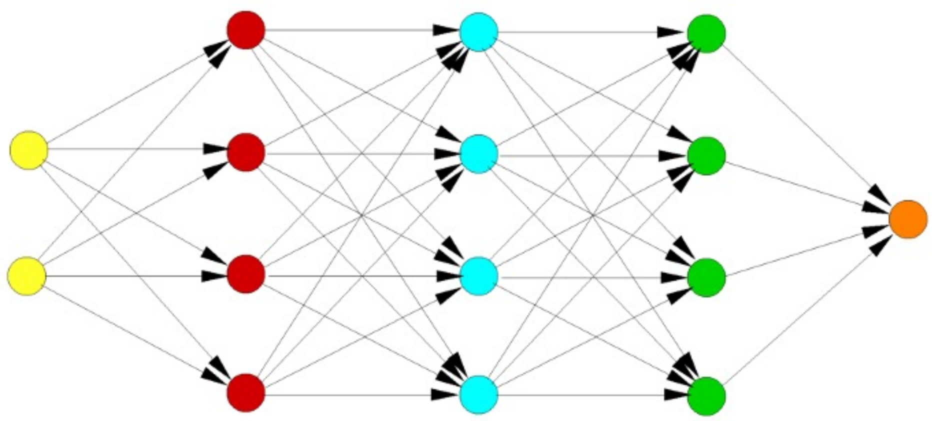
Data-Driven Construction of Iterative Algebraic Reynolds Stress Models using Model-Derived Turbulence Variables



John Andrew Evans

Smead Department of Aerospace Engineering Sciences, University of Colorado Boulder

Collaborators: B. Parmar, A. Prakash, R. Balin, E. Peters, K. Jansen

Motivation

The Ultimate Goal:

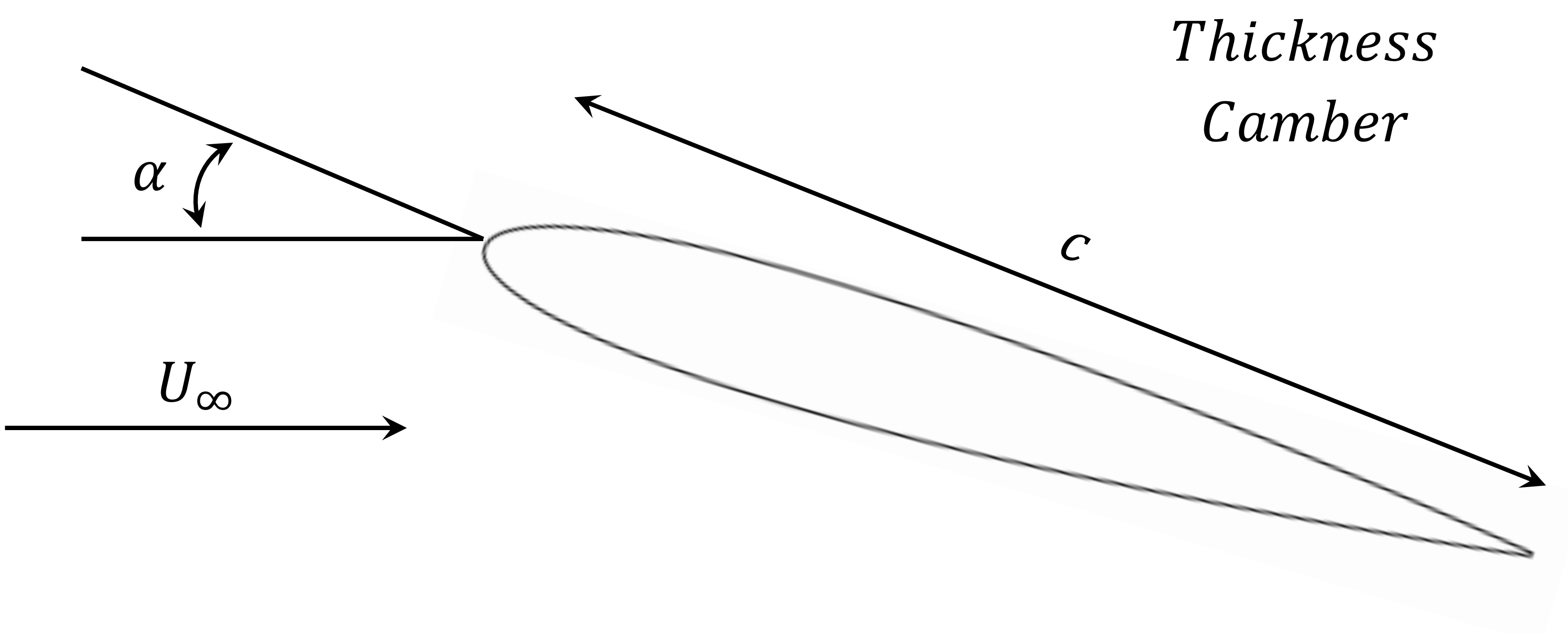
Data-driven construction of a RANS model that is universal in the "sense it can be used by anyone and applied to as many flows as possible without concern for unusual or dentrimental behavior".

C.L. Rumsey, G.N. Coleman, and L. Wang. "In search of data-driven improvements to RANS models applied to separated flows." *AIAA SCITECH 2022 Forum*. 2022.

Our Far Less Ambitious Goal:

Construction of a data-driven framework capable of yielding RANS models that are accurate and stable for design space exploration about a nominal design configuration.

