Application of Augmented Fourier-Neural Operator Methods to Conjugate Heat Transfer Problems [*]

Rishabh Iyer, Xinrui Xiang, Hongwei Sun
College of Engineering
Northeastern University
Boston, MA
`{iyer.ris, xiang.xinr, ho.sun}@northeastern.edu`

Abstract

High fidelity numerical simulations of the Navier-Stokes Equations coupled with the Heat-Advection equation for various Conjugate Heat Transfer (CHT) problems has proven to be of high cost computationally. These numerical simulations prove crucial to many engineering applications from the electronics industry to gas turbine simulation to aid with design optimization. Recent efforts have been made to apply machine learning as a surrogate model for such applications using various methods such as Variational Auto-Encoders (VAEs) or Physics Informed Neural Networks (PINNs), which require fixed-size inputs and only learn a single instance of a partial differential equation (PDE). The Fourier Neural Operator is a relatively new architecture which utilizes fast-fourier-transform compression to map between continuous function spaces, which allows for a mesh-independent architecture. We apply this to conjugate heat transfer problems by providing additional features as input, including a domain mask, and a masked loss function, which allows the model to render multi-physics, multi-domain problems with a single input. In less that 10 ms, and with an accuracy of 0.992 on the test set, this modified Fourier Neural Operator should be a crutch for future repeatable conjugate heat transfer simulations in engineering.

Keywords Fourier Neural Operator · Partial Differential Equations · Conjugate Heat Transfer

1 Introduction

Conjugate heat transfer (CHT) simulations are important to various industries where temperature is a design criterion, including automotive, aerospace, electronics, etc. In these simulations, design optimization can be a very inefficient process due to the high computational load of solving the coupled partial differential equations (PDEs) present in CHT. Effective compression using machine learning is of great interest to these industries, where topological optimization could be done in orders of magnitude less time by using surrogate models.

Traditional Numerical Methods In traditional numerical methods for PDEs, a mesh must be constructed reflecting the problem, such that the solution is represented as an operator in the original coordinate space. This creates a curse of dimensionality associated with large high resolution meshes when solving complex problems with finite differences, finite element or finite volume methods [1]. Spectral methods have a rich history in fluid dynamics, mainly due to the ease of calculation compared to other collocation methods [2]. With only a few degrees of freedom, spectral methods can achieve accuracy on-par with lower-order based methods like the finite-difference family of methods. In a time where compute was sparse, the speedups were invaluable.
It is also possible to directly augment the traditional numerical methods with machine learning. [3] augments traditional numerical solvers by utilizing a neural network to generate a fast multi-grid solver.

Surrogate Models The choice of surrogate model is important to retain architectural features that are consistent with the current state-of-the-art of the field. Data-driven methods that incorporate machine learning have become

[*] *Citation*: **Authors. Title. Pages.... DOI:000000/11111.**

popularized over the last decade [4, 5]. Many architectures such as Variational Auto-Encoders [6] or Physics Informed Neural Networks [7, 8] have been used in-place of simulations associated with coupled PDEs, but these architectures struggle with mesh-size dependence. Transformers and convolutional U-Nets have also been used for learned surrogate models in PDEs despite being designed for other input data structures, displaying results of varying performance learning a single instance of a PDE. In traditional machine learning methods, physics-based constraints can always be implemented via a loss [9, 7, 10] using auto-differentiation with libraries such as PyTorch [11]. Physics based methods are often used in CHT problems when sparse data is available, but mainly when conservation laws need preservation [8]. Quasi mesh-invariant reduced order methods are also historically associated with the Navier-Stokes equations, where proper orthogonal decomposition (POD) has been used to achieve simulation speedups. This type of reduced order compression often comes at the cost of accuracy directly.
Many of these neural network surrogate models utilize convolutions due to their natural ability to aggregate neighboring information, which is a useful property for physics problems that have traditionally used finite-differencing techniques. In many PDEs, the existing state-of-the-art traditional numerical method aggregates information from only a few points to update each cell in the grid (finite-differencing, finite volume), and given a uniform stencil, convolutional neural networks (CNNs) are the easiest way to approximate a single instance of a PDE solution [12, 13]. Since there is a natural correspondance between image frames and scalar physical states in a structured grid, one might expect transformers' dominance to extend to this field as well. Transformers [14, 15] have shown to be effective on images, particularly vision-transformers ViT, which utilize convolutions to generate query, key and value tensors. These tensors contain the griddata alongside the embeddings, which can then be utilized to compute the pairwise importance of every node in the grid to every other node. This is useful in determining global, long range dependence between data, which might be useful for multi-physics problems, or when more grid points are needed to resolve a state update or solution for any particular point in the grid. For turbulence, this sort of global, long-range accurate model is very important [16].

Ideally, we would like to generalize across multiple instances of a PDE, with quasi mesh-invariance, with speed and high retained accuracy.

