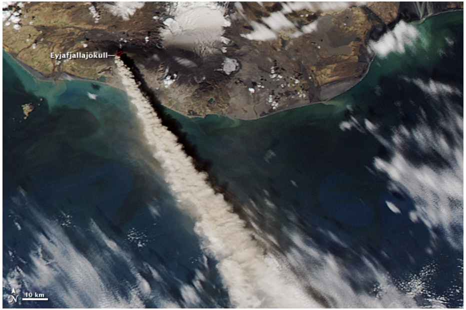
Leveraging physics information in neural networks for fluid flow problems

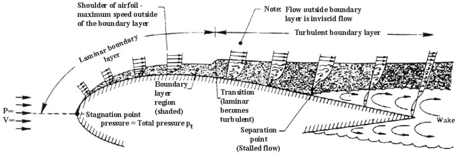
Akshay Subramaniam

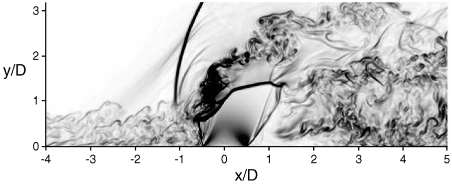
Dept. of Aeronautics & Astronautics
Stanford University

March 17, 2021

Turbulence

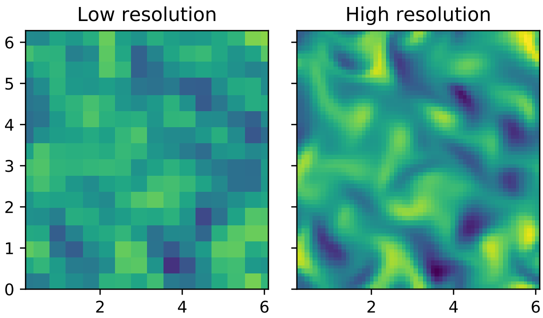

Eyjafjallajökull
10 km


Shoulder of airfoil - maximum speed outside of the boundary layer
Note: Flow outside boundary layer is inviscid flow
Turbulent boundary layer
Laminar boundary layer
Boundary layer region (shaded)
Transition (laminar becomes turbulent)
P_∞
V_∞
Stagnation point pressure = Total pressure p_t
Separation point (Stalled flow)
Wake


y/D
x/D

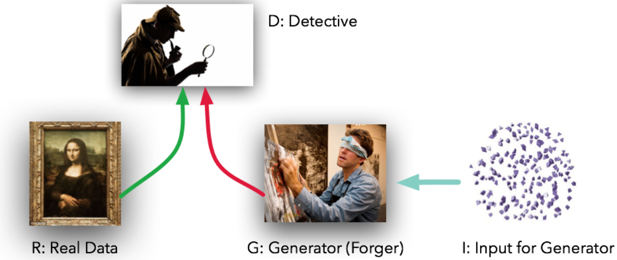
Turbulence Enrichment

Given a low-resolution turbulent flow field (LES), recover the high-resolution field (DNS) in a pointwise sense



Generative Adversarial Networks

- A class of generative models introduced by Goodfellow et. al.[1]

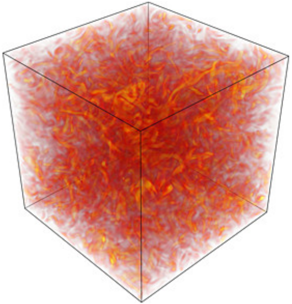


- Mainly used for generating photorealistic images
- Also used previously for scientific datasets[2][3]

[1] Ian Goodfellow et al. "Generative adversarial nets". In: *Advances in neural information processing systems.* 2014, pp. 2672–2680.
[2] Mustafa Mustafa et al. "Creating Virtual Universes Using Generative Adversarial Networks". In: *arXiv preprint arXiv:1706.02390* (2017).
[3] Shing Chan and Ahmed H Elsheikh. "Parametrization and Generation of Geological Models with Generative Adversarial Networks". In: *arXiv preprint arXiv:1708.01810* (2017).

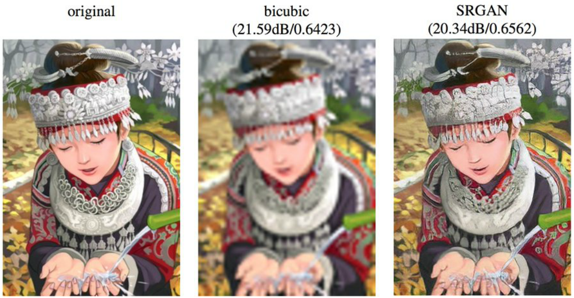
Data

- 4 3D fields: 3 components of velocity and pressure
- Each field is on a $64 \times 64 \times 64$ grid
- Low resolution data: filtered and downsampled to $16 \times 16 \times 16$
- Homogeneous Isotropic Turbulence (HIT) that is stationary in time
- A fairly low Reynolds number case to keep computational costs low



SRGAN

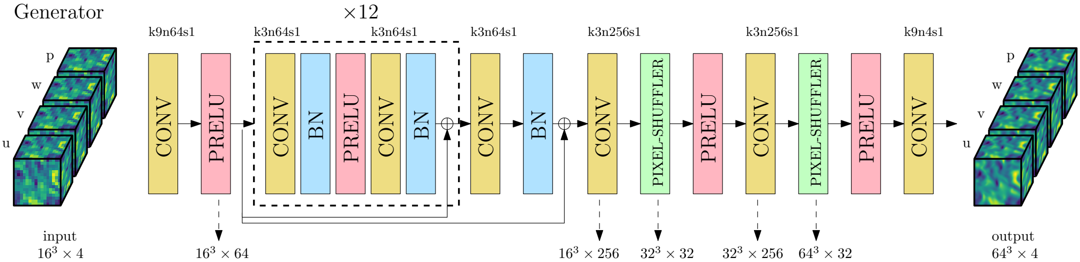
- SRGAN: super-resolution for images[4]



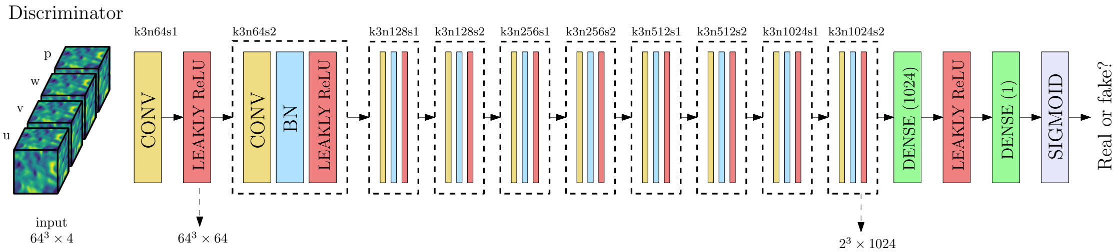
- Residual network with convolutional layers
- Two upsampling layers
- Model architecture used in this work is inspired by SRGAN

[4] Christian Ledig et al. "Photo-realistic single image super-resolution using a generative adversarial network". In: *arXiv preprint* (2016).

