 (July 2019), pp. 15344–15349.

[Koch21] Dmitrii Kochkov et al. Machine learning accelerated computational fluid dynamics. 2020.

[Zhu20] Jiawei Zhuang et al. “Learned discretizations for passive scalar advection in a two-dimensional turbulent flow”.

References (3)

[Ran20] Rishikesh Ranade, Chris Hill, and Jay Pathak. DiscretizationNet: A Machine Learning based solver for Navier-Stokes Equations using Finite Volume Discretization. 2020.

[Rai19] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. “Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations”. In: J. Comput. Phys. 378 (2019), pp. 686–707.

[Titi90] Edriss S Titi. “On approximate Inertial Manifolds to the Navier-Stokes equations”. In: Journal of Mathematical Analysis and Applications 149.2 (1990), pp. 540–557.

[MAT] Reinforcement Learning Agents. Mathworks. Official MatLab Blog

[Hao20] Karen Hao “AI has cracked a key mathematical puzzle for understanding our world”, MIT, Technology Review

[Wang17] Prof. Qiqi Wang. Lecture notes of course Numerical Methods for PDEs. Last accessed on 2022-01-03. url:https://www.youtube.com/watch?v= prqI73vq9sk.

References (4)

[Bre19] Prof. Brenner Presentation of Data-driven Discretization Method. Last accessed on 2022-01-03. URL: https://www.youtube.com/watch?v=9Rycnz2O-aY&t=1031s

[Burg] Burger's Equation. Wikipedia Article. Last accessed on 2022-01-03. URL: https://en.wikipedia.org/wiki/Burgers%27_equation

Back-Up

Theory-guided Approaches: Roadmap

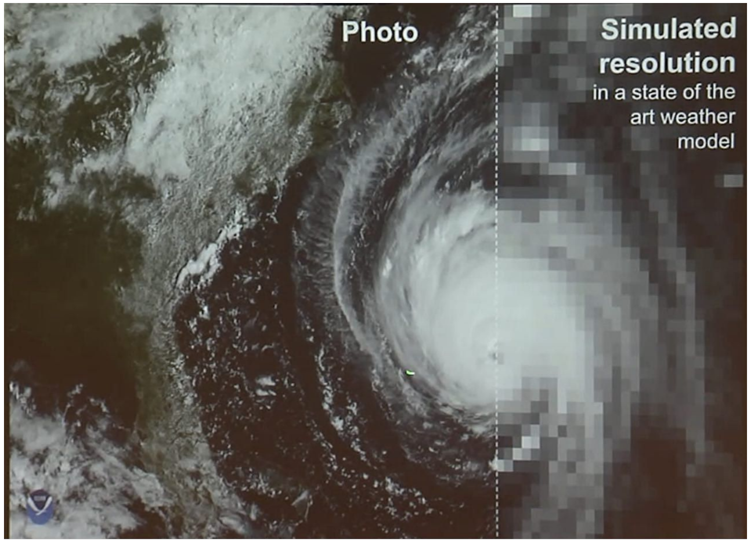
- Theory-guided: include underlying physics of the process
- Also formulated as *physics-informed* neural networks (PINNs)
- Research field in many areas

  Image?

- Linear vs. Non-linear vs. Quasi-linear
- Elliptic vs. Parabolic vs. Hyperbolic
- Continuous vs. discrete:
  - continuous: Fully-Connected NN approximate continuous function
  - discrete: CNN learns directly spatiotemporal solutions based on discretization scheme
    - Data-driven vs. Data-free
    - FDM vs. FVM vs. FEM

Data-driven Discretization

- Typical PDE, that simulates a physical process $$\frac{\partial v}{\partial t} + v\frac{\partial v}{\partial x} = \eta\frac{\partial^2 v}{\partial x^2}$$
- Discretize in space and represent on the mesh $$\frac{\partial^2 v}{\partial x^2} \approx \frac{v_{i+1} - 2v_i + v_{i-1}}{\Delta x^2}$$
- Results in a large set of ODEs to solve $$\frac{dv_i}{dt} + v_i\frac{v_i - v_{i-1}}{\Delta x} = \eta\frac{v_{i+1} - 2v_i + v_{i-1}}{\Delta x^2}$$
- Classical Applied Math: new *effective methods*



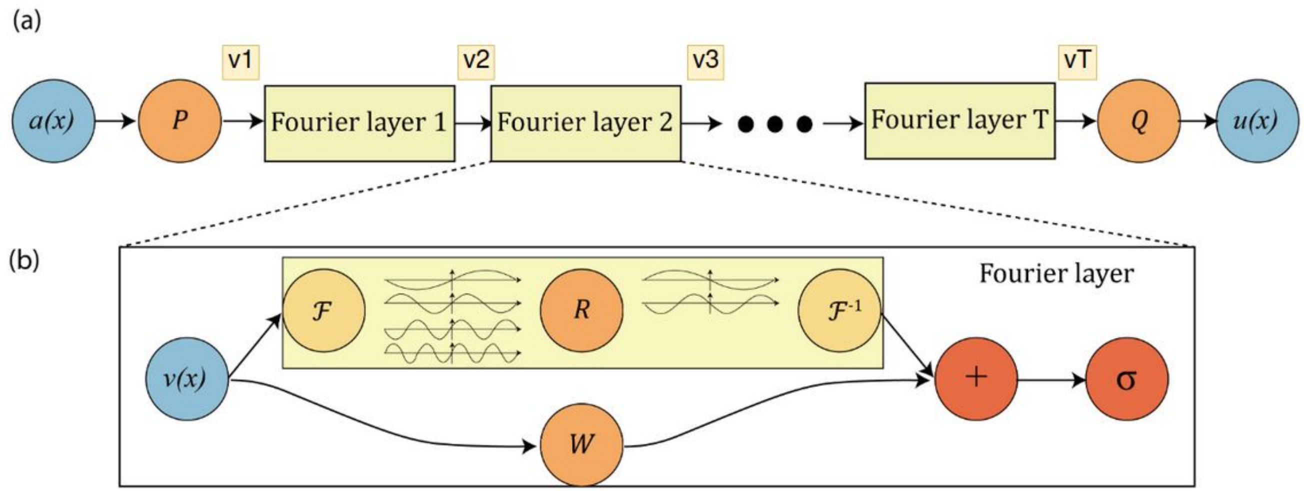
FNO Method with Formulas

- Fourier Layer:
  - Goal: Iterative update
  - Kernel Integral operator:
    - parameterized by NN
    - Convolve the space
  - Impose restrictions to get FNO:
- FNO:
  - Transform into Fourier Space and back
  - In Fourier Space: Convolving is just Multiplication with R

$$v_{t+1}(x) := \sigma\Big(W v_t(x) + \big(\mathcal{K}(a;\phi)v_t\big)(x)\Big)$$

$$\big(\mathcal{K}(a;\phi)v_t\big)(x) := \int_D \kappa\big(x, y, a(x), a(y); \phi\big) v_t(y)\mathrm{d}y,$$

$$\big(\mathcal{K}(\phi)v_t\big)(x) = \mathcal{F}^{-1}\Big(R_\phi \cdot (\mathcal{F}v_t)\Big)(x)$$



FNO: Training Data

- How to train this?
  - Generate data with classical numerical solver
  - Different types of IC & BC for PDE
- **Limitations**:
  - Need classical solver for dataset
  - Engineering choices:
    - # Fourier Layers
    - Cut away the top modes
  - Fourier applicable not for all types of problems