Neural Operators Neural operators generate infinite dimensional mappings between the original domain and output without mesh dependence or high latency [17]. Neural operators achieve this by defining a kernel integral, which integrates out the original coordinate space to generate the solution. This is not so different from canonical methods in CFD, where the solution operator is expressed as solving a linear system [1]. Here, instead of a linear system solution, we express the solution operator as an infinite-dimensional inner product that can be learned [18].
Popular alternative methods currently in this subset of scientific machine learning include the Latent Spectral Model (LSM) [19] and the Graph Kernel Operator [20]. LSMs utilize attention based aggregation to a latent space where trignometric basis functions are used to represent the signal in this latent space. The Graph Kernel Operator utilizes message passing to represent the kernel integral, which is a natural way to deal with physics problems, given most physics problems are isotropic.
This is a similar approach to the most popular neural operator in PDEs today: the Fourier Neural Operator (FNO) [21]. The FNO introduces a spectral convolution layer, which leverages the fast fourier transform (FFT) as a means of compressing the continuous domain into a small number of frequencies. The original JPEG image compression algorithm actually utilizes this with patching and cosine basis functions, rather than both sine and cosine basis functions [22]. This can effectively be thought of as a low pass filter.
The convolution theorem can then be applied such that a linear map in the frequency domain will represent a convolution in the original coordinate space after inverse FFT. These layers can be stacked to represent a neural operator which maps the original coordinates and parameters to the solution in a quasi-mesh independent way. This allows many mesh resolutions as input, and also allows a general solution to be learned for multiple sets of parameters. This type of model leverages the speedups of spectral compression methods with the accuracy of CNNs for physics problems.

The FFT expects periodic boundary conditions, which are unnatural for many engineering problems of interest. Dirichlet, Neumann, and Robin Boundary conditions with spectral methods in PDEs are an interesting problem explored by many, since spectral methods are the original highly compressed surrogate model. It is well documented that the best way to deal with such boundary conditions is by utilizing padding within the domain in the original coordinate space, similar to requiring several ghost points to resolve boundary conditions in finite-difference based stencil methods.
For complex geometries that can be represented as a torus, a simple linear-map on the coordinates as a perturbed domain is all that is needed to learn an operator for the PDE [23].

Our Contribution Our work is an implementation of a mixture of above methods, mainly the FNO, where

we suggest a framework for applying neural operator methods to CHT problems. We solve the coupled fluid, heat transfer problem by passing a boundary mask alongside the query coordinates, which gives the model information regarding which parts of the domain are solid and fluid. We then allow for mask-based padding of the domain to better resolve the first order discontinuities of velocity and zeroth order discontinuities of mechanical pressure for the internal points of the domain. This is further emphasized by a masked loss function, which allows the points outside the fluid domain to be utilized as padding, effectively reusing parts of the solid domain as ghost points for the fluid domain. To our knowledge, this is the first application of neural operators to CHT problems with immersed boundaries.
We also add an additional branch to the spectral convolution of the FNO, where we compute spectral-attention, which is attention in the frequency domain. In order to do this we require usage of the traditional methods in [14, 15], alongside the phase non-linear activation functions described in complex-valued neural networks [24].

2 Governing Equations

2.1 Conjugate Heat Transfer

Conjugate

$$\rho\left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u}\right) = -\nabla p + \mu\nabla^2\mathbf{u} + \mathbf{f} \tag{1}$$

$$\nabla\cdot\mathbf{u} = 0 \tag{2}$$

$$\rho c_p\left(\frac{\partial T}{\partial t} + \mathbf{u}\cdot\nabla T\right) = k\nabla^2 T + q \tag{3}$$

where ρ is the density, $\mathbf{u}$ is the velocity vector, t is time, p is the pressure, μ is the dynamic viscosity, $\mathbf{f}$ represents body forces, c_p is the specific heat capacity, T is the temperature, k is the thermal conductivity, and q is a heat source term.

Equation 1 represents the momentum conservation (Navier-Stokes) equations, which describe the motion of fluids. The left-hand side represents the inertial forces, while the right-hand side includes the pressure gradient, viscous forces, and body forces.

Equation 2 is the continuity equation, which ensures mass conservation in the fluid domain. It states that the divergence of the velocity field must be zero for an incompressible fluid.

Equation 3 is the heat advection-diffusion equation, which describes the transport of heat in the fluid and solid domains. The left-hand side represents the advection of heat by the fluid, while the right-hand side includes heat diffusion and any heat source terms.

In conjugate heat transfer problems, these equations are coupled at the fluid-solid interfaces. The velocity is assumed to be zero at the solid boundaries (no-slip condition), and the temperature and heat flux are continuous across the interface:

$$T_f = T_s, \quad k_f\frac{\partial T_f}{\partial n} = k_s\frac{\partial T_s}{\partial n} \tag{4}$$

where the subscripts f and s denote the fluid and solid sides of the interface, respectively, and n is the normal direction to the interface. In practice the heat advection-diffusion equation can be solved everywhere in the domain, since the advection velocity in the solid is zero.

The coupled system of equations 1-4 fully describes conjugate heat transfer problems. However, solving these equations directly using traditional numerical methods can be computationally expensive, especially for complex geometries and high-resolution simulations that require unstructured meshes. This motivates the use of surrogate models, such as the Fourier Neural Operator, to efficiently approximate the solution while maintaining accuracy of the physics involved.
For boundary conditions, in most CHT problems that do not have infinite length spatial dimensions or periodicity, either temperature (Dirichlet) or heat flux (Robin or Neumann BC) most be specified at the boundary values. For the NS equations, there are boundary conditions for the fluid domain that consist of no-slip walls (dirichlet velocity), velocity inlet (dirichlet), and pressure outlet with fully developed flow (dirichlet pressure, neumann velocity).

