Solving Forward and Inverse Problems in High-Speed Boundary Layers Using Machine Learning

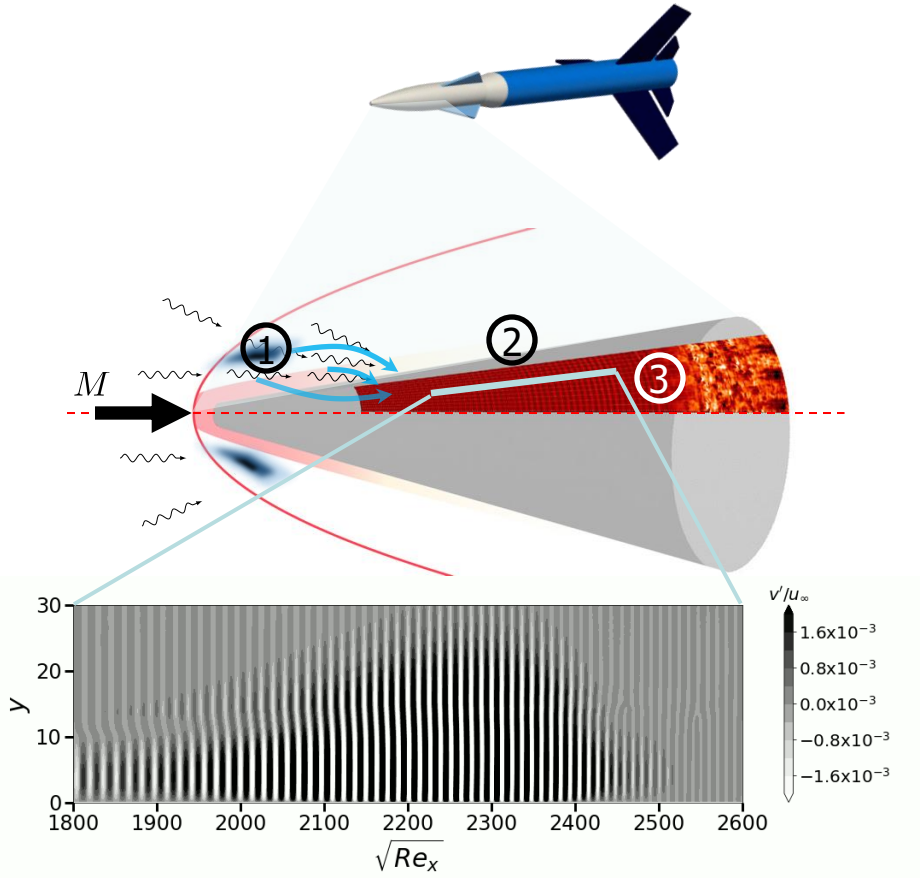
Yue Hao

Advisors: Charles Meneveau & Tamer A Zaki

Advanced Modeling & Simulation (AMS) Seminar Series
NASA Ames Research Center, April 10th, 2025

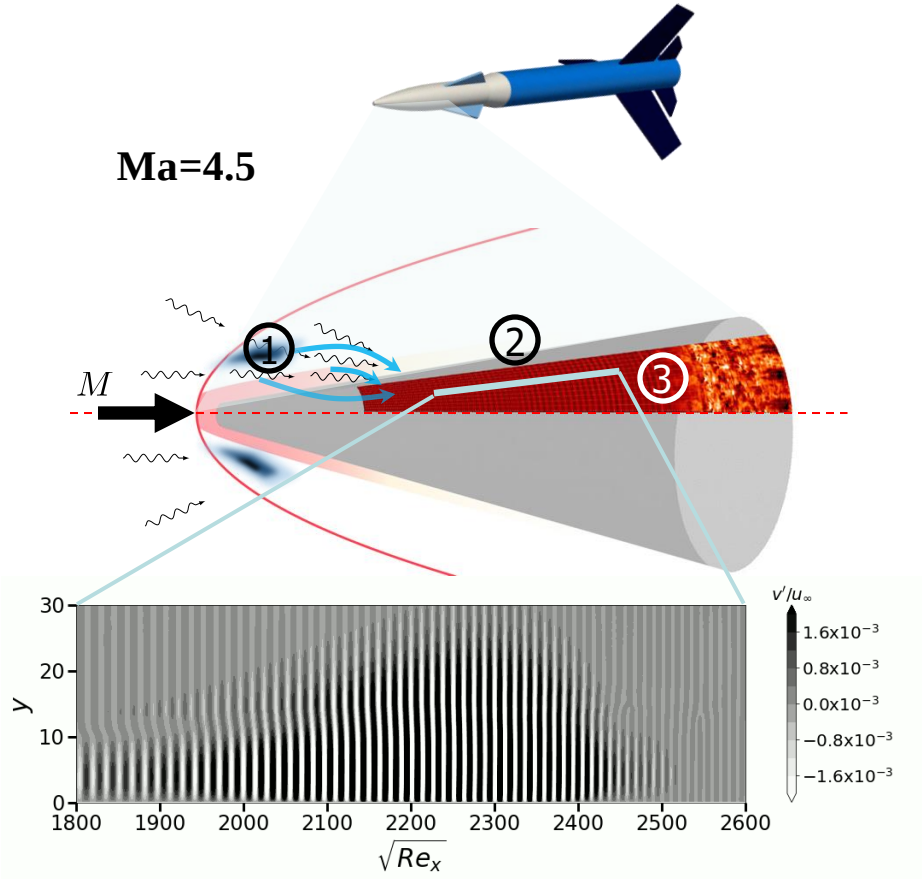
The US Air Force Office of Scientific Research, (Grant FA9550-21-1-0345, , FA9550-23-1-0327).
DARPA/CompMods (Grant HR00112090062).

Transition in hypersonic boundary layer



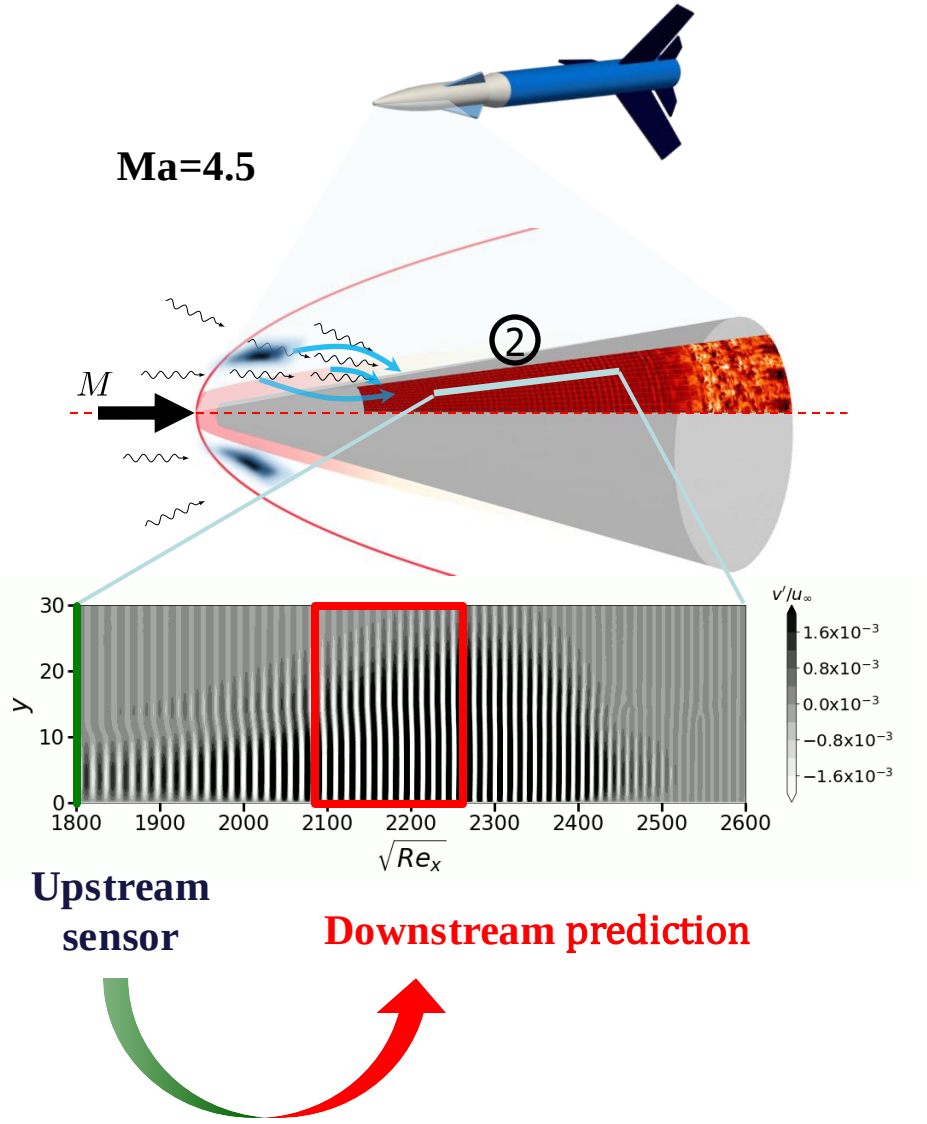
- **1) Shock & gas dissociation**
  - At high Mach numbers, the gas dissociates downstream of the shock with chemical reaction rates commensurate with the flow
- **2) Amplification of instability waves**
  - Instability waves interact with the boundary layer on the flight vehicle and can amplify exponentially
- **3) Transition to turbulence**
  - Once finite amplitude perturbations form, nonlinear interaction spur the generation of harmonics and ultimately break down to turbulence

Transition in hypersonic boundary layer

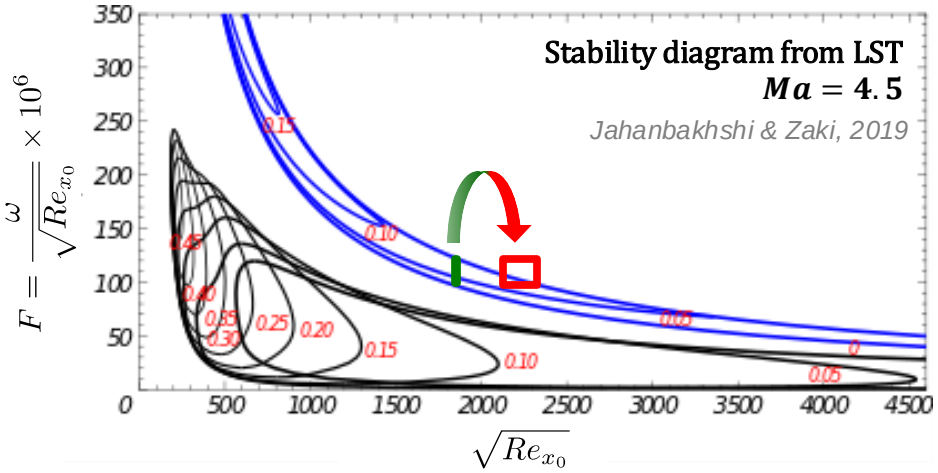


- **1) Shock & gas dissociation**
  - At high Mach numbers, the gas dissociates downstream of the shock with chemical reaction rates commensurate with the flow
- **2) Amplification of instability waves**
  - Instability waves interact with the boundary layer on the flight vehicle and can amplify exponentially
- **3) Transition to turbulence**
  - Once finite amplitude perturbations form, nonlinear interaction spur the generation of harmonics and ultimately break down to turbulence

Fast flow estimation in hypersonic boundary layer



- **Fast estimation of instability waves**
  - **Predict the fate of instability waves from upstream measurements**



- **Challenge**
  - **Sensor is isolated from downstream location**
  - **Full DNS is computationally expensive**

$$\frac{\partial \tilde{\rho}}{\partial t} + \tilde{\nabla} \cdot (\tilde{\rho}\tilde{\boldsymbol{v}}) = 0,$$

$$\tilde{\rho}\left[\frac{\partial \tilde{\boldsymbol{v}}}{\partial t} + (\tilde{\boldsymbol{v}} \cdot \tilde{\nabla})\tilde{\boldsymbol{v}}\right] = -\tilde{\nabla}\tilde{p} + \frac{1}{Re_0}[\tilde{\nabla} \cdot \{\tilde{\lambda}(\tilde{\nabla} \cdot \tilde{\boldsymbol{v}})\boldsymbol{I} + \tilde{\mu}(\tilde{\nabla}\tilde{\boldsymbol{v}} + \tilde{\nabla}\tilde{\boldsymbol{v}}^{\mathrm{T}})\}],$$

$$\tilde{\rho}\left[\frac{\partial \tilde{T}}{\partial t} + (\tilde{T} \cdot \tilde{\nabla})\tilde{\boldsymbol{v}}\right] = \frac{1}{Re_0 Pr_0}\tilde{\nabla} \cdot (\tilde{k}\tilde{\nabla}\tilde{T}) + E_0\left[\frac{\partial \tilde{p}}{\partial t} + (\tilde{\boldsymbol{v}} \cdot \tilde{\nabla})\tilde{p} + \tilde{\Phi}\right],$$

$$\gamma_0 M_0^2 \tilde{p} = \tilde{\rho}\tilde{T}.$$

Fast flow estimation in hypersonic boundary layer

- **Up/down stream flow mapping**
  - **Upsteam measurement**

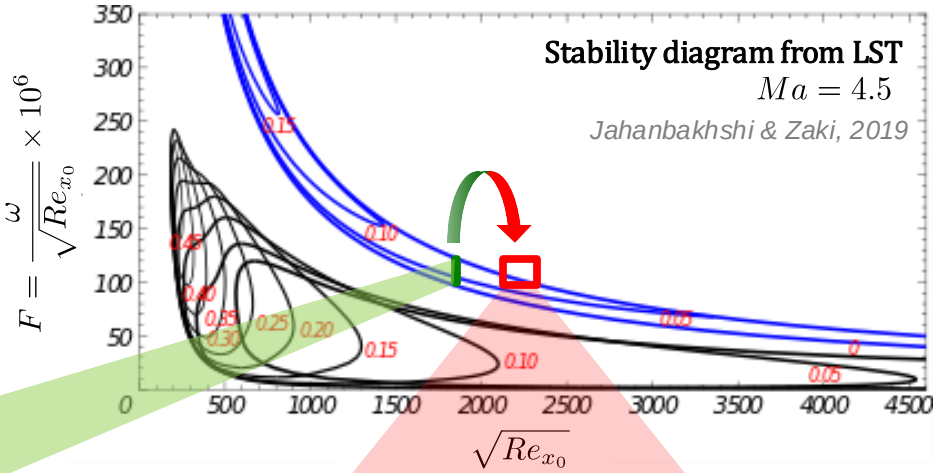
$$p_w\left(x_0, t\right)$$

"N-S" Operator *G*

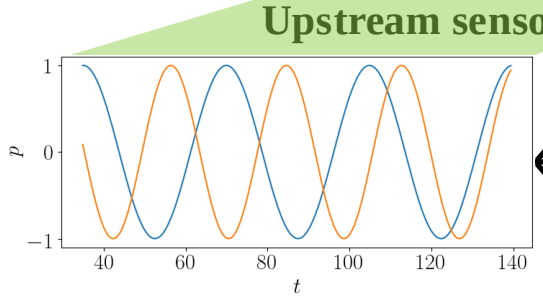
  - **Downstream flow**

$$q\left(x, y, t\right)$$

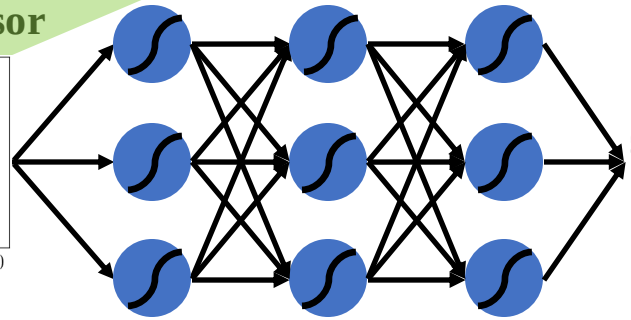
- **Fast estimation of instability waves**
  - **Predict the fate of instability waves from upstream measurements**



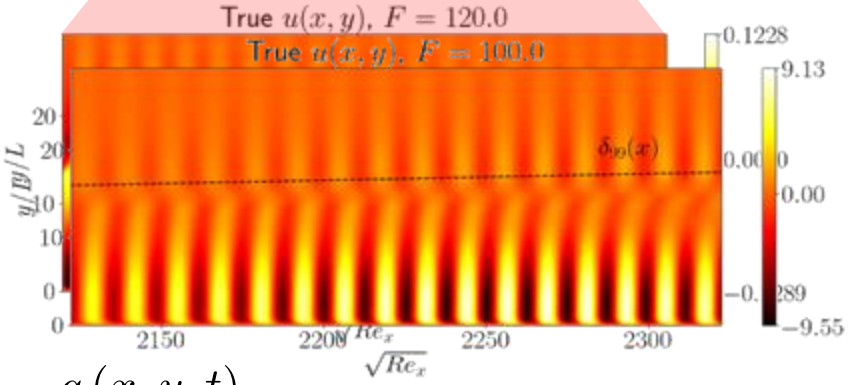
Upstream sensor



$p_w\left(x_0, t\right)$



Downstream prediction



$q\left(x, y, t\right)$

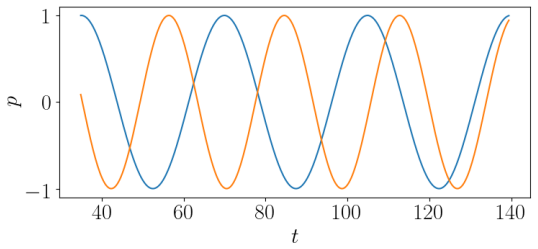
Method

- **Universal approximation theorem of Operators**
  - Input function / signal: p_w; Coordinates of interest: x, y, t
  - Output function: $G(p_w)(x,y,t) = \sum_{k=1}^{p}\sum_{i=1}^{n} c_j^k \sigma\left(\sum_{j=1}^{m} \xi_{ij}^k p_w(x_0,t_j) + \theta_j^k\right)\sigma\left(w_k \cdot [x,y,t] + \zeta_k\right)$

- **Deep Operator Neural Networks (DeepONet)**

- Branched networks can approximate any operator on bounded domains
- Functional input does not have to be regular
- Branched architecture prevents lopsidedness

input function / signal

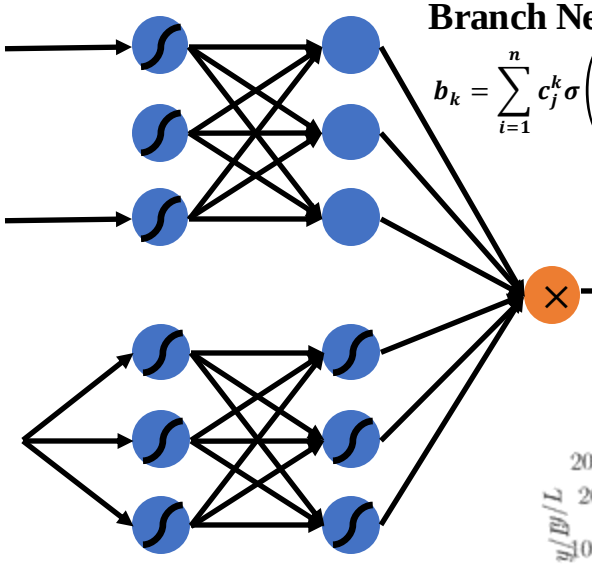


$p_w(x_0, t_1)$

...

$p_w(x_0, t_m)$

coordinates of interest (x, y, t)



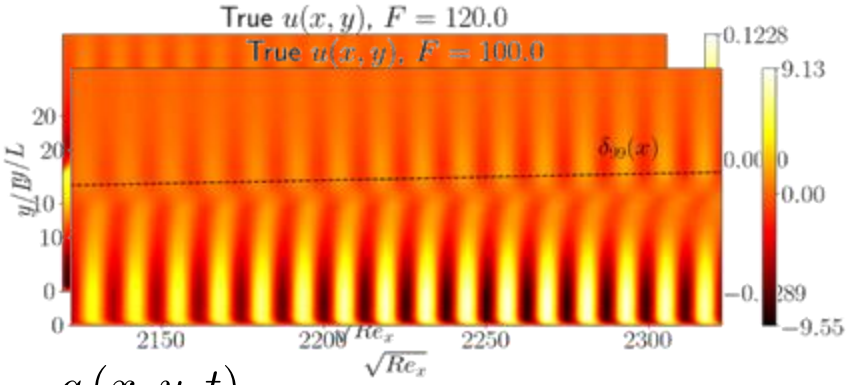
Branch Network:

$$b_k = \sum_{i=1}^{n} c_j^k \sigma\left(\sum_{j=1}^{m} \xi_{ij}^k p_w(x_0,t_j) + \theta_j^k\right)$$

Trunk Network:

$$t_k = \sigma(w_k \cdot [x,y,t] + \zeta_k)$$

$$G(p_w)(x,y,t) = \sum_{k=1}^{p} b_k t_k$$

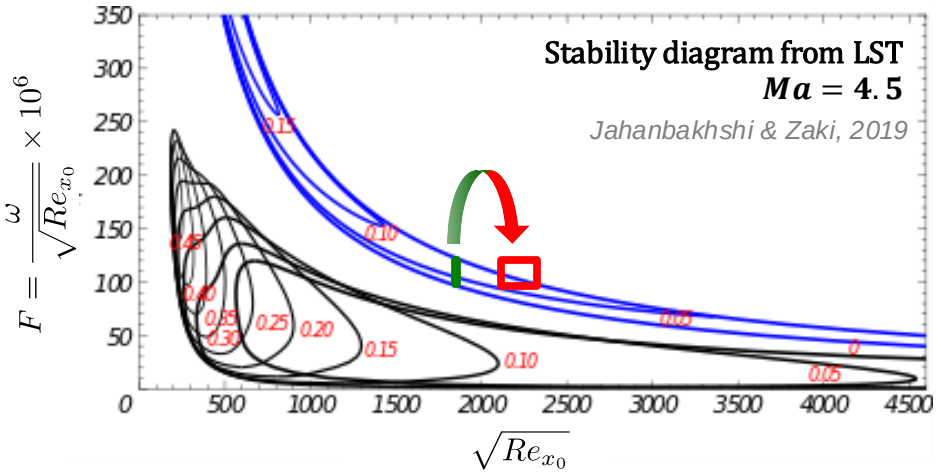


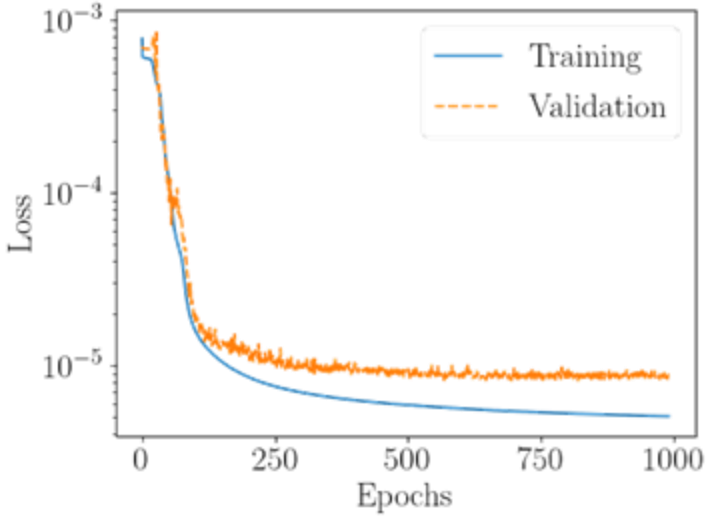
$q(x, y, t)$

Universal function approximation theorem (Cybenko, 1989)
Universal operator approximation theorem (Chen & Chen, 1993)
Deep Operator Networks (Lu et al, 2020)

Datset

- **Data generation**
  - DNS of Ma=4.5 hypersonic boundary layer
- **Training data**
  - 51 instability waves with frequencies in the range F=[100, 125]
  - Data were augmented by varying the phase of the inputs and associated outputs
- **Testing/validation data**
  - 8 independent instabilities were reserved for validation
- **Architecture**
  - 6 layers with 100 units in each branch
  - Activation function: ELU