Motivation

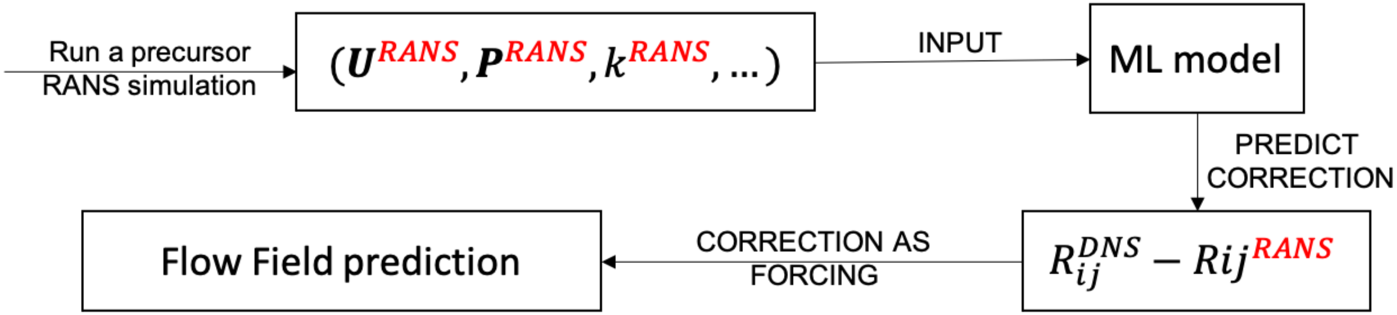


Our particular objective is to learn an effective RANS model for the entire design space using data coming from a very small number of high-fidelity simulations (ideally one simulation).

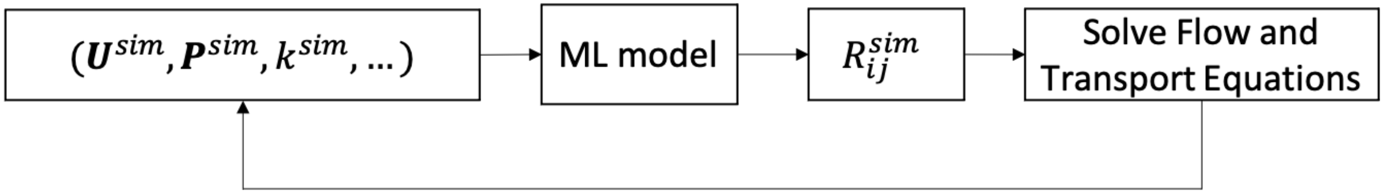
More concisely, we seek a model that extrapolates beyond the training set of designs – but still within a limited design space.

Motivation

This last objective has pushed us to consider iterative models than more commonly employed corrective models.



Corrective Model Workflow



Iterative Model Workflow

Motivation

Our framework has also been driven by the following:

1. We seek to learn as few model terms as possible.

   We start with a baseline RANS model and only learn an improved Reynolds stress model.

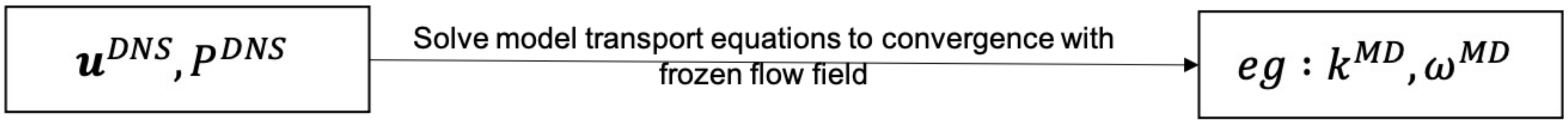
2. We seek to run as few RANS simulations as possible during training.

   We train using turbulence variables that are derived from the baseline RANS model and the high-fidelity data.

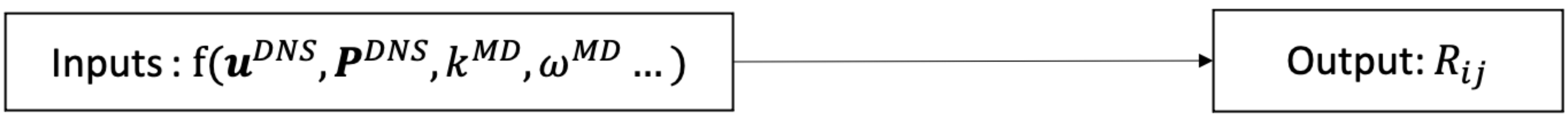
3. We seek to build algebraic Reynolds stress models that accommodate an arbitrary number of tensor inputs.

   We learn the components of the Reynolds stress tensor in a particular flow- and geometry-dependent reference frame in terms of inputs in this same frame.

Training with Model-Derived Variables



Step 1: Generate Model-Derived Turbulence Variables



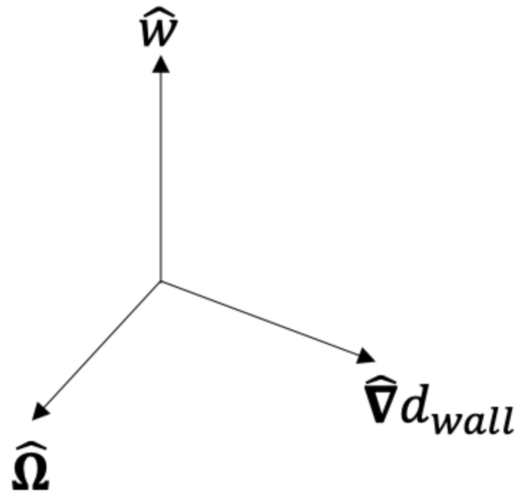
Step 2: Train Reynolds Stress Model Using High-Fidelity Flow Field Data and Model-Derived Turbulence Variables

Note this training methodology can be used in conjunction with any baseline RANS model.

N-Frame Reynolds Stress Representation

As previously mentioned, we learn the components of the Reynolds stress tensor in a particular flow- and geometry-dependent reference frame in terms of inputs of this same frame.

In particular, we employ a frame constructed from vorticity and the gradient of the distance to the wall:



Note this automatically yields a model form with rotational and reflectional invariance.

