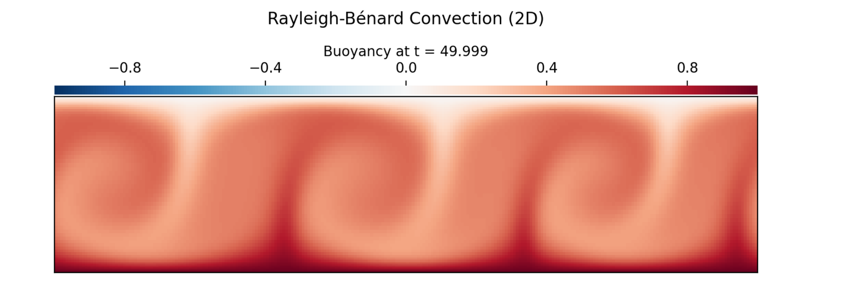
Harnessing Fourier Neural Operator For Rayleigh–Bénard Convection

Chelsea John*°, Andreas Herten*, Stefan Kesselheim*, Daniel Ruprecht°

***Jülich Supercomputing Center, °Technical University of Hamburg**

Rayleigh-Bénard Convection (RBC)

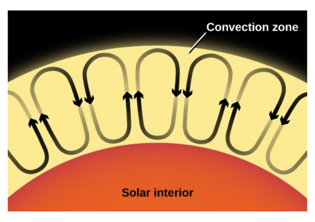
- Natural buoyancy-driven convective motion occurring in fluid layers due to temperature gradient
- Modelled by incompressible Navier-Stokes equations under Boussineq approximation



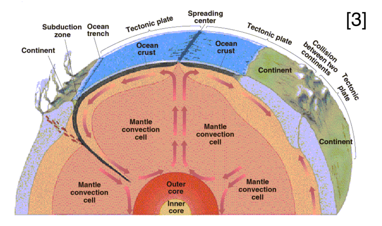
Simulation: Two-dimensional Rayleigh–Bénard convection with Rayleigh Number Ra = 10^4 at time t = 49.99, generated using Dedalus software[1]

[1]: https://dedalus-project.org/

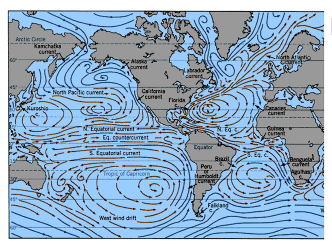
Applications



Astrophysics [2]



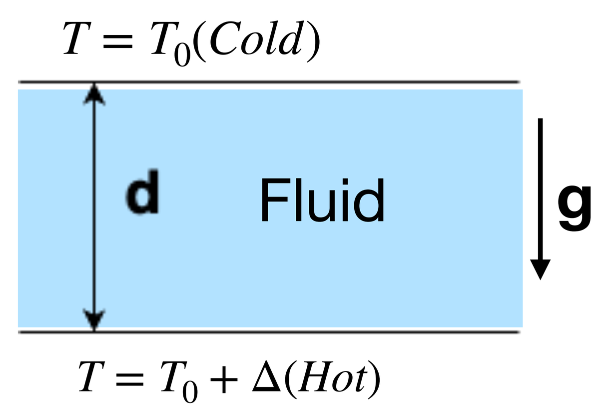
Geothermal Dynamics [3]



Atmospheric Flows [4]

[2]: LumenLearning - The Solar Interior: Theory
[3]: UniversityOfSydney - Geothermal Energy
[4]: MyNasaData - Ocean Circulation Pattern

RBC Model Equations



Dimensionless Equations

$$\frac{1}{Pr}\left(\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u}\right) = -\nabla p + \theta\hat{\mathbf{z}} + \nabla^2\mathbf{u}$$

$$\frac{\partial\theta}{\partial t} + (\mathbf{u}\cdot\nabla)\theta = Ra(\mathbf{u}\cdot\hat{\mathbf{z}}) + \nabla^2\theta$$

$$\nabla\cdot\mathbf{u} = 0$$

Where:

- $\mathbf{u} = (u_x, u_y, u_z)$: Flow velocity
- θ: Temperature deviation from conduction state
- p: Pressure deviation from conduction state
- Δ: Temperature difference between boundaries
- ν: Kinematic viscosity of fluid
- α: Thermal expansion coefficient
- κ: Thermal diffusivity of fluid
- $Pr = \nu/\kappa$: Prandtl number
- $Ra = \rho_0 \alpha g \Delta d^3/\nu\kappa$: Rayleigh Number

Fourier Neural Operators (FNO)[5]