Generator architecture



Discriminator architecture



Physics informed learning

Stanford University

- The generated results need to be physical
- Respect conservation laws governing the system

$$\nabla \cdot \boldsymbol{u} = 0,$$
$$-\nabla^2 p = \nabla \boldsymbol{u} : \nabla \boldsymbol{u}^T$$

- Can do so by penalizing the generator using the residual of the equations above

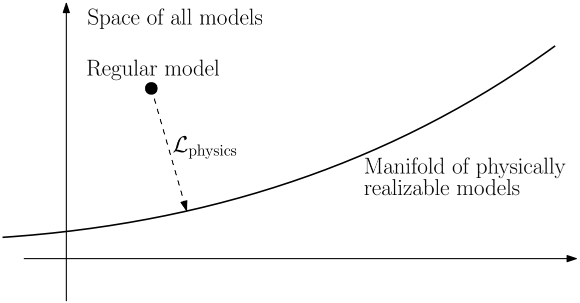
$$\mathcal{L}_{\text{continuity}} = \int_\Omega (\nabla \cdot \boldsymbol{u})^2 \, \mathrm{d}\Omega,$$
$$\mathcal{L}_{\text{pressure}} = \int_\Omega \left(\nabla^2 p + \nabla \boldsymbol{u} : \nabla \boldsymbol{u}^T\right)^2 \mathrm{d}\Omega$$

Physics loss

- Add a physics loss to the generator training

$$\mathcal{L}_{\text{physics}} = (1 - \lambda_{\text{C}})\,\mathcal{L}_{\text{pressure}} + \lambda_{\text{C}}\mathcal{L}_{\text{continuity}}$$

- Enforces better compatibility with physics
- Acts as a regularizer for the model



Loss function

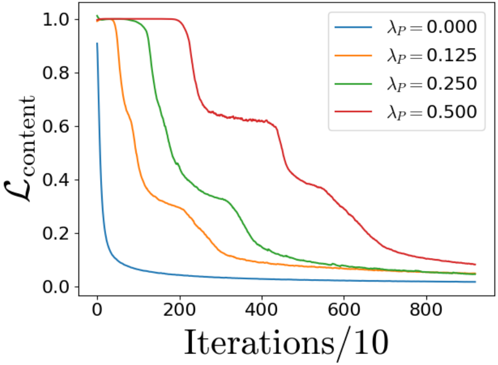
- The loss function $\mathcal{L}_{\mathrm{GAN}}$ is given by

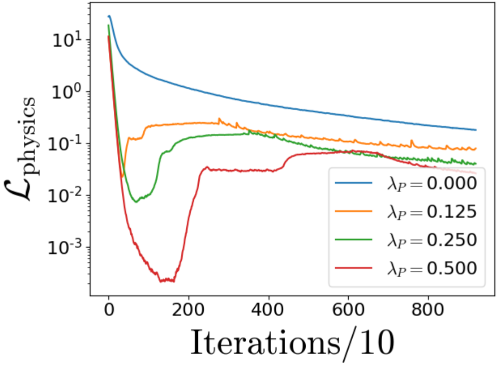
$$\begin{aligned}
\mathcal{L}_{\mathrm{GAN}} &= (1-\lambda_{\mathrm{A}})\,\mathcal{L}_{\mathrm{resnet}} + \lambda_{\mathrm{A}}\mathcal{L}_{\mathrm{adversarial}} \\
\mathcal{L}_{\mathrm{resnet}} &= (1-\lambda_{\mathrm{P}})\,\mathcal{L}_{\mathrm{content}} + \lambda_{\mathrm{P}}\mathcal{L}_{\mathrm{physics}} \\
\mathcal{L}_{\mathrm{content}} &= (1-\lambda_{\mathrm{E}})\,\mathcal{L}_{\mathrm{MSE}} + \lambda_{\mathrm{E}}\mathcal{L}_{\mathrm{enstrophy}} \\
\mathcal{L}_{\mathrm{physics}} &= (1-\lambda_{\mathrm{C}})\,\mathcal{L}_{\mathrm{pressure}} + \lambda_{\mathrm{C}}\mathcal{L}_{\mathrm{continuity}}
\end{aligned}$$

- MSE and enstrophy sensitize the model to large and small scale features respectively
- Four hyperparameters to tune: λ_{A}, λ_{P}, λ_{E} and λ_{C}

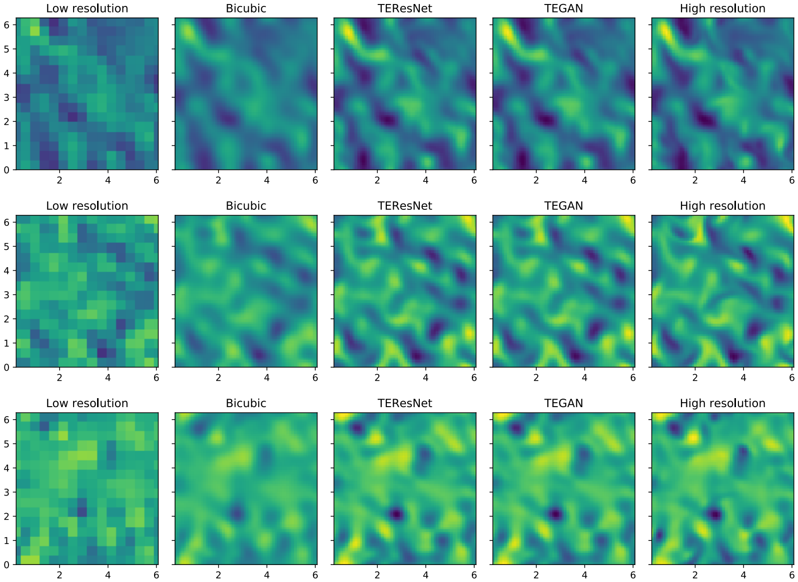
Impact of physics loss on training

Stanford University





- Increasing λ_{P} improves the physics loss by an order of magnitude but compromises content loss
- Higher λ_{P} creates a strong local minimum at the trivial solution
- Choose $\lambda_{\mathrm{P}} = 0.125$ as a compromise


Low resolution
Bicubic
TEResNet
TEGAN
High resolution
Low resolution
Bicubic
TEResNet
TEGAN
High resolution
Low resolution
Bicubic
TEResNet
TEGAN
High resolution