N-Frame Reynolds Stress Representation

Suppose we have the following Galilean invariant model form:

$$R_{ij} = R_{ij}^{model}(\boldsymbol{S}, \boldsymbol{\Omega}, k, \omega, \boldsymbol{\nabla} k, \nu, d, \boldsymbol{\nabla} d)$$

Then our chosen Reynolds stress representation yields the following rotationally, reflectionally, and Galilean invariant model form in 2D:

$$R_{ij}^{N} = R_{ij}^{N,model}(S_{11}^{N}, S_{12}^{N}, |\boldsymbol{\Omega}|, k, \omega, (\boldsymbol{\nabla} k)_1^N, (\boldsymbol{\nabla} k)_2^N, \nu, d)$$

Only four additional inputs are included in 3D. Any additional vector inputs introduce two new inputs in 2D and three new inputs in 3D, while any additional tensor inputs introduce four new inputs in 2D and nine new inputs in 3D.

N-Frame Reynolds Stress Representation

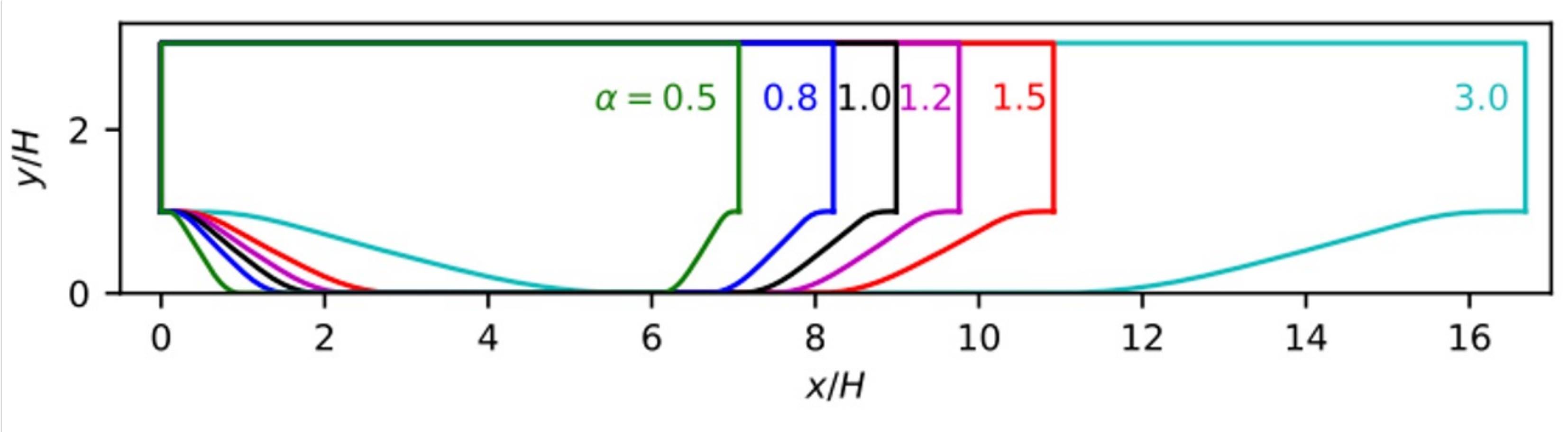
We can further use the Buckingham Pi theorem to arrive at a rotationally, reflectionally, Galilean, and unit invariant model form. This further reduces the number of inputs.

In the model employed in this presentation, the following non-dimensional inputs and outputs were employed:

Input 1	$\frac{\nu}{\nu+\nu_T}$	Output 1	$\frac{a_{11}^N}{(\nu_T + min(\sqrt{k}d_{wall}, \nu))\omega}$
Input 2	$\frac{\sqrt{k}d_{wall}}{\sqrt{k}d_{wall}+\nu}$	Output 2	$\frac{a_{22}^N}{(\nu_T + min(\sqrt{k}d_{wall}, \nu))\omega}$
Input 3,4	$\frac{S_{11}^N}{\lvert\lvert\boldsymbol{S}\rvert\rvert+\lvert\boldsymbol{\Omega}\rvert+\omega}, \frac{S_{12}^N}{\lvert\lvert\boldsymbol{S}\rvert\rvert+\lvert\boldsymbol{\Omega}\rvert+\omega}$	Output 3	$\frac{a_{12}^N}{(\nu_T + min(\sqrt{k}d_{wall}, \nu))\omega}$
Input 5	$\frac{\lvert\boldsymbol{\Omega}\rvert^N}{\lvert\lvert\boldsymbol{S}\rvert\rvert+\lvert\boldsymbol{\Omega}\rvert+\omega}$		
Input 6,7	$\frac{\nabla k_1^N}{\lvert\boldsymbol{\nabla}k\rvert+\sqrt{k}\omega}, \frac{\nabla k_2^N}{\lvert\boldsymbol{\nabla}k\rvert+\sqrt{k}\omega}$		

Demonstration Example

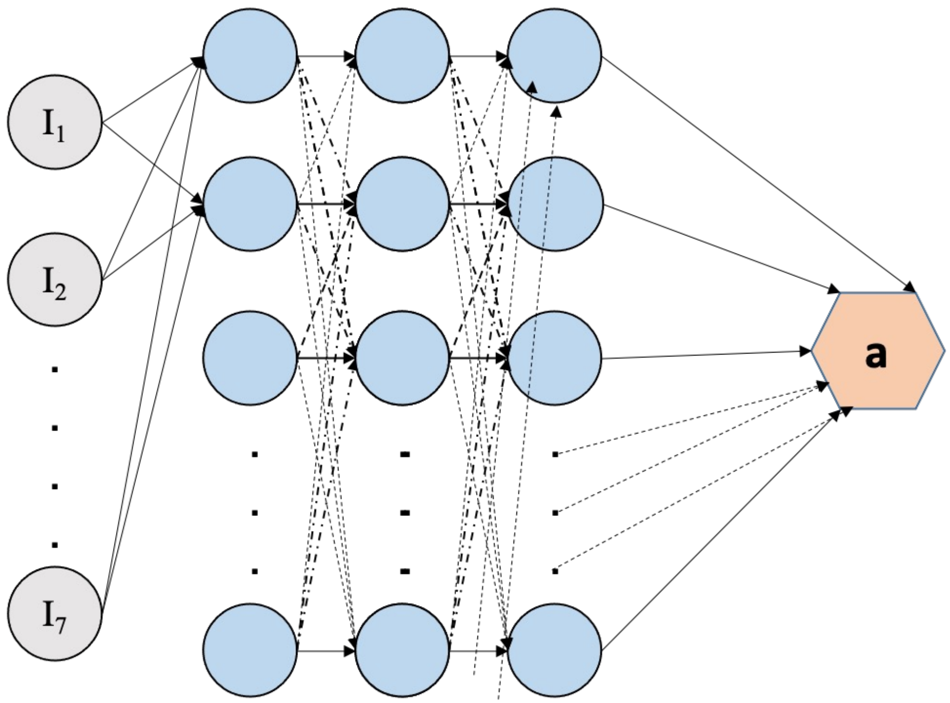
To demonstrate the utility of our framework, we apply it to the periodic hill problem at *Re* = 5600 with SST as the baseline RANS model:



H. Xiao, et al. "Flows over periodic hills of parameterized geometries: A dataset for data-driven turbulence modeling from direct simulations." Computers & Fluids 200 (2020): 104431.

We train using the $\alpha = 1.0$ (baseline) geometry and test using the $\alpha = 0.5$ (short) and $\alpha = 1.5$ (long) geometries.

Neural Network Architecture



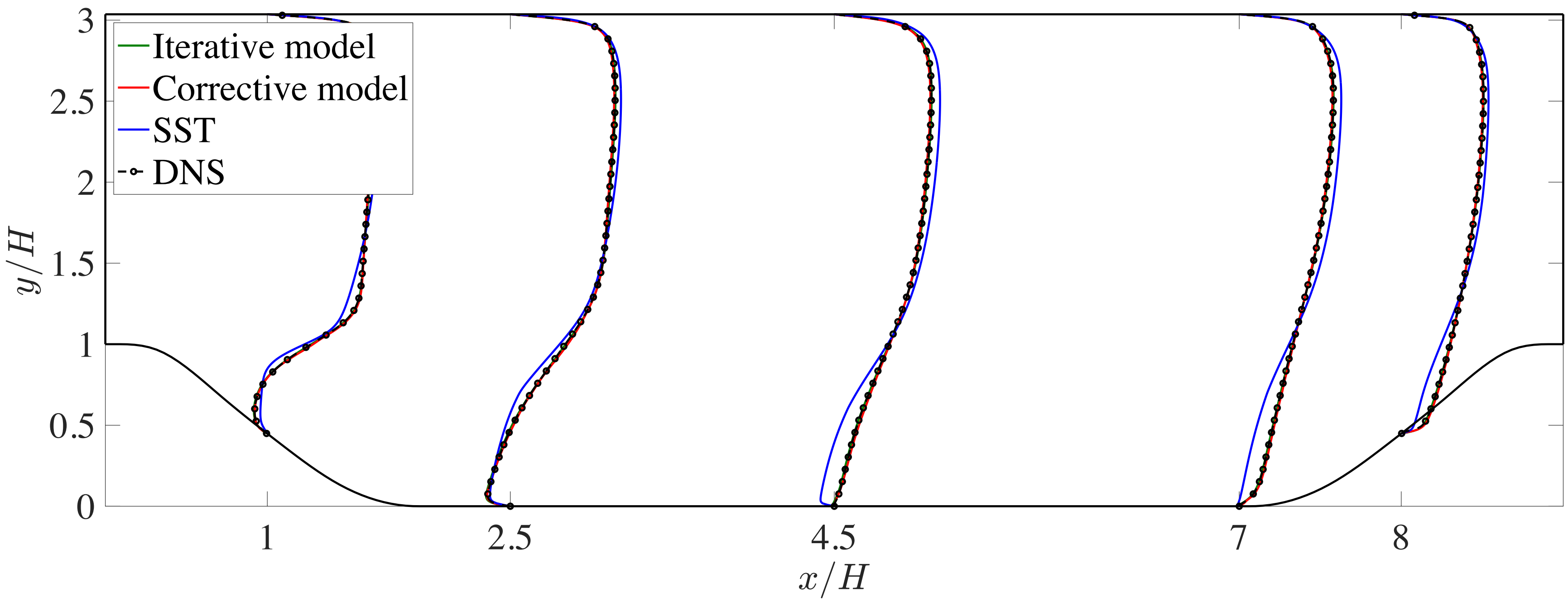
3 Hidden Layers

100 Neurons Per Layer

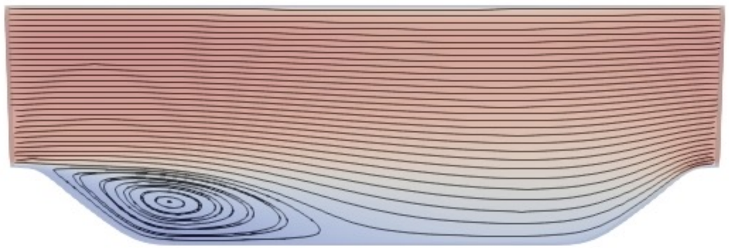
Leaky ReLU Activation

Neural network trained using the Adam optimizer using a standard MSE loss function with L2, Gaussian noise, and dropout regularization.