- Neural Operators are able to learn function space mappings
- Neural Operator when parametrised by integral kernel in **Fourier space** gives Fourier Neural Operator
- **Resolution-invariant**
- **Quasi-linear complexity** with FFT

Let v be input vector and u output vector, then deep neural network with K_i layer and σ_i activation has the form:

$$u = (K_l \circ \sigma_l \circ \ldots \circ \sigma_1 \circ K_0)v$$

For Fourier Neural operator, v and u are functions with discretisation. Let x, y be points in domain D then $K : v_t \mapsto v_{t+1}$ is parameterised as:

$$v_{t+1}(x) = \mathcal{F}^{-1}\left(\mathcal{F}(k)\cdot\mathcal{F}(v_t)\right)(x) + Wv_t(x),\ \forall x \in D$$

Where k is a periodic function in D and , $\mathcal{F}, \mathcal{F}^{-1}$ are Fourier transform and it's inverse respectively

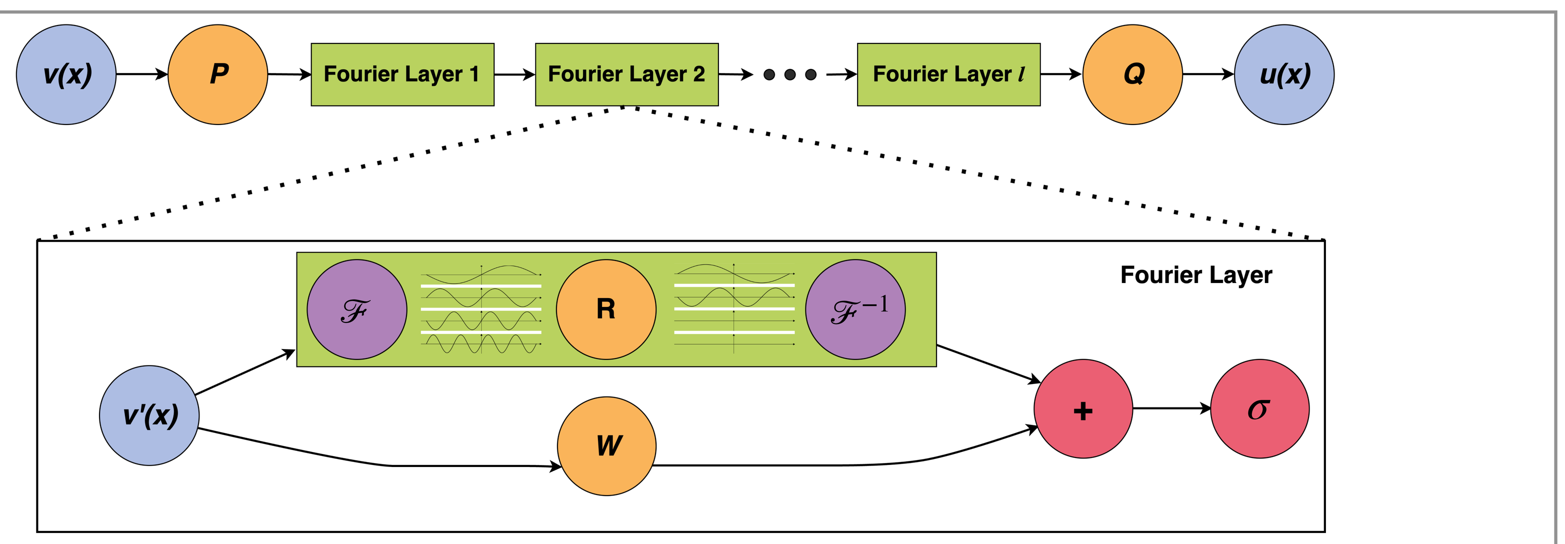


Fig: Fourier neural operator architecture with input v, lifted to higher channel space by neural network P, passed through Fourier layers and activation function σ, then projected back to target dimension by neural network Q to give output u. Fourier layer takes input v' and applies Fourier transform $\mathcal{F}$, linear transform R on lower Fourier modes, filtering out higher modes; applies inverse Fourier transform $\mathcal{F}^{-1}$ then concatenates the output with local linear transform W which is passed through activation function σ.