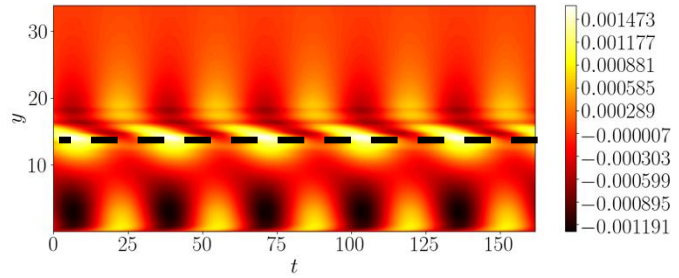


Results

- **Field comparison: DeepONets prediction vs. DNS**

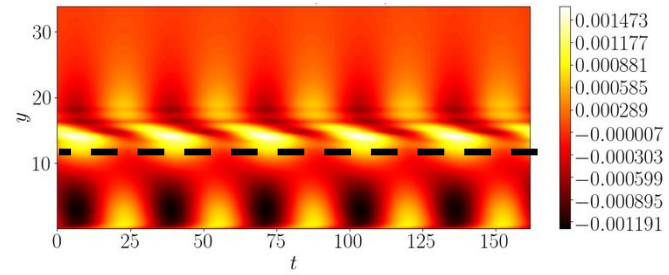
DeepONets prediction
$u(x_0, y, t), F = 107.75$

$y = y_0$

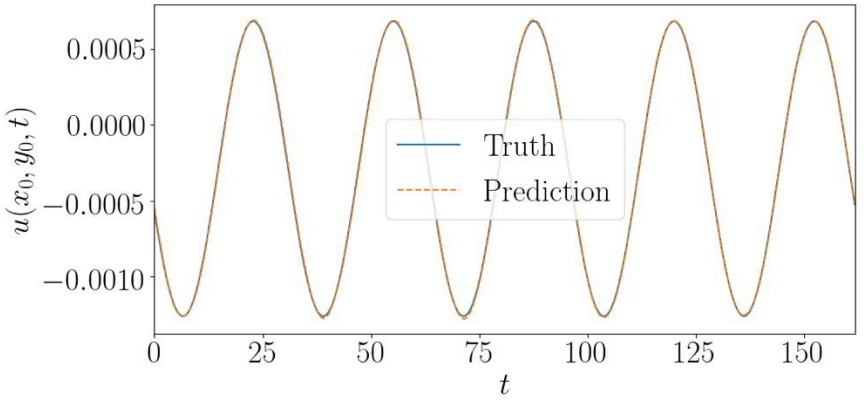


DNS result
$u(x_0, y, t), F = 107.75$

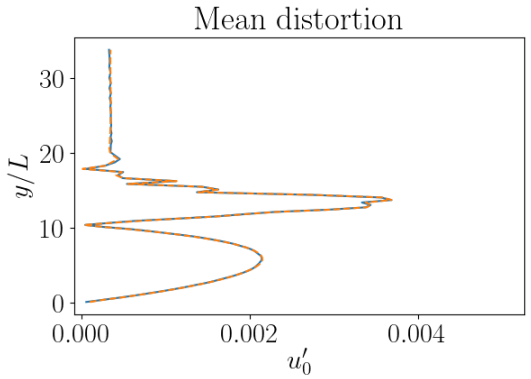
$y = y_0$

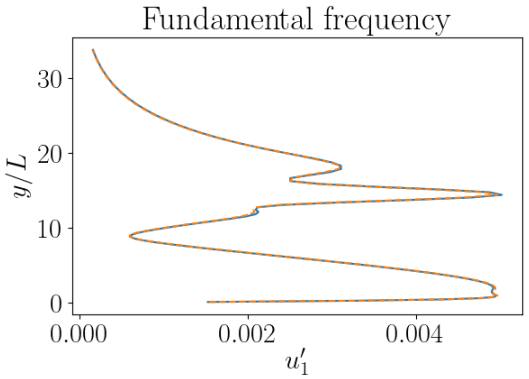


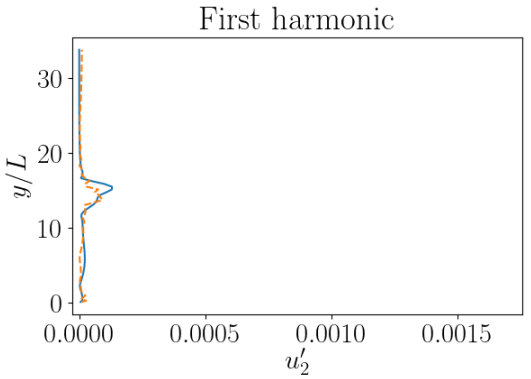
Comparison at $y = y_0$



- **Accuracy in Fourier space**
  - **Accurate prediction of the base-state distortion, and both the fundamental and harmonic waves**





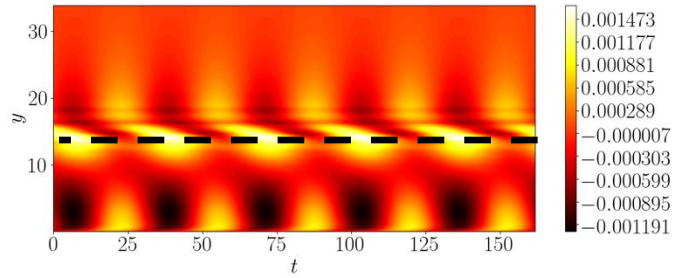


Results

- **Field comparison: DeepONets prediction vs. DNS**

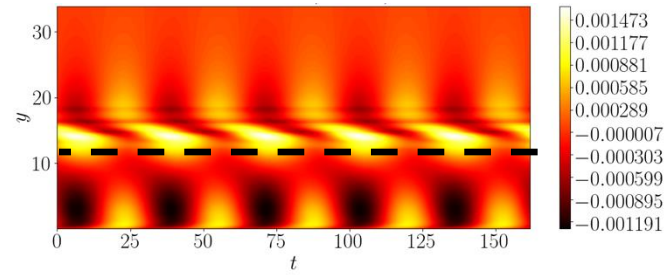
DeepONets prediction
$u(x_0, y, t), F = 107.75$

$y = y_0$

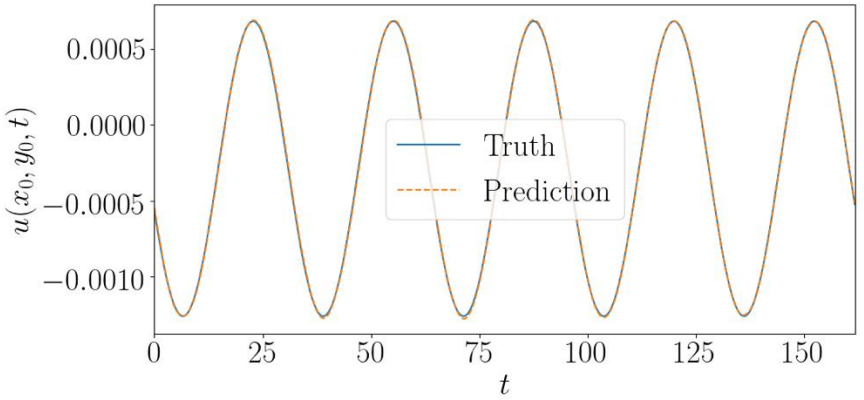


DNS result
$u(x_0, y, t), F = 107.75$

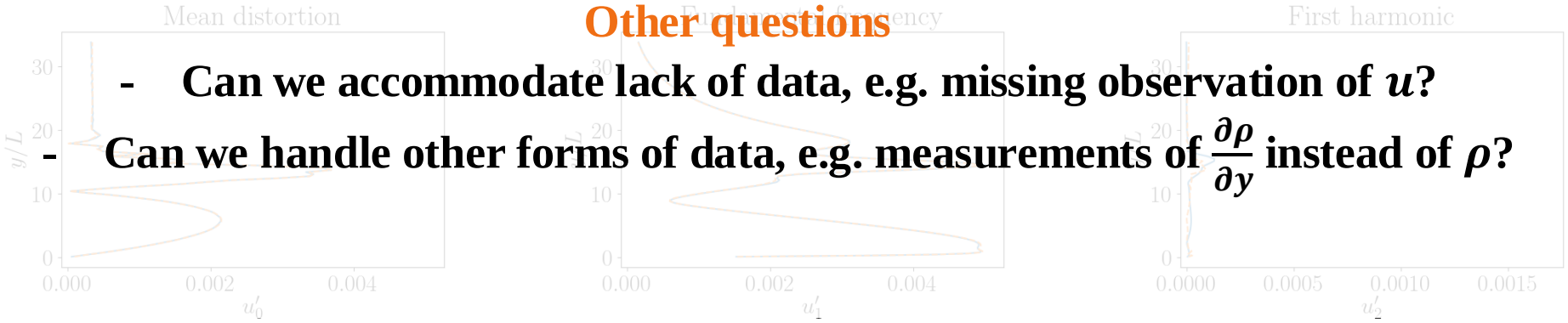
$y = y_0$



Comparison at $y = y_0$



- **Accuracy in Fourier space**
  - **Accurate prediction of the base-state distortion, and both the fundamental and harmonic waves**



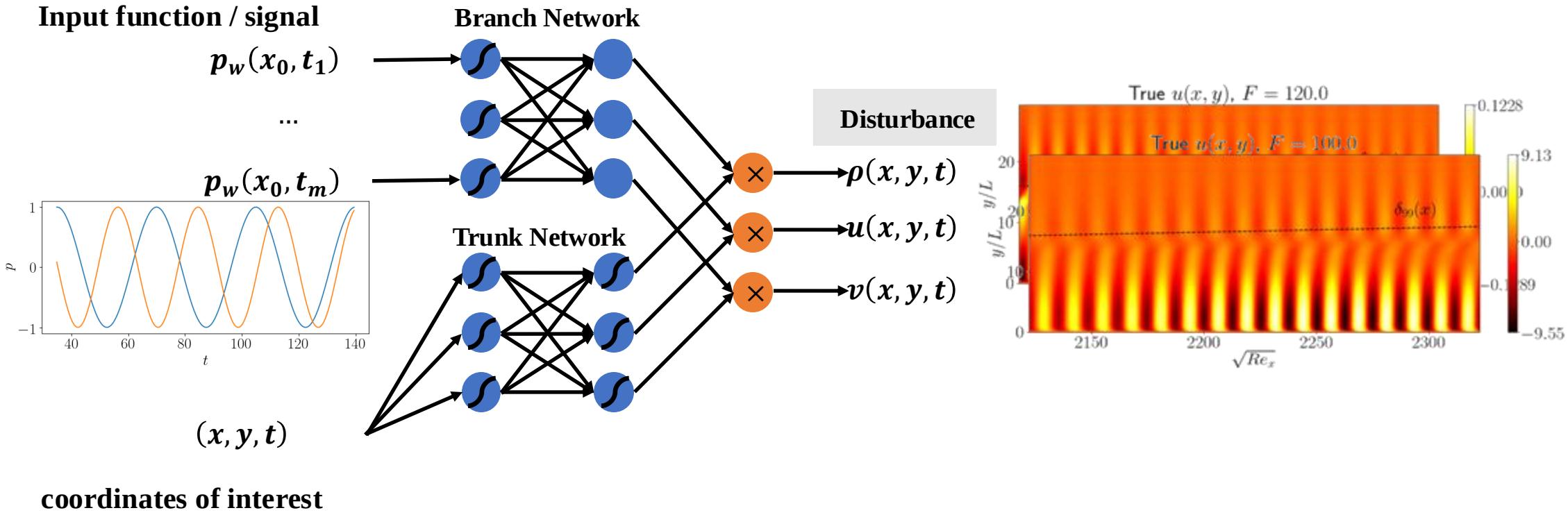
Other questions

- **Can we accommodate lack of data, e.g. missing observation of u?**
- **Can we handle other forms of data, e.g. measurements of $\frac{\partial \rho}{\partial y}$ instead of ρ?**

Method

- **Physics Informed DeepONet**
  - **Helps regularize the network**

$$L = \frac{1}{N_s} \sum_{i \in \Omega_s} \left[(\rho_i - \rho_i^*)^2 + (u_i - u_i^*)^2 + (v_i - v_i^*)^2 \right]$$



Method

- **Physics Informed DeepONet**
  - **Helps regularize the network**

$$L = \frac{1}{N_s}\sum_{i\in\Omega_s}\left[(\rho_i-\rho_i^*)^2+(u_i-u_i^*)^2+(v_i-v_i^*)^2\right]+\frac{1}{N_c}\sum_{i\in\Omega_c}|f_i|^2$$

$$f = \partial_t\tilde{\rho}+\partial_x(\tilde{\rho}\tilde{u})+\partial_x(\tilde{\rho}\tilde{v})$$

Full field

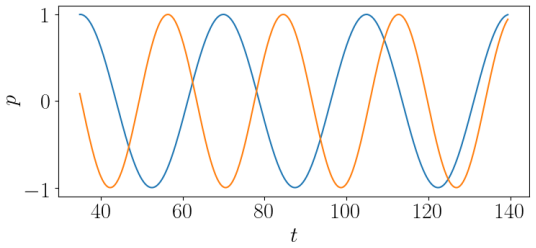
$$\rho_B+\rho=\tilde{\rho}$$
$$u_B+u=\tilde{u}$$
$$v_B+v=\tilde{v}$$

Input function / signal

$p_w(x_0,t_1)$

...

$p_w(x_0,t_m)$



(x,y,t)

coordinates of interest

Branch Network

Trunk Network

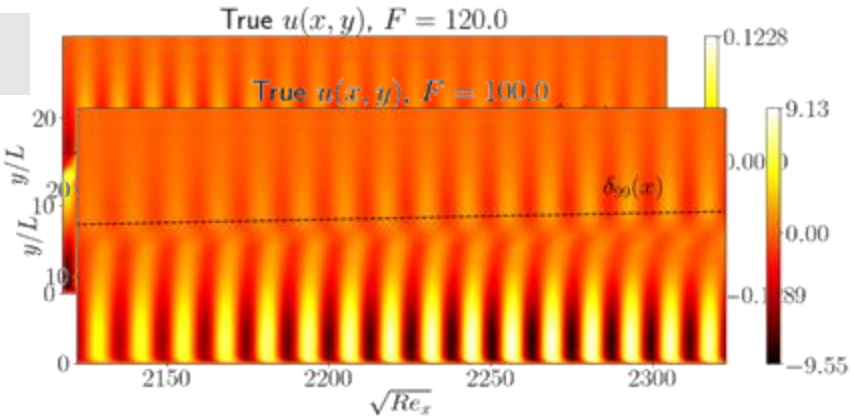
Baseflow PINN

Disturbance

$\rho(x,y,t)$

$u(x,y,t)$

$v(x,y,t)$

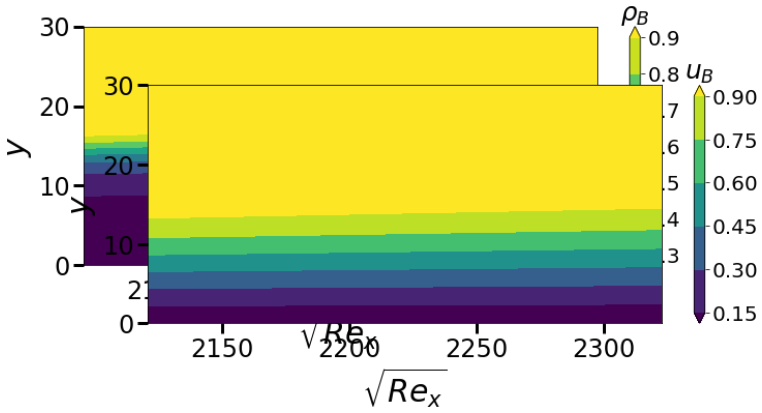


Base flow

$\rho_B(x,y)$

$u_B(x,y)$

$v_B(x,y)$



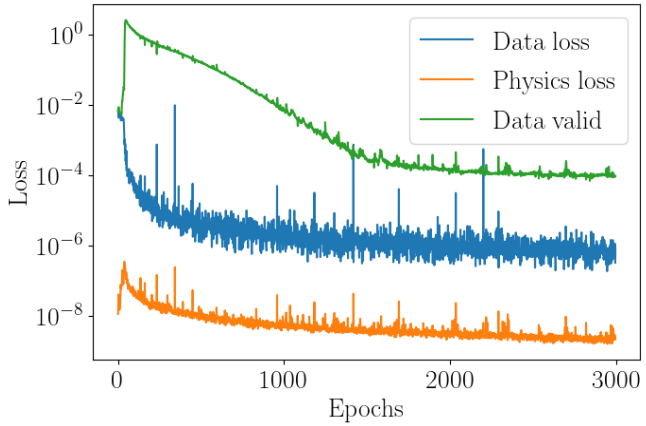
Results

- **Test1**
  - The equations can be adopted in lieu of data.
  - In this case no data on u was given during training.
  - Loss function

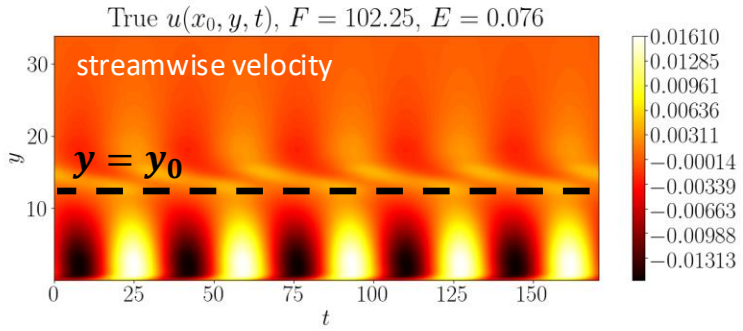
$$L = \frac{1}{N_s}\sum_{i\in\Omega_s}\left[(\rho_i-\rho_i^*)^2+(v_i-v_i^*)^2\right]+\frac{1}{N_c}\sum_{i\in\Omega_c}|f_i|^2$$

  - The network can infer the missing field

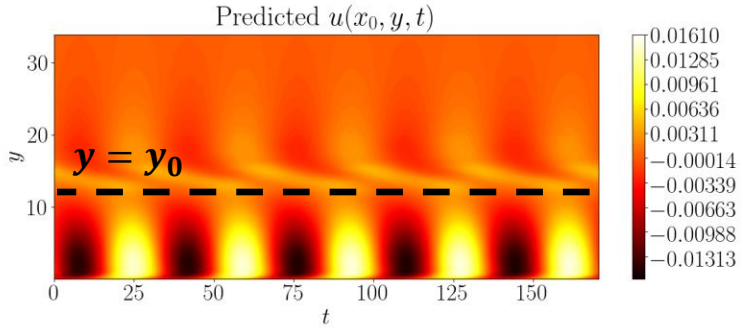
Training history



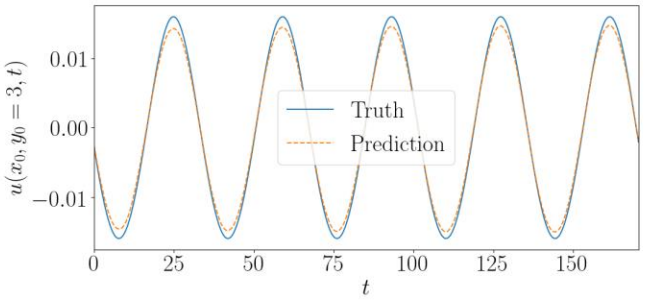
DNS result



DeepONets prediction



Comparison at $y = y_0$

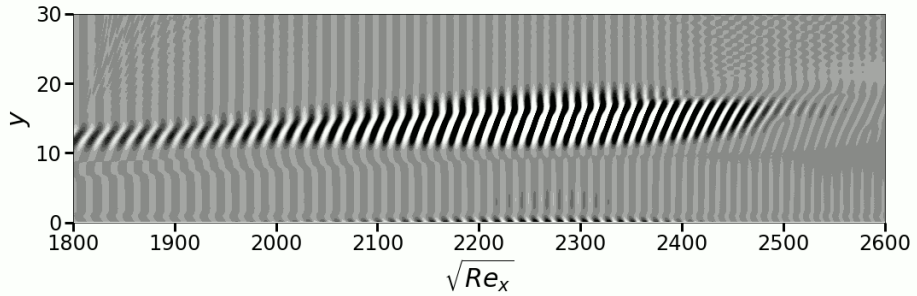


Results

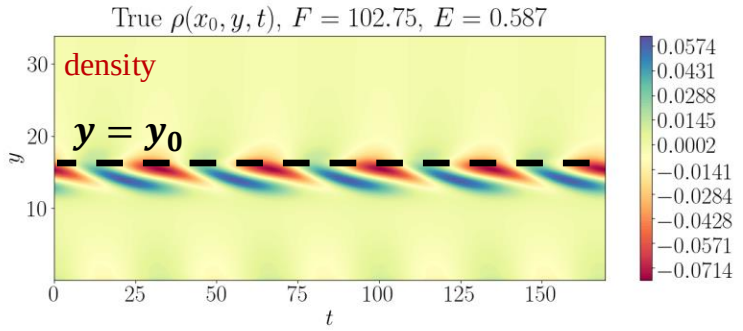
- **Test2**
  - **PI-DeepONets can also take other forms of data**
  - **Here we supply data over all the spatiotemporal domain but of the vertical gradient of the density**

$$L = \frac{1}{N_s}\sum_{i\in\Omega_s}\left[\left(\frac{\partial \rho_i}{\partial y} - \frac{\partial \rho_i^*}{\partial y}\right)^2 + (u_i - u_i^*)^2 + (v_i - v_i^*)^2\right] + \frac{1}{N_c}\sum_{i\in\Omega_c}|f_i|^2$$

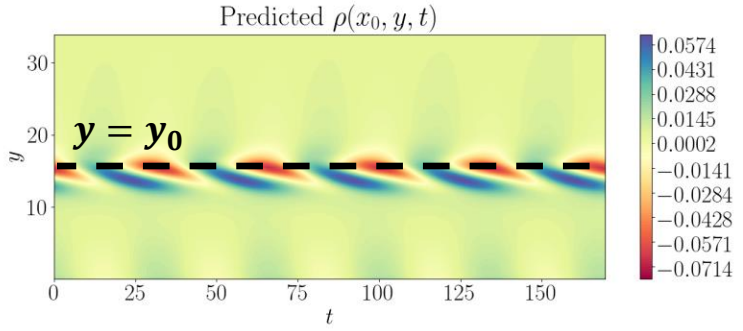
  - **This mimics the type of data that is obtained in Schlieren measurements**