Evaluation

Stanford University

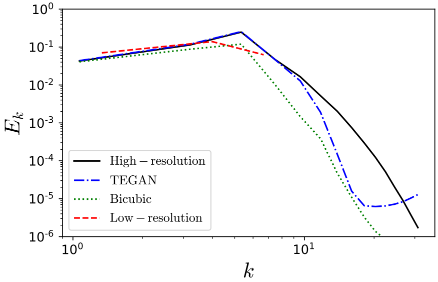
	$\mathcal{L}_{\text{content}}$		$\mathcal{L}_{\text{physics}}$	
	Dev	Test	Dev	Test
TEResNet	0.049	0.050	0.078	0.085
TEGAN	0.047	0.047	0.070	0.072
% Difference	4.1	6.0	10.3	15.2

- TEGAN has consistently lower content and physics losses
- TEGAN also generalizes to the dev set better

Energy spectra

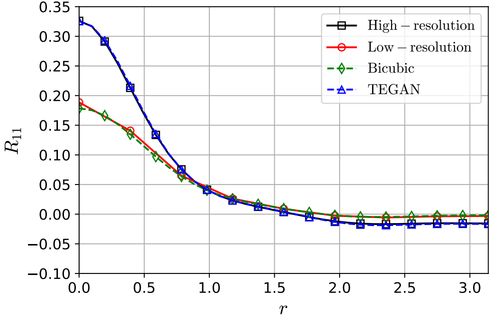
Stanford University

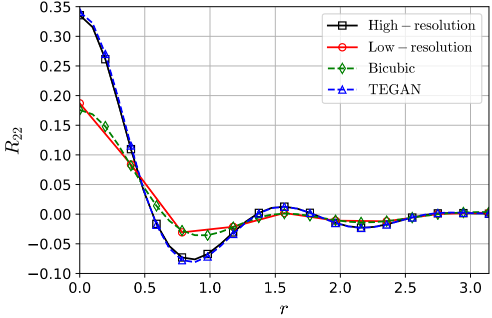
- Energy spectra of the velocity field is a fundamental statistical quantity in turbulence



- The low resolution data is very coarse
- TEGAN is able to recover the spectrum very well in the intermediate wavenumbers
- Not as effective at capturing the finest dissipative scales

Second order two-point correlations

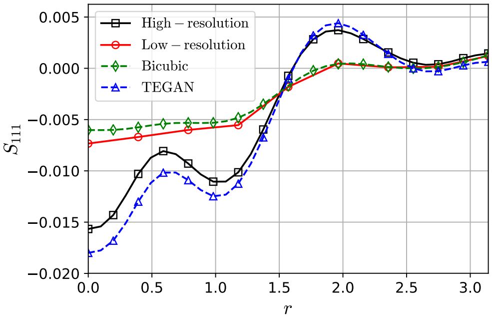


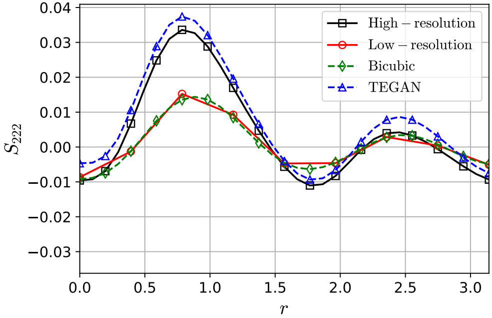


- $R_{11}(r = 0)$ and $R_{22}(r = 0)$ are the variances of the longitudinal and transverse velocity components respectively
- TEGAN captures both the longitudinal and transverse correlations virtually perfectly

Third order two-point correlations

Stanford University

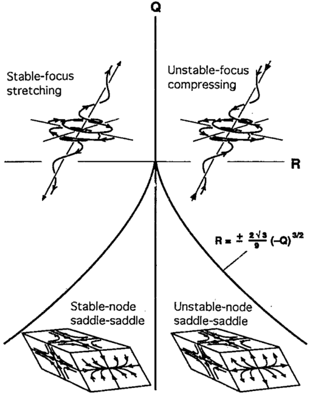




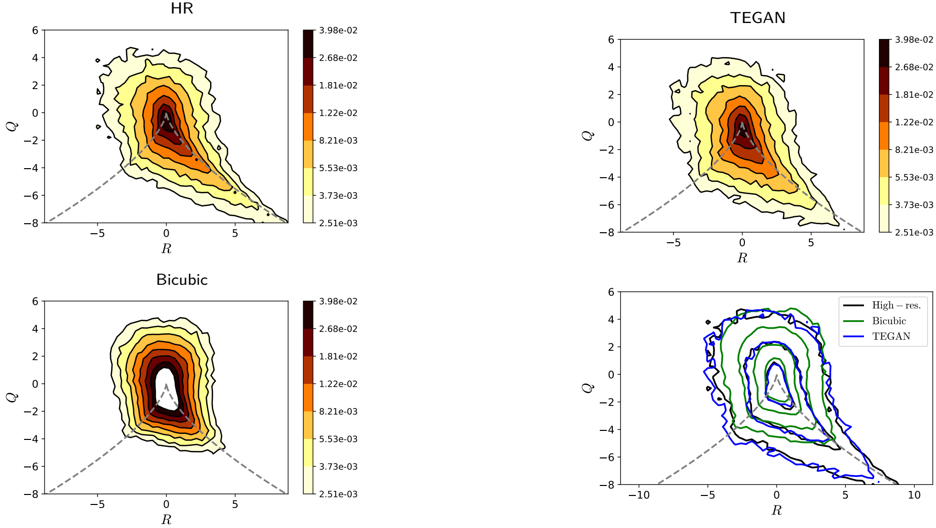
- TEGAN captures the qualitative structure of third order correlations
- $\approx 15\%$ overprediction of the longitudinal correlations
- Third order correlations are harder for models to capture

QR diagram

- The velocity gradient tensor gives a picture of the local flow structure
- Represent using the second and third invariants of the tensor: Q and R



QR diagram


HR
TEGAN
Bicubic
Q
R
3.98e-02
2.68e-02
1.81e-02
1.22e-02
8.21e-03
5.53e-03
3.73e-03
2.51e-03
High – res.
Bicubic
TEGAN

Acknowledgements

- Man-Long Wong
- Raunak Borker
- Sravya Nimmagadda
- Sanjiva Lele

Physics-Informed Neural Networks (PINNs)

http://developer.nvidia.com/simnet

SOLVING PDEs WITH PINNs

Neural network approximates solution to partial differential equation.