[5]: https://arxiv.org/abs/2010.08895

Solving RBC in 2D with FNO

- **Problem**: Given $\mathcal{S}_t$ (state at t), predict $\mathcal{S}_{t'}$ (state at $t' = t + \delta t$)
- **FNO Model Input**: $\mathcal{S}_t = (u_{x,t}, u_{z,t}, p_t, b_t)$
- **FNO Model Output**: $\mathcal{S}_{t'} = (u_{x,t'}, u_{z,t'}, p_{t'}, b_{t'})$
- **Code**: Implemented in Python using PyTorch
- **Device**: NVIDIA A100 (40GB) GPU[6]

Where:
$u_{x,t}$: velocity x-component at time t
$u_{z,t}$: velocity z-component at time t
p_t: pressure at time t
b_t: buoyancy at time t

Data Card

Source: Dedalus[1]
Samples: {train:1499, val:1600, test:1600}
Initial state, $\mathcal{S}_0$= (0, 0 , 0, random.normal())
Grid(x, z): (64,64)
Pr: 1
Ra: 10^4

Training Parameters

Learning_rate: 0.00039
Optimiser: AdamW
Loss: torch.nn.SmoothL1Loss()
Epoch: 2000
Batch_size: 30

FNO Model Card

Fourier_layer: 2
x_fourier_modes: 32
z_fourier_modes: 32
Activation: ReLU
layer_width: 128
projection_width: 32

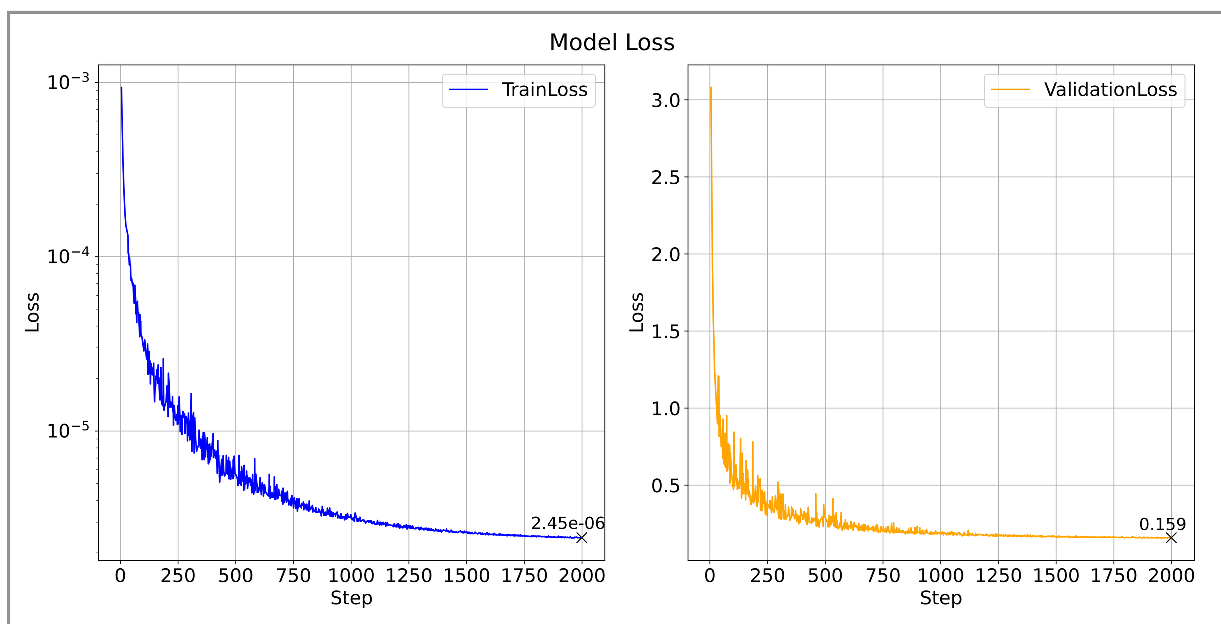


Fig (Left): FNO model training loss on y-axis (log scale) and epoch step on x-axis for 2D RBC problem with moderate turbulence $Ra = 10^4$. Trained till loss $\approx \mathcal{O}(10^{-6})$

Fig (Right): FNO model validation loss on y-axis and epoch step on x-axis for 2D RBC problem with moderate turbulence $Ra = 10^4$. Validation loss $\approx \mathcal{O}(10^{-1})$

[6]: JUWELS Booster, https://jlsrf.org/index.php/lsf/article/view/171

FNO Result For 2D RBC

For $t \in [0,374.256]$ (In Training Distribution Data)