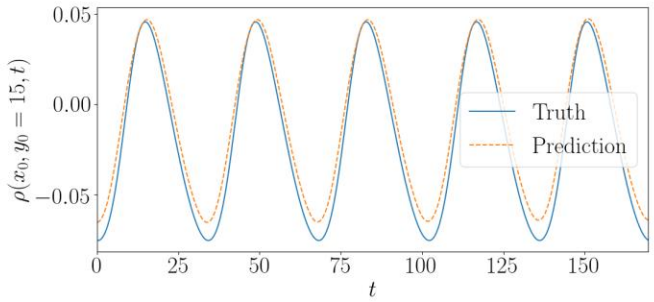
DNS result



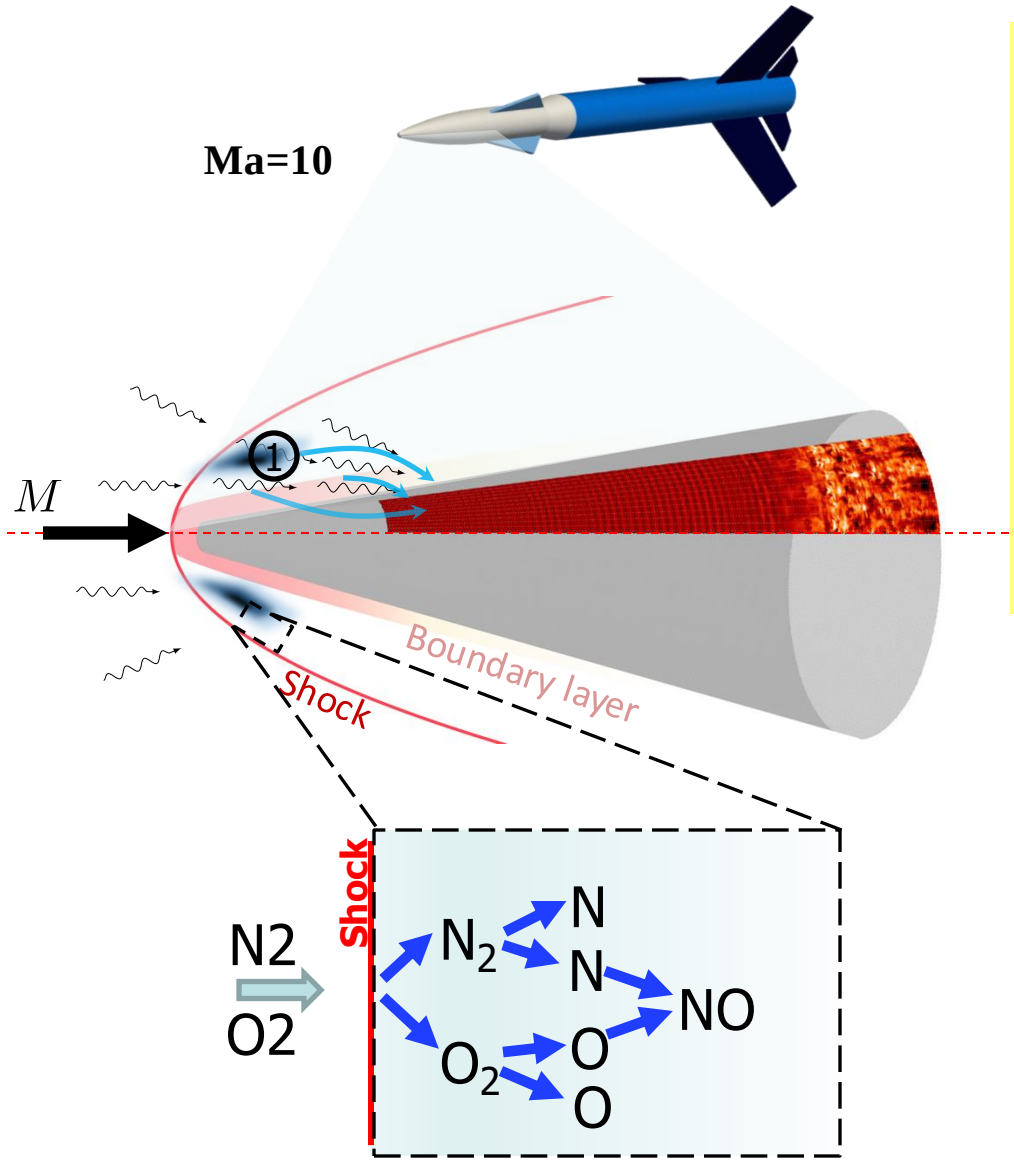
DeepONets prediction



Comparison at $y = y_0$

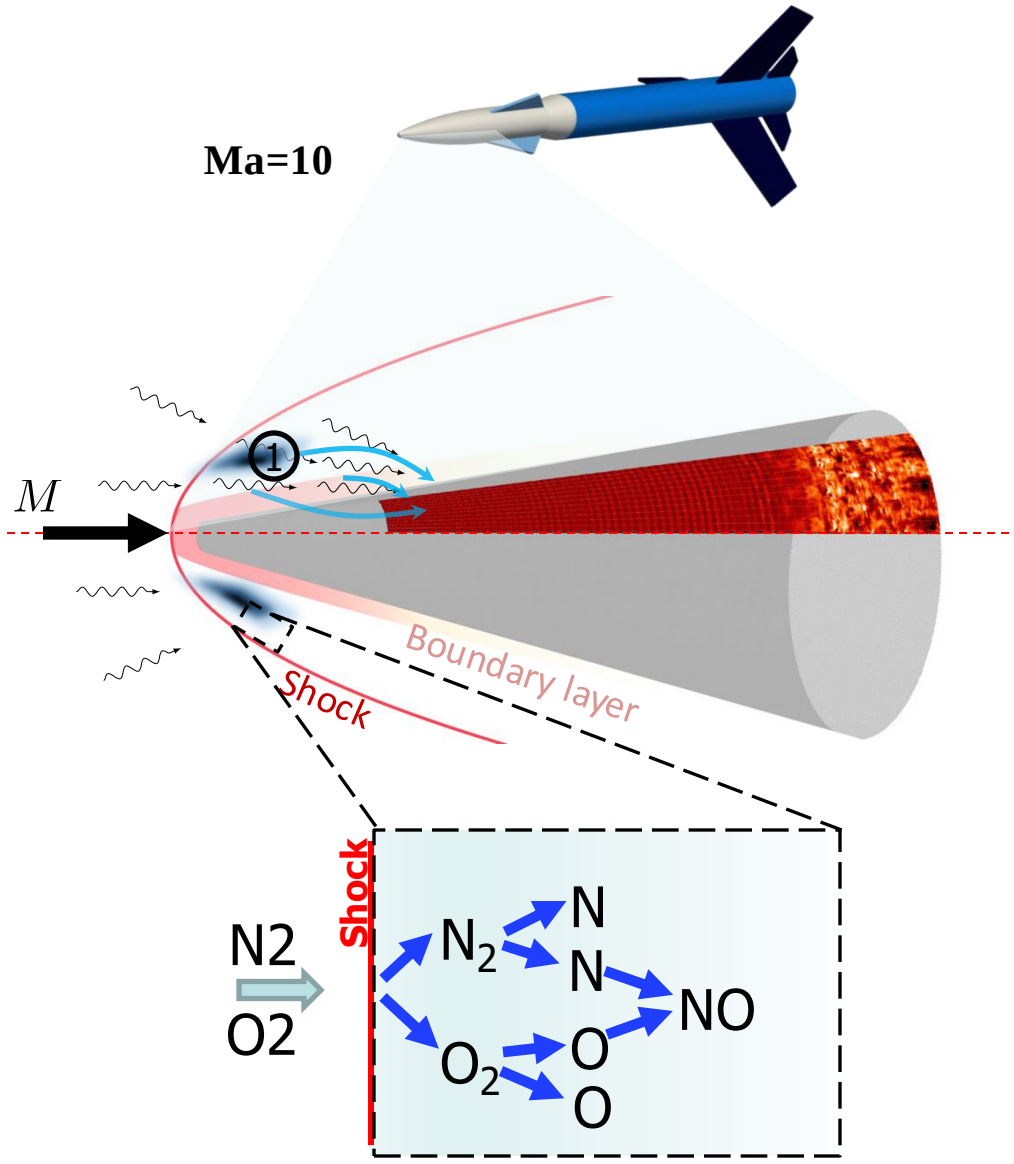


Transition in hypersonic boundary layer



- 1) Shock & gas dissociation
  - At high Mach numbers, the gas dissociates downstream of the shock with chemical reaction rates commensurate with the flow
- 2) Amplification of instability waves
  - Instability waves interact with the boundary layer on the flight vehicle and can amplify exponentially
- 3) Transition to turbulence
  - Once finite amplitude perturbations form, nonlinear interaction spur the generation of harmonics and ultimately break down to turbulence

Transition in hypersonic boundary layer



- **Compressible Navier Stokes equations with chemically reacting gas**

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{u}) = 0,$$

$$\frac{\partial \rho^s}{\partial t} + \nabla \cdot (\rho \boldsymbol{u} + \boldsymbol{J^s}) = \dot{w}^s, \qquad s = 1, \dots, NS,$$

$$\frac{\partial \rho \boldsymbol{u}}{\partial t} + \nabla \cdot (\rho \boldsymbol{u} \otimes \boldsymbol{u}) = -\nabla p + \frac{1}{Re_\infty} \nabla \cdot [\lambda (\nabla \cdot \boldsymbol{u}) \boldsymbol{I} + \mu (\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T)],$$

$$\frac{\partial E}{\partial t} + \nabla \cdot [(E + p)\boldsymbol{u}] = -\nabla \cdot \left(-\frac{1}{Re_\infty Pr_\infty} k \nabla T + \sum_s h^s \boldsymbol{J^s} \right),$$

$$p^s = \chi^s p = \frac{1}{\gamma M_\infty^2} \rho^s T,$$

where $\rho = \sum_s \rho^s$, $p = \sum_s p^s$, $E = \frac{\rho e}{Ec_\infty} + \frac{1}{2} \rho \boldsymbol{u} \cdot \boldsymbol{u}$

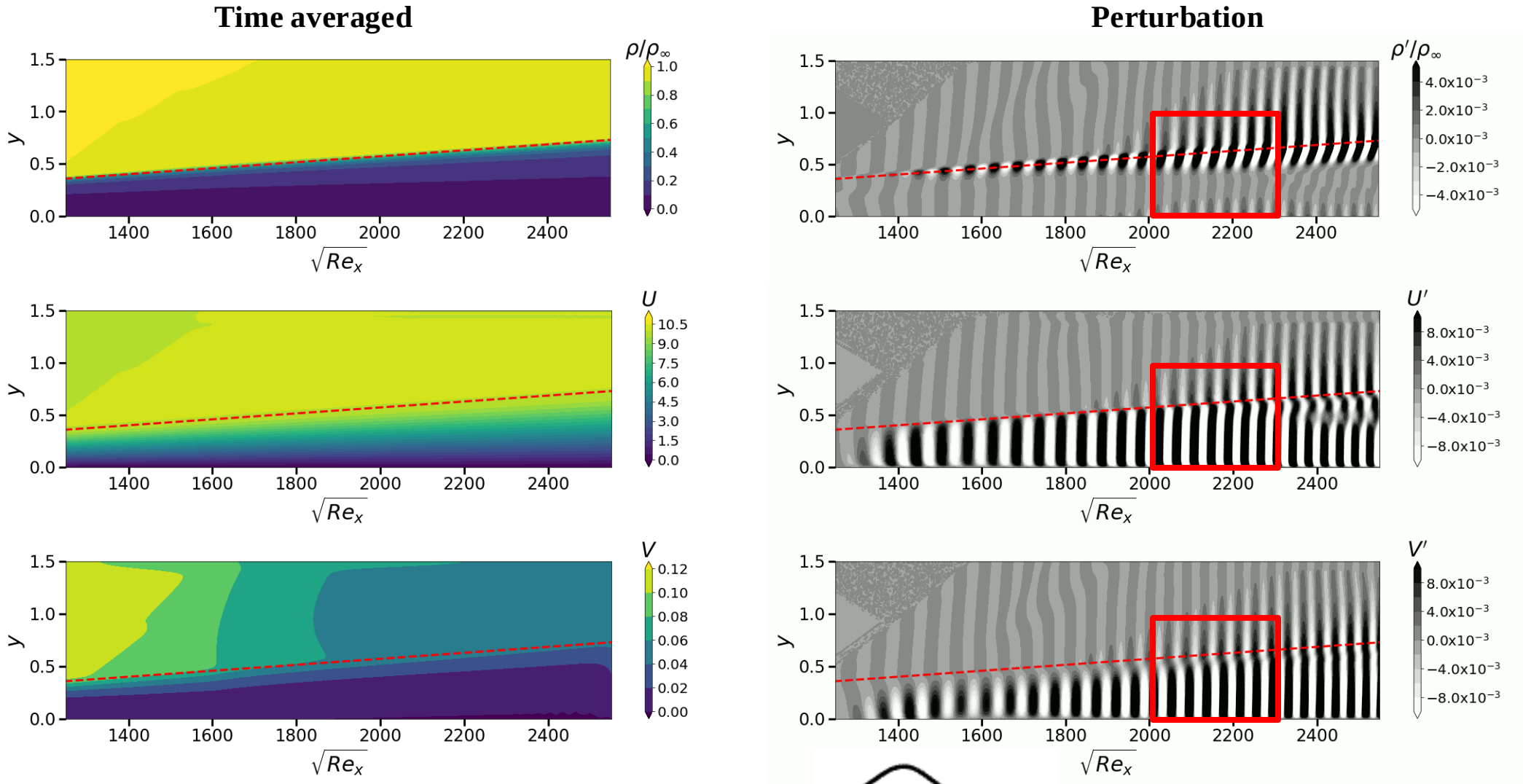
$k, \mu, \boldsymbol{J^s}$ (mass diffusion flux) solved from material property equations,

reaction rate $\dot{w}^s = \frac{M^{s*} \Omega^{s*}(T^*, \rho^*, p^*, \chi^s)}{\rho_\infty^* a_\infty^* / L_{ref}^*}$ is computed from Arrhenius law,

$$\Omega^{s*} = \sum_{s=1}^{NS} \left(\nu_{P,\eta}^s - \nu_{R,\eta}^s \right) \left[k_{f,\eta}^* \prod_{r=1}^{NS} \left(\frac{\rho^{r*}}{M^{r*}} \right)^{\nu_{R,\eta}^s} - k_{b,\eta}^* \prod_{r=1}^{NS} \left(\frac{\rho^{r*}}{M^{r*}} \right)^{\nu_{P,\eta}^s} \right].$$

Results

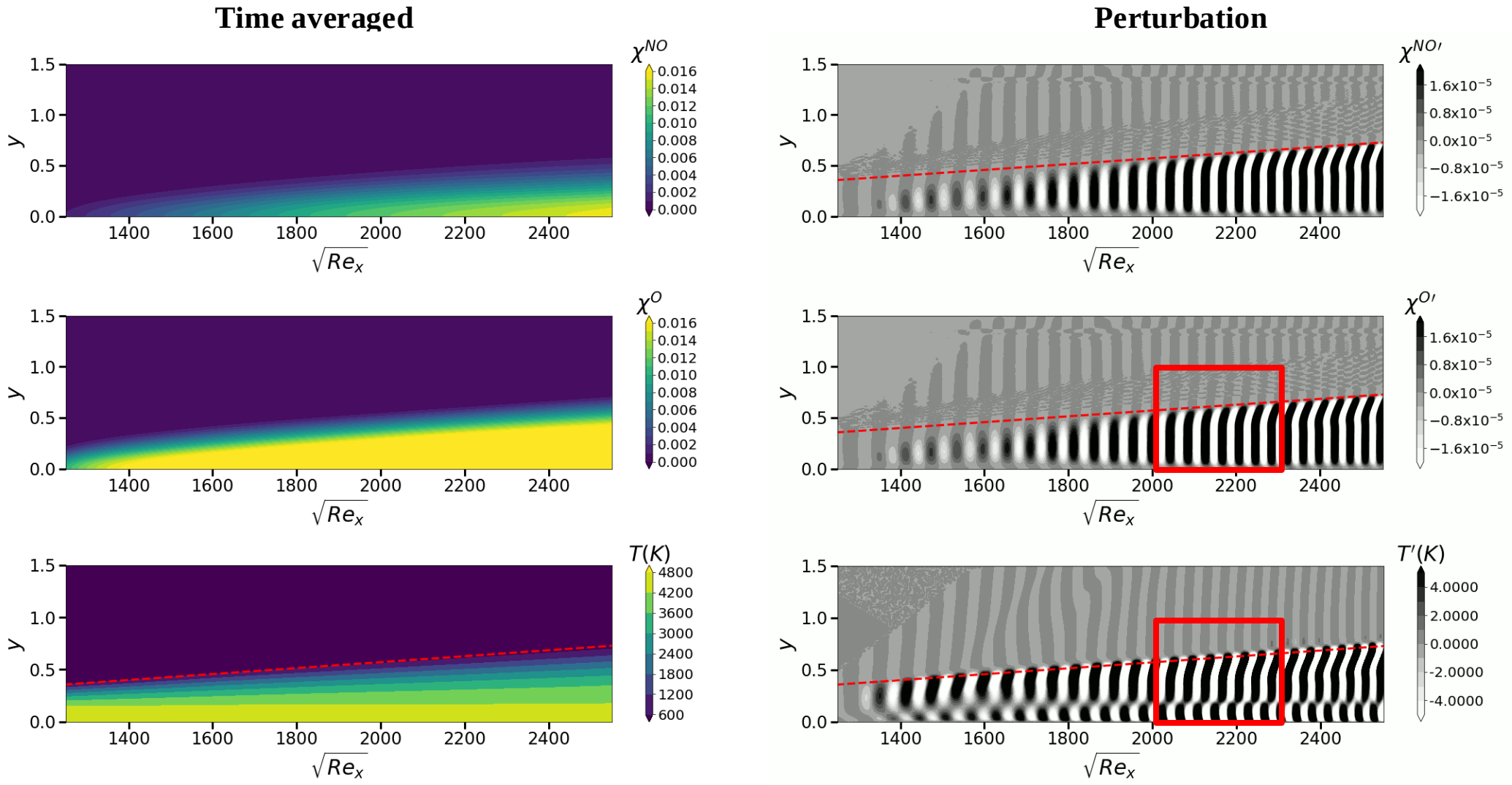
- **Ma=10 boundary layer simulation**



$$(\rho v)_{wall} = A_v \sin(2\pi\xi - \omega t) exp(-0.4\xi^2)$$

Results

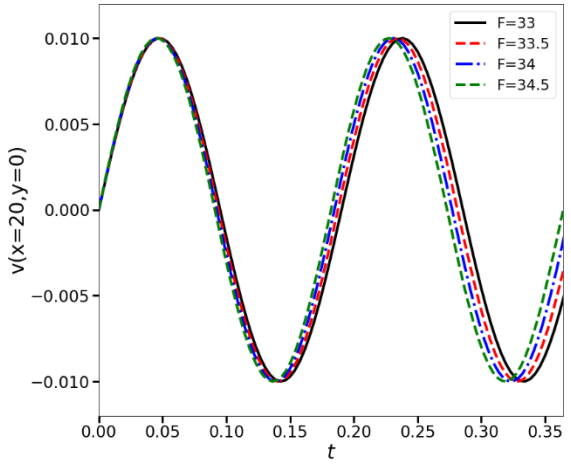
- **Ma=10 boundary layer simulation**

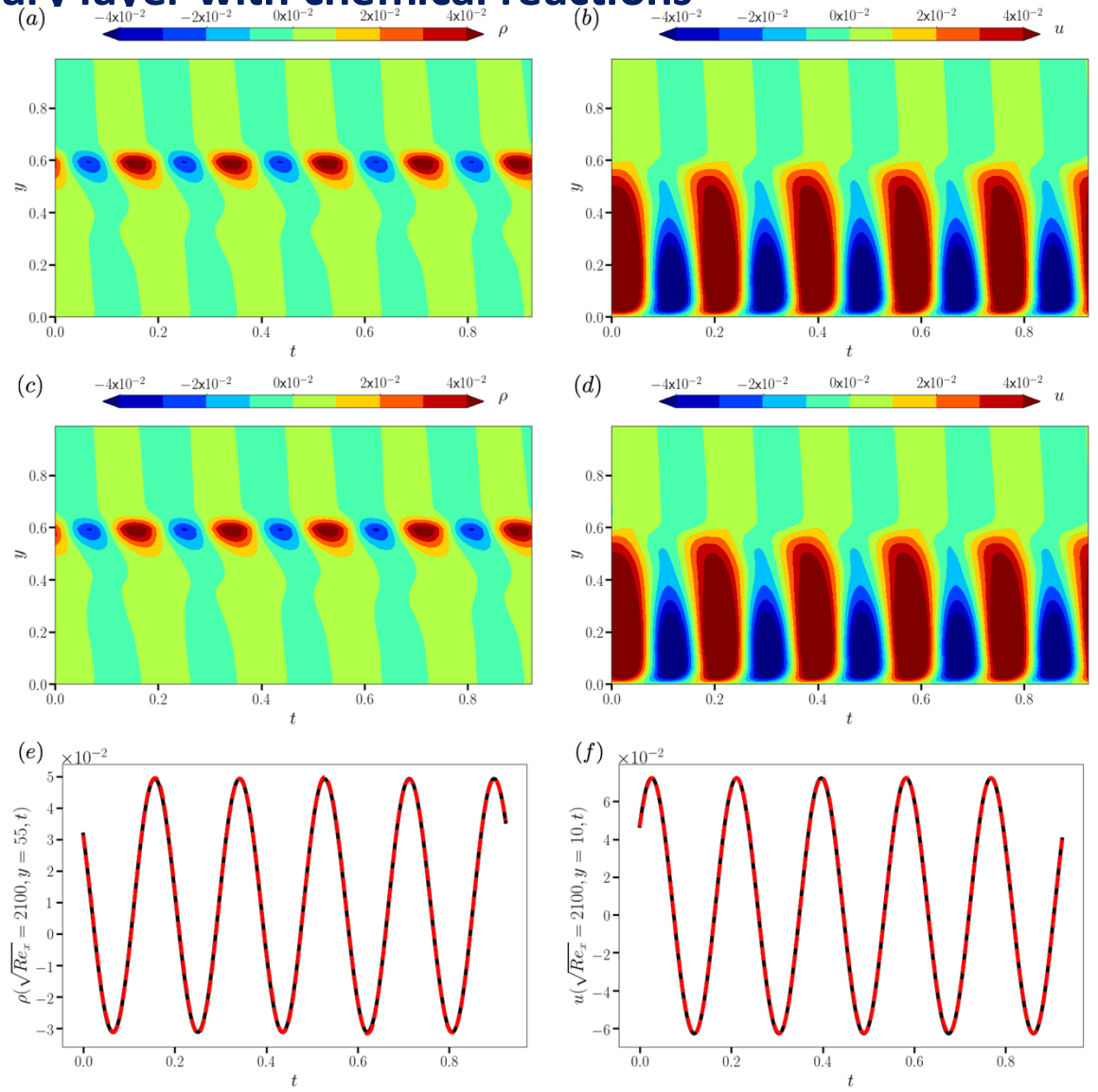


Results

- **Forward problem to Ma=10 boundary layer with chemical reactions**
  - **Prediction of testing data**

Input: velocity perturbation at center of wall forcing

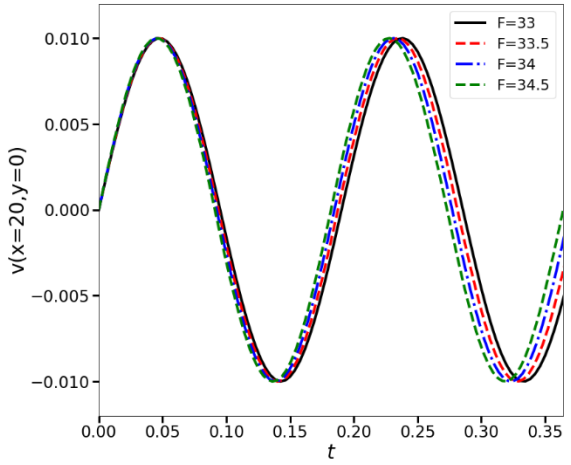


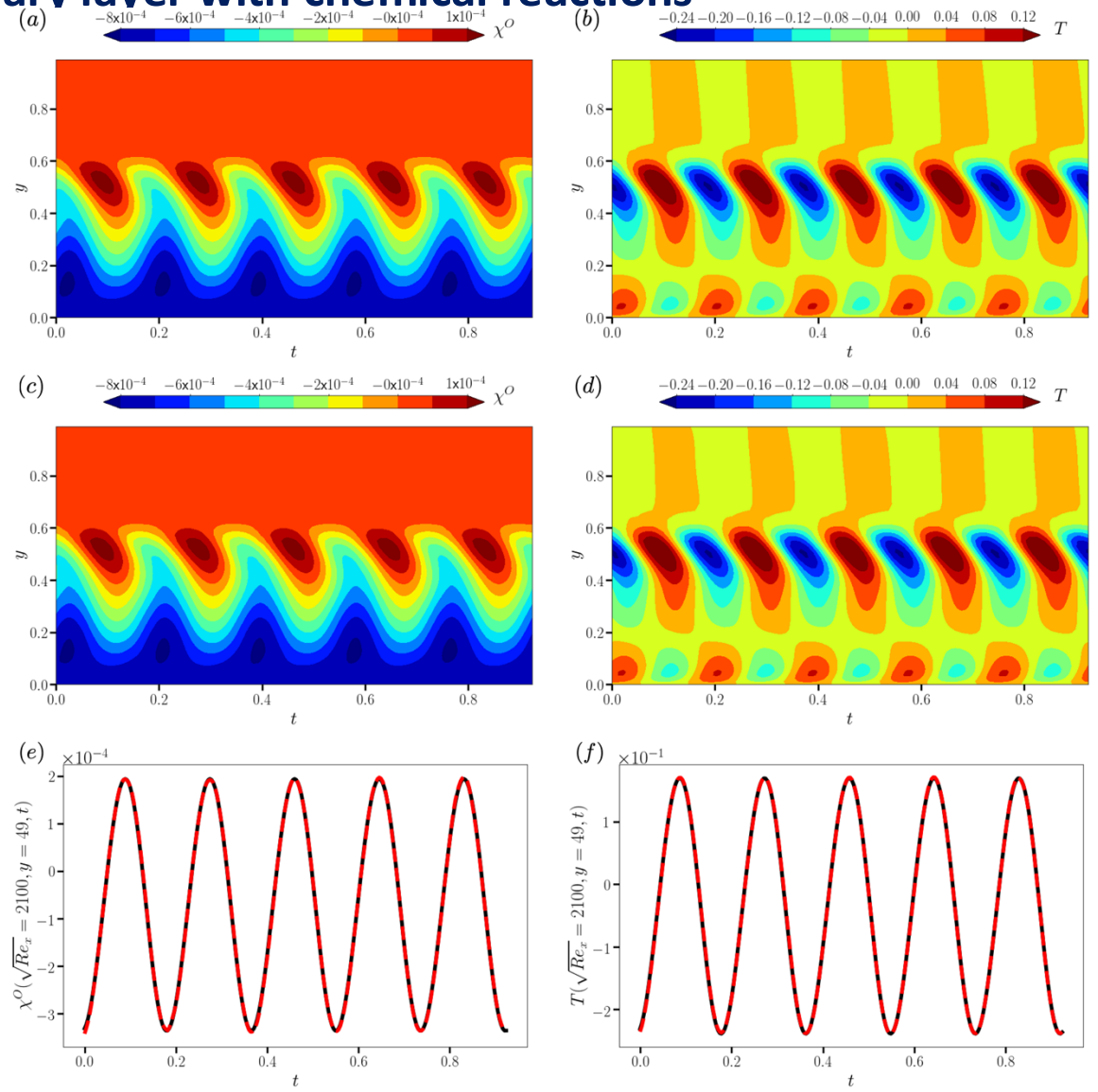


Results

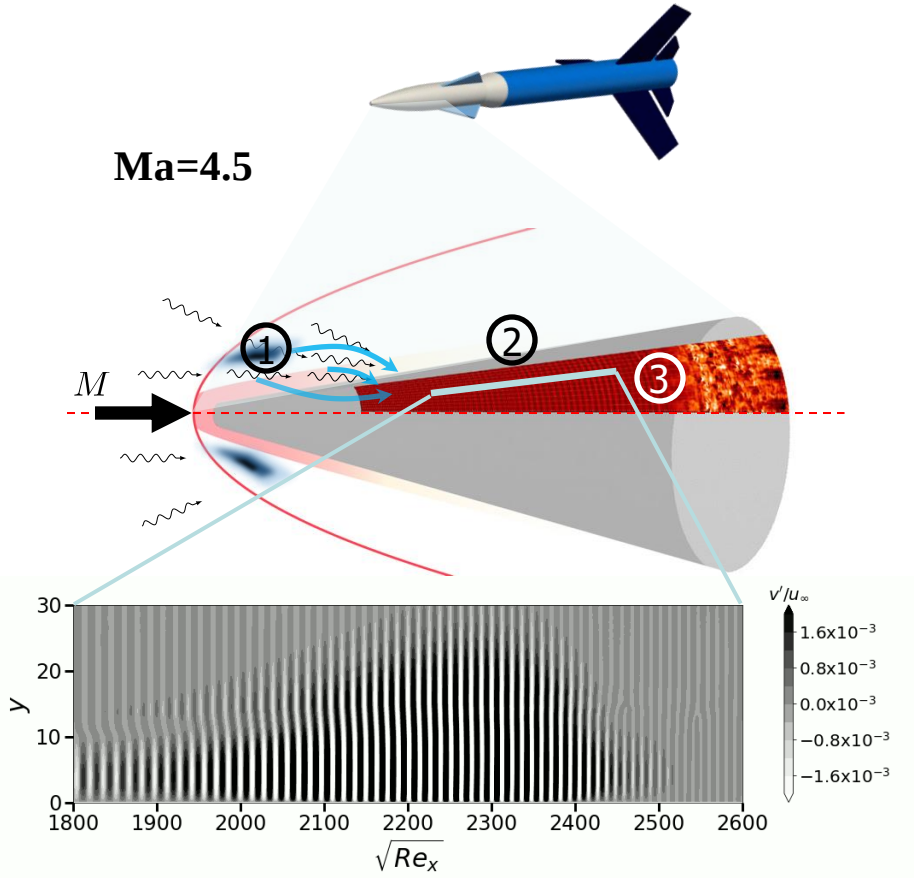
- **Forward problem to Ma=10 boundary layer with chemical reactions**
  - **Prediction of testing data**