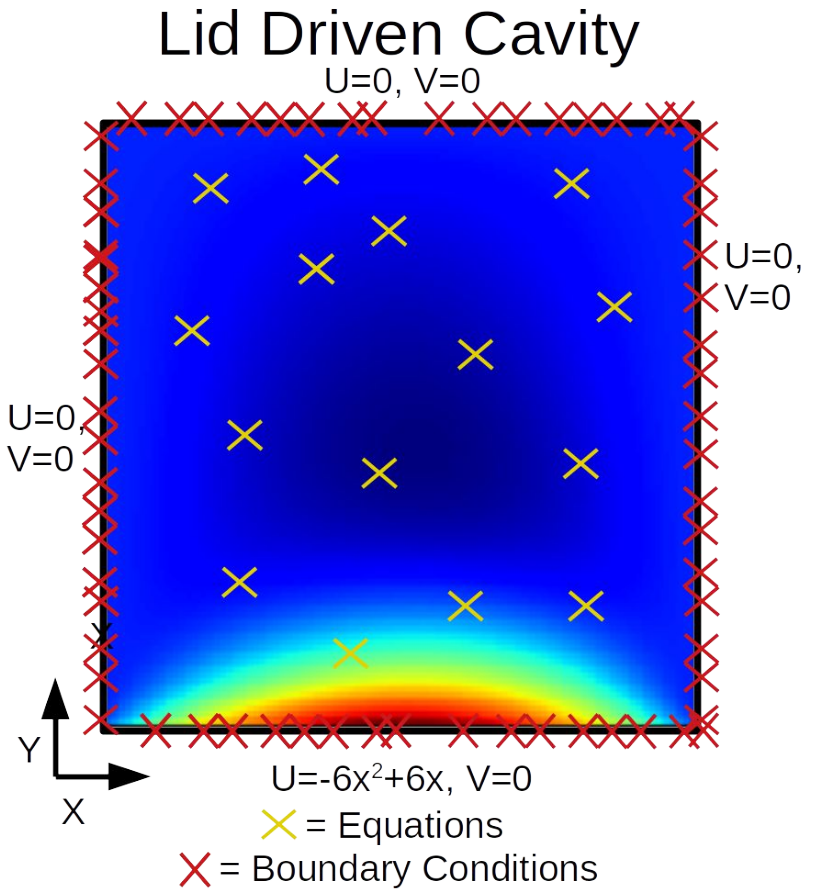
$$u_{net}(x, y) \rightarrow (u, v, p)$$

Minimize loss from boundary conditions and equations.

$$0 = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}$$

$$0 = u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + \frac{\partial p}{\partial x} - \nu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right)$$

$$0 = u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + \frac{\partial p}{\partial y} - \nu\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right)$$



NEURAL NETWORK SOLVER METHODOLOGY

The idea is to use a neural network to approximate the solution to given differential equation and boundary conditions.
Example Problem,

$$\mathbf{P} : \begin{cases} \frac{\delta^2 u}{\delta x^2}(x) = f(x), \\ u(0) = u(1) = 0, \end{cases} \tag{1}$$

Construct a deep multi-layer perception $u_{net}(x) \rightarrow u$. $x \in \mathbb{R}$. Assume that $u_{net} \in C^\infty$. This means using activation functions like *tanh*, *swish*, *sin*, *sigmoid*... [1]

NEURAL NETWORK SOLVER METHODOLOGY

Construct a Loss function to train $u_{net}(x)$. We can compute $\frac{\delta^2 u_{net}}{\delta x^2}(x)$ using automatic differentiation.

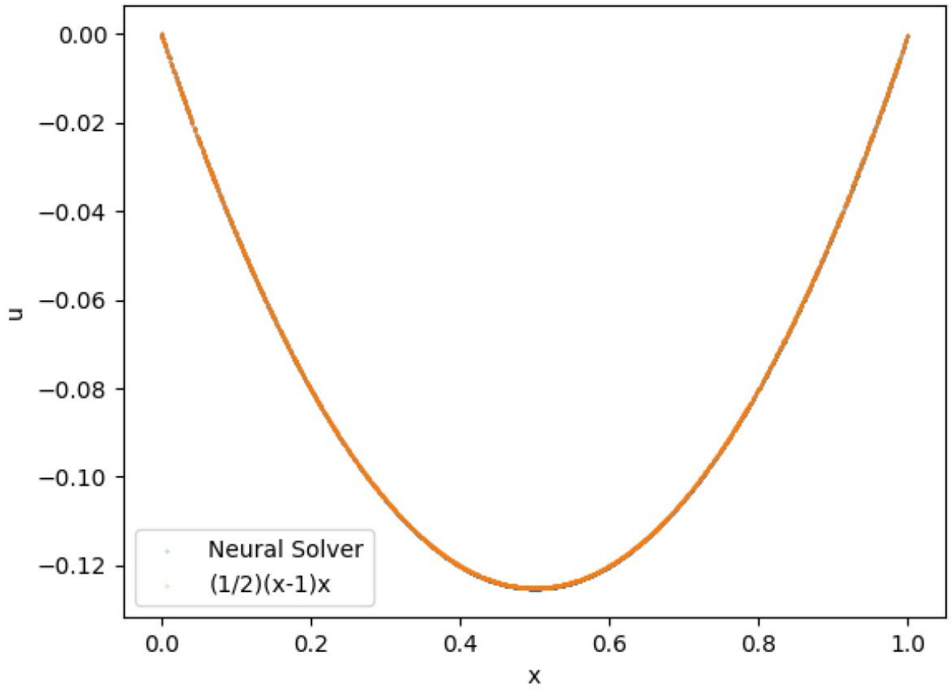
$$L_{BC} = u_{net}(0)^2 + u_{net}(1)^2 \quad (2)$$

$$L_{residual} = \frac{1}{N}\sum_{i=0}^{N}(\frac{\delta^2 u_{net}}{\delta x^2}(x_i) - f(x_i))^2; x_i \in (0,1) \quad (3)$$

$$L = L_{BC} + L_{residual} \quad (4)$$

NEURAL NETWORK SOLVER METHODOLOGY

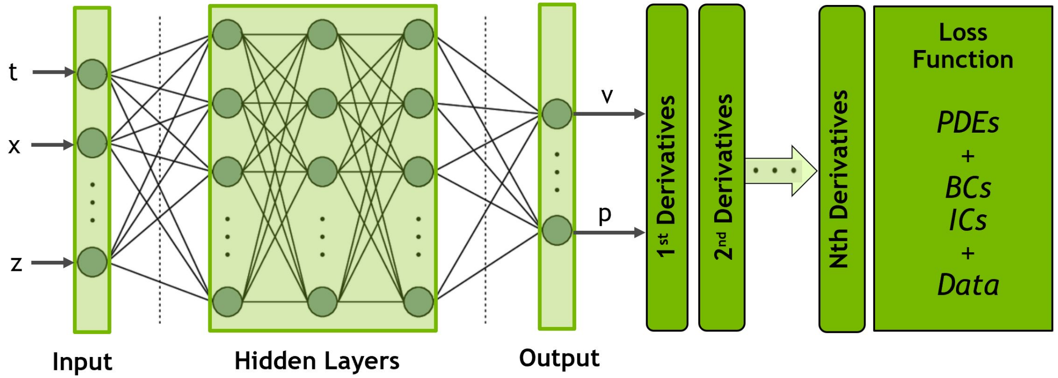
$$f(x) = 1$$



Physics Informed Neural Network

Architectures

- Fully Connected (FC)



