

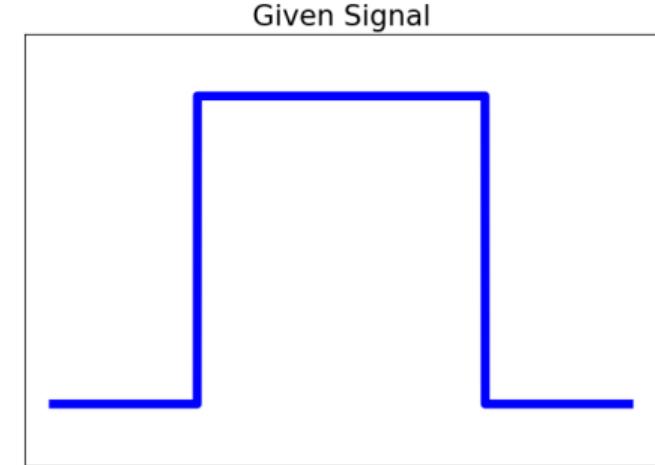
# Fourier Neural Operators

A Fourier Approach for Operator Learning

Janek Gödeke, Nick Heilenkötter, Tom  
Freudenberg  
Renningen, 21.11.2025

# Fourier Transform

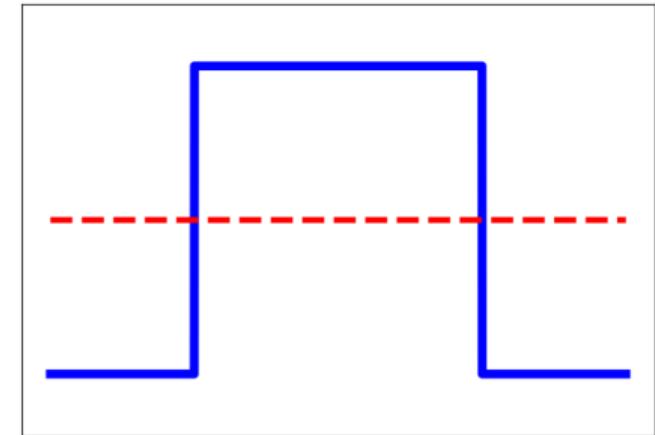
- Reconstruct a function  $f$   
by superposition of cosine & sine waves



# Fourier Transform

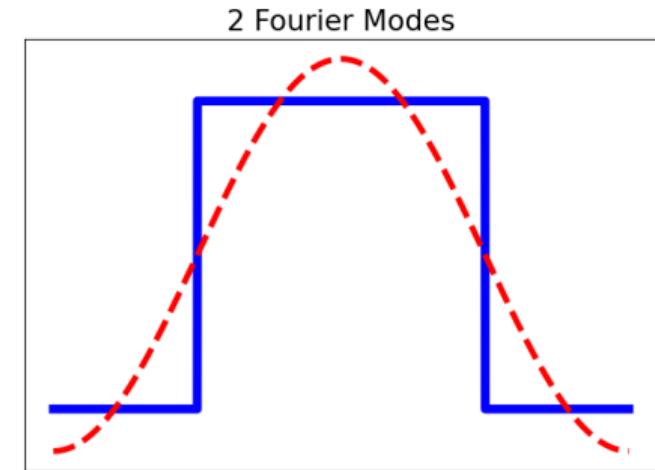
- Reconstruct a function  $f$   
by superposition of cosine & sine waves