2.2 Fourier Transform

The Fourier transform is a mathematical tool that decomposes a function into its constituent frequencies. For a function $f(x)$ defined on the real line, the Fourier transform $\mathcal{F}f$ is given by:

$$\mathcal{F}f = \int_{-\infty}^{\infty} f(x)e^{-2\pi ikx}dx \tag{5}$$

where k is the frequency variable, and i is the imaginary unit. The Fourier transform maps a function from the spatial (or time) domain to the frequency domain. The inverse Fourier transform, which maps a function from the frequency domain back to the spatial domain, is given by:

$$f(x) = \int_{-\infty}^{\infty} \mathcal{F}fe^{2\pi ikx}dk \tag{6}$$

The Fourier transform utilizes complex exponential basis functions to represent the original signal as a series of sines and cosines through the inner products of the FT and its inverse. This can be exploited when the original signal and its frequency domain counterpart are discretized. For a discrete function $f[n]$ with N samples, the discrete Fourier transform (DFT) is defined as:

$$F[k] = \sum_{n=0}^{N-1} f[n]e^{-2\pi ikn/N}, \quad k = 0, 1, \dots, N-1 \tag{7}$$

The Fast Fourier Transform (FFT) is an efficient algorithm for computing the DFT and its inverse. The computational complexity of the FFT is $O(N \log N)$, which has made the FFT a fundamental tool in signal processing throughout history. This is due to a natural bias continuous, and some discrete phenomena contain in the frequency domain, where much of the higher frequency bands contain irrelevant noise to the signal we are interested in.
Since the FFT uses trigonometric basis functions, special treatment is needed for the boundary conditions. Sines and cosines are periodic functions, thus an un-padded FFT spectral solution to any PDE will not respect Dirichlet or Neumann BCs, unless they can also be periodic. Not only that, but when modeling boundary value problems with non-periodic BCs, the basis functions will register larger coefficients for high frequency components because of the boundary conditions, which increase the inaccuracies associated with compressing the signal. A traditional solution to this issue is to pad the domain with zeros or with the value of the Dirichlet BC to give the lower frequency basis functions some space to return to modeling the signal accurately. This can effectively be thought of as utilizing several "ghost points".

2.3 Physics Informed Loss

It is important here to cover physics informed neural networks (PINNs), since our treatment of the CHT examples involve utilizing physics based losses to reduce violation of the conservation laws present in the problem.

Physics Informed Neural Networks (PINNs) [7] are a class of deep learning models that incorporate physical laws and domain knowledge into the learning process. By encoding the governing partial differential equations (PDEs) into the loss function, PINNs can learn solutions that satisfy the underlying physical principles.

In the context of conjugate heat transfer problems, PINNs can be used to enforce the conservation of mass, momentum, and energy, as described by the Navier-Stokes equations (Eqs. 1-2) and the heat advection-diffusion equation (Eq. 3). The idea is to construct a neural network that represents the solution variables (e.g., velocity, pressure, and temperature) parameterized by the input variables and use automatic differentiation to compute the derivatives required in the governing equations.

Let $\mathbf{u}(\mathbf{x})$, $p(\mathbf{x})$, and $T(\mathbf{x})$ denote the neural network approximations of the velocity, pressure, and temperature fields, respectively. The physics-informed loss function for the conjugate heat transfer problem can be defined as:

$$\mathbb{L} = \mathbb{L}\text{data} + \mathbb{L}\text{physics} \tag{8}$$

where $\mathbb{L}$data is the data mismatch loss, which measures the discrepancy between the neural network predictions and the available data (e.g., boundary and initial conditions), and $\mathbb{L}$physics is the physics-based loss, which enforces the satisfaction of the governing equations.

The physics-based loss can be computed by substituting the neural network approximations into the governing equations and evaluating the residuals at a set of collocation points $(t_i, \mathbf{x}i)i = 1^N$:

$$\mathbb{L}\text{physics} = \frac{1}{N}\sum i = 1^N |\mathcal{N}[\mathbf{u}(\mathbf{x}_i), p(\mathbf{x}i)]|^2 + \frac{1}{N}\sum i = 1^N |\mathcal{H}[\mathbf{u}(\mathbf{x}_i), T(\mathbf{x}_i)]|^2 \tag{9}$$

Here, $\mathbf{u}(\mathbf{x})$, $p(\mathbf{x})$, and $T(\mathbf{x})$ are the neural network approximations of the velocity, pressure, and temperature fields, respectively. The Navier-Stokes operator is denoted as $\mathcal{N}$ and the heat advection-diffusion operator as $\mathcal{H}$, where the operators are non-linear differential operators that set the PDEs as homogenous for non-source problems.

The derivatives in the physics-based loss can be computed using automatic differentiation, which allows for efficient and accurate computation of the gradients required for training the neural network.

By minimizing the total loss function (Eq. 8), PINNs can learn solutions that simultaneously fit the available data and satisfy the governing physical equations. This physics-informed approach has been shown to be effective in solving a wide range of PDEs, including those arising in fluid dynamics and heat transfer problems. This same procedure can also be applied to ensure boundary value accuracy as additional loss terms in the physics based loss.

In the context of the Fourier Neural Operator, incorporating physics-informed losses can help to further constrain the learned solution operator and improve the model's accuracy to physics [25]. The combination of the FNO's efficient spectral representation and the physics-informed loss functions can lead to a powerful and sample-efficient approach for solving conjugate heat transfer problems, especially when computing numerical simulations for the dataset is computationally taxing.

2.4 Fourier Neural Operator

The Fourier Neural Operator (FNO) [21, 23] is used here as an infinite-dimensional mapping between two function spaces, allowing for the learning of solution operators for parametric PDEs. The FNO is formulated as an architecture that maps an input function $a \in \mathcal{A}$ to an output function $u \in \mathcal{U}$ through a series of composed convolutional layers in the spectral domain and point-wise non-linear activation functions.