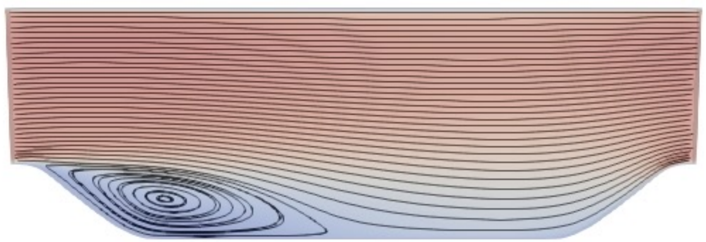
Performance for Baseline Geometry



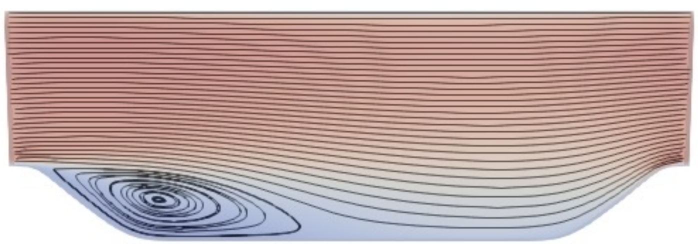
Performance for Baseline Geometry



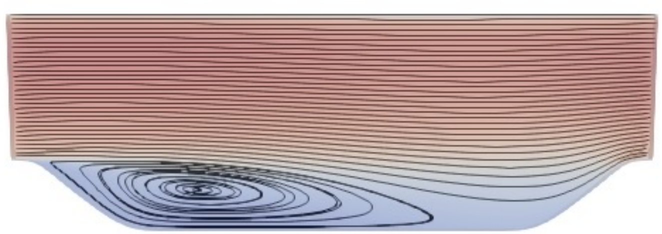
Iterative Model



DNS

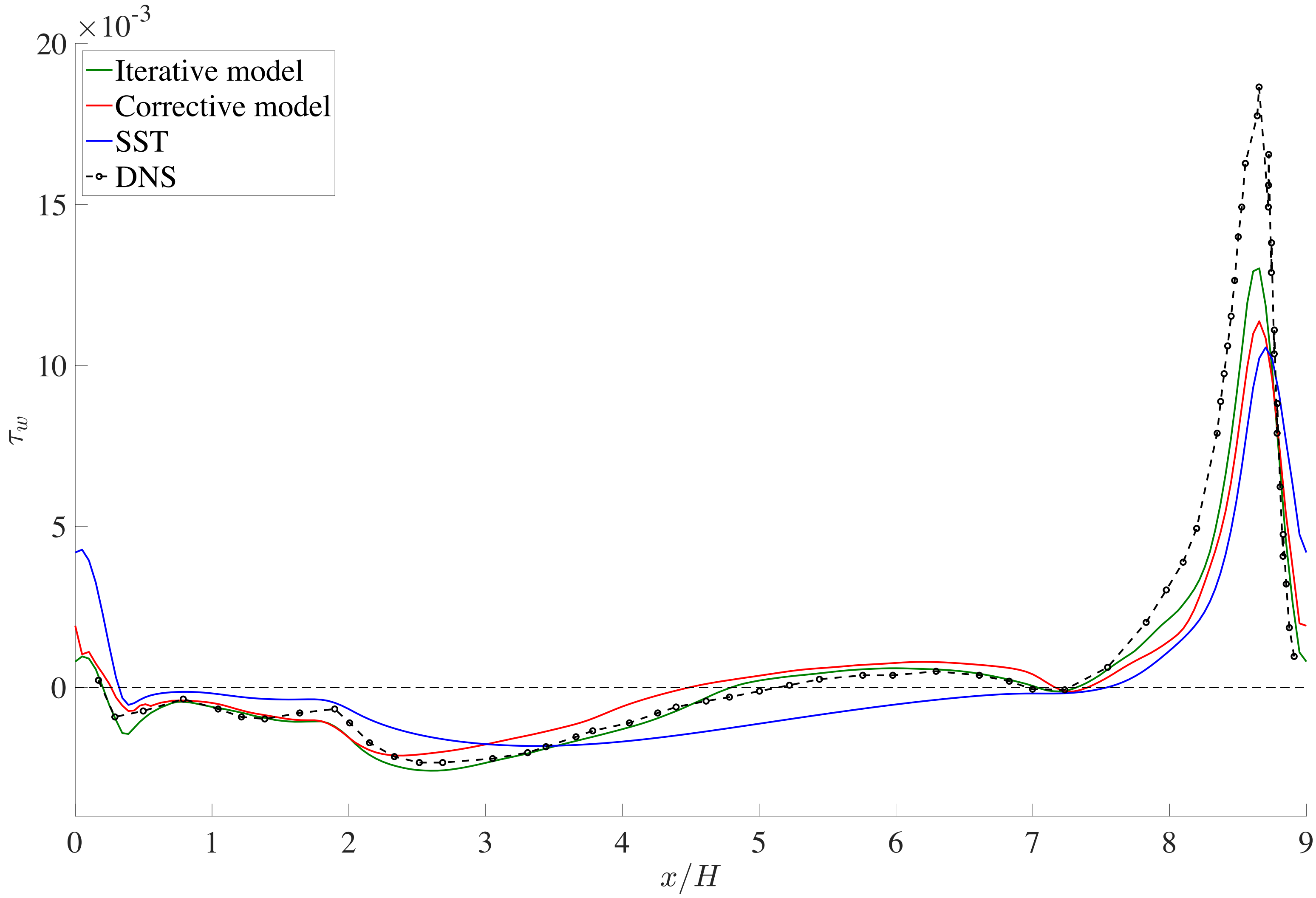


Corrective Model

