



Universität
Bremen

Center for Industrial
Mathematics (ZeTeM)

Faculty 03

Mathematics / Computer science

Fourier Neural Operators

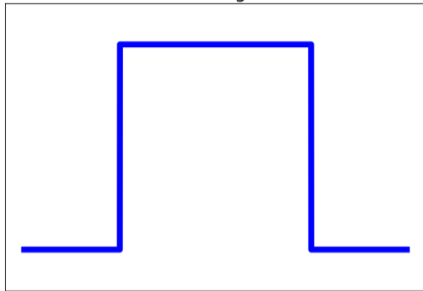
A Fourier Approach for Operator Learning

Janek Gödeke, Nick Heilenkötter, Tom
Freudenberg
Renningen, 21.11.2025

Fourier Transform

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

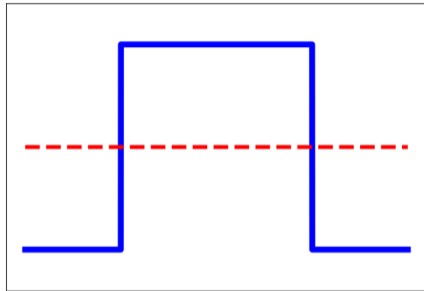
Given Signal



Fourier Transform

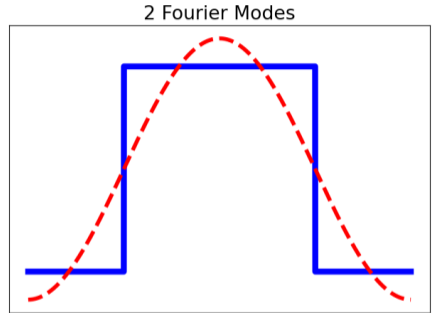
- Reconstruct a function f
by superposition of cosine & sine waves

1 Fourier Modes



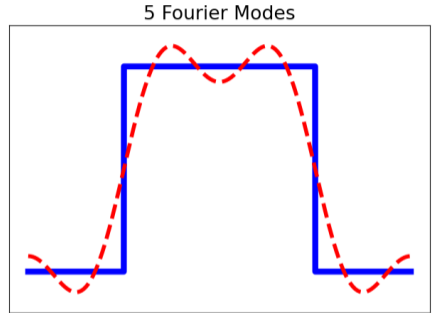
Fourier Transform

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



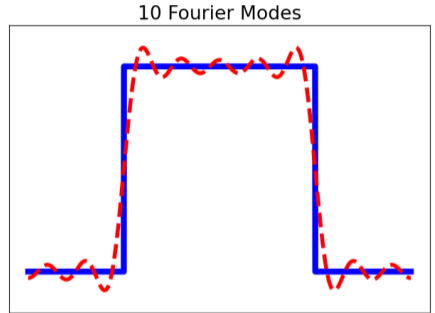
Fourier Transform

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



Fourier Transform

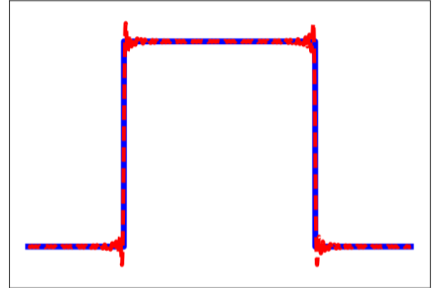
- Reconstruct a function f
by superposition of cosine & sine waves



Fourier Transform

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

100 Fourier Modes



Fourier Transform

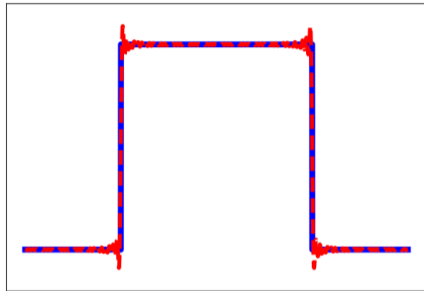
- Reconstruct a function f by superposition of cosine & sine waves
- Fourier transform ($k \in \mathbb{Z}$):

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

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

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

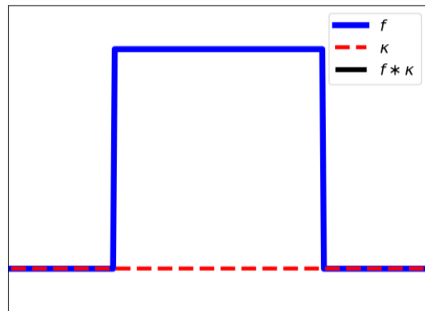
100 Fourier Modes



Convolution Theorem

- For a function f and **convolution kernel** (function) κ :

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



Convolution Theorem

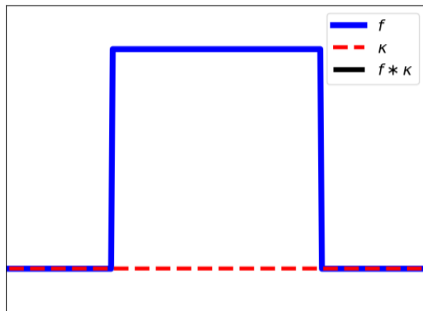
- For a function f and **convolution kernel** (function) κ :