The key component of the FNO is the kernel integral operator $\mathcal{K}$ that maps between function spaces:

$$(\mathcal{K}(a;\phi)v_t)(x) := \int_D \kappa(x, y, a(x), a(y); \phi)v_t(y)dy, \quad \forall x \in D \tag{10}$$

where $\kappa_\phi : \mathbb{R}^{2(d+d_a)} \to \mathbb{R}^{d_v \times d_v}$ is a neural network parameterized by $\phi \in \Theta_\mathcal{K}$, $D \subset \mathbb{R}^d$ is a bounded domain, and v_t is the function representation at iteration t.

The original FNO architecture consists of lifting the input function a to a higher-dimensional representation v_0 using a simple multi-layer perceptron network P, then directly mapping the input the kernel integral operator and pointwise non-linearities, and then projecting the final representation v_T to the output function u using another pointwise neural network Q:

$$v_0(x) = P(a(x)) \tag{11}$$

$$v_{\ell+1}(x) = \sigma(Wv_\ell(x) + (\mathcal{K}(a;\phi)v_\ell)(x)) \tag{12}$$

$$u(x) = Q(v_L(x)) \tag{13}$$

where:

P is a pointwise neural network (MLP) that lifts the input function $a(x)$ to a higher-dimensional representation $v_0(x)$. $v_\ell(x)$ is the function representation at layer ℓ. σ is a pointwise non-linearity (e.g., ReLU or sigmoid). W is a learnable linear transformation matrix. $\mathcal{K}(a;\phi)$ is the kernel integral operator parameterized by ϕ. L is the total number of layers (i.e., the number of spectral convolution layers). Q is a pointwise neural network (MLP) that projects the final representation $v_L(x)$ to the output function $u(x)$. In this formulation, the FNO takes the input function $a(x)$ and directly maps it to the output function $u(x)$ through a series of L layers, each consisting of a linear transformation W, a kernel integral operator $\mathcal{K}(a;\phi)$, and a pointwise non-linearity σ. The input function $a(x)$ is first lifted to a higher-dimensional representation $v_0(x)$ using the pointwise neural network P. The lifted representation then passes through the L layers, where each layer updates the representation using the kernel integral operator and the pointwise non-linearity. Finally, the last layer's representation $v_L(x)$ is projected back to the output function $u(x)$ using the pointwise neural network Q.

This zero-shot formulation allows the FNO to learn a direct mapping from the input function to the output function without the need for iterative updates over time.

To efficiently parameterize and compute the kernel integral operator, the FNO uses a spectral convolution operator defined in Fourier space:

$$(\mathcal{K}(\phi)v)(x) = \mathcal{F}^{-1}(R_\phi \cdot (\mathcal{F}v))(x), \quad \forall x \in D \tag{14}$$

where $\mathcal{F}$ and $\mathcal{F}^{-1}$ denote the Fourier transform and its inverse, and R_ϕ is the Fourier transform of a learnable periodic function κ.

By parameterizing the kernel directly in Fourier space and using the Fast Fourier Transform (FFT) to efficiently compute convolutions using the compression techniques shown earlier, the FNO can learn complex solution operators while maintaining a computationally efficient architecture.

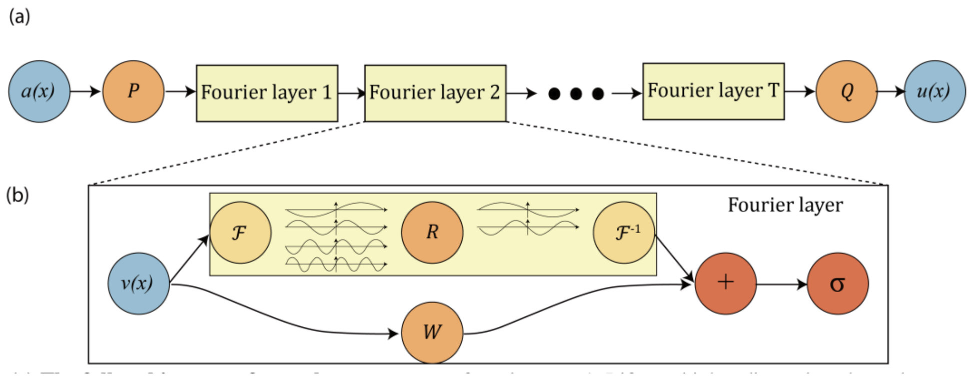

Figure 1: Fourier Neural Operator Architecture, [21]

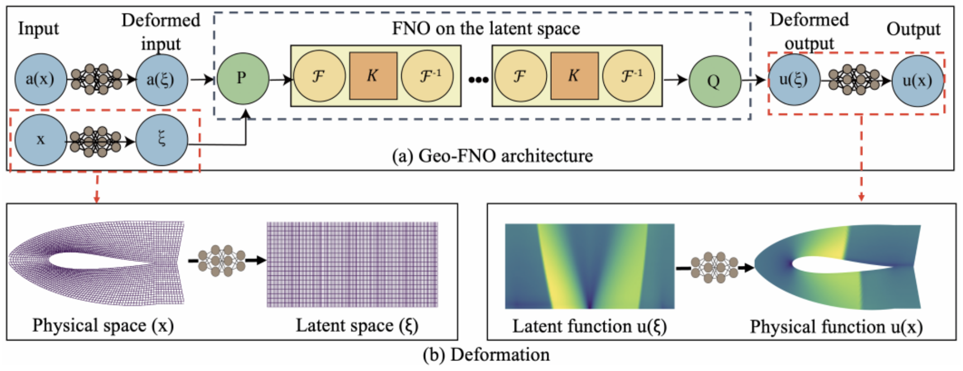

Figure 2: Geo-FNO Architecture, [23]

For our example problem, we have to use an architecture that perturbs the domain using a linear map, since the original fluid domain contains an irregular shaped channel [23]. We can utilize a linear map to deform the domain into a regularly shaped domain for ease of use within the spectral convolution layers, but also for the model to understand how to map irregular geometry to a solution function.