- Fourier Features (FN) – Axis, Partial, Full or Random Spectrum
- Sinusoidal Representation (SiReNs)
- Modified Fourier Features (mFN)
- Deep Galerkin Method (DGM)
- Modified Highway Networks

INVERSE PROBLEM

Inverse problems start with effects and then calculate the causes. Suppose we are given the solution $u_{true}(x)$ at 100 random points between 0 and 1 and we want to determine the $f(x)$ that is causing it.

$$L_{residual} \approx (\int_0^1 dx)\frac{1}{N}\sum_{i=0}^{N}(\frac{\delta^2 u_{net}}{\delta x^2}(x_i) - f_{net}(x_i))^2; x_i \in (0,1) \tag{10}$$

$$L_{data} = \frac{1}{100}\sum_{i=0}^{100}(u_{net}(x_i) - u_{true}(x_i))^2 \tag{11}$$

NVIDIA

INVERSE PROBLEM

For $u_{true}(x) = \frac{1}{48}(8x(-1+x^2) - (3sin(4\pi x))/\pi^2)$ the solution for $f(x)$ is $x + sin(4\pi x)$.

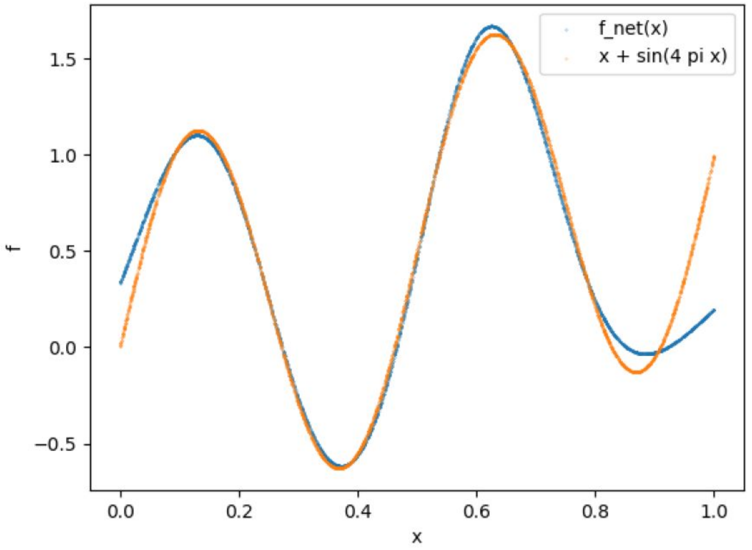


Figure: Comparison of true solution for $f(x)$ and the function inverted out.

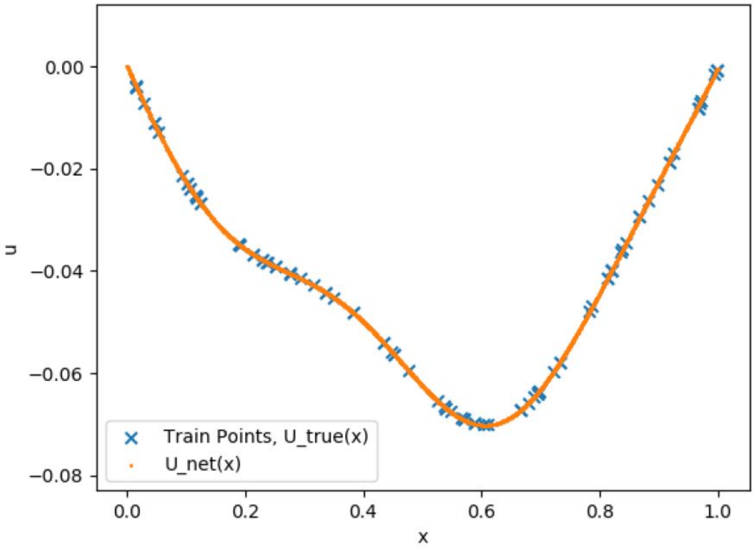


Figure: Comparison of $u_{net}(x)$ and train points from u_{true}.

SOLVING PARAMETERIZED PDES

We can solve parameterized geometries [2]. Suppose we want to know how the solution changes as we move the position on the boundary condition $u(1) = 0$.

$$\mathbf{P}: \begin{cases} \frac{\delta^2 u}{\delta x^2}(x) = f(x), \\ u(0) = u(l) = 0, l \in [1, 2] \end{cases} \tag{7}$$

Now train a network $u_{net}(x, l)$ on the losses,

$$L_{residual} \approx \left(\int_1^2 \int_0^l dx dl\right) \frac{1}{N} \sum_{i=0}^{N} \left(\frac{\delta^2 u_{net}}{\delta x^2}(x_i, l_i) - f(x_i)\right)^2 \tag{8}$$

$$L_{BC} \approx \left(\int_1^2 dl\right) \frac{1}{N} \sum_{i=0}^{N} (u_{net}(0, l_i))^2 + (u_{net}(l_i, l_i))^2 \tag{9}$$