1 Fourier Modes



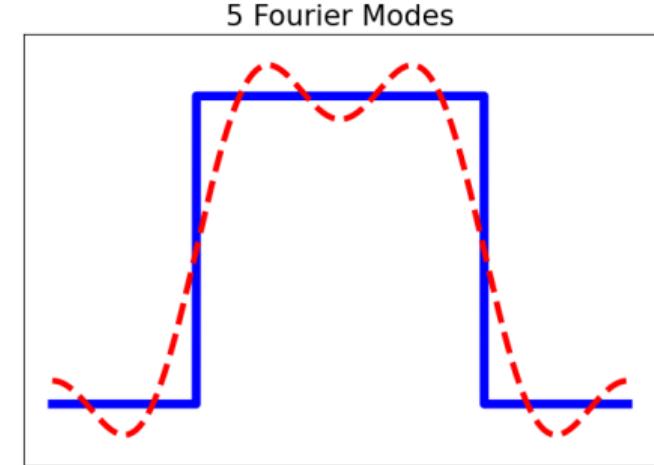
# Fourier Transform

- Reconstruct a function  $f$   
by superposition of cosine & sine waves



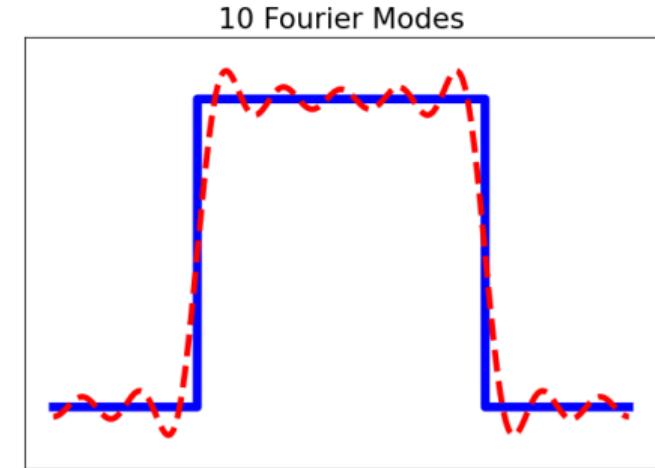
# Fourier Transform

- Reconstruct a function  $f$   
by superposition of cosine & sine waves



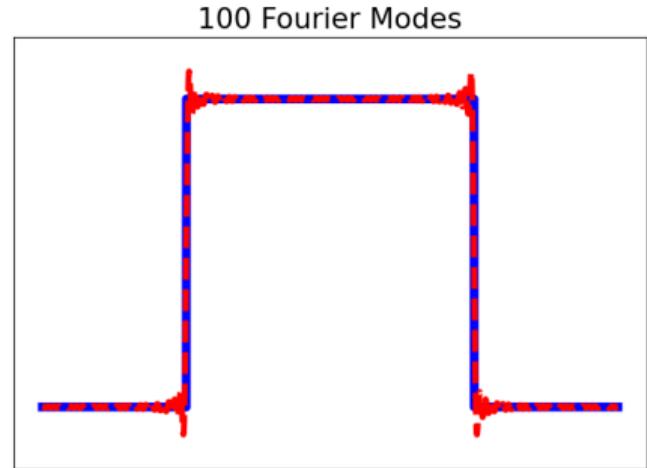
# Fourier Transform

- Reconstruct a function  $f$   
by superposition of cosine & sine waves



# Fourier Transform

- Reconstruct a function  $f$   
by superposition of cosine & sine waves



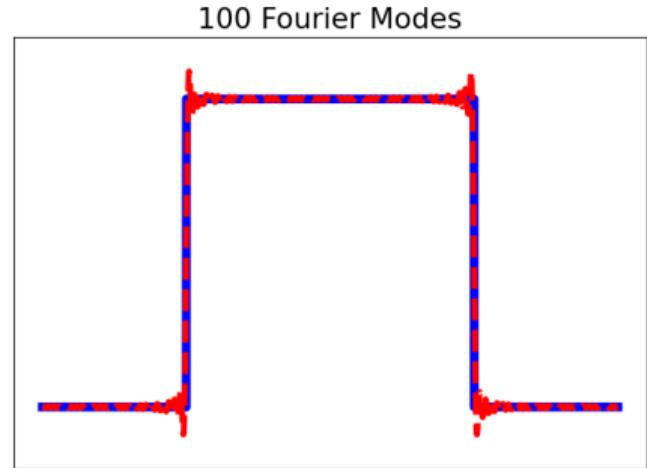
# Fourier Transform

- Reconstruct a function  $f$  by superposition of cosine & sine waves
- Fourier transform ( $k \in \mathbb{Z}$ ):

$$\mathcal{F}(f)[k] := \int_{-\infty}^{\infty} f(x) e^{-i2\pi kx} dx$$

- Approximate reconstruction (for  $K \in \mathbb{N}$ ):