3 Architecture Details

Our architecture consists of the Geometry-Aware FNO detailed above, where the input is deformed into a uniform resolution structured grid more suitable for convolution operations, spectral or otherwise. We then concatenate the original grid, the deformed grid, and a mask which is a grid-valued function that masks the domain in 0,1 to denote domain type, where 1 indicates fluid. This mask gives information on how to treat subsets of the input image differently. Then, we pad the domain for the FFT operations that come next. We then use an MLP to lift the input channel (5) to vectors of 128. Lower dimensional configurations were tested with decent results, but increasing the width of the neural network is generally useful to capture more information in the data, depending on if there is enough data present.

Here we need to introduce our novel spectral-attention block, which computes the query, key and value tensors using convolutions in the original domain, but then takes the FFT filtering and computes attention in the frequency domain before giving the model information about which embedding frequencies are the most important.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{\mathcal{F}(Q) \cdot \mathcal{F}(K)^T}{\sqrt{d_k}}\right) \cdot \mathcal{F}(V) \tag{15}$$

where Q, K, and V are the query, key, and value tensors, respectively, computed using convolutions in the original domain. $\mathcal{F}$ denotes the Fourier transform, which is applied to each tensor before computing the attention. This is very similar to the Galerkin-Attention shown in [26]. The attention is computed as the complex valued softmax of the scaled dot-product between the Fourier transforms of the query and key tensors, multiplied by the Fourier transform of the value tensor. d_k is a scaling factor, typically set to the square root of the dimension of the key tensor. The spectral-attention block allows the model to learn which frequency components are most important for the task at hand, enabling it to focus on the relevant parts of the input signal in the frequency domain. We can then inverse FFT in the same way as the spectral-convolution block in the FNO.

The total architecture consists of 3 spectral-convolution layers that include point-wise convolutions to model boundary conditions and other point-wise phenomena, with a single branch of spectral-attention on the last spectral-convolution layer; then a separate MLP brings the channel dimensions back down to our output size of 4 (x-velocity,y-velocity,pressure, temperature).

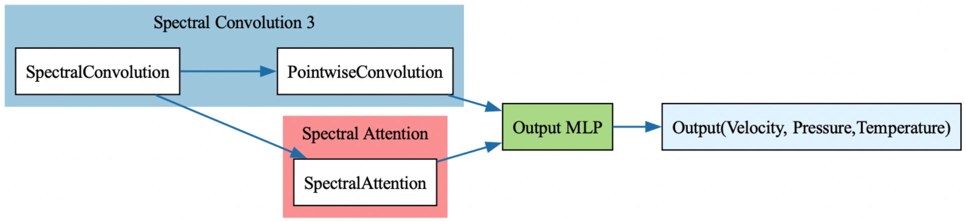

Figure 3: Final Layers of the Neural Network Architecture

We utilize a relative loss function, which penalizes based on relative difference, which is useful for multiscale phenomena. We also include a physics-informed loss that ensures the continuity-equation is adhered for mass conservation within the fluid. Because our fluid has multiple domain types, yet we treat the multi-domain as a single domain that encompasses all of the multiphysics, we utilize a novel mask-based loss, which only penalizes terms for x-velocity (u), y-velocity (v) and pressure (P) within the masked input. This not only emphasizes the importance of the mask to predict fluid outputs correctly, but it also encourages the model to utilize non-fluid parts of the domain as non-essential padding, or quasi ghost-points. The model can utilize these points to give the lower frequencies the requisite space to accurately model the internal points, while maintaining their periodicity by utilizing the hidden parts of the domain. Of course, temperature in CHT problems does not get this benefit, but it doesn't have to, since we pad the entire domain within the architecture itself. The main point of this masked loss function is to handle the 0th and 1st order discontinuities in velocity and pressure elegantly. PINNs and PINN based loss functions tend to optimize poorly and fall into local minimums given sparse data [27], but this masked loss can help combat this common failure mode by focusing on the relevant patches of the domain.
This architecture allows for blazing fast speeds and high physical accuracy due to the physics informed loss that enforces mass conservation, along with the masked loss.

4 Dataset and Experiments

Neural operators generally require lots of training data, which can be costly, possibly outweighing the benefits of the surrogate model speed. Our model is sample efficient due to the physics informed losses. For our toy problem dataset, we generated 1000 solutions to 2D conjugate heat transfer problems that cool hot water from 1000 degrees celsius in an aluminum box in room-temperature, free-convection air.
Our method was to first solve the Navier-Stokes equations for 2-dimensional channel flow using a high resolution mesh.