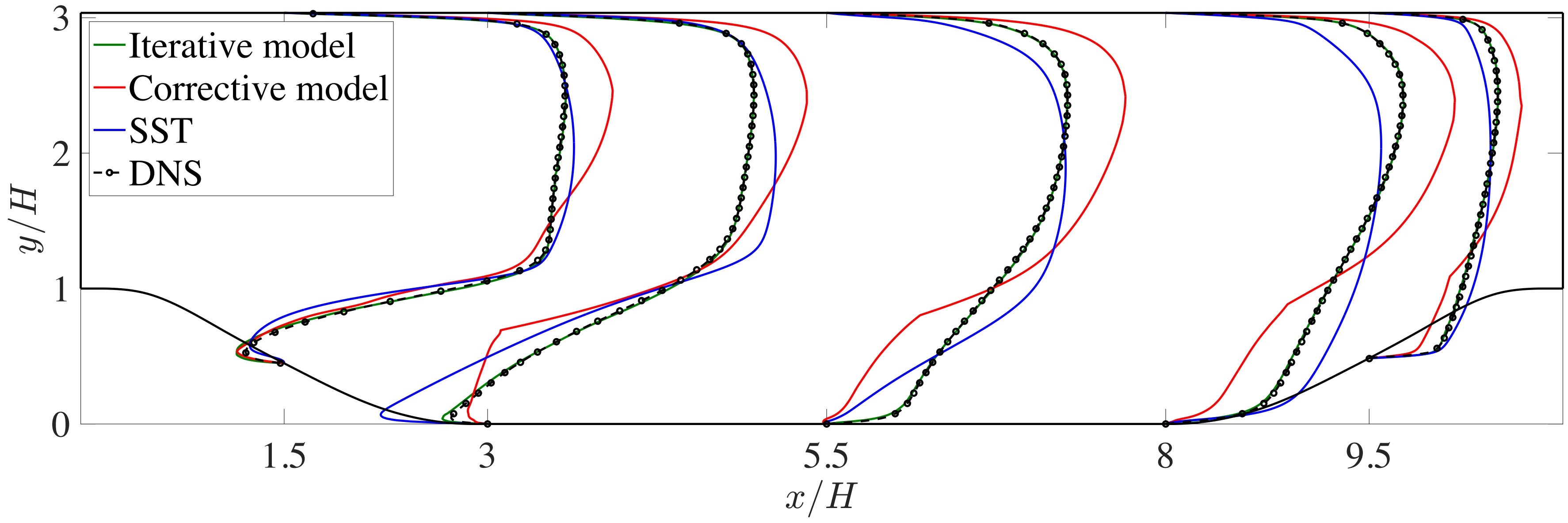


SST

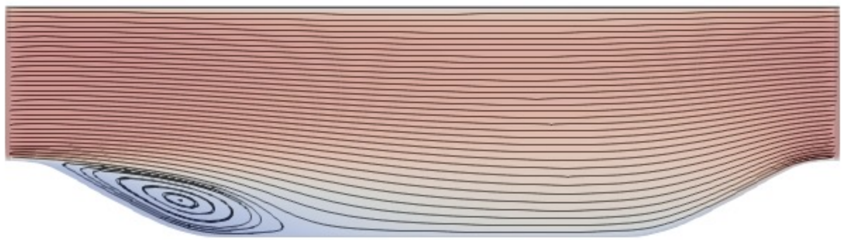
Performance for Baseline Geometry



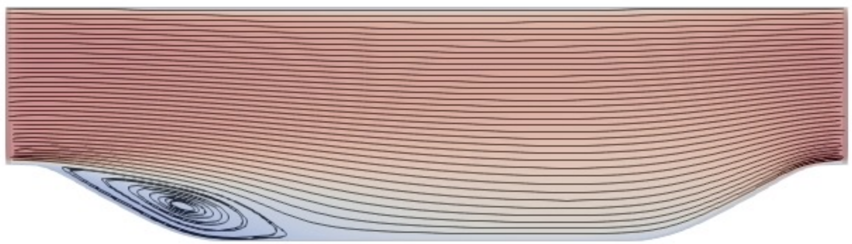
Performance for Long Geometry



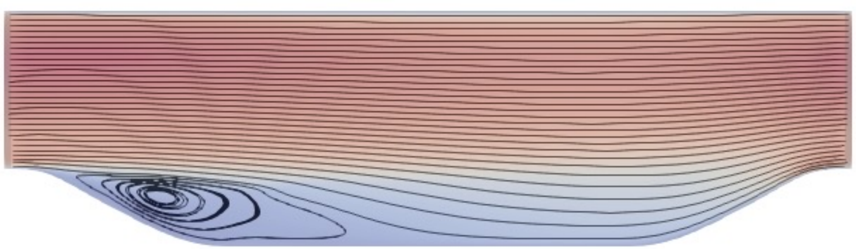
Performance for Long Geometry