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

- 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 κ with

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

\Rightarrow **Learn** κ, A, M, b

Back to Operator Learning

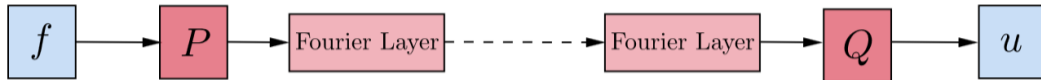
- **Why** is this helpful?
- **Goal:** Learn solution operator $S(f) = u$
- **Theory:** There is a kernel κ with

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

⇒ **Learn** κ, A, M, b

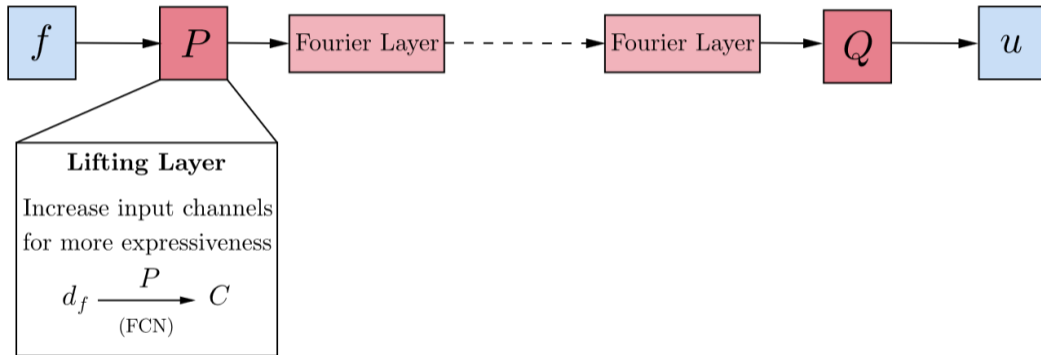
- **However:** Computing convolutions is expensive!
→ Fast Fourier transform and convolution theorem to **speed up** computation!

Fourier Neural Operator (FNO)¹



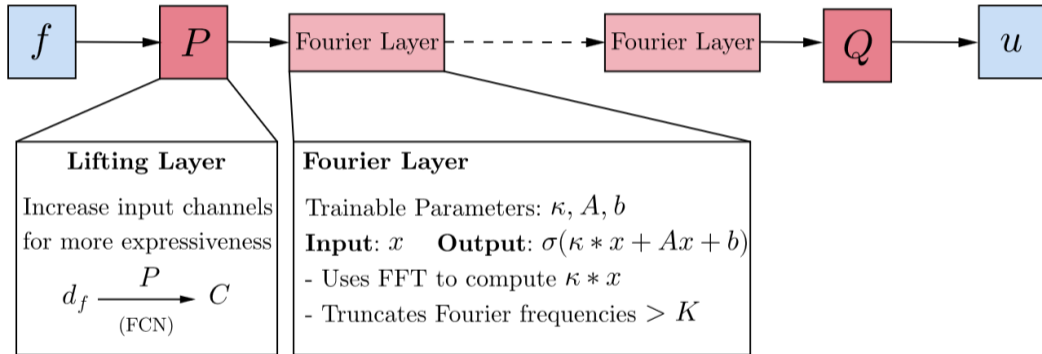
¹ Li et al, *Fourier neural operator for parametric partial differential equations*, 2021

Fourier Neural Operator (FNO)¹



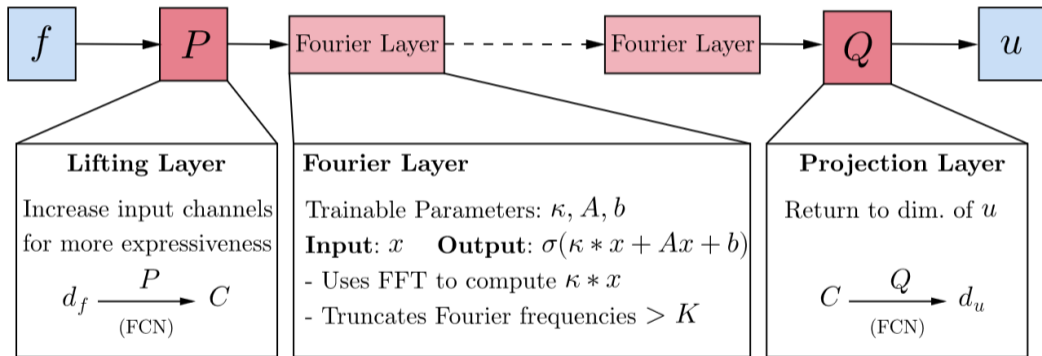
¹ Li et al, *Fourier neural operator for parametric partial differential equations*, 2021

Fourier Neural Operator (FNO)¹



¹ Li et al, *Fourier neural operator for parametric partial differential equations*, 2021

Fourier Neural Operator (FNO)¹



¹ 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

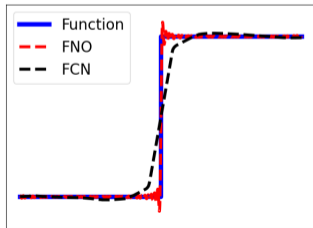
Benefit for Operator Learning

Convolutions have high expressiveness

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

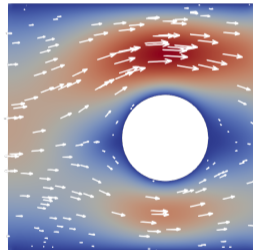
Additional advantages:

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