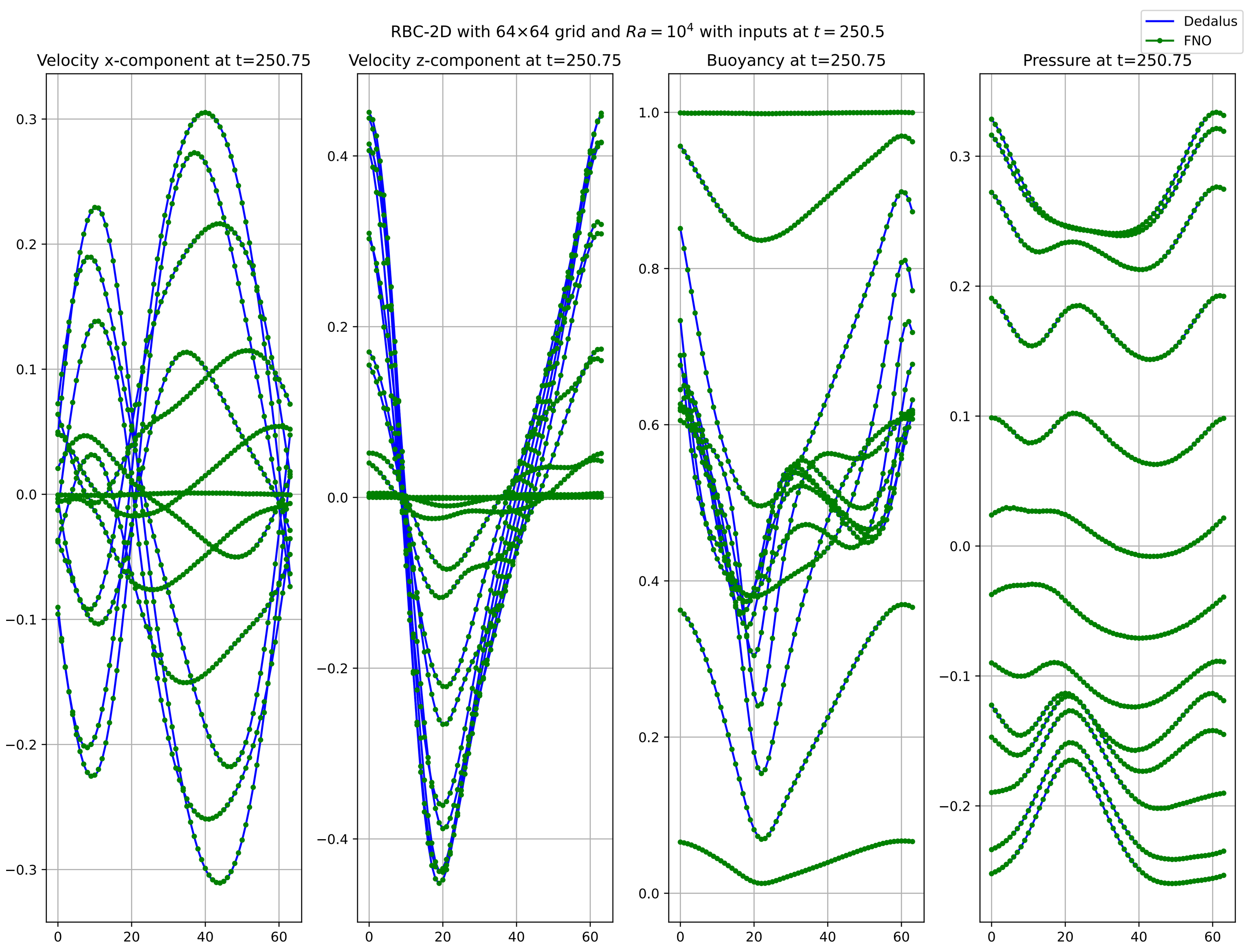


Fig: Velocity, Buoyancy and Pressure output of FNO model against Dedalus result at time $t = 250.75 \in [0,374.256]$ on 64×64 grid for 2D RBC problem with moderate turbulence $Ra = 10^4$. Dedalus output and FNO model output match with error $\approx \mathcal{O}(10^{-5})$

For $t \notin [0,374.256]$ (Out Of Training Distribution Data)