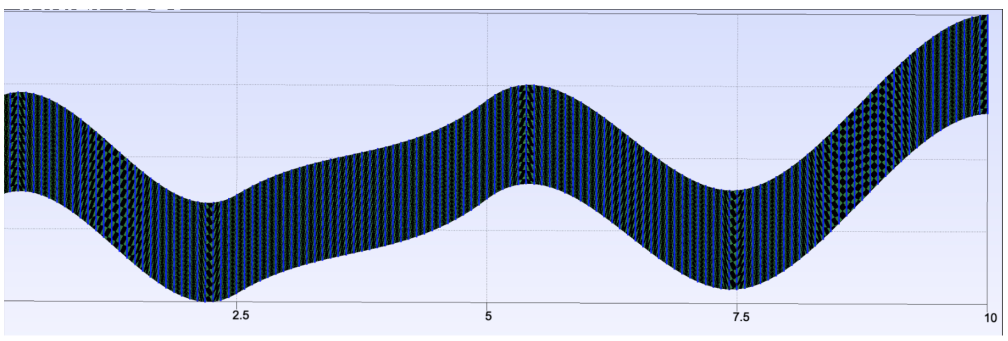

Figure 4: Example mesh, 129x129 quadrilateral cell mesh

The N-S equations were solved through a channel with height of 1 and length of 10, for water with no-slip boundary conditions, velocity inlet (max x-velocity=0.25), and zero-pressure outlet. We then can interpolate these solved channel flows onto a much coarser structured grid, since the heat convection-diffusion equations are far simpler to solve using traditional numerical methods when decoupled from the N-S equations. Then, we proceed to solve the conjugate heat transfer equations for a box of Aluminum with height 4 and width 10. The boundary conditions for the thermal model were Dirichlet for fluid inlet (T=1000), fully developed temperature Neumann condition for the fluid outlet, and free-air convection Robin BCs (h=20) everywhere else. We use a mask to denote solid vs fluid domains for the model as well. The coarse structured grid allows for very fast data generation that utilizes GPU acceleration in JAX [28] when given data from a flow solver. Note: this does not have to be done for the operator to work, this was done to accelerate the training process.

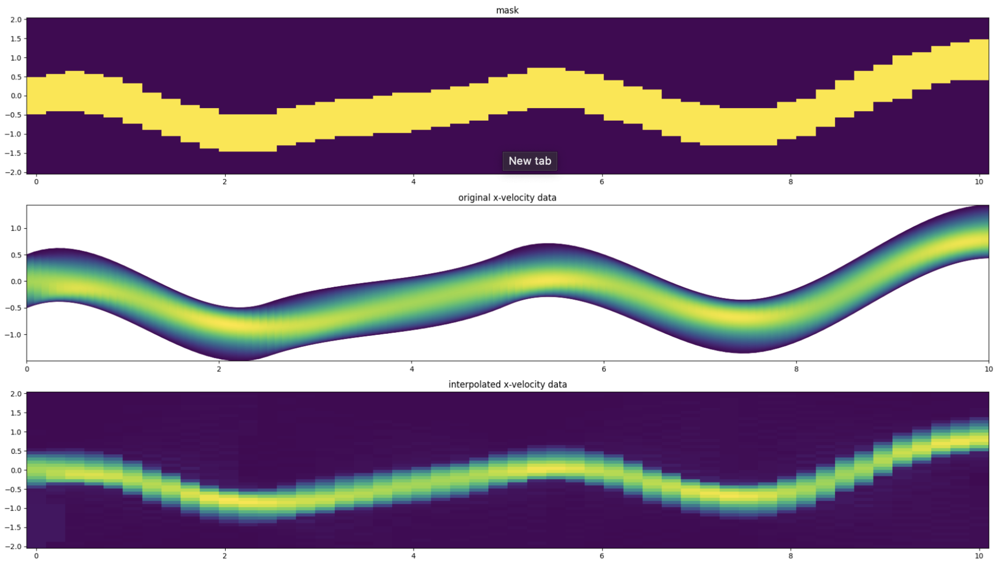

Figure 5: Example velocity interpolation and mask

Here we show an example of a solved channel alongside ground truth. This model with an input of 50x50 grid resolution reached a solution through a forward pass of the network in less than 10 ms, compared to a similar geometry used in the dataset which took 20 min with gpu accelerated capability via JAX. The accuracy of this run on a mesh from the test set was 0.992, which is astounding for a physics model attempting to capture such complex behavior in the 2D Navier-Stokes equations coupled with the Heat Convection-Diffusion equation. We find that the spectral-attention