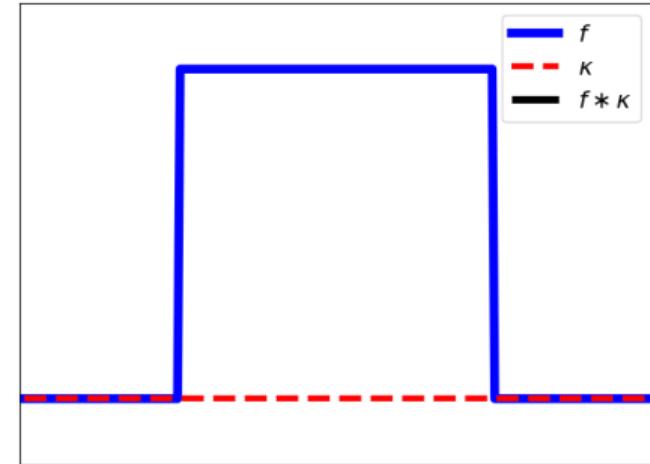
$$f(x) \approx \int_{-K}^{K} \mathcal{F}(f)[k] e^{i2\pi kx} dk$$



# Convolution Theorem

- For a function  $f$  and **convolution kernel** (function)  $\kappa$ :

$$(f * \kappa)(x) := \int_{-\infty}^{\infty} f(y)\kappa(y - x)dy$$



# Convolution Theorem

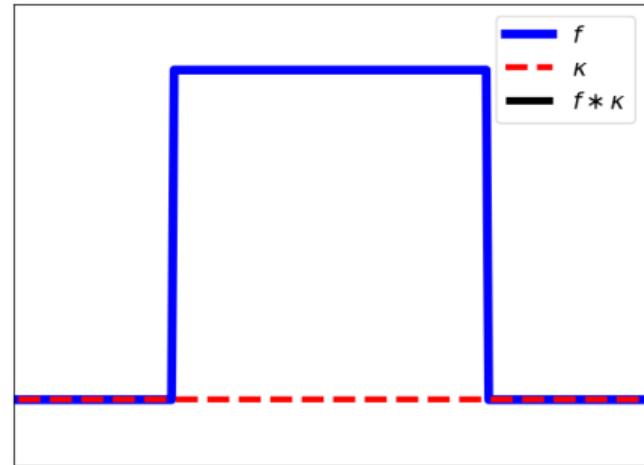
- For a function  $f$  and  
**convolution kernel** (function)  $\kappa$ :

$$(f * \kappa)(x) := \int_{-\infty}^{\infty} f(y)\kappa(y - x)dy$$

- Convolution Theorem:

$$\mathcal{F}(f * \kappa)[k] = \mathcal{F}(f)[k] \cdot \mathcal{F}(\kappa)[k]$$

→ Convolution in space "≡" Multiplication of frequencies



# Back to Operator Learning

- **Why** is this helpful?
- **Goal:** Learn solution operator  $S(f) = u$
- **Theory:** There is a kernel  $\kappa$  with

$$M \underbrace{\sigma(f * \kappa)}_{\text{conv.}} + \underbrace{Af + b}_{\text{lin. Layer}} \approx S(f) = u$$

⇒ **Learn**  $\kappa, A, M, b$

# Back to Operator Learning

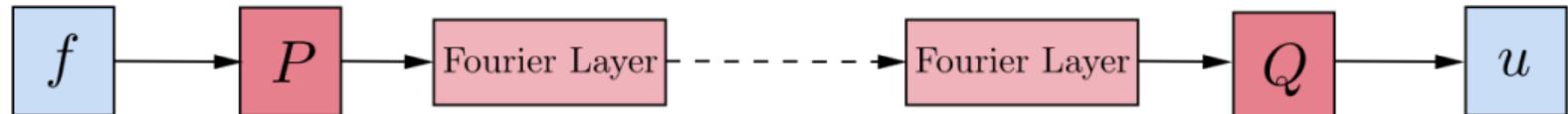
- **Why** is this helpful?
- **Goal:** Learn solution operator  $S(f) = u$
- **Theory:** There is a kernel  $\kappa$  with

$$M \underbrace{\sigma(f * \kappa)}_{\text{conv.}} + \underbrace{Af + b}_{\text{lin. Layer}} \approx S(f) = u$$