Input: velocity perturbation at center of wall forcing





Transition in hypersonic boundary layer



- **1) Shock & gas dissociation**
  - At high Mach numbers, the gas dissociates downstream of the shock with chemical reaction rates commensurate with the flow

- **2) Amplification of instability waves**
  - Instability waves interact with the boundary layer on the flight vehicle and can amplify exponentially

- **3) Transition to turbulence**
  - Once finite amplitude perturbations form, nonlinear interaction spur the generation of harmonics and ultimately break down to turbulence

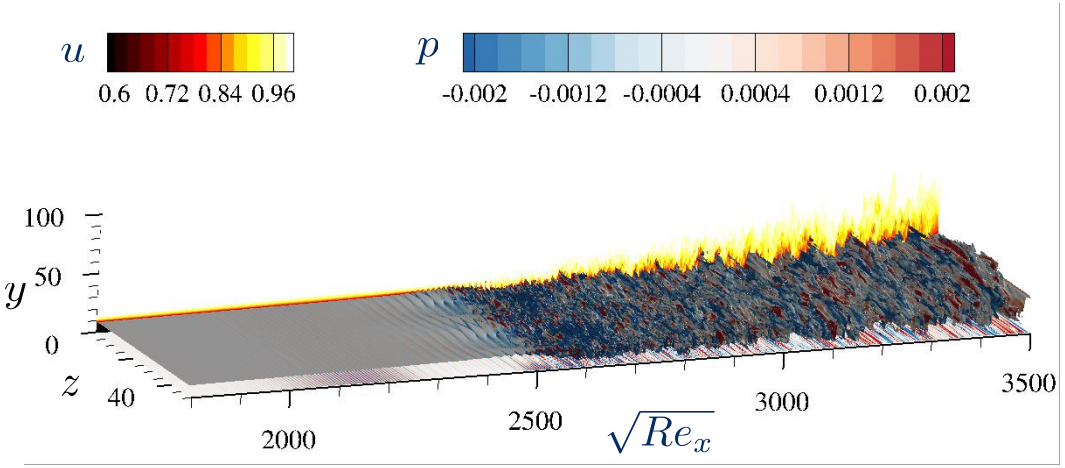
Transition prediction in high-speed boundary layer

- **Fast estimation of transition**
  - Fastly estimate downstream mean skin friction from upstream velocity disturbance

- **Problem formulation**
  - Solving forward problem

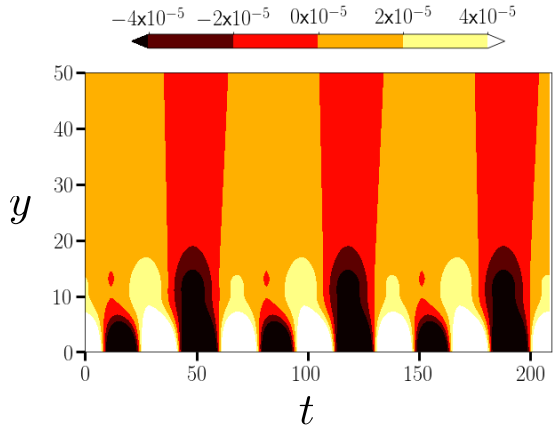
$$q_L \rightarrow C_f\left(\sqrt{Re_x}\right)(q)$$

- **Challenges**
  - Full DNS is expensive
  - Inflow to skin friction mapping is governed by NS operator difficult to find an explicit expression



upstream flow disturbance

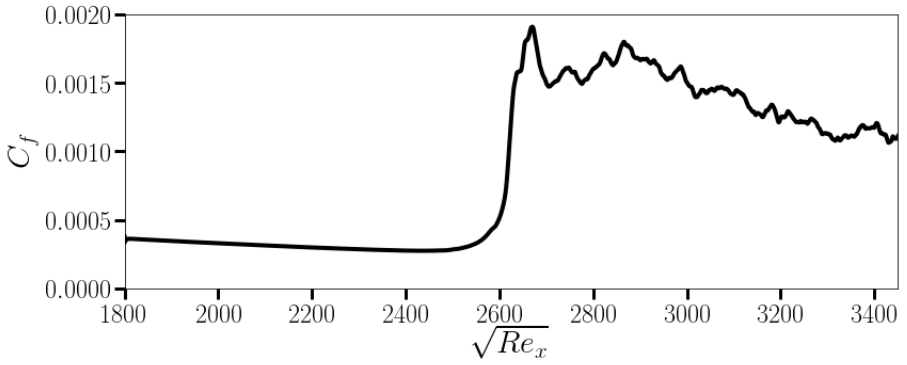
$$q_L = [\rho_L, u_L, v_L, w_L, T_L]$$



estimate

Downstream skin friction

$$C_f\left(\sqrt{Re_x}\right)(q)$$



Simulation setup

- **DNS of flat plate boundary layer**
  - **Compressible N-S equation**

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{u}) = 0$$

$$\frac{\partial \rho \boldsymbol{u}}{\partial t} + \nabla \cdot (\rho \boldsymbol{u}\boldsymbol{u} + p\boldsymbol{I} - \boldsymbol{\tau}) = 0$$

$$\frac{\partial E}{\partial t} + \nabla \cdot (\boldsymbol{u}[E + p] + \boldsymbol{\theta} - \boldsymbol{u} \cdot \boldsymbol{\tau}) = 0$$

- **Inflow instability waves**
  - **total energy** $E_0 = 2 \times 10^{-5}$
  - **2D mode** $(F, \beta = 0), E_{2D} = 0.57 \times E_0$
  - **3D mode** $(F/2, \beta = \beta_0), E_{2D} = 0.43 \times E_0$

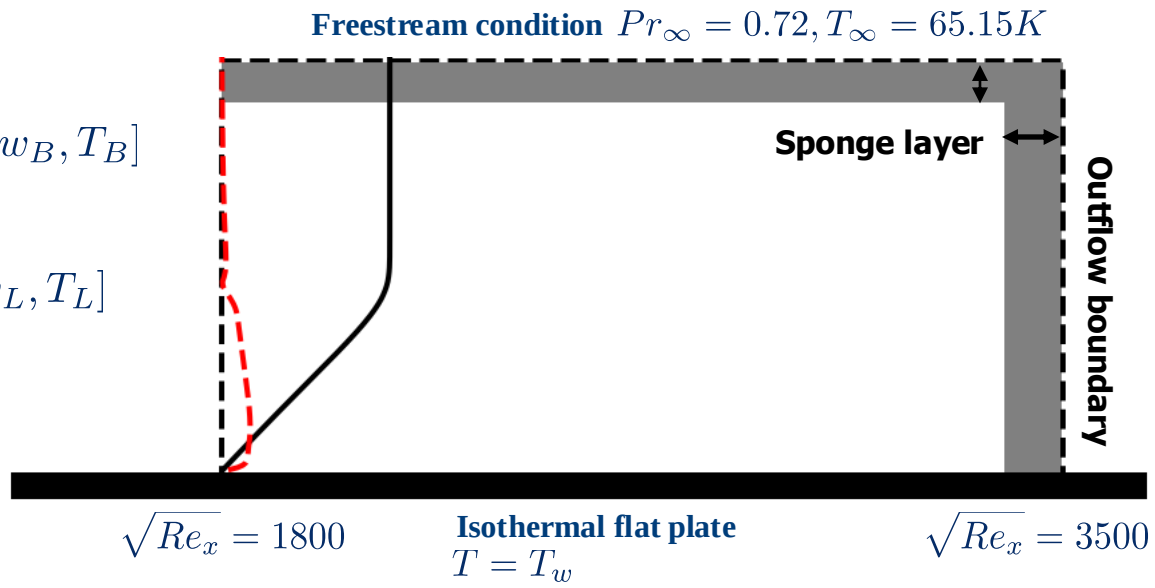
- **Solver: mixed 6th order central difference & 5th order WENO scheme**
- **Boundary conditions**

Blasius base flow $\boldsymbol{Q_B} = [\rho_B, u_B, v_B, w_B, T_B]$

+

Instability waves $\boldsymbol{q_L} = [\rho_L, u_L, v_L, w_L, T_L]$

$$\boldsymbol{q}(x_0, y, z, t) = \boldsymbol{Q_B} + \boldsymbol{q_L}$$

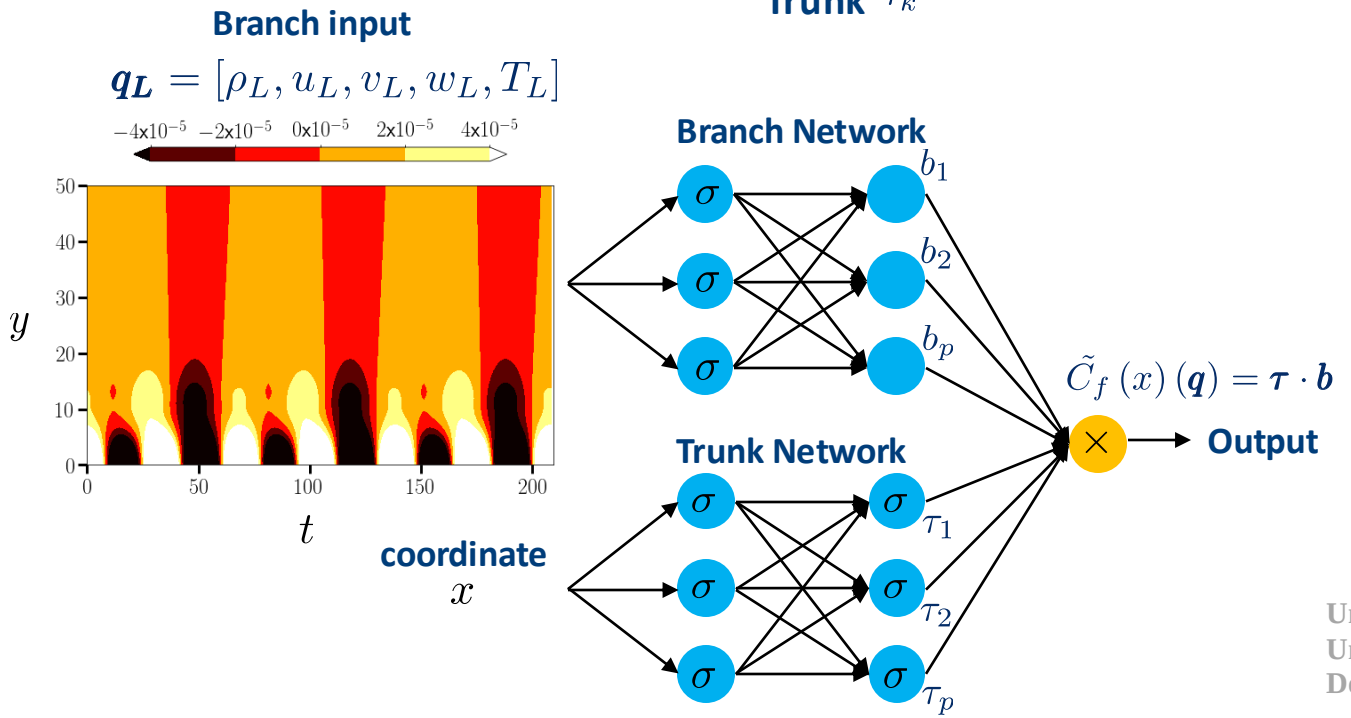


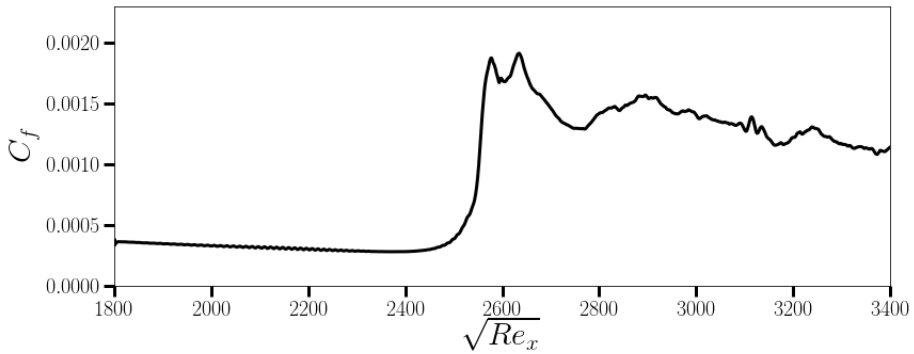
Deep Operator Neural Networks

- **DeepONets**
  - inflow $\boldsymbol{q}_L(y,t)$ skin friction $\tilde{C}_f(x)(\boldsymbol{q}_L)$
  - Universal approximation theorem for operators
    - Any continuous operator $G(x)(\boldsymbol{q}_L)$ can be approximated by $\tilde{C}_f(x)(\boldsymbol{q}_L) = \sum_k \underbrace{\left(\sum_i c_i^k \sigma(\xi_{ij}^k q_{L,j} + \theta_i^k)\right)}_{\text{Trunk } \tau_k} \underbrace{\sigma(w_k x + \zeta_k)}_{\text{Branch } b_k}$

- **Loss function**
  - Mean squared error

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}(\tilde{G}(q_{L_i}, \mathbf{w}) - C_{f_i})^2$$



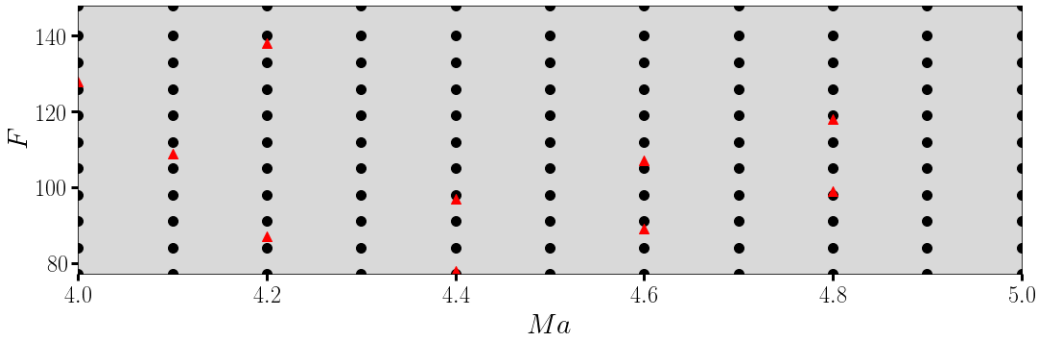


Universal function approximation theorem (Cybenko, 1989)
Universal operator approximation theorem (Chen & Chen, 1993)
Deep Operator Networks (Lu et al, 2020)

Transition prediction

- **Training & validation dataset**
  - **# training data: 121**
  - **Black dots: location of training data**

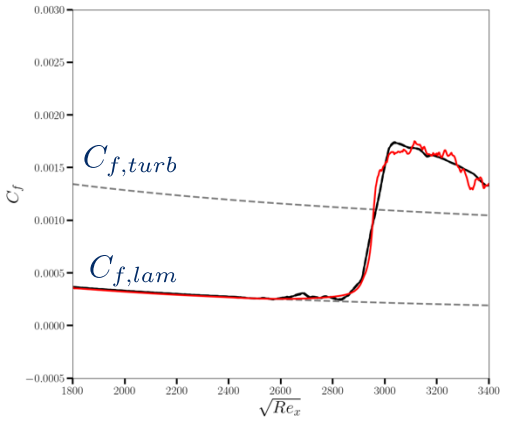
    $\Delta Ma = 0.1, \Delta F = 7$
  - **Red triangle: location of validation data**



- **DeepONets prediction**
  - **Validation case**

    $Ma = 4.1$

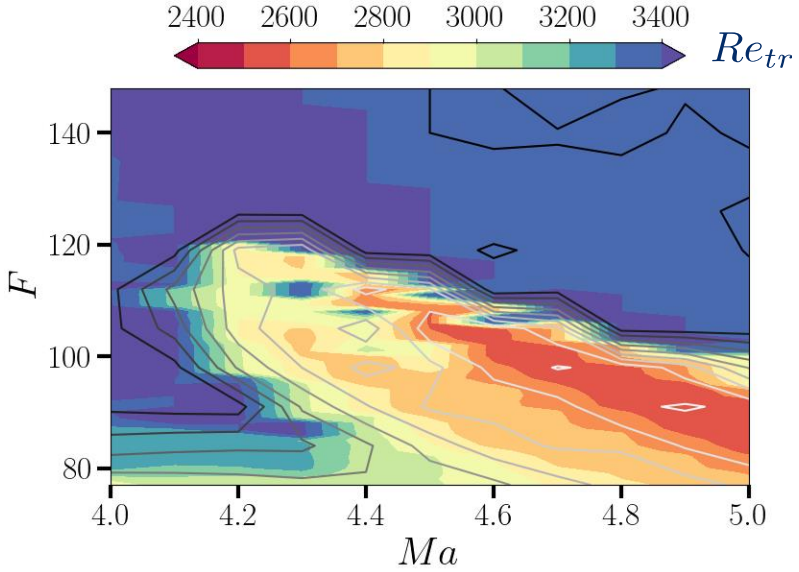
    $F = 109$



- **Transition Reynolds number**

$$C_{f,tr} = \frac{1}{2}(C_{f,lam} + C_{f,turb})$$

$$Re_{tr} = \arg\min_{\sqrt{Re_x}} \mid C_f - C_{f,tr} \mid$$

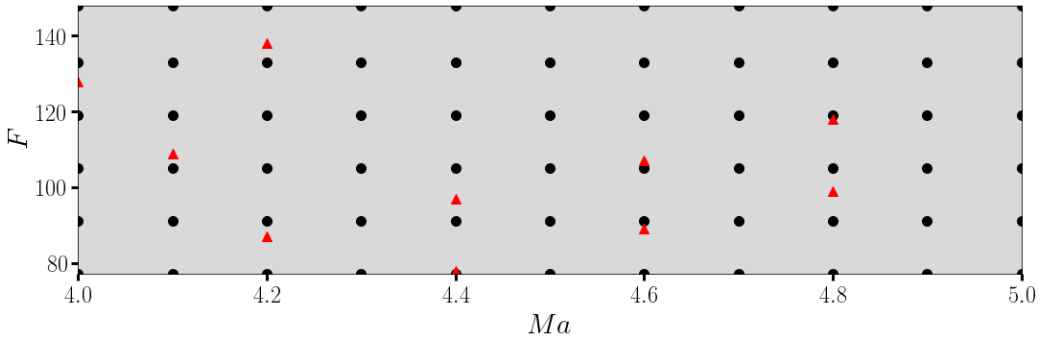


Color contour: DeepONets prediction
Lines contour: DNS

Transition prediction

- **Training & validation dataset**
  - **# training data: 66**
  - **Black dots: location of training data**

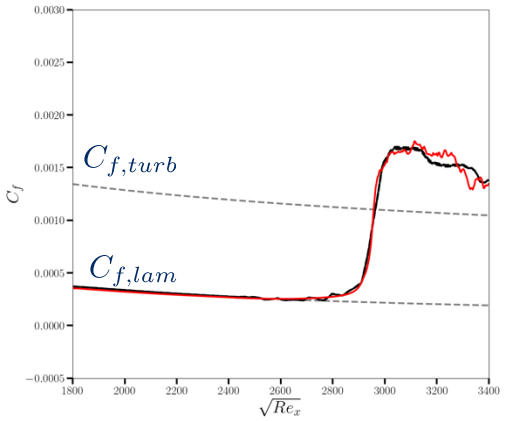
    $\Delta Ma = 0.1, \Delta F = 14$
  - **Red triangle: location of validation data**



- **DeepONets prediction**
  - **Validation case**

    $Ma = 4.1$

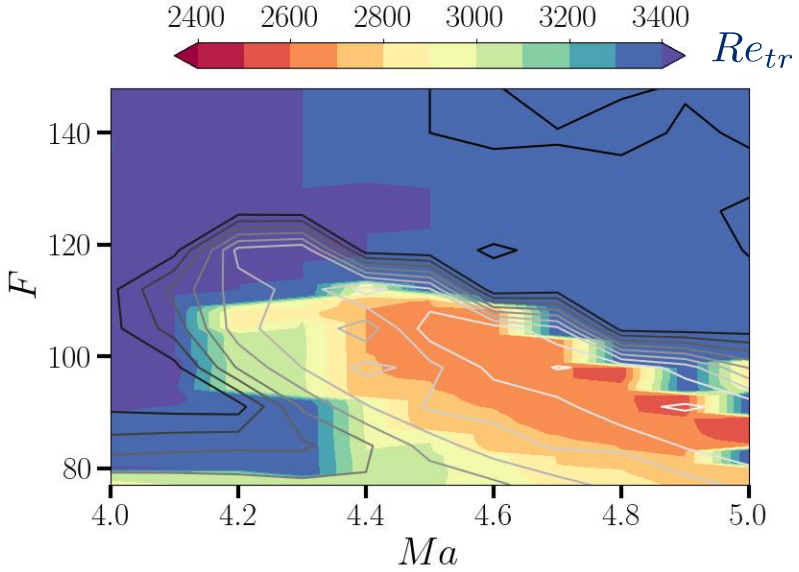
    $F = 109$



- **Transition Reynolds number**

$$C_{f,tr} = \frac{1}{2}(C_{f,lam} + C_{f,turb})$$

$$Re_{tr} = \arg\min_{\sqrt{Re_x}} \mid C_f - C_{f,tr} \mid$$

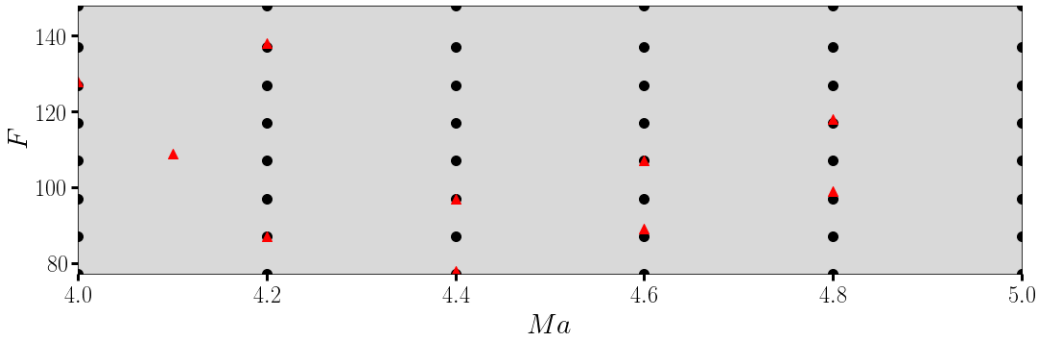


Color contour: DeepONets prediction
Lines contour: DNS

Transition prediction

- **Training & validation dataset**
  - **# training data: 48**
  - **Black dots: location of training data**