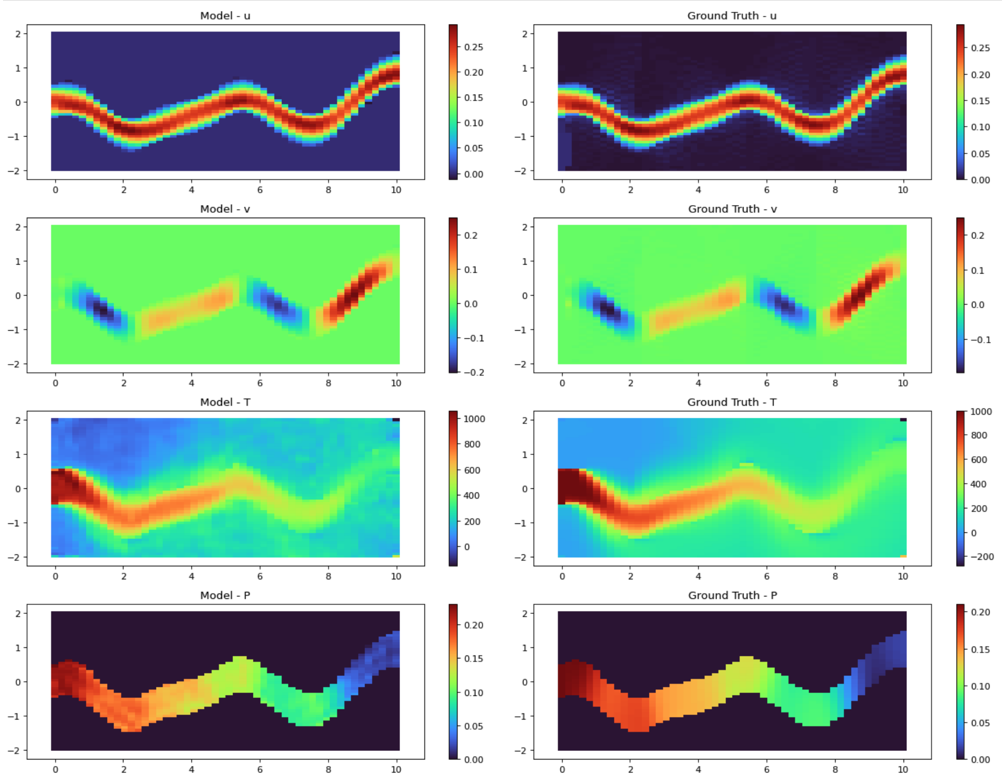

Figure 6: Example Inference Using Modified Mask FNO on CHT

augmented network slightly outperforms the network when spectral-attention is ablated. In a comparison on the test set

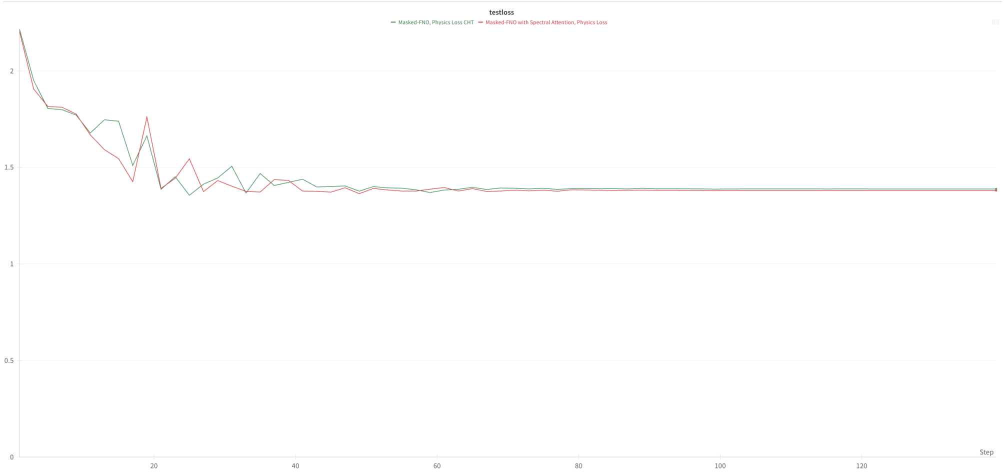

Figure 7: Ablation Study for Spectral-Attention Module

of the masked loss function FNO with the unmasked version, the masked method nearly slices the error in half, while retaining the speed of the surrogate model.