⇒ **Learn**  $\kappa, A, M, b$

- **However:** Computing convolutions is expensive!  
→ Fast Fourier transform and convolution theorem to **speed up** computation!

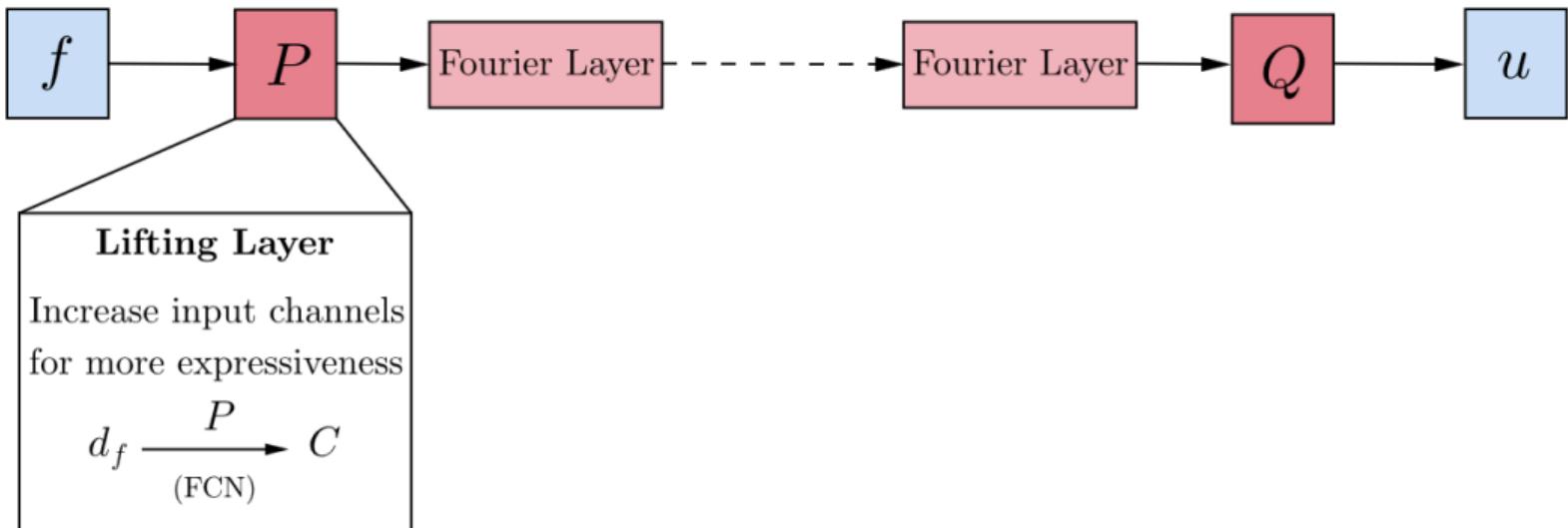
# Fourier Neural Operator (FNO)<sup>1</sup>