SOLVING PARAMETERIZED PDES

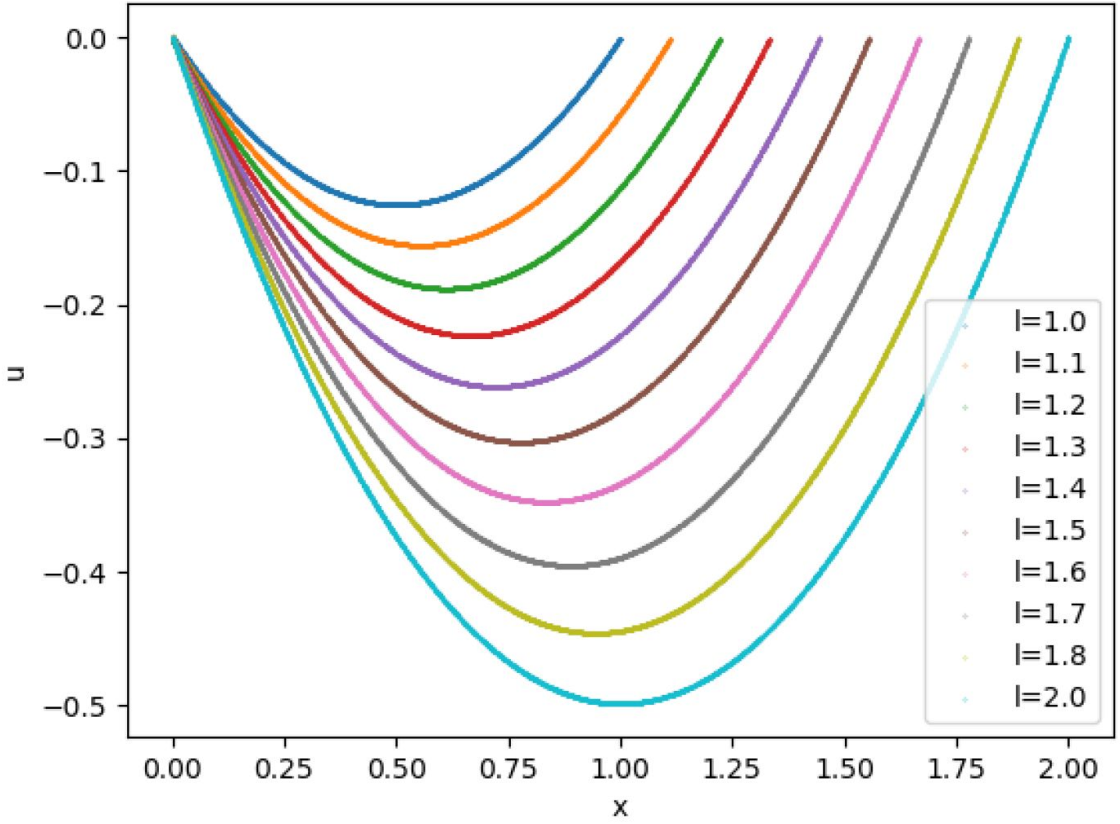
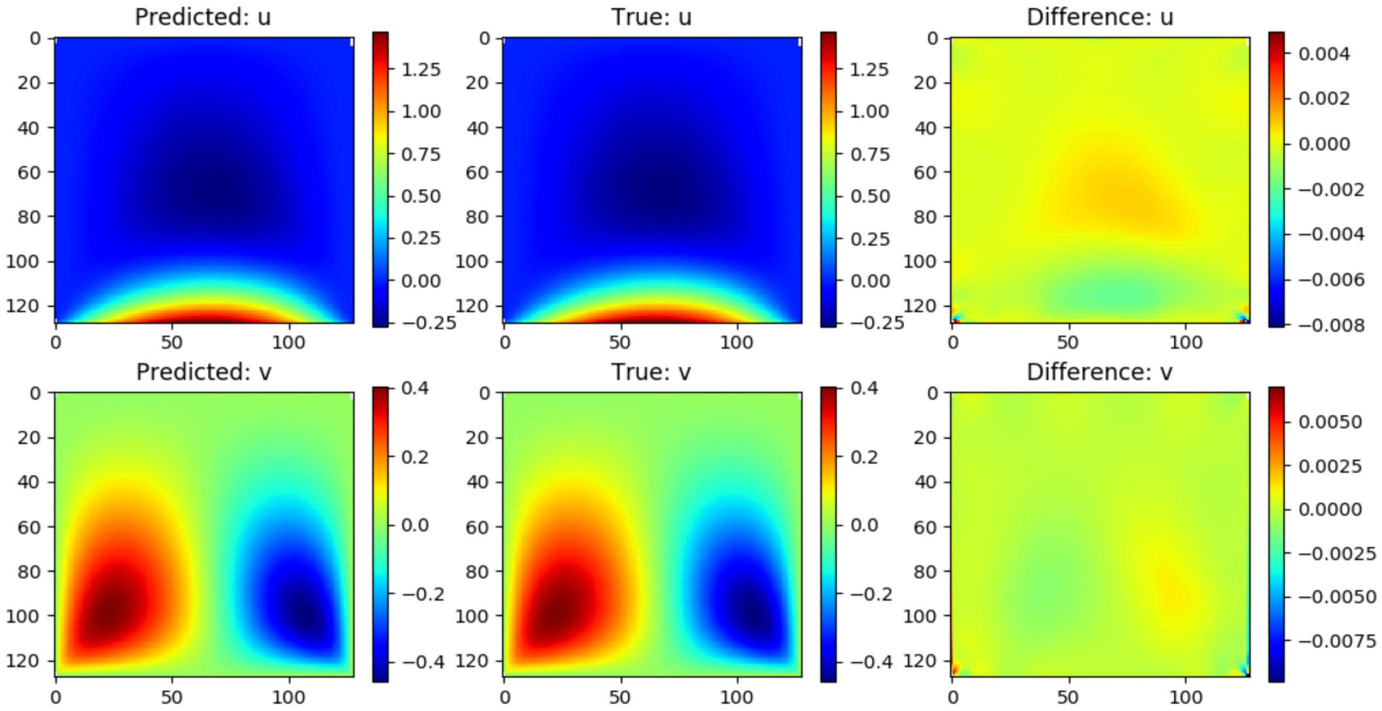


Figure: SimNet solving parameterized differential equation problem.

LID DRIVEN CAVITY

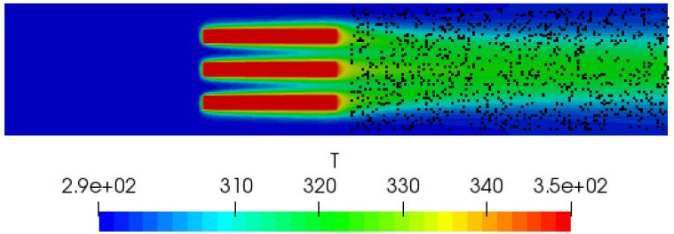
SimNet versus OpenFOAM



U velocity difference = 0.2%
V velocity difference = 0.4%

INVERSE PROBLEM APPLICATION

Finding Unknown Coefficients of a PDE: Heat Sink



Fluid Heat Convection: $0 = \nabla \cdot (D_{fluid} \nabla \theta_{fluid}) - \nabla \cdot (U \theta_{fluid})$ $\quad D_{fluid} = \frac{k_{fluid}}{\rho_{fluid} c_{pfluid}}$

Solid Heat Conduction: $0 = \nabla \cdot (k_{solid} \nabla \theta_{solid})$ $\quad D_{solid} = \frac{k_{solid}}{\rho_{solid} c_{psolid}}$

Interface Conditions:

$$\theta_{solid} = \theta_{fluid}$$

$$k_{solid}(N \cdot \nabla \theta_{solid}) = k_{fluid}(N \cdot \nabla \theta_{fluid})$$

Results:

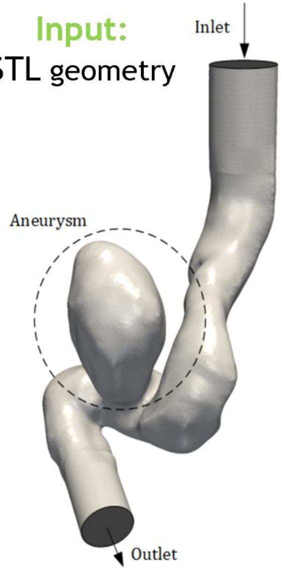
Property	OpenFOAM (True)	SimNet (Predicted)
Kinematic Viscosity (m^2/s)	1.00×10^{-2}	1.03×10^{-2}
Thermal Diffusivity (m^2/s)	2.00×10^{-3}	2.19×10^{-3}