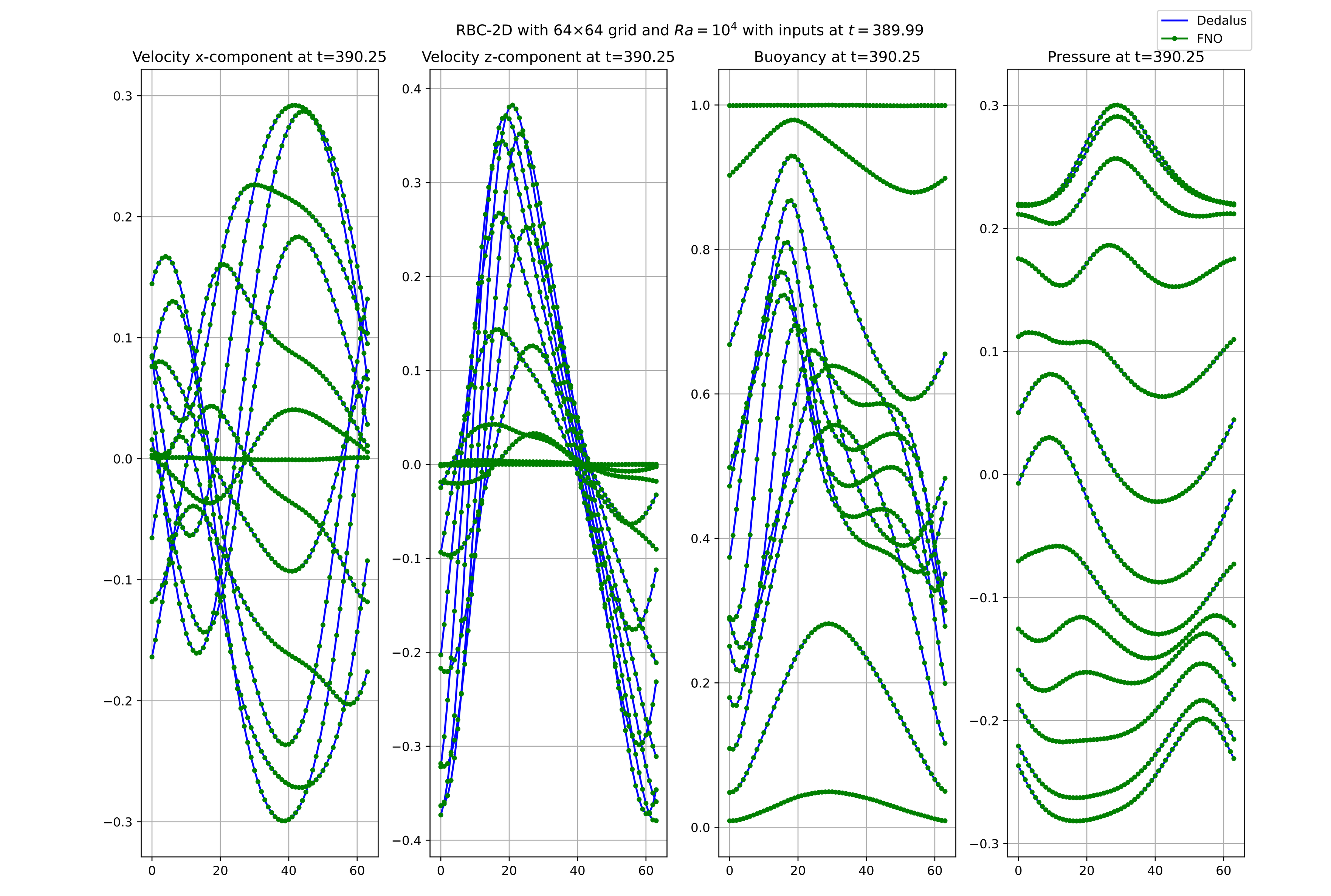


Fig: Velocity, Buoyancy and Pressure output of FNO model against Dedalus result at time $t = 390.25 \notin [0,374.256]$ on 64×64 grid for 2D RBC problem with moderate turbulence $Ra = 10^4$. Dedalus output and FNO model output matches with error $\approx \mathcal{O}(10^{-5})$

Conclusion

- FNO shows promising results for solving two dimensional Rayleigh-Bénard Convection
- FNO model does not give results for initial evolution of system when Kinetic Energy < $\mathcal{O}(10^{-5})$
- Verification through Dedalus (PDE solver)

Next Steps

- Solve Rayleigh-Bénard Convection for high turbulence with $Ra > 10^4$ and liquids with $Pr > 1$ or < 1
- Solve Rayleigh-Bénard Convection in three dimensions
- Optimise memory storage
- Optimise dense convolution in Fourier space
- Implement parallel training strategies

Acknowledgements

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. Compute time on the GCS Supercomputer JUWELS Booster at JSC is provided through the Gauss Center for Supercomputing e.V.

JÜLICH Forschungszentrum

Funded by the European Union

EuroHPC Joint Undertaking

TUHH Technische Universität Hamburg