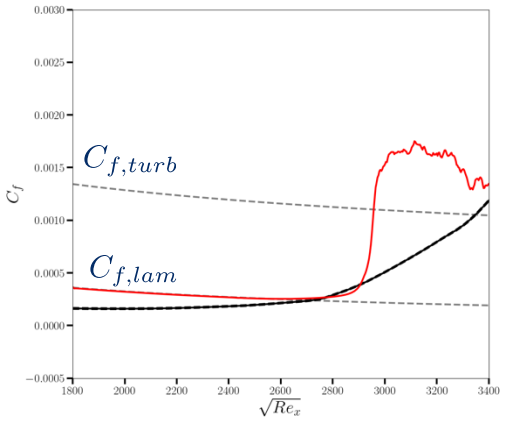
    $\Delta Ma = 0.2, \Delta F = 10$
  - **Red triangle: location of validation data**



- **DeepONets prediction**
  - **Validation case**

    $Ma = 4.1$

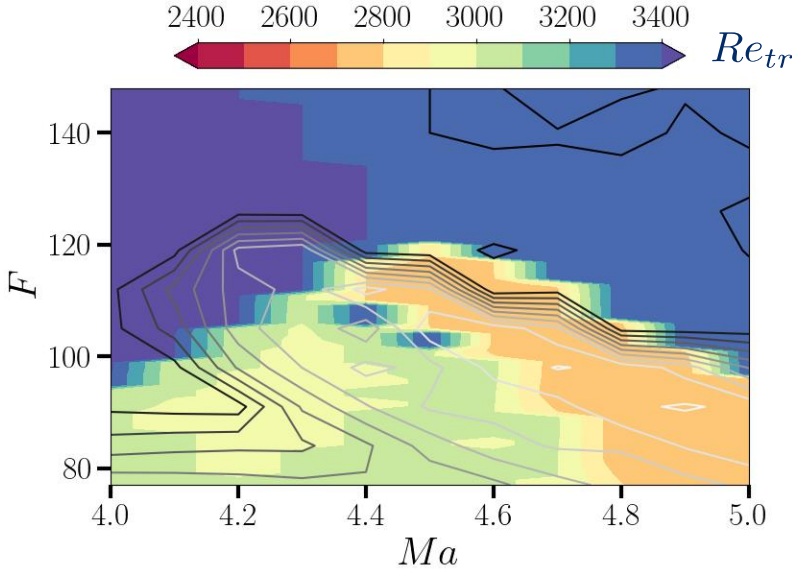
    $F = 109$



- **Transition Reynolds number**

$$C_{f,tr} = \frac{1}{2}(C_{f,lam} + C_{f,turb})$$

$$Re_{tr} = \arg\min_{\sqrt{Re_x}} \mid C_f - C_{f,tr} \mid$$

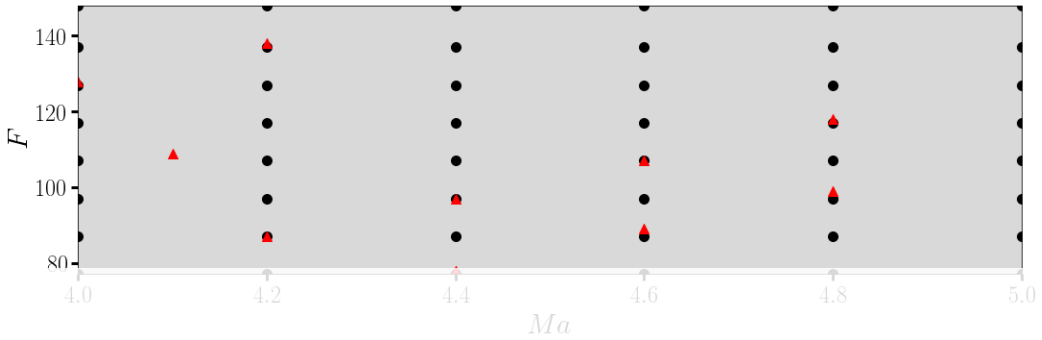


Color contour: DeepONets prediction
Lines contour: DNS

Transition prediction

- **Training & validation dataset**
  - **# training data: 48**
  - **Black dots: location of training data**

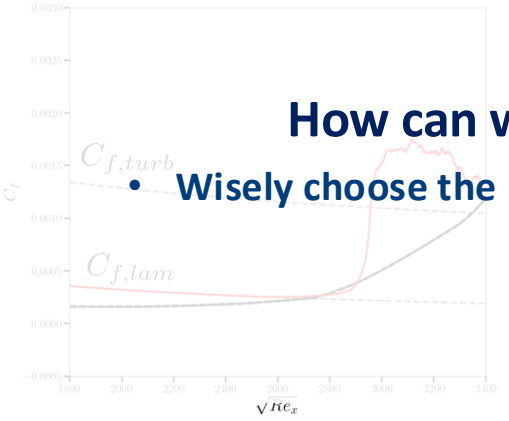
    $\Delta Ma = 0.2, \Delta F = 10$
  - **Red triangle: location of validation data**



- **DeepONets prediction**
  - **Validation case**

    $Ma = 4.1$

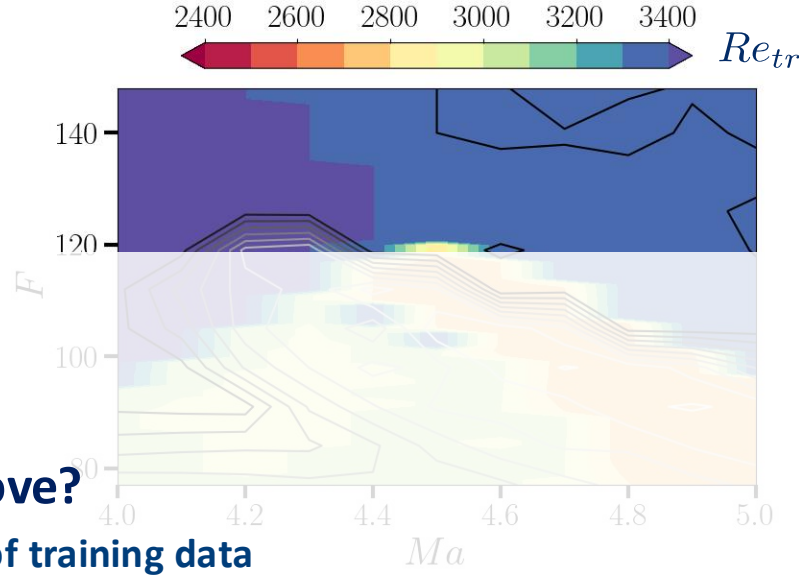
    $F = 109$



- **Transition Reynolds number**

$$C_{f,tr} = \frac{1}{2}(C_{f,lam} + C_{f,turb})$$

$$Re_{tr} = \arg\min_{\sqrt{Re_x}} \mid C_f - C_{f,tr} \mid$$



Color contour: DeepONets prediction
Lines contour: DNS

How can we improve?

- **Wisely choose the locations of training data**

Bayesian Deep Operator Neural Networks

- **Bayesian DeepONets**
  - inflow $\boldsymbol{q}_L(y,t)$ skin friction $\tilde{C}_f(x)(\boldsymbol{q}_L)$
  - Universal approximation theorem for operators
    - Any continuous operator $G(x)(\boldsymbol{q}_L)$ can be approximated by $\tilde{C}_f(x)(\boldsymbol{q}_L) = \sum_k \underbrace{\left(\sum_i c_i^k \sigma(\xi_{ij}^k q_{L,j} + \theta_i^k)\right)}_{\text{Trunk } \tau_k} \underbrace{\sigma(w_k x + \zeta_k)}_{\text{Branch } b_k}$

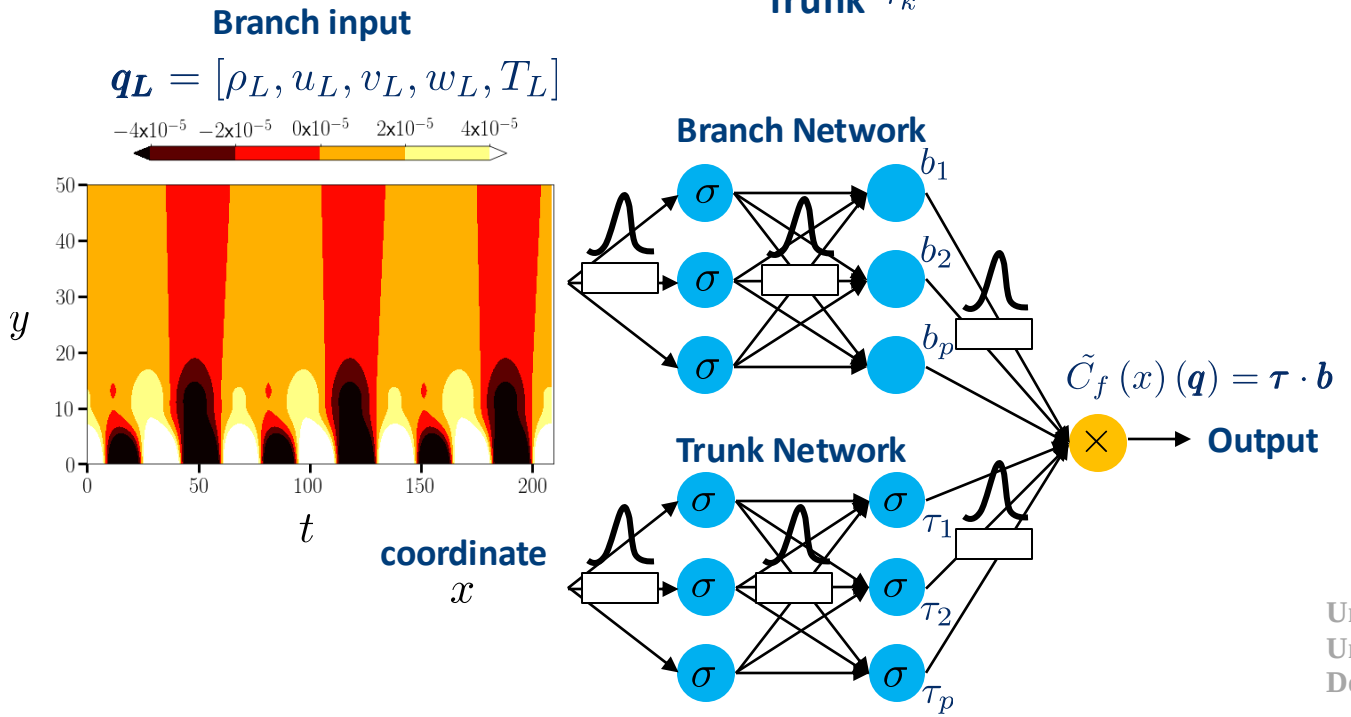
- **Loss function**
  - KL divergence between posterior $P(\mathbf{w}|\mathbf{D})$ and assumption $r(\mathbf{w}|\theta)$

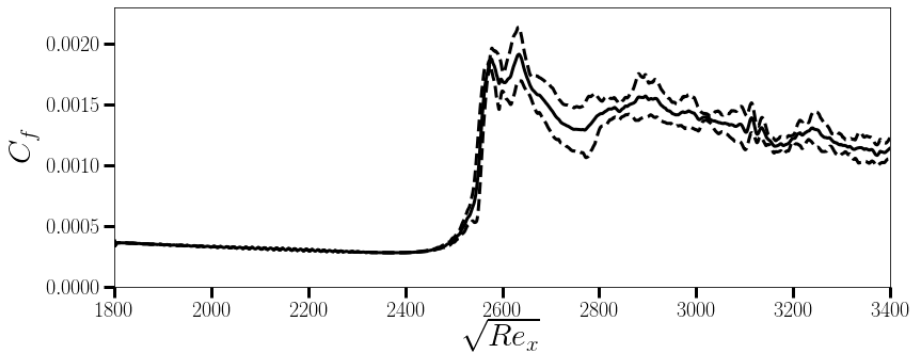
$$\mathcal{L} = \min_\theta \underbrace{\mathrm{KL}(r(\mathbf{w}|\theta)||P(\mathbf{w}))}_{\text{Analytically evaluated for Gaussian distributions}} - \mathbb{E}_{r(\mathbf{w}|\theta)}[\log P(\mathbf{D}|\mathbf{w})]$$

- Regularizes

$$P(\mathbf{D}|\mathbf{w}) = \prod_{i=1}^{N} \mathcal{N}(\tilde{G}(q_L, \mathbf{w}), \sigma)$$

$$logP(\mathbf{D}|\mathbf{w}) = \sum_{i=1}^{N} \frac{1}{2}\log\sigma^2 + \underbrace{\frac{(\tilde{G}(q_{L_i}, \mathbf{w}) - C_{f_i})^2}{\sigma^2}}_{\text{M.S.E}}$$

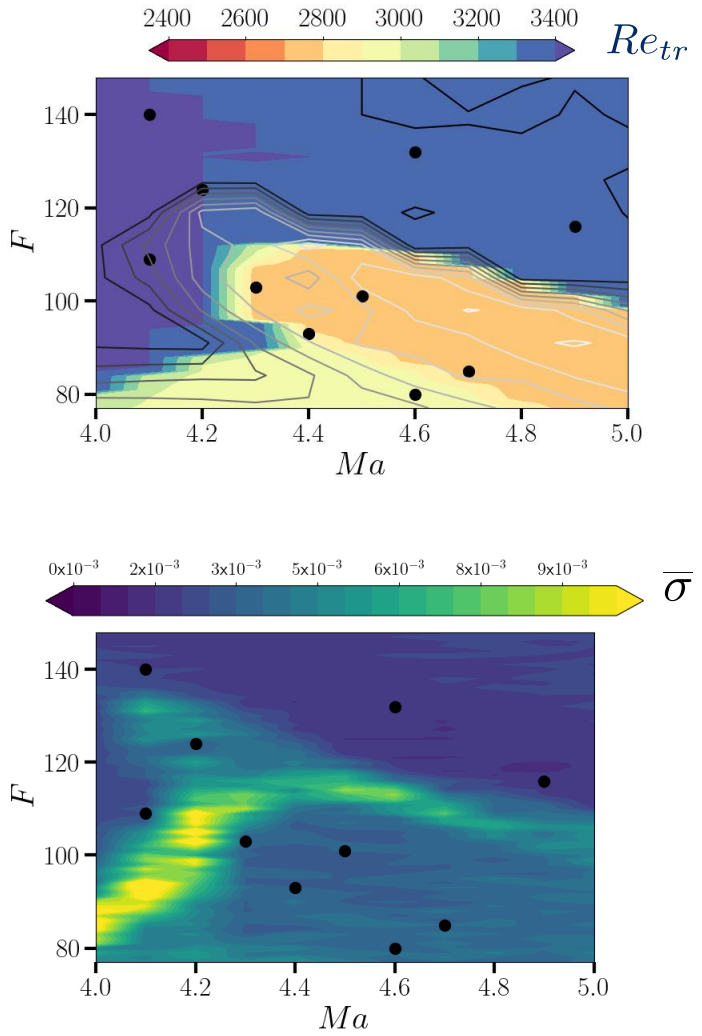




Universal function approximation theorem (Cybenko, 1989)
Universal operator approximation theorem (Chen & Chen, 1993)
Deep Operator Networks (Lu et al, 2020)

Method: Active learning

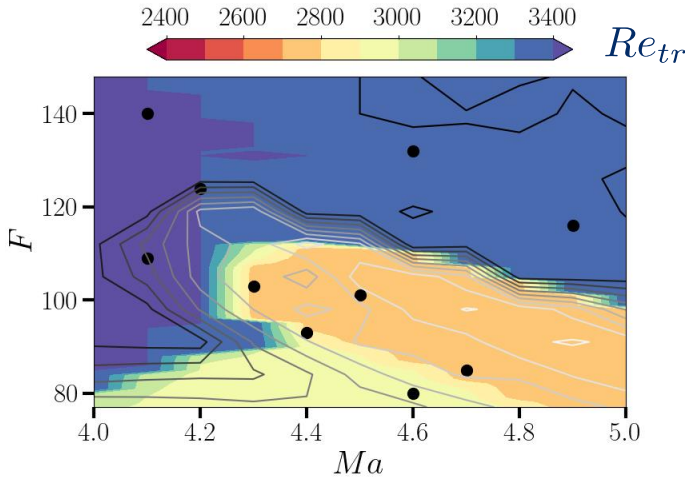
- **Training point selection strategy, Pseudo code**
  - Initial training points at n_0 locations
  - Make predictions $\widetilde{Cf}(c,x) \sim \mathcal{N}(\mu, \sigma^2)$
  - Compute averaged uncertainty $\overline{\sigma^2}$ (normalized)
  - While $\overline{\sigma^2} \geq \sigma_0^2$ do
    - Generate new training points following a distribution $\gamma(c) \propto \sigma(c)$
    - Retrain for prediction $\widetilde{Cf}_i(c,x) \sim \mathcal{N}(\mu, \sigma^2)$
  - End while
  - Return a solution: accurate prediction of transition in domain of interest