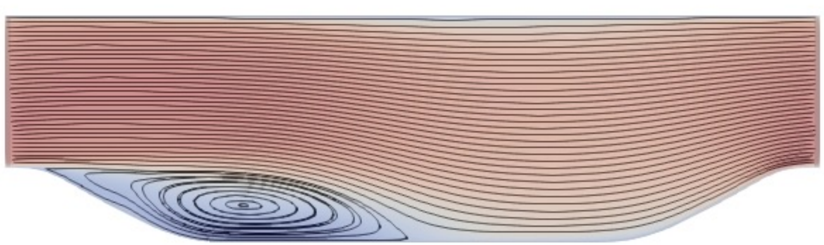
Iterative Model



DNS

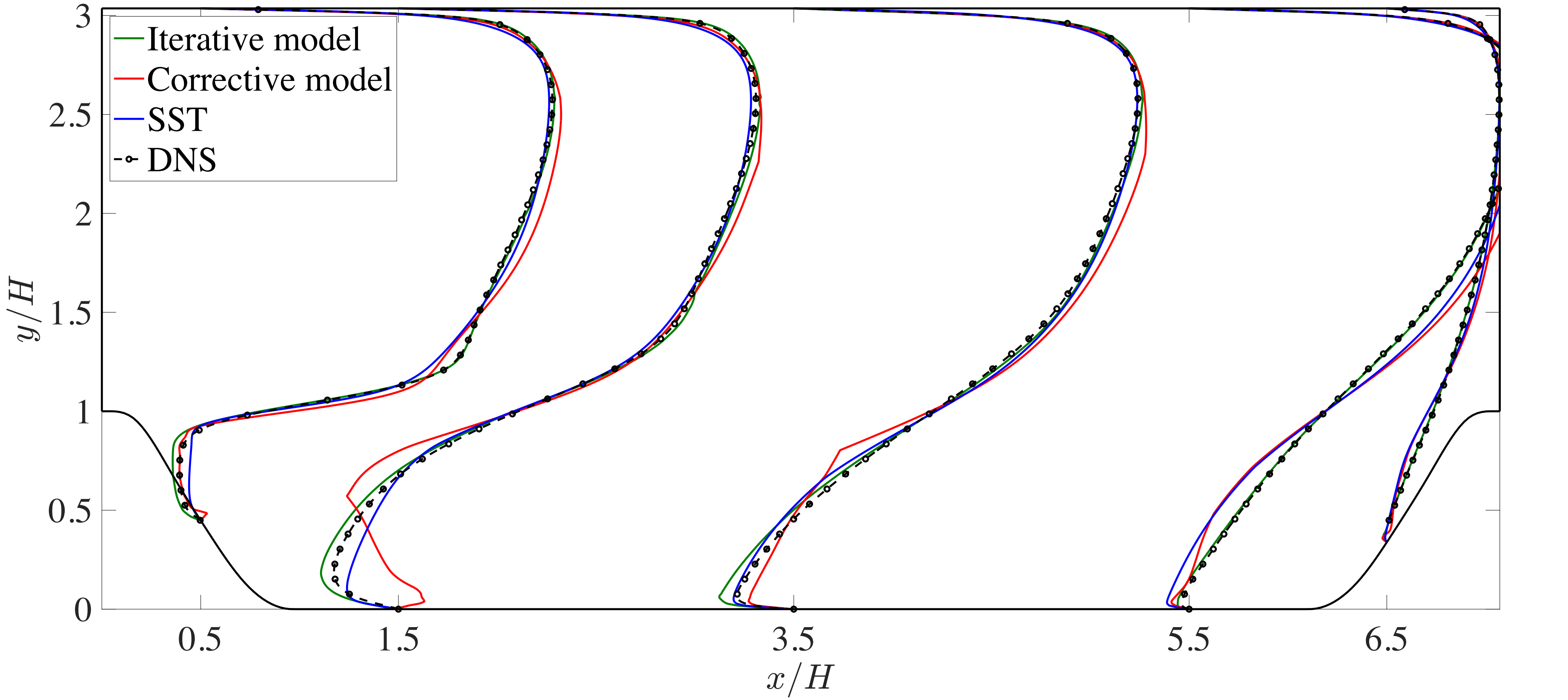


Corrective Model

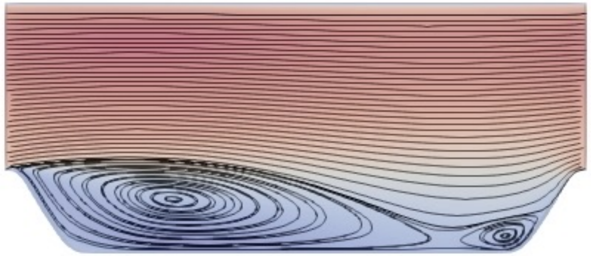


SST

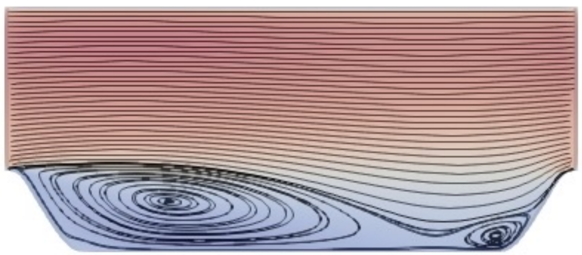
Performance for Short Geometry



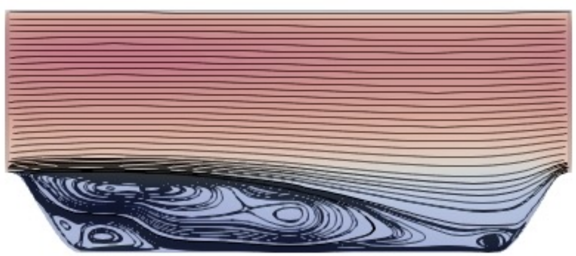