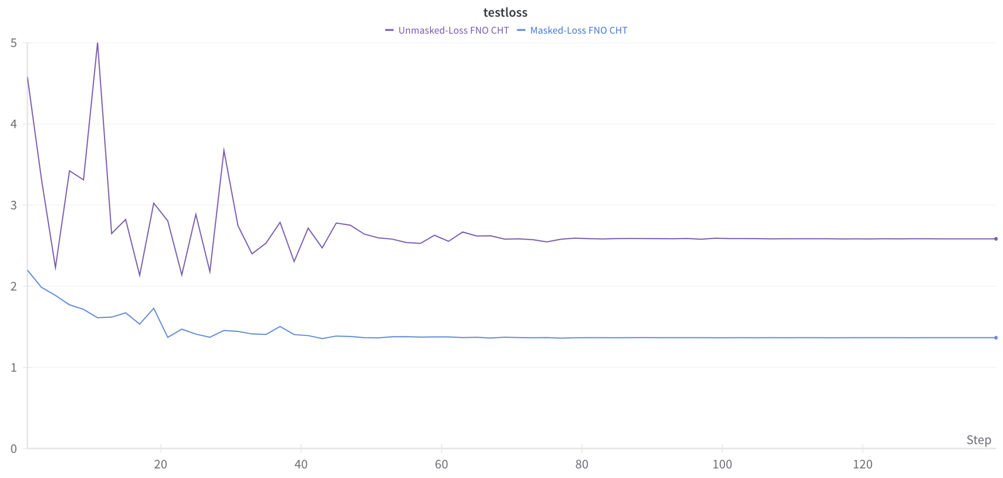

Figure 8: Test Loss over epochs for FNO methods

5 Conclusion

In this work, we extend the capabilities of the Fourier Neural Opearator and introduce a novel architecture by applying a novel masking method on a single domain to generate solutions to 2-dimensional conjugate heat transfer problems that consist of the Navier-Stokes equations and the Heat Convection-Diffusion equations. We accomplish this by adding a mask that describes cells that are within the fluid domain versus solid domain, and also computing the Navier-Stokes based data and physics losses within the mask. This masking loss function effectively creates additional padding in the form of quasi-ghost points to resolve the fluid channel. We also get benefits from applying a novel spectral-attention module that computes attention in the frequency domain as an additional branch in the network, and it provides a slight increase in performance at this data scale on top of the benefits of the masked-input and masked loss function.
We show that this modified FNO resolves the dynamics for all output variables (velocity, pressure, temperature) with high accuracy 0.99, and with blazing fast speed at 10 ms with a 50x50 grid. In comparison to a GPU-accelerated solver, the time decrease on the same resolution is from 20 minutes down to 10 ms. Comparing with traditional CPU based numerical solvers, this order of magnitude difference in speed only increases. This is an important step forward in utilizing these operator methods at scale for engineering problems.
Future Work Potential future work includes expanding on the spectral-attention block, as it is an innovative concept that could either be potentially improved upon for multi-physics surrogate models, or for other fields entirely (video, optics). Also, solving turbulence closure with machine learning is an ongoing problem in CFD, which could be aided dramatically by a neural operator, especially one that does not need a specific grid resolution. This would likely require additional branches to the FNO beyond the scope of this paper. Given datasets of higher fidelity, potentially spectral-attention could aid greatly where convolutions fail.

Acknowledgments

This was was supported in part by Northeastern University, with fruitful discussions from Xinrui Xiang and Prof. Hongwei Sun.

References

[1] C. Grossmann, H.G. Roos, and M. Stynes. *Numerical Treatment of Partial Differential Equations*. Universitext. Springer Berlin Heidelberg, 2007.

[2] David I. Gottlieb and Steven A. Orszag. Numerical analysis of spectral methods : theory and applications. 1977.

[3] Daniel Greenfeld, Meirav Galun, Ronen Basri, Irad Yavneh, and Ron Kimmel. Learning to optimize multigrid PDE solvers. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2415–2423. PMLR, 09–15 Jun 2019.

[4] Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937, March 2016.

[5] Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937, 2016.

[6] Takiah Ebbs-Picken, David A. Romero, Carlos M. Da Silva, and Cristina H. Amon. Deep convolutional encoder-decoder hierarchical neural networks for conjugate heat transfer surrogate modeling. 2023.

[7] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations, 2017.

[8] Shengze Cai, Zhicheng Wang, Sifan Wang, Paris Perdikaris, and George Em Karniadakis. Physics-Informed Neural Networks for Heat Transfer Problems. *Journal of Heat Transfer*, 143(6):060801, 04 2021.

[9] Ravi G. Patel, Nathaniel A. Trask, Mitchell A. Wood, and Eric C. Cyr. A physics-informed operator regression framework for extracting data-driven continuum models. *Computer Methods in Applied Mechanics and Engineering*, 373:113500, January 2021.

[10] M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[11] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019.

[12] Kun Wang, Yu Chen, Mohamed Mehana, Nicholas Lubbers, Kane Bennett, Qinjun Kang, H. Viswanathan, and Timothy Germann. A physics-informed and hierarchically regularized data-driven model for predicting fluid flow through porous media. *Journal of Computational Physics*, 443:110526, 06 2021.

[13] Junhyuk Kim and Changhoon Lee. Prediction of turbulent heat transfer using convolutional neural networks. *Journal of Fluid Mechanics*, 882:A18, 2020.

[14] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

[15] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *CoRR*, abs/2010.11929, 2020.

[16] Steeven JANNY, Aurélien Bénéteau, Madiha Nadri, Julie Digne, Nicolas THOME, and Christian Wolf. EAGLE: Large-scale learning of turbulent fluid dynamics with mesh transformers. In *The Eleventh International Conference on Learning Representations*, 2023.

[17] Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, March 2021.

[18] Tianping Chen and Hong Chen. Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. *IEEE Transactions on Neural Networks*, 6(4):911–917, 1995.

[19] Haixu Wu, Tengge Hu, Huakun Luo, Jianmin Wang, and Mingsheng Long. Solving high-dimensional pdes with latent spectral models, 2023.

[20] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Graph kernel network for partial differential equations, 2020.

[21] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. 2021.

[22] A Léger. Implementations of fast discrete cosine transform for full color videotex services and terminals. In *Proceedings of the IEEE Global Telecommunications Conference, IEEE Communications Society*, pages 333–337, 1984.

[23] Zongyi Li, Daniel Zhengyu Huang, Burigede Liu, and Anima Anandkumar. Fourier neural operator with learned deformations for pdes on general geometries, 2022.

[24] Chiheb Trabelsi, Olexa Bilaniuk, Ying Zhang, Dmitriy Serdyuk, Sandeep Subramanian, João Felipe Santos, Soroush Mehri, Negar Rostamzadeh, Yoshua Bengio, and Christopher J Pal. Deep complex networks, 2018.

[25] Zongyi Li, Hongkai Zheng, Nikola Kovachki, David Jin, Haoxuan Chen, Burigede Liu, Kamyar Azizzadenesheli, and Anima Anandkumar. Physics-informed neural operator for learning partial differential equations, 2023.

[26] Shuhao Cao. Choose a transformer: Fourier or galerkin, 2021.

[27] Aditi S. Krishnapriyan, Amir Gholami, Shandian Zhe, Robert M. Kirby, and Michael W. Mahoney. Characterizing possible failure modes in physics-informed neural networks. *CoRR*, abs/2109.01050, 2021.

[28] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018.