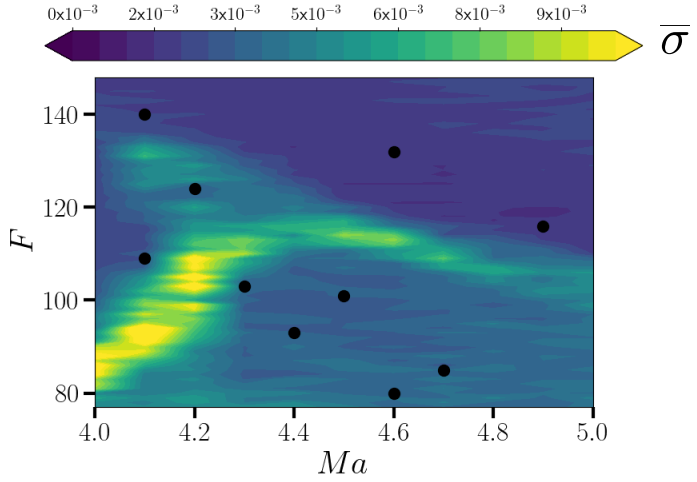


Transition prediction

- **Results at iteration 0**
  - **Black dots: location of training data**

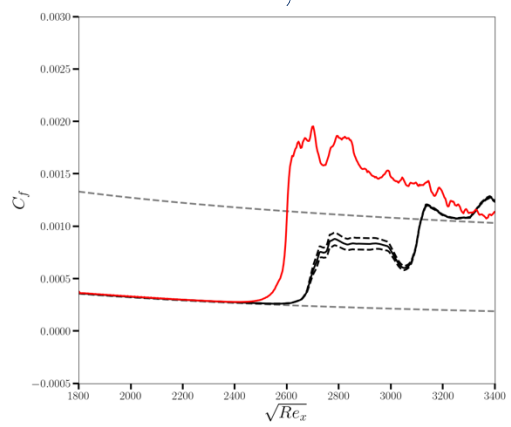
Cumulative # training data: 10



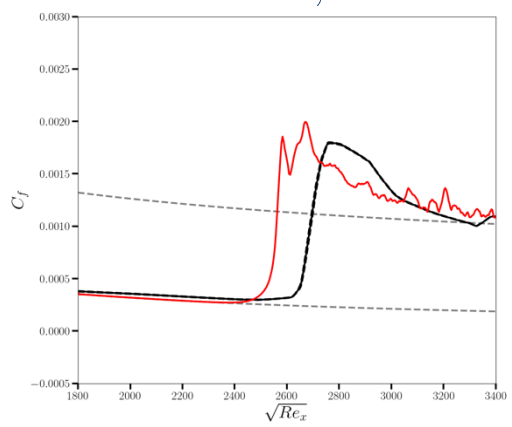


- **DeepONets prediction**
  - **Validation cases**

$Ma = 4.6, F = 107$



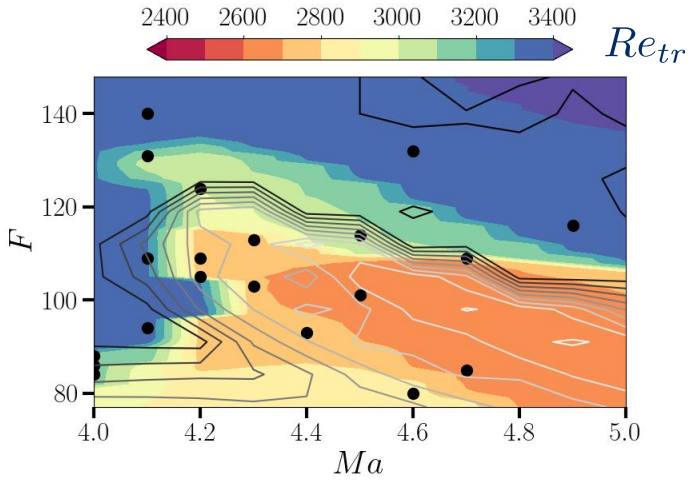
$Ma = 4.8, F = 99$

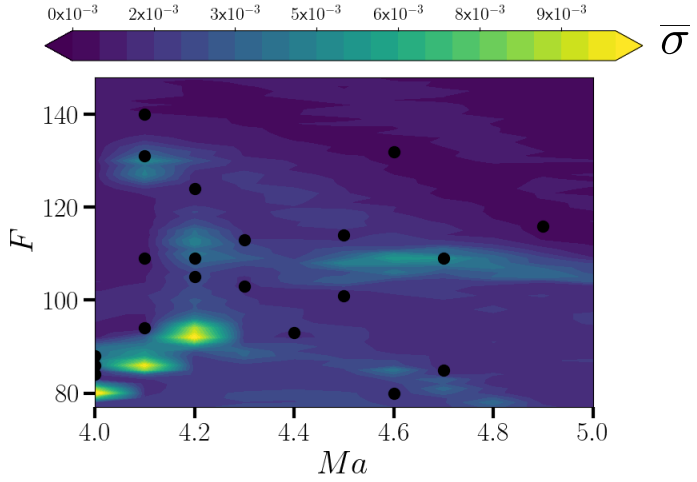


Transition prediction

- **Results at iteration 1**
  - **Black dots: location of training data**

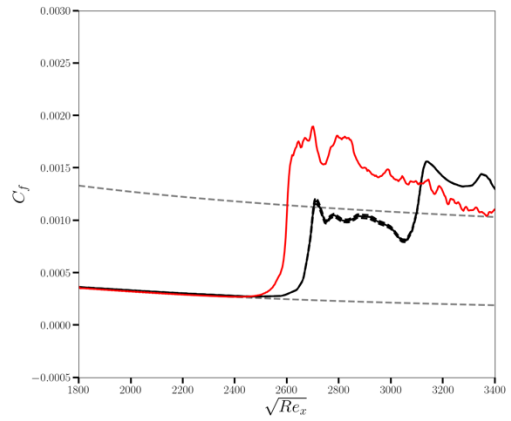
Cumulative # training data: 12



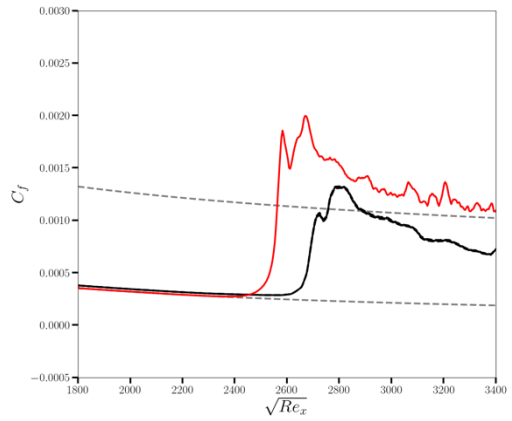


- **DeepONets prediction**
  - **Validation cases**

$Ma = 4.6, F = 107$



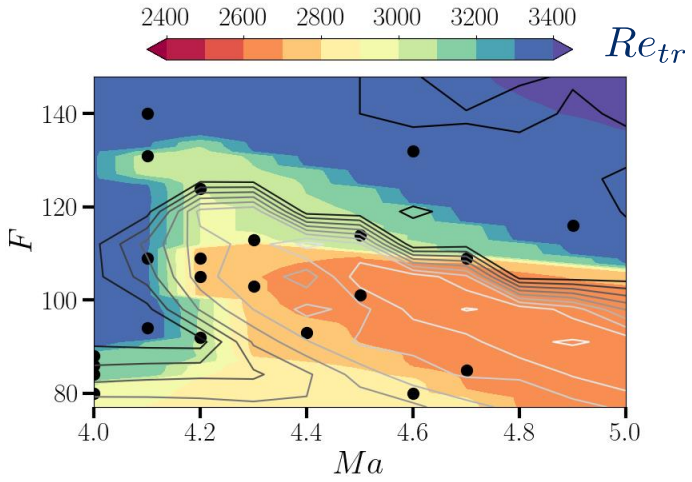
$Ma = 4.8, F = 99$

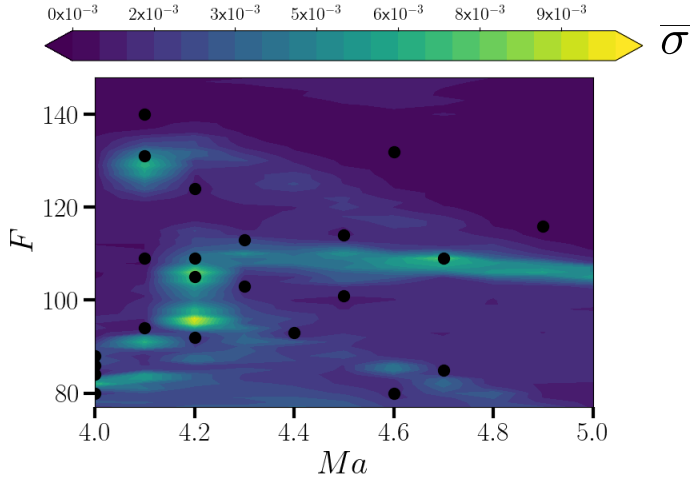


Transition prediction

- **Results at iteration 2**
  - **Black dots: location of training data**

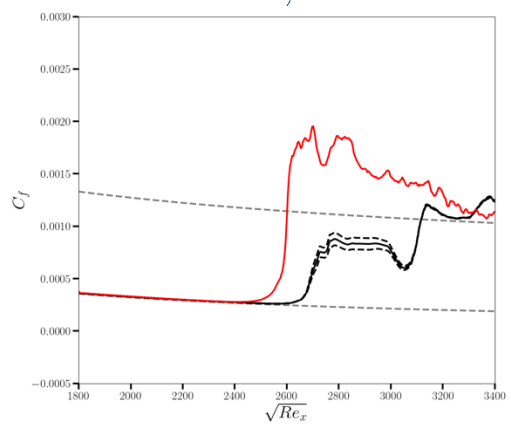
Cumulative # training data: 15



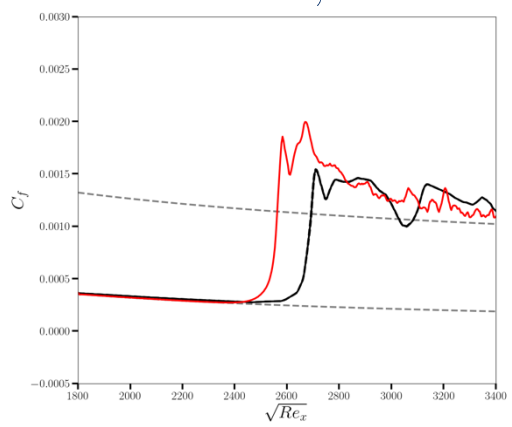


- **DeepONets prediction**
  - **Validation cases**

$Ma = 4.6, F = 107$



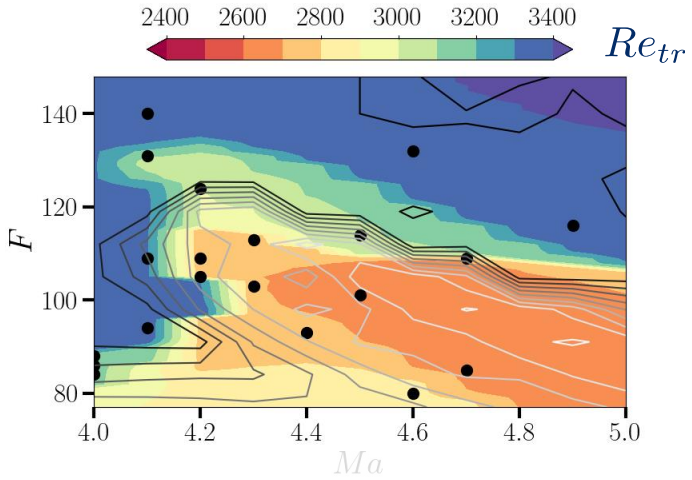
$Ma = 4.8, F = 99$

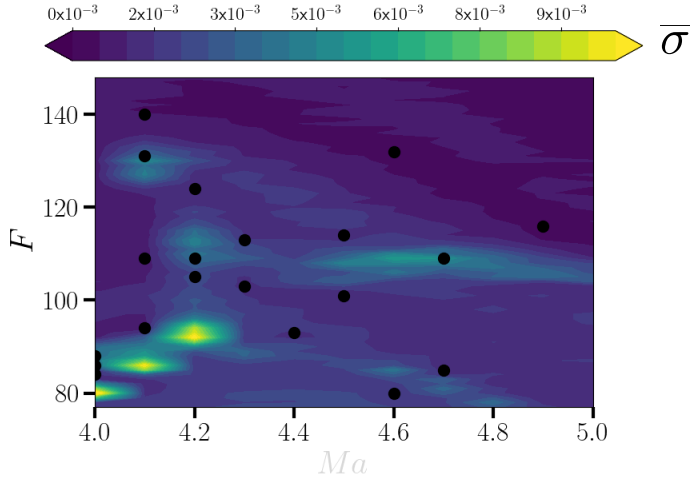


Transition prediction

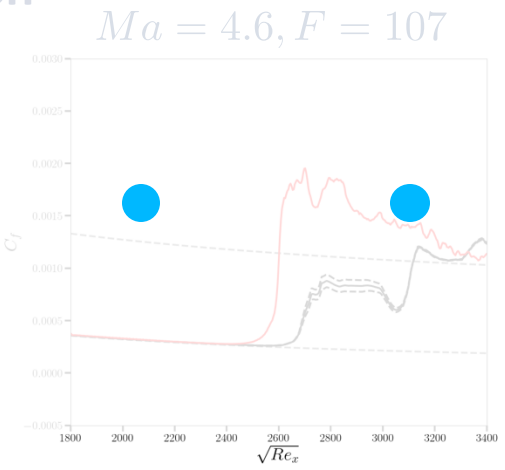
- **Results at iteration 2**
  - **Black dots: location of training data**

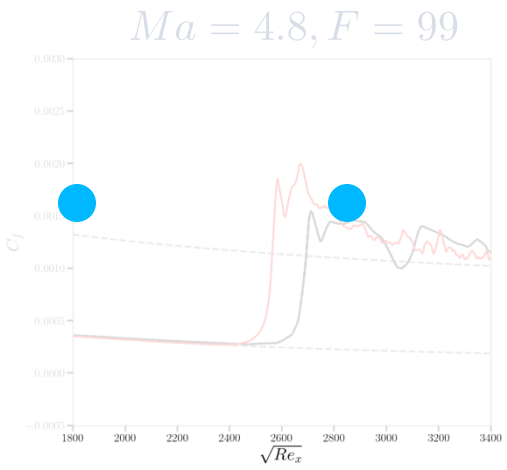
Cumulative # training data: 15





- **DeepONets prediction**
  - **Validation cases**

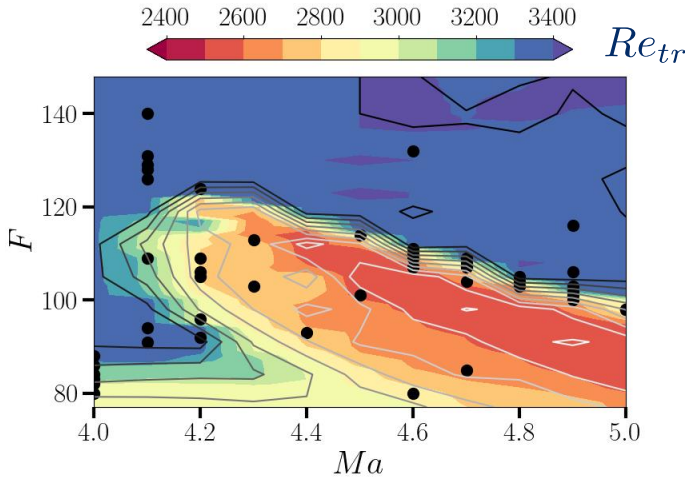


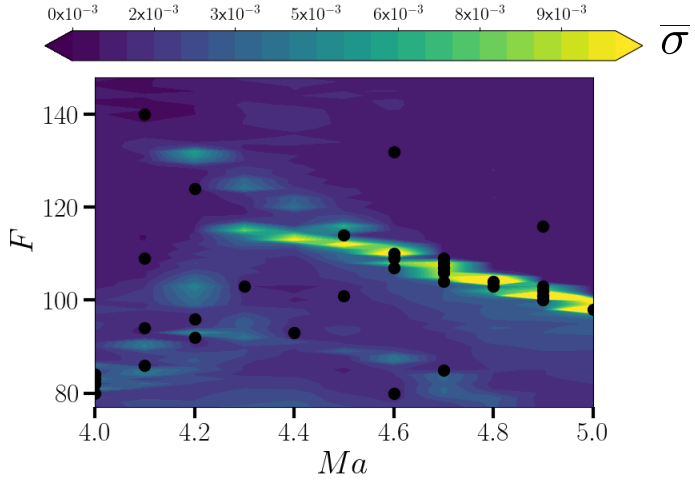


Transition prediction

- **Results at iteration 9**
  - **Black dots: location of training data**

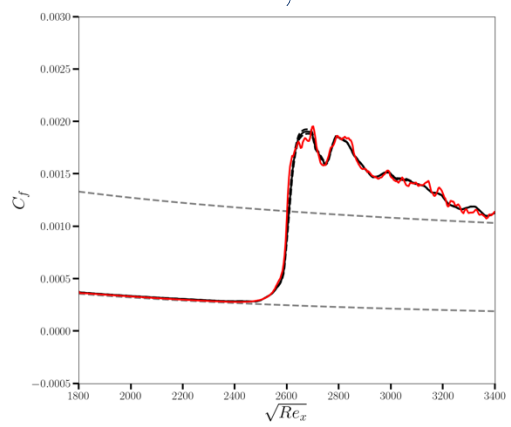
Cumulative # training data: 35



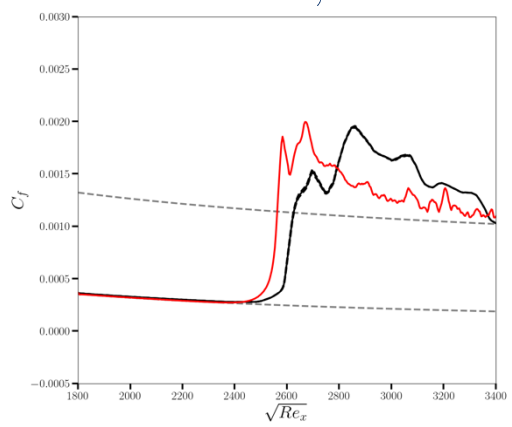


- **DeepONets prediction**
  - **Validation cases**

$Ma = 4.6, F = 107$

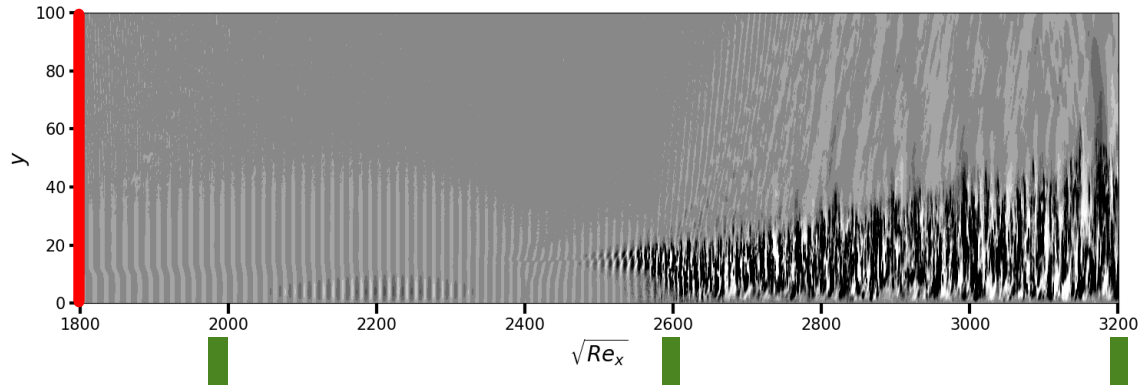


$Ma = 4.8, F = 99$



Inverse problem

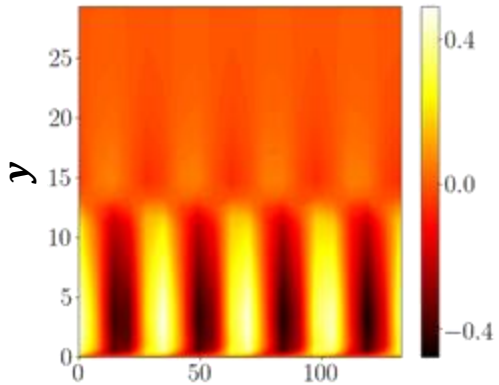
- **Estimation of oncoming instability waves**
  - Discover unknown upstream velocity disturbance from downstream measurements of wall-pressure
- **Problem formulation**
  - Oncoming instability waves
    - control vector c
  - Downstream pressure data
    - measurement m
  - Inverse problem
    - Solve $m \rightarrow c$
- **Challenges**
  - Discrete sensors located at far downstream
  - No explicit approach to solve the problem

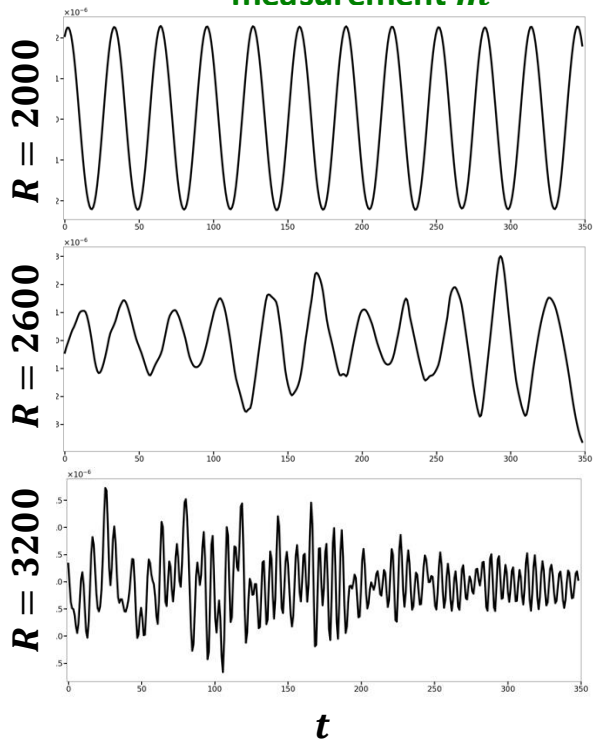


Unknown upstream velocity disturbance
Control vector c

discover

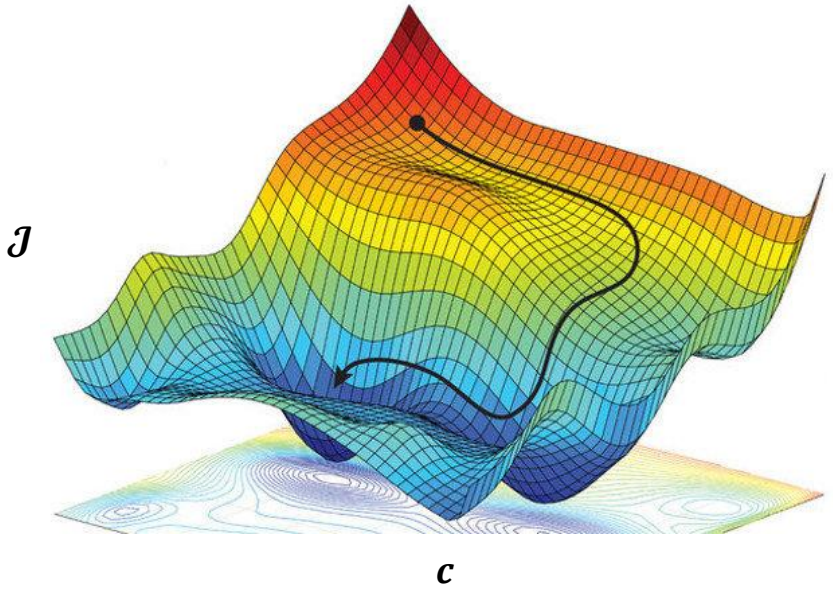
Downstream wall-pressure data measurement m





Estimation of oncoming instability waves

- **Problem formulation**
  - Inverse problem: solve $m \rightarrow \mathbf{c}$
  - Formulate it as an optimization problem
    - model observation: $\mathcal{M}(c)$
    - objective function: $\mathcal{J}(c) = \frac{1}{2}\|m - \mathcal{M}(c)\|^2$
    - Objective: estimate c that minimizes $\mathcal{J}$

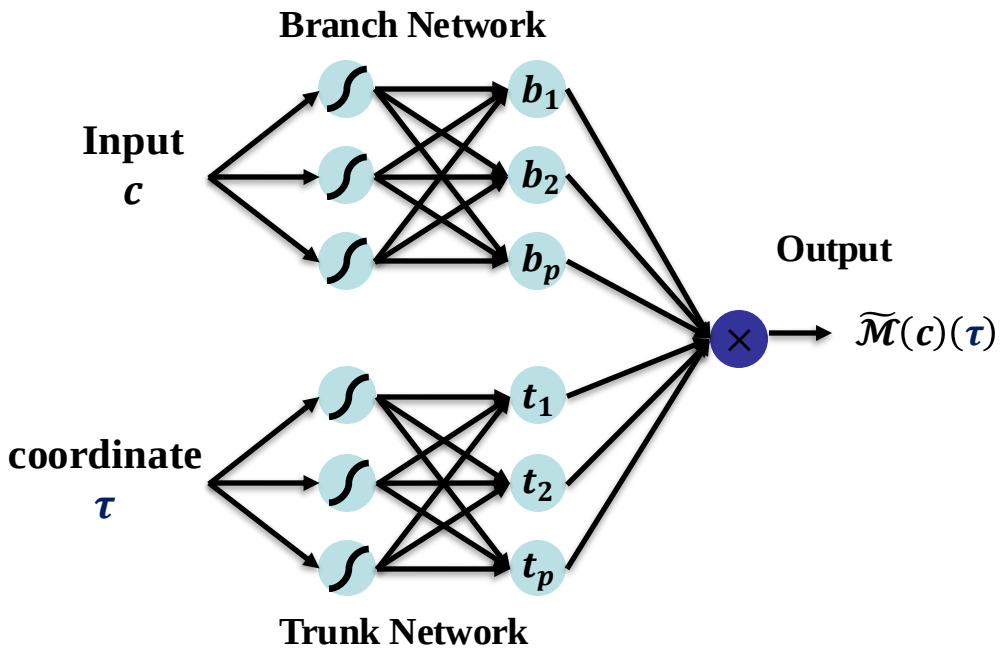


- **Methods to minimize $\mathcal{J}$**
  - Ensemble variational method (Butcha & Zaki, JFM 2021,2022)
    - Gradient based method $c_{i+1} = c_i - \alpha \nabla \mathcal{J}$
    - Pros: does no require adjoint model; non-intrusive to forward solver; can be used with long time horizon
    - Cons: requires large number of DNS ; do not generate a surrogate model at end
  - Adjoint variational method (Wang et al, JCP 2019)
    - minimize $\mathcal{J}$ with Navier-Stokes constraint $q^{n+1} = G^n q^n$
    - Pros: high efficiency, high accuracy
    - Cons: require adjoint model; requires significant storage; forward-adjoint duality not guaranteed for long horizon; do not generate a surrogate model at end

Method

- **Formulated optimization problem**
  - loss function: $\mathcal{J}(c) = \frac{1}{2}\|m - \mathcal{M}(c)\|^2$
  - Proposed idea: have a fast estimator of $\mathcal{M}(c)$

- **Deep Operator Neural networks (DeepONets)**
  - Input function: c
  - Output coordinate: τ
  - Branch net output: $b_k = \sum_{i=1}^{n} c_j^k \sigma\left(\sum_{j=1}^{m} \xi_{ij}^k c(j) + \theta_j^k\right)$
  - Trunk net output: $t_k = \sigma(w_k \cdot \tau + \zeta_k)$
  - Output function: $\widetilde{\mathcal{M}}(c)(\tau) = \sum_{k=1}^{p} b_k t_k$

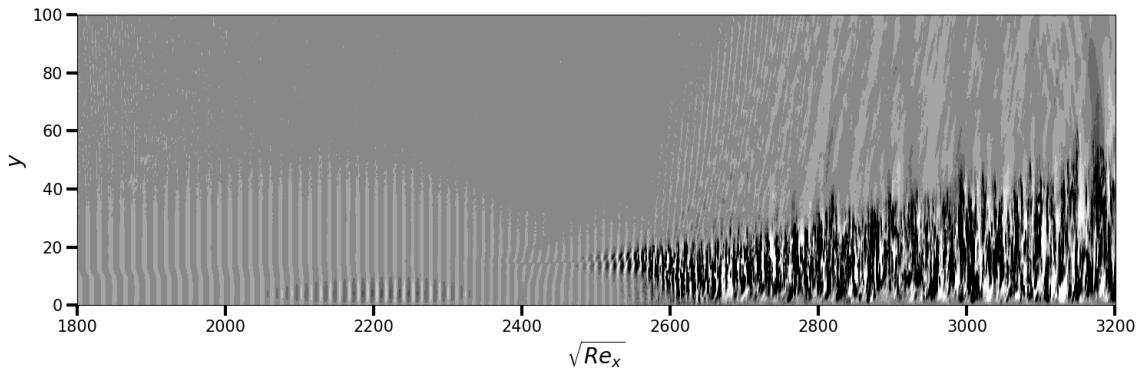


- **Universal approximation theorem**
  - DeepONet prediction $\widetilde{\mathcal{M}}(c)(t)$ approximates the truth $\mathcal{M}(c)(t)$
  - DeepONet modeled loss function $\widetilde{\mathcal{J}}(c) = \frac{1}{2}\left\|m - \widetilde{\mathcal{M}}(c)\right\|^2$ approximates true loss function $\mathcal{J}(c) = \frac{1}{2}\|m - \mathcal{M}(c)\|^2$
  - $\widetilde{\mathcal{J}}(c)$ can be cheaply computed from c
  - We can switch to estimate c that minimizes $\widetilde{\mathcal{J}}(c)$, which is much easier

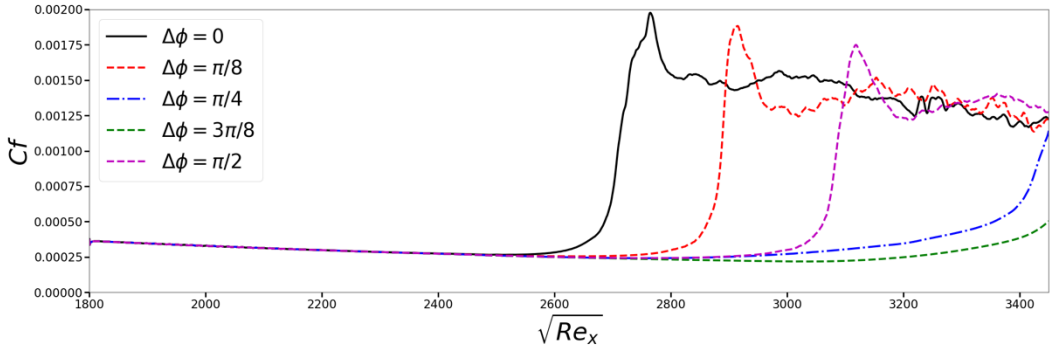
Universal function approximation theorem (Cybenko, 1989)
Universal operator approximation theorem (Chen & Chen, 1993)
Deep Operator Networks (Lu et al, 2020)