TRANSIENT: MEDICAL IMAGING OF AN ICA

A Data Assimilation Problem

https://www.youtube.com/watch?v=QjY_8xFjsgE

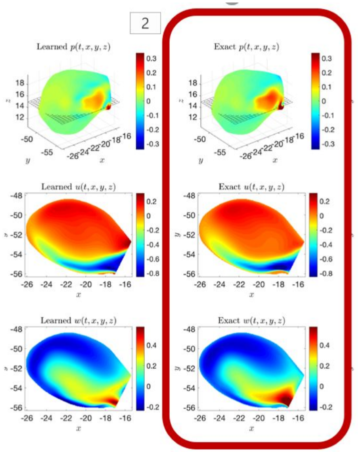
Input:
STL geometry



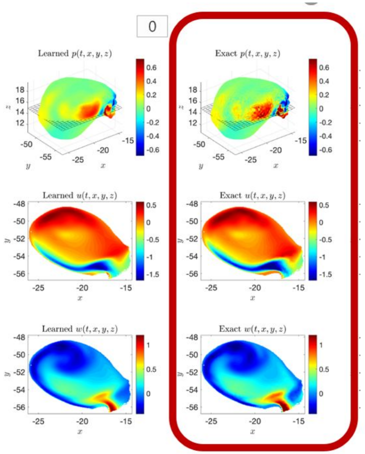
Intracranial Cerebral Aneurysm

SimNet

Nvidia SDF library (Ray Tracing - TC) → **Output:** Point Cloud, location relative to surface, SDF & SDF derivatives for SimNet → **Neural Physics Solver**



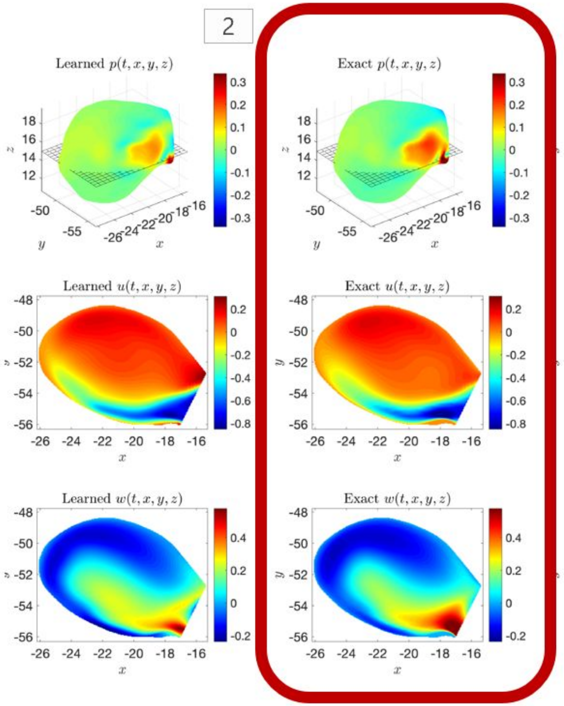
SimNet vs. OpenFOAM



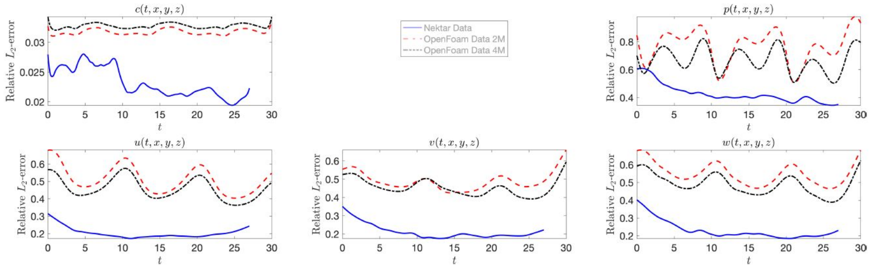
SimNet vs. Nektar++

ICA — COMPARISON BETWEEN SIMNET & CFD SOLVERS

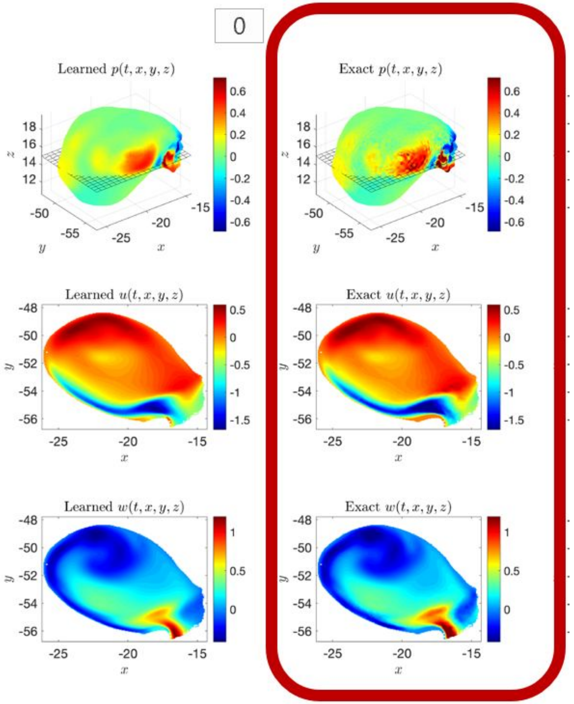
Inverse Solution



OpenFOAM vs.
Neural Networks



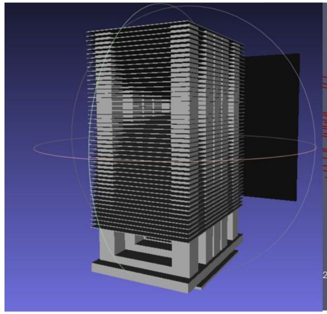
Nektar++ is a higher fidelity solver (implicit, h- & p- method based finite element CFD code) and provides higher quality results with less diffusion



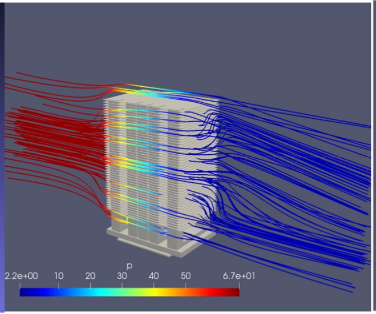
Nektar++ vs.
Neural Networks

NVIDIA DGX-A100 NVSWITCH HEAT SINK

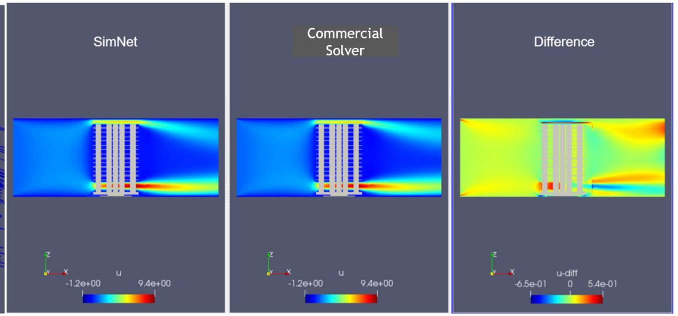
Multi-Physics Application: Fluids + Heat Transfer