---

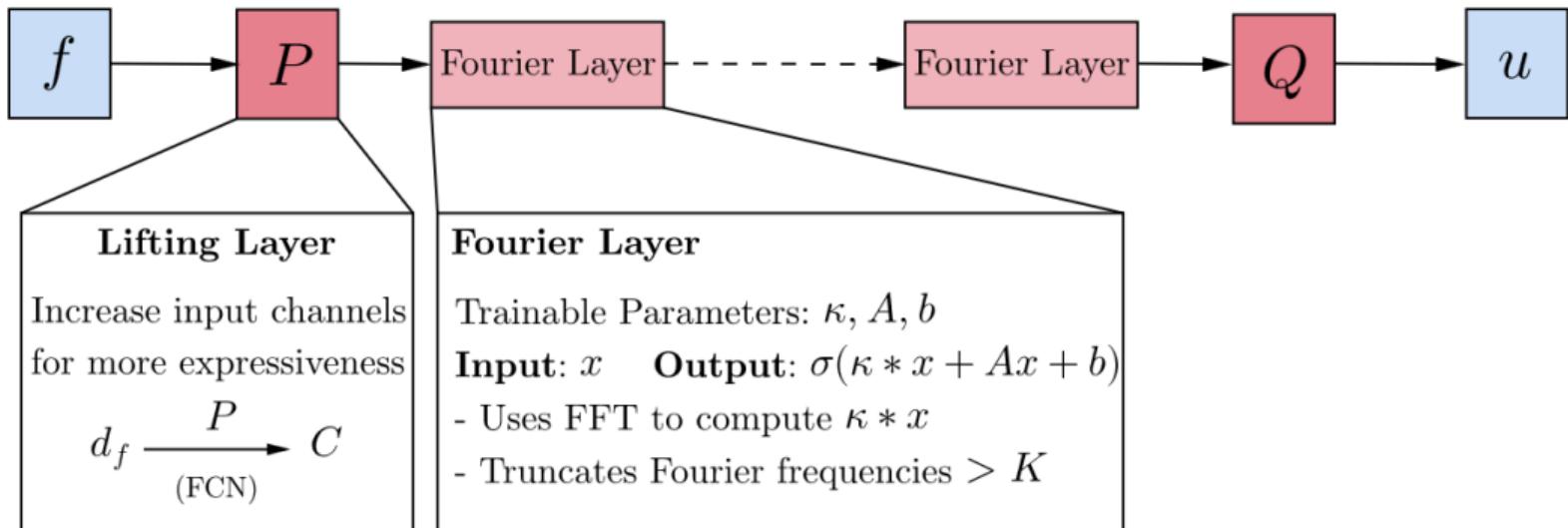
<sup>1</sup> Li et al, *Fourier neural operator for parametric partial differential equations*, 2021

# Fourier Neural Operator (FNO)<sup>1</sup>



<sup>1</sup> Li et al, *Fourier neural operator for parametric partial differential equations*, 2021

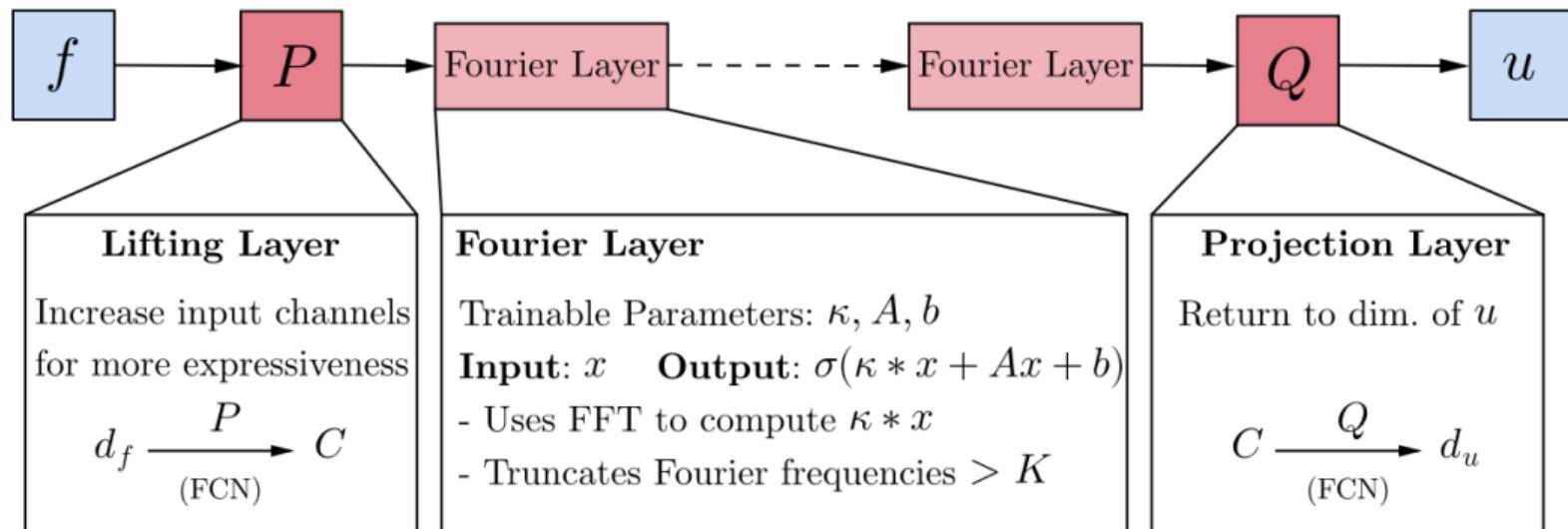
# Fourier Neural Operator (FNO)<sup>1</sup>