Test problem

- **Problem definition**
  - $Ma = 4.5$ high speed boundary layer
  - Inflow $\sqrt{Re_x} = 1800$
  - Inflow disturbance, total energy $E_0 = 2 \times 10^{-5}$
    - 2D mode $(F, 0)$, $E_{2D} = 0.57 \times E_0$
    - 3D mode $(F/2, \beta)$, $E_{3D} = 0.43 \times E_0$
    - Phase difference $\Delta\Phi$ between 2D & 3D mode
  - Control vector: $c = (F, \Delta\Phi)$
  - Measurements
    - 9 wall pressure probe data at $\sqrt{Re_x} = 2300$



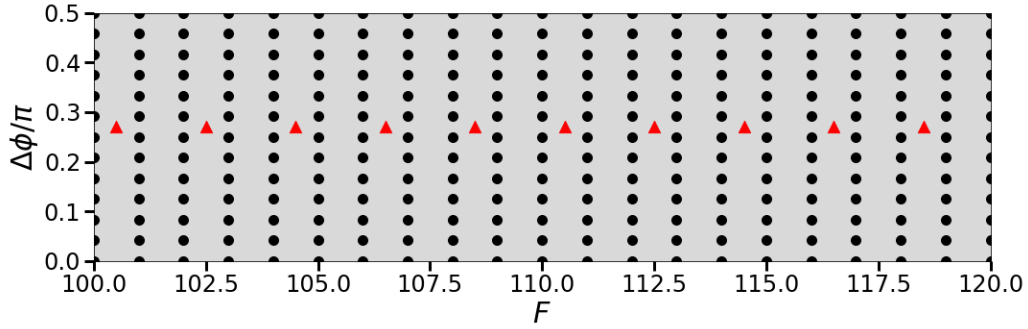
Skin friction of $F = 112$ cases



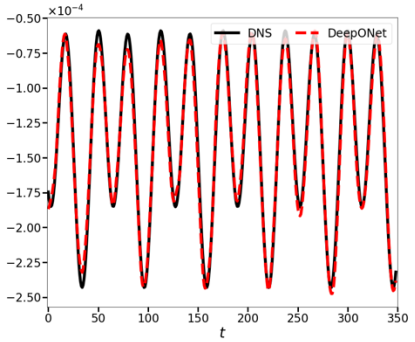
- **Transition point is sensitive to both $F, \Delta\Phi$**
  - Critical Reynolds number
  - $Re_{cr} = \underset{\sqrt{Re_x}}{\operatorname{argmin}} Cf\left(\sqrt{Re_x}\right)$

Result

- **Training dataset**
  - $\Delta F = 1, \Delta\phi = \pi/24$
  - Total # of cases: $21 \times 13 = 273$

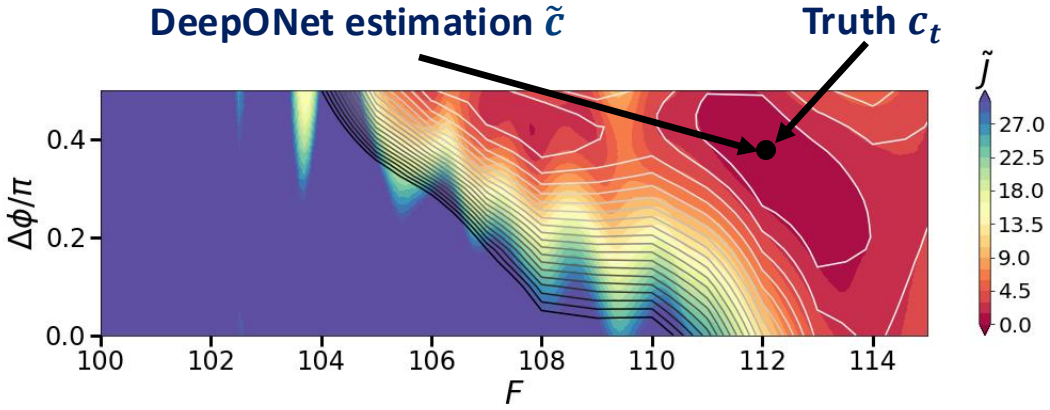


- **DeepONet predicted wall pressure**



- **Inflow estimation by DeepONet**
  - DeepONet estimated control vector
    - $\tilde{c} = \underset{c \in A}{\operatorname{argmin}} \tilde{\mathcal{J}}(c)$
  - Truth $c_t = (112, 0.38\pi)$

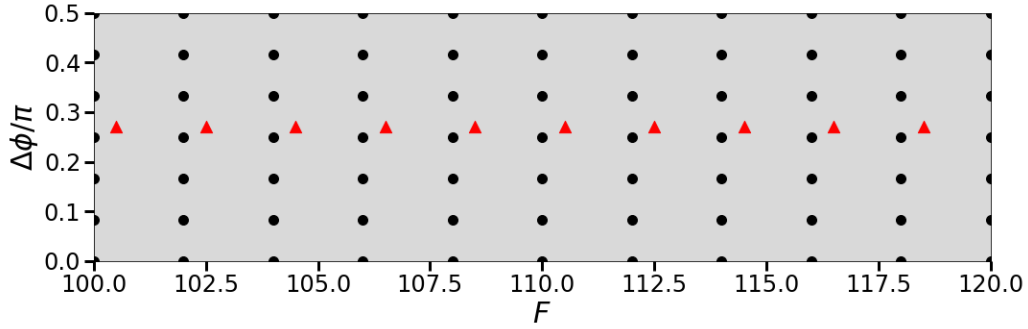
Loss function landscape
Contour: DeepONets modeled loss function $\tilde{\mathcal{J}}(c)$
Lines: True loss function $\mathcal{J}(c)$



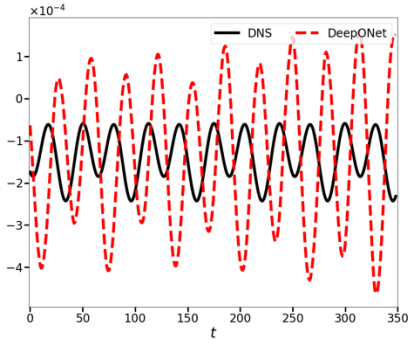
- DeepONet estimation $\tilde{c} = (112, 0.38\pi)$

Result

- **Training dataset**
  - $\Delta F = 2, \Delta\phi = \pi/12$
  - Total # of cases: $11 \times 7 = 77$

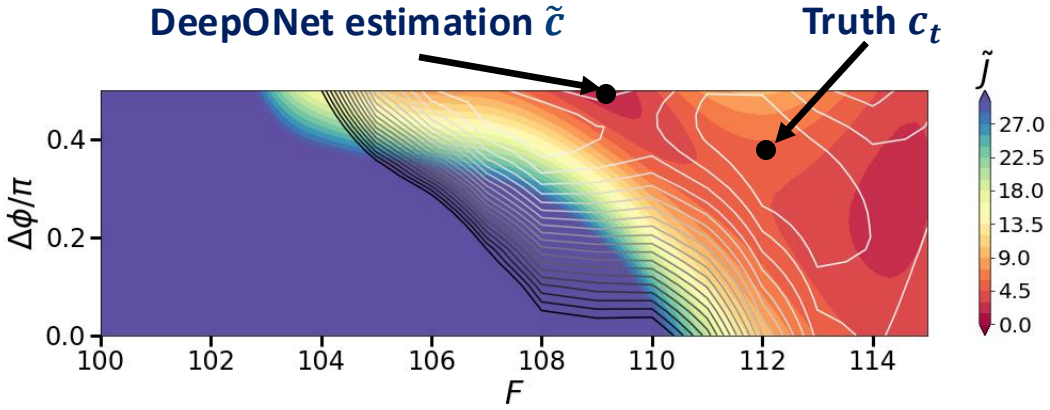


- **DeepONet predicted wall pressure**



- **Inflow estimation by DeepONet**
  - DeepONet estimated control vector
    - $\tilde{c} = \underset{c \in A}{\operatorname{argmin}} \tilde{\mathcal{J}}(c)$
  - Truth $c_t = (112, 0.38\pi)$

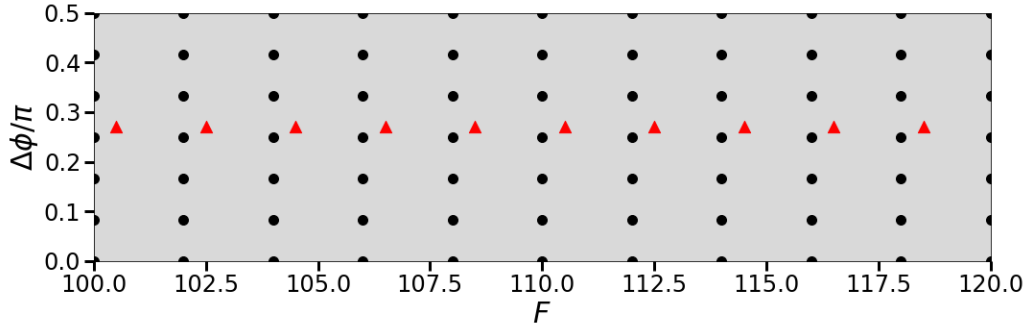
Loss function landscape
Contour: DeepONets modeled loss function $\tilde{\mathcal{J}}(c)$
Lines: True loss function $\mathcal{J}(c)$



- DeepONet estimation $\tilde{c} = (109.6, 0.5\pi)$

Result from smaller training set

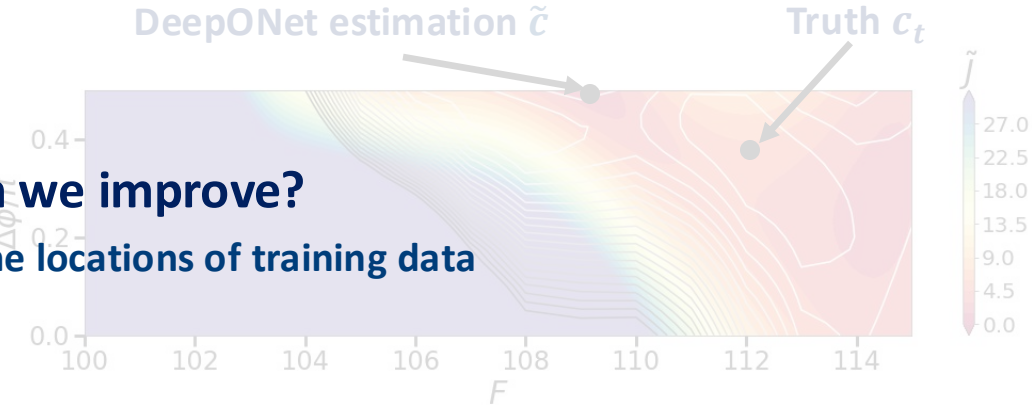
- **Training dataset**
  - $\Delta F = 2, \Delta\phi = \pi/12$
  - Total # of cases: $11 \times 7 = 77$



- **DeepONet predicted wall pressure**

- **Inflow estimation by DeepONet**
  - DeepONet estimated control vector
    - $\tilde{c} = \underset{c \in A}{\operatorname{argmin}} \tilde{\mathcal{J}}(c)$
  - Truth $c_t = (112, 0.38\pi)$

Loss function landscape
Contour: DeepONets modeled loss function $\tilde{\mathcal{J}}(c)$
Lines: True loss function $\mathcal{J}(c)$



- DeepONet estimation $\tilde{c} = (109.6, 0.5\pi)$

How can we improve?

- Wisely choose the locations of training data

Method

- **Training point selection strategy:**

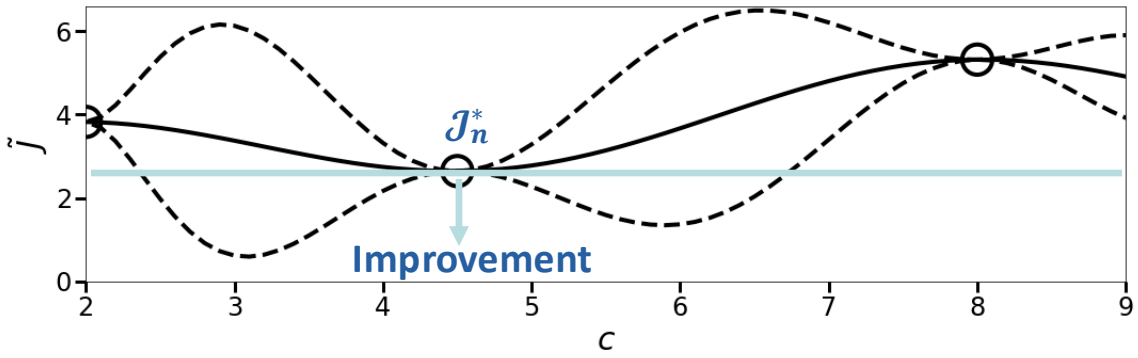
  Bayesian Optimization

  - **Observe $\mathcal{J}$ at n_0 points according to an initial space-filling experimental design**
  - **Set $n = n_0$**
  - **While $n \leq N$ do**
    - **Fit a response surface $\log\tilde{\mathcal{J}} \sim \mathcal{N}(\mu, \sigma^2)$ of $\mathcal{J}$**
    - **Compute expected improvement**

      $$\mathrm{EI}_n(c) = E_{\log\tilde{\mathcal{J}} \sim \mathcal{N}(\mu,\sigma^2)}\left[max\left(\mathcal{J}_n^* - \tilde{\mathcal{J}}, 0\right)\right]$$

    - **Let c_n be a maximizer of $\mathrm{EI}_n(c)$**
    - **Observe $\mathcal{J}_n = f(c_n)$**
    - **Increment n**
  - **End while**
  - **Return a solution: the point with the smallest posterior mean**

Fitted posterior distribution
Circle: observation of $\mathcal{J}$
Solid line: posterior mean; Dash line: 95%credible interval
Minimal observation at itration n: $\mathcal{J}_n^*$



Acquisition Function $\mathrm{EI}_n(c)$

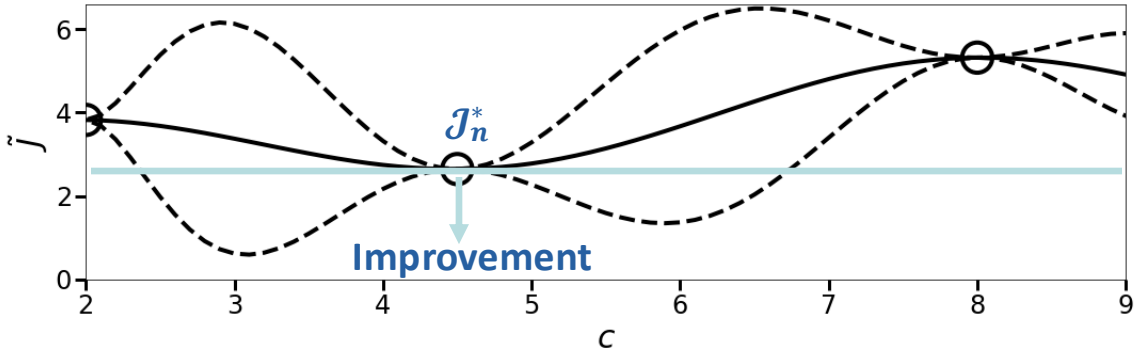
Method

- **Training point selection strategy:**

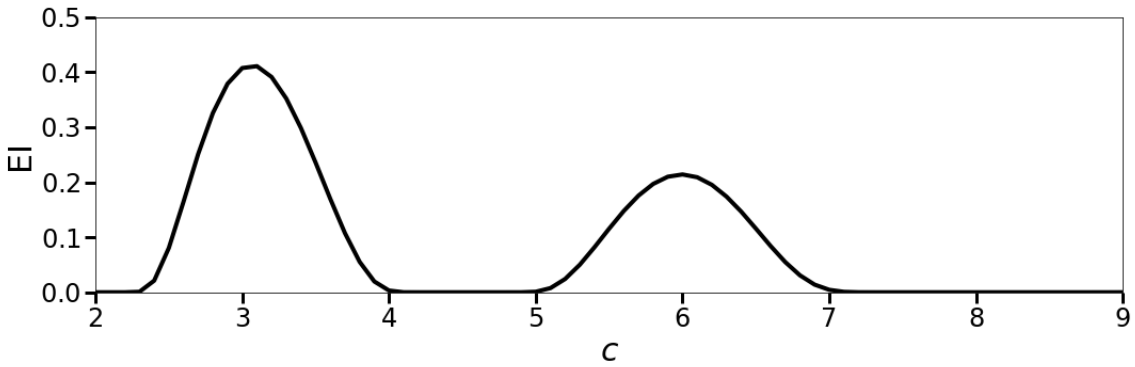
  Bayesian Optimization
  - Observe $\mathcal{J}$ at n_0 points according to an initial space-filling experimental design
  - Set $n = n_0$
  - While $n \leq N$ do
    - Fit a response surface $\log\tilde{\mathcal{J}} \sim \mathcal{N}(\mu, \sigma^2)$ of $\mathcal{J}$
    - Compute expected improvement

      $$\mathrm{EI}_n(c) = E_{\log\tilde{\mathcal{J}} \sim \mathcal{N}(\mu,\sigma^2)}\left[max\left(\mathcal{J}_n^* - \tilde{\mathcal{J}}, 0\right)\right]$$
    - Let c_n be a maximizer of $\mathrm{EI}_n(c)$
    - Observe $\mathcal{J}_n = f(c_n)$
    - Increment n
  - End while
  - Return a solution: the point with the smallest posterior mean

Fitted posterior distribution
Circle: observation of $\mathcal{J}$
Solid line: posterior mean; Dash line: 95%credible interval
Minimal observation at itration n: $\mathcal{J}_n^*$



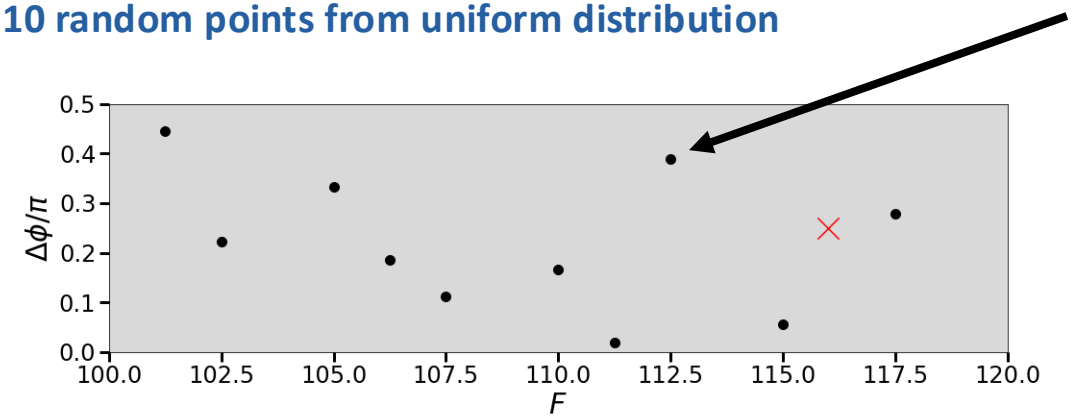
Acquisition Function $\mathrm{EI}_n(c)$



Method

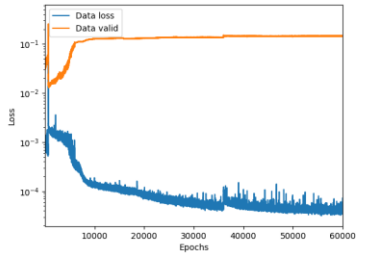
- **Statistical model**
  - **Gaussian process regression**
    - **Prediction assumed** $\log\tilde{\mathcal{J}} \sim \mathcal{N}(\mu, \sigma^2)$
    - **Correlation between** $\tilde{\mathcal{J}}(c^{(a)}), \tilde{\mathcal{J}}(c^{(b)})$: $\exp\left[-d\left(\phi^{(a)} - \phi^{(b)}\right)\right]$
    - **Fit parameters from current training data**

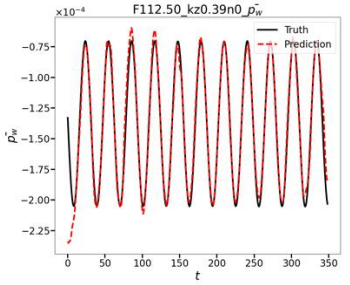
Initial training data:
10 random points from uniform distribution



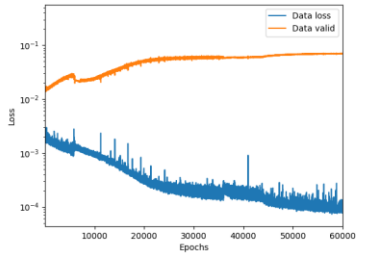
- **DeepONets + Deep ensemble uncertainty**
  - **Quantify uncertainty from N independent DeepONets**

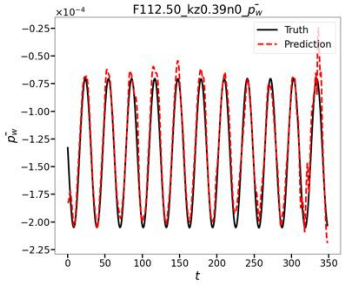
DeepONets 1





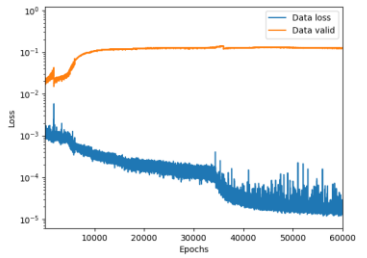
DeepONets 2

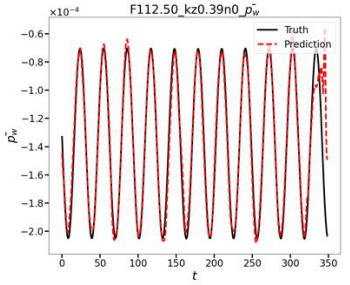




……

DeepONets N

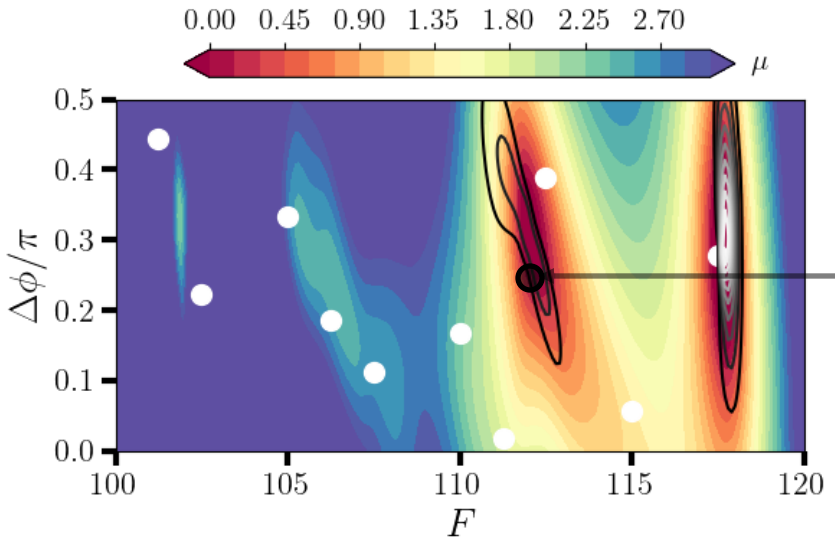




Result

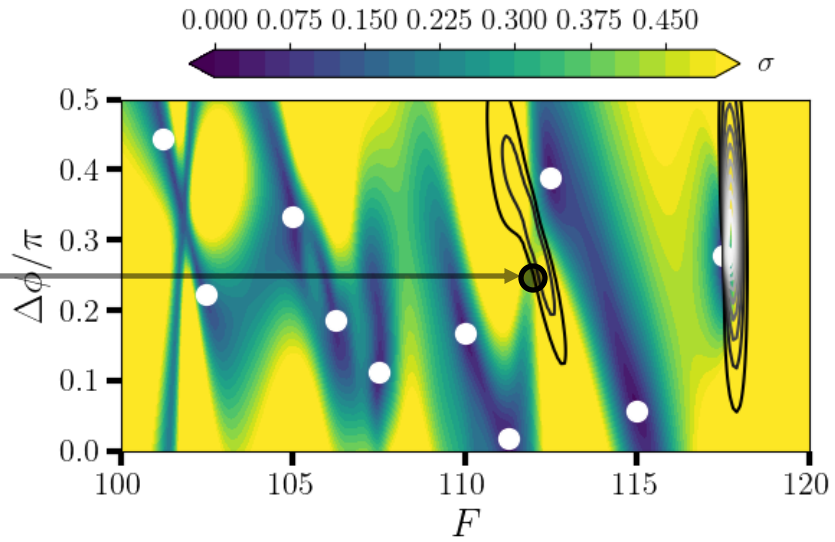
- **Fitted posterior distribution at iteration 0**
  - Color contour: posterior mean, standard deviation
  - Line contour: expected improvement