Performance for Short Geometry



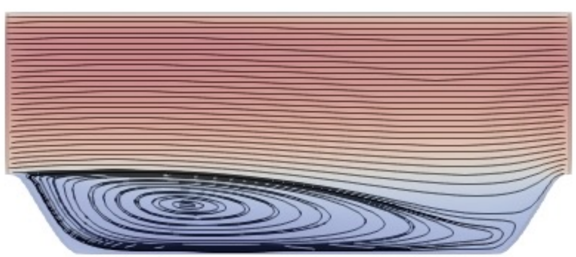
Iterative Model



DNS

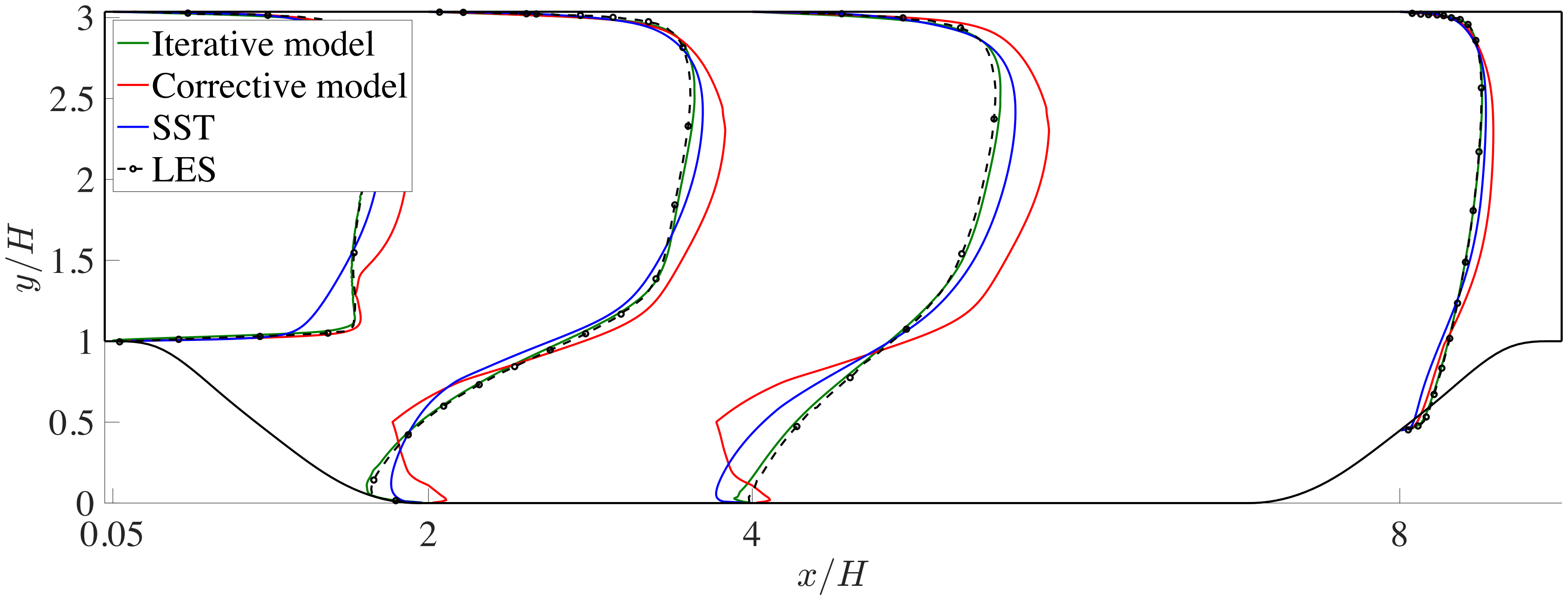


Corrective Model

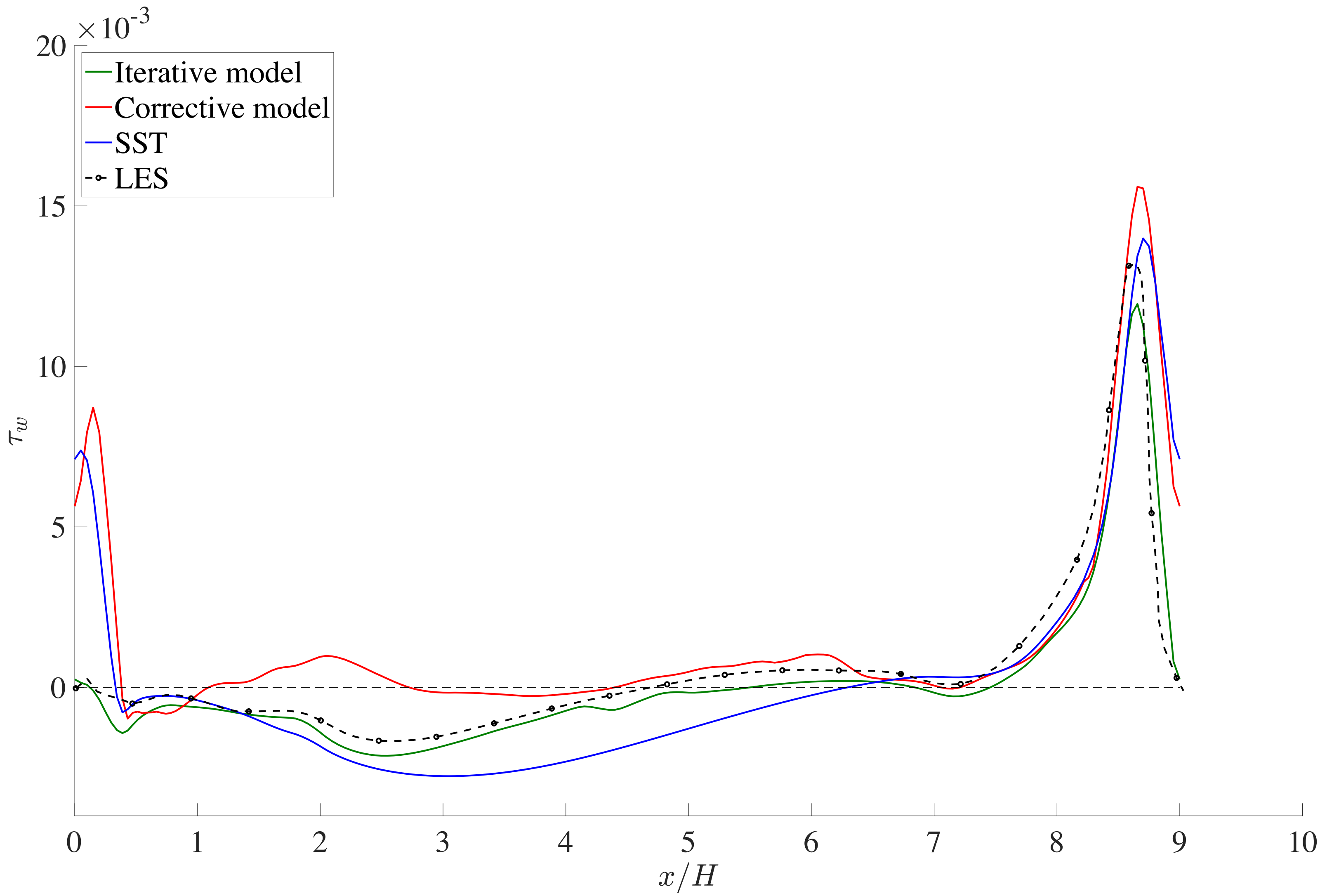


SST

Performance at *Re* = 10,595



Performance at *Re* = 10,595



Thank You For Your Time!