---

<sup>1</sup> Li et al, *Fourier neural operator for parametric partial differential equations*, 2021

# Fourier Neural Operator (FNO)<sup>1</sup>



<sup>1</sup> Li et al, *Fourier neural operator for parametric partial differential equations*, 2021

# Benefit for Operator Learning

## Convolutions have high expressiveness

- Often in imaging tasks (*local* convolutions)
- Fast *local & global* convolution  $f * \kappa$  via FFT
- PDE solutions sometimes connected to convolutions

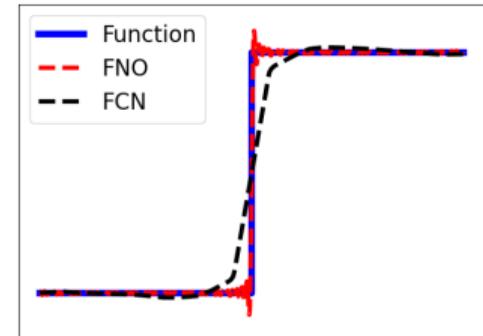
# Benefit for Operator Learning

## Convolutions have high expressiveness

- Often in imaging tasks (*local* convolutions)
- Fast *local & global* convolution  $f * \kappa$  via FFT
- PDE solutions sometimes connected to convolutions

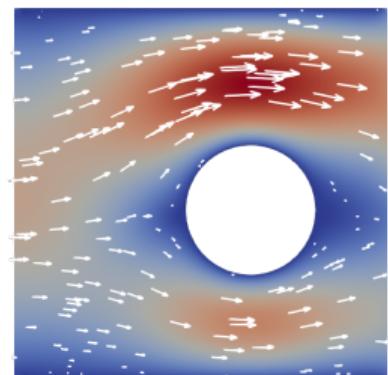
## Additional advantages:

- Truncation of frequencies → reduction of noise
- Fourier transform allows learning of *non-regular* functions



# Using FNOs in TORCHPHYSICS

- ① A joined exercise to see the general implementation:  
`Introduction_FNOs.ipynb`
  
- ② Solving the Stokes equations on different domains:  
`Exercise_8.ipynb`



Stokes solution

	Default FCN	PCA-Net	FNO
Advantages	<ul style="list-style-type: none"><li>• Flexible (arbitrary data)</li></ul>	<ul style="list-style-type: none"><li>• Interpretable PCA-Basis</li><li>• Fast and stable training</li></ul>	<ul style="list-style-type: none"><li>• Leverages spectral information</li><li>• Handles complex functions (e.g., non-smooth)</li></ul>
Disadvantages	<ul style="list-style-type: none"><li>• Scalability issues</li><li>• Complicated to train</li></ul>	<ul style="list-style-type: none"><li>• PCA inherently linear</li><li>• Basis dependent</li></ul>	<ul style="list-style-type: none"><li>• Grid/Domain limitations for Fourier transform</li><li>• Less interpretable</li></ul>