Posterior mean



Truth
$c_t = (112, 0.25\pi)$

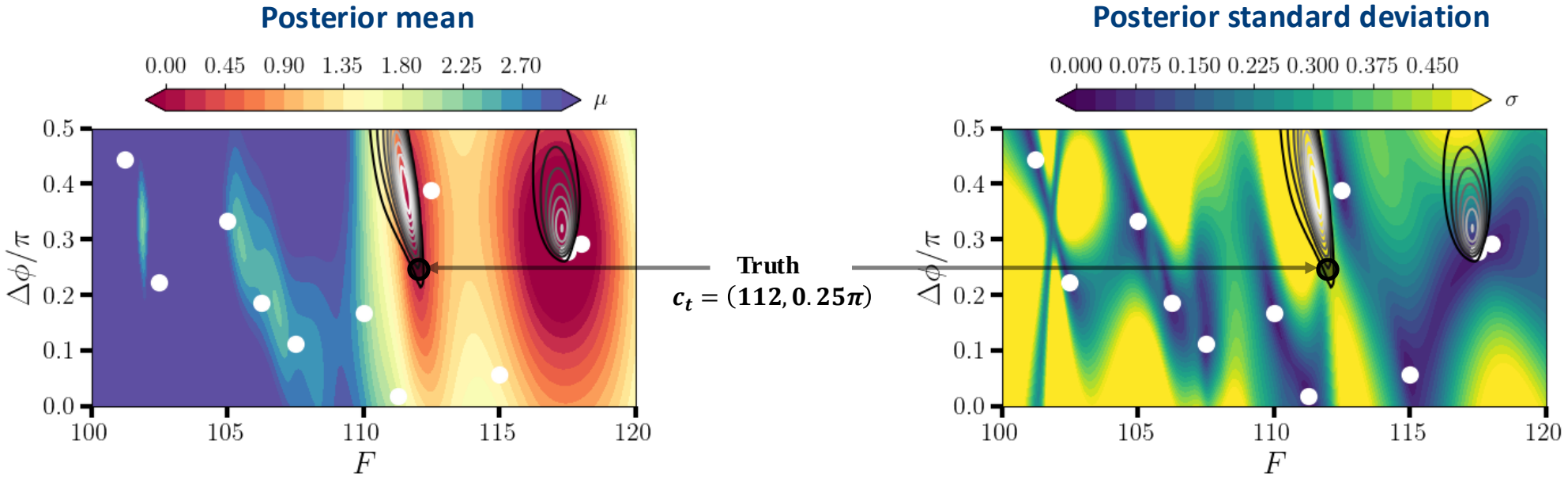
Posterior standard deviation



- **Expected Improvement at iteration 0**
  - $\mathbf{EI}_0(c) = E_{Y\sim\mathcal{N}(\mu,\sigma^2)}[max(\mathcal{J}_0^* - Y, 0)]$
  - $\mathbf{EI}_0(c)$ maximized at $c_0 = (118.0, 0.29\pi)$

Result

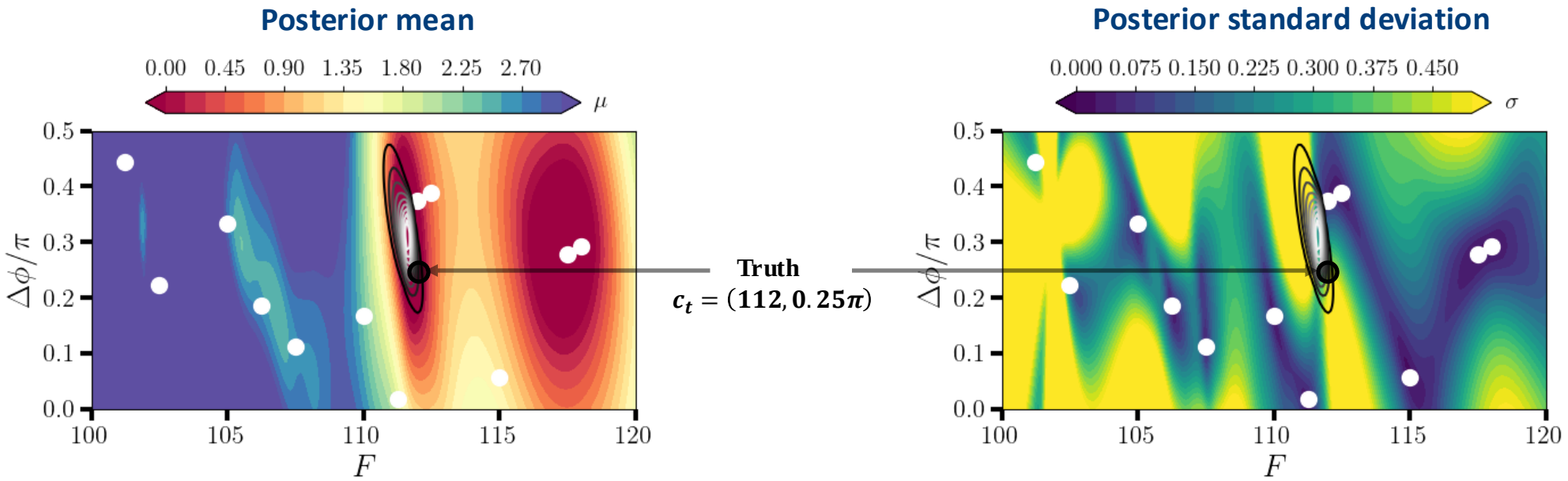
- **Fitted posterior distribution at iteration 1**
  - Color contour: posterior mean, standard deviation
  - Line contour: expected improvement



- **Expected Improvement at iteration 1**
  - $\mathbf{EI}_1(c) = E_{Y\sim\mathcal{N}(\mu,\sigma^2)}[max(\mathcal{J}_1^* - Y, 0)]$
  - $\mathbf{EI}_1(c)$ maximized at $c_1 = (112.0, 0.38\pi)$

Result

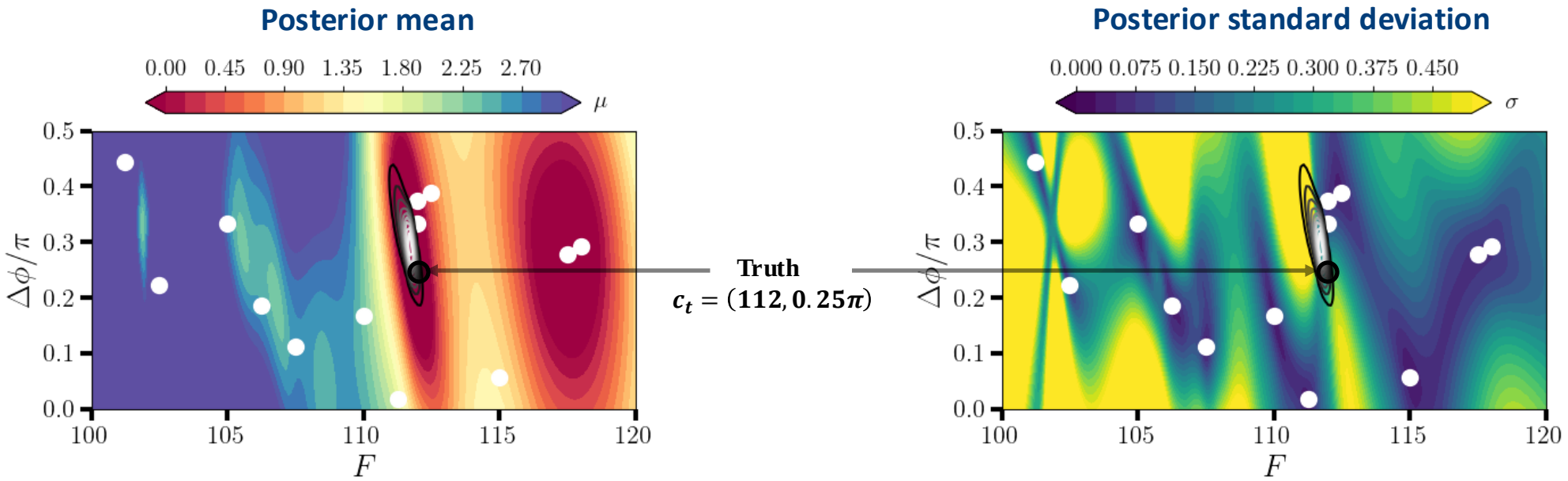
- **Fitted posterior distribution at iteration 2**
  - Color contour: posterior mean, standard deviation
  - Line contour: expected improvement



- **Expected Improvement at iteration 2**
  - $\mathbf{EI}_2(c) = E_{Y\sim\mathcal{N}(\mu,\sigma^2)}[max(\mathcal{J}_2^* - Y, 0)]$
  - $\mathbf{EI}_2(c)$ maximized at $c_2 = (112.0, 0.33\pi)$

Result

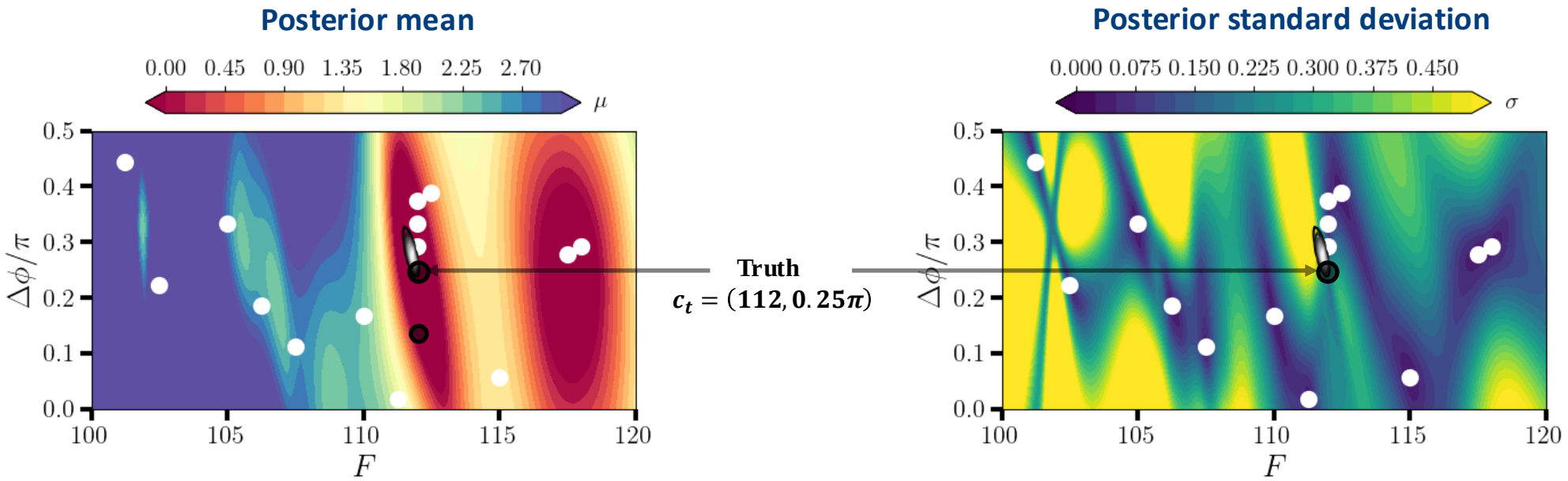
- **Fitted posterior distribution at iteration 3**
  - Color contour: posterior mean, standard deviation
  - Line contour: expected improvement



- **Expected Improvement at iteration 3**
  - $\mathbf{EI}_3(c) = E_{Y\sim\mathcal{N}(\mu,\sigma^2)}[max(\mathcal{J}_3^* - Y, 0)]$
  - $\mathbf{EI}_3(c)$ maximized at $c_3 = (112.0, 0.29\pi)$

Result

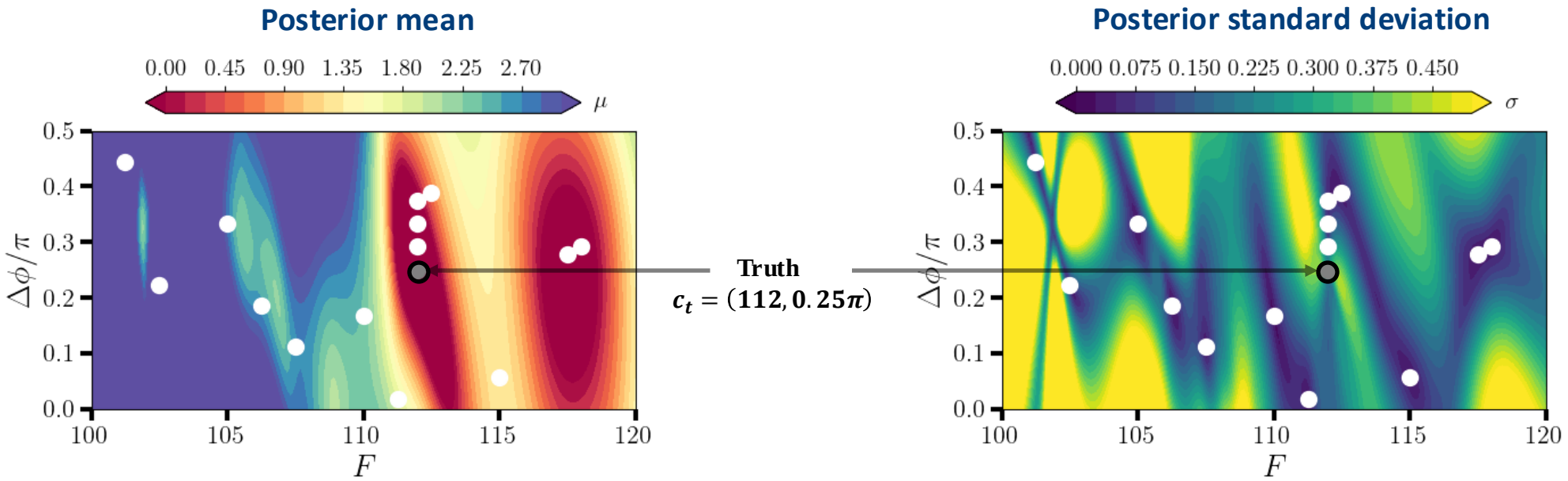
- **Fitted posterior distribution at iteration 4**
  - Color contour: posterior mean, standard deviation
  - Line contour: expected improvement



- **Expected Improvement at iteration 4**
  - $\mathbf{EI}_4(c) = E_{Y\sim\mathcal{N}(\mu,\sigma^2)}[max(\mathcal{J}_4^* - Y, 0)]$
  - $\mathbf{EI}_4(c)$ maximized at $c_4 = (112.0, 0.25\pi)$

Result

- **Fitted posterior distribution at iteration 5**
  - Color contour: posterior mean, standard deviation
  - Line contour: expected improvement

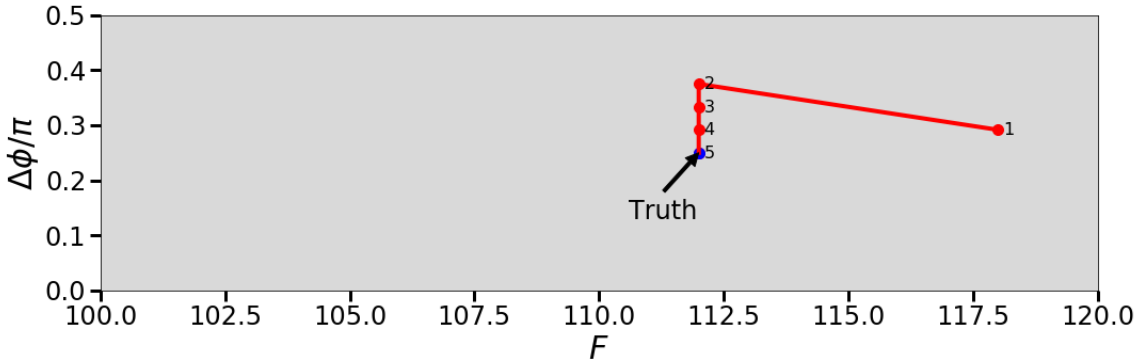


- **Observation**
  - Minimum error: $\epsilon = \sqrt{\mathcal{J}_5^*} < 5\%$, successfully converged!
  - Estimation $\tilde{c} = c_t = (112, 0.25\pi)$ is accurate

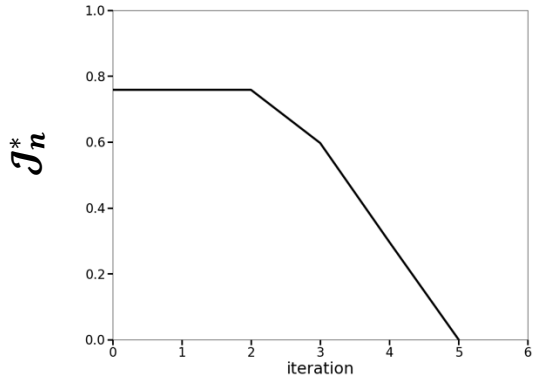
Result

- **Convergence history**

Added training data

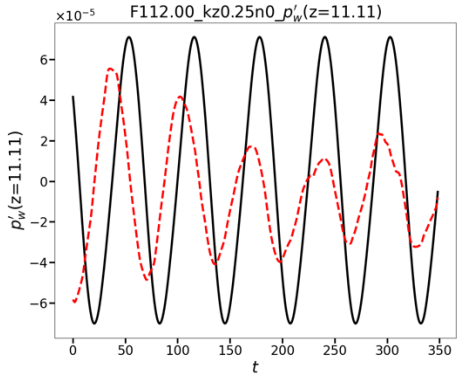


Objective function

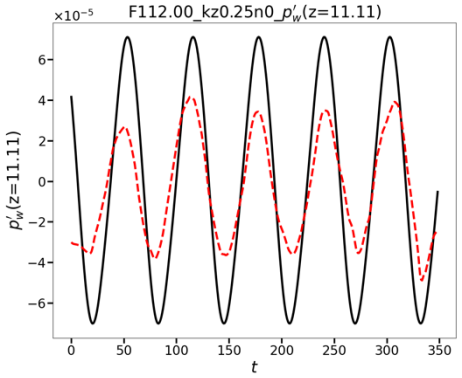


- **Predicted wall pressure on true control vector $c_t = (F, \Delta\phi) = (112.0, 0.25)$**

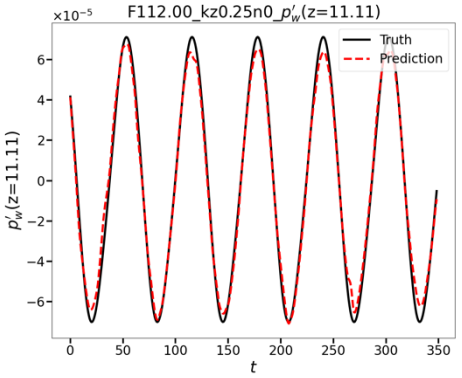
Iteration 1



Iteration 3

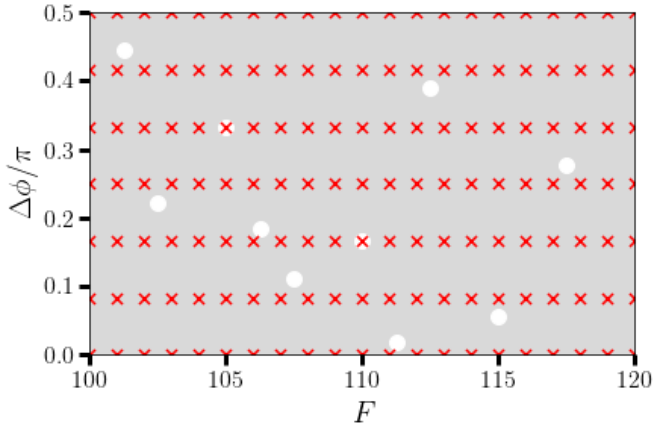


Iteration 5

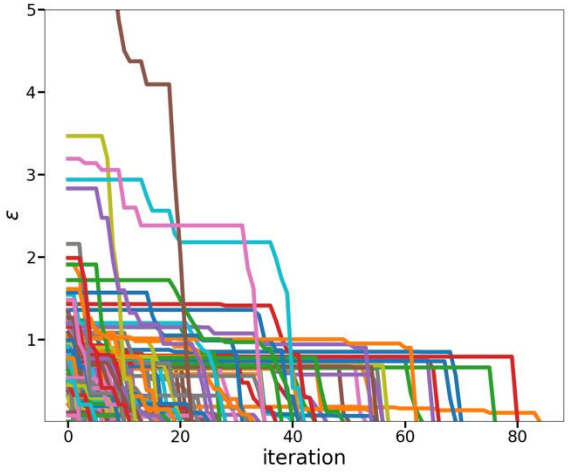


Result

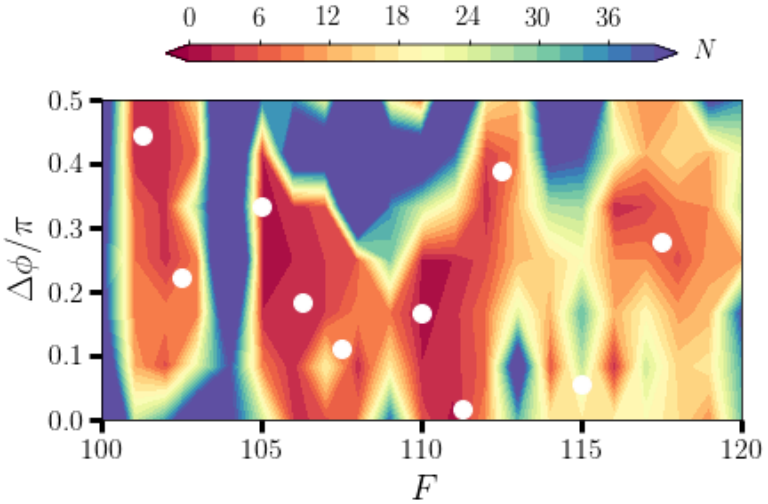
- **Other test cases**
  - $F, \Delta\phi$ in range $[100 - 120] \times [0 - 0.5\pi]$
  - White dots: initial training points
  - Red cross: testing cases



Objective function



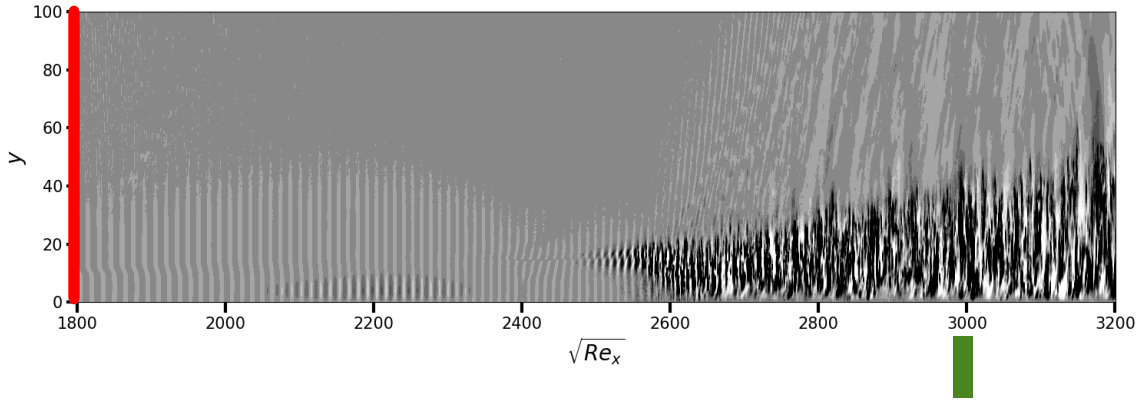
Convergence speed



  - Comparison with Gaussian process regression(GPR):
    - DeepONet converges in 22 steps
    - GPR converges in 81 steps
    - DeepONet shows ~ 3.7 times acceleration from Kriging

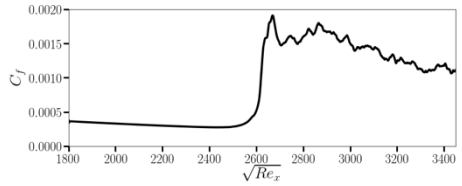
Conclusions

- **Forward and inverse problem in high-speed boundary layer**
  - **DeepONet prediction of evolution of instability waves of Ma=4.5 & Ma=10 boundary layer**
  - **Transition prediction using DeepONet, accelerated by active learning**
  - **Solving inverse problem using DeepONet & Bayesian optimization**



Forward problem

Upstream flow disturbance → **Skin friction prediction**



Inverse problem

Unknown upstream velocity disturbance Control vector c ← **discover** ← **Downstream wall-pressure data from sensor array measurements m**