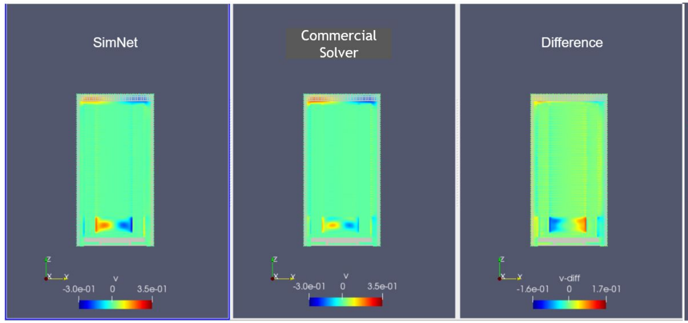
Nvidia DGX A100 Heatsink



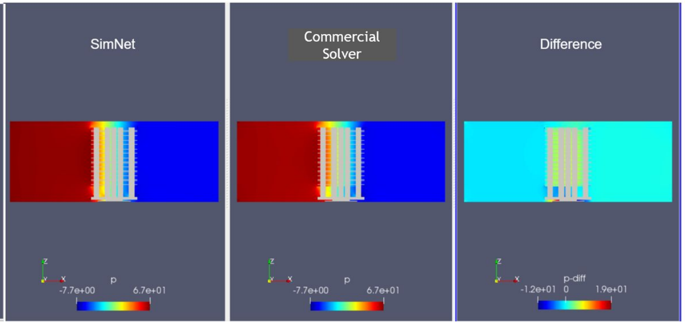
Pressure color coded flow streamlines



U-velocity (transverse direction) comparison



V-velocity (transverse direction) comparison

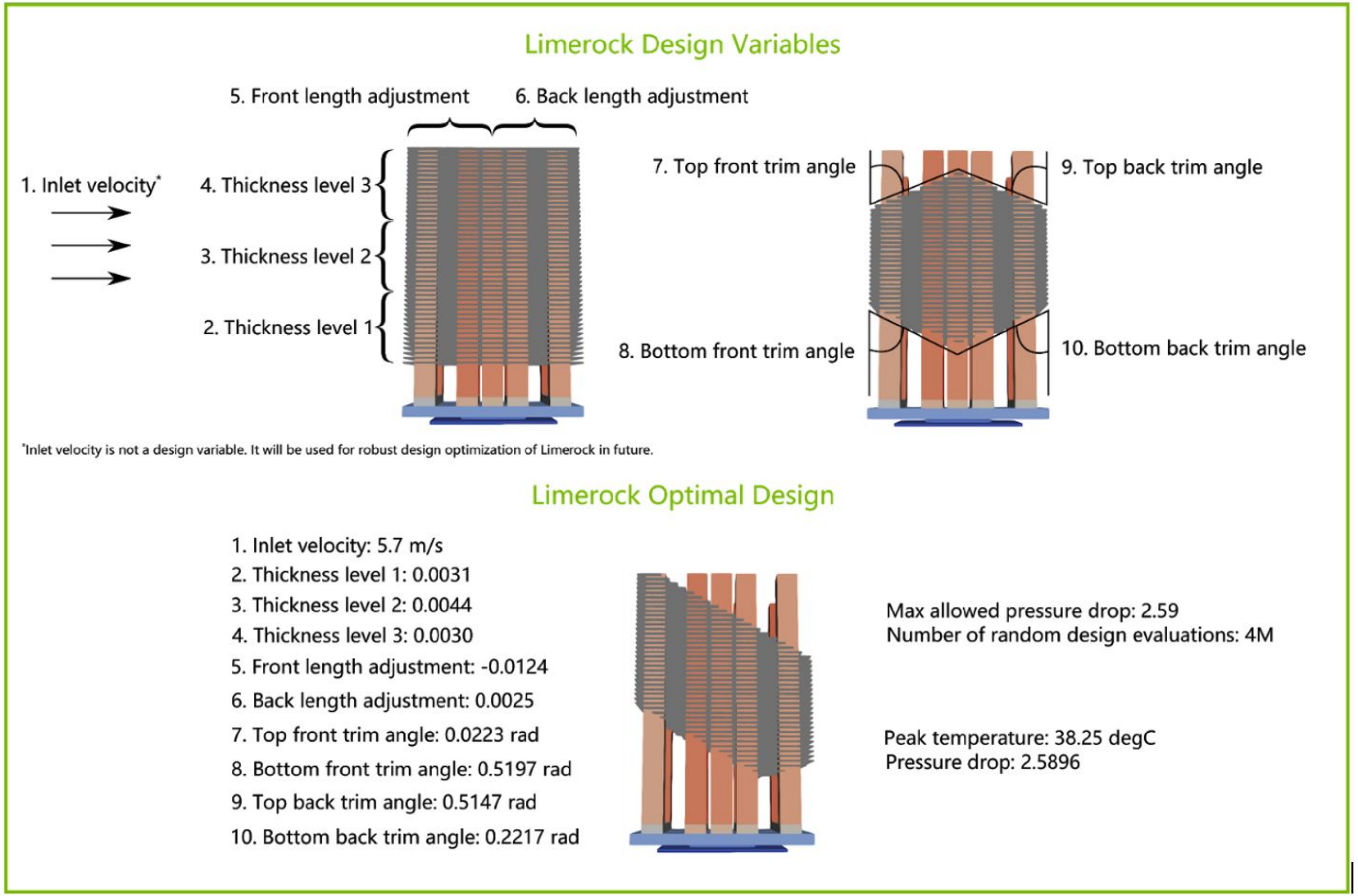


Pressure comparison

Turbulent Flow (Re=20,687)		
	Temperature	Pressure Drop
SimNet - Fourier Network	43.1°C	3.56
Commercial Solver	43.5 °C	3.6
OpenFOAM	41.6 °C	4.58

PARAMETERIZED DGX-A100 NVSWITCH HEAT SINK

Multi-Physics Application: Fluids + Heat Transfer



Computational Times (10 parameters, 3 values per parameter)	
SimNet	1000 V100 GPU hrs.
Traditional Solver (OpenFOAM) 59,049 separate runs (26 wall hours on 12 CPU cores)	18.4 Million CPU hrs.

Co-authors:

- Oliver Hennigh
- Susheela Narasimhan
- Mohammad Amin Nabian
- Kaustubh Tangsali
- Max Rietmann
- Jose del Aguila Ferrandis
- Wonmin Byeon
- Zhiwei Fang
- Sanjay Choudhry

NVIDIA®