A Framework for Turbulence Modeling using Big Data

Karthik Duraisamy
Department of Aerospace Engineering
Univ of Michigan

NASA Aeronautics Research Mission Directorate (ARMD)

2015 LEARN/Seedling Technical Seminar

January 13-15, 2015

Team

Karthik Duraisamy (Michigan): Turbulence Modeling, Inverse Modeling, Integration, Machine Learning
Juan Alonso (Stanford): Machine Learning, Integration
Paul Durbin (Iowa State): Turbulence modeling
Kaushik Das (Pivotal): Scalability of Machine Learning
Philippe Spalart (Boeing): Advisory

Students:
Brendan Tracey, Helen Zhang, Anand Pratap Singh

Postdocs:
Shivaji Medida

Pivotal:
Hulya Farinas, Grace Gee

Turbulent flows & their computation

"Perhaps the single, most critical area in CFD simulation capability that will remain a pacing item by 2030 in the analysis and design of aerospace systems is the ability to adequately predict viscous turbulent flows with possible boundary layer transition and flow separation present"

Vision 2030 CFD Report (NASA CR 2014-218178)

Turbulent simulations & Modeling hierarchies

Direct Numerical Simulations (DNS):	No modeling required
Large Eddy Simulations (LES) :	Modeling of small scales
Reynolds Averaged Navier—Stokes (RANS) :	Modeling all scales

Resolution requirements for aircraft wing ($Re_c = 2x10^7$)

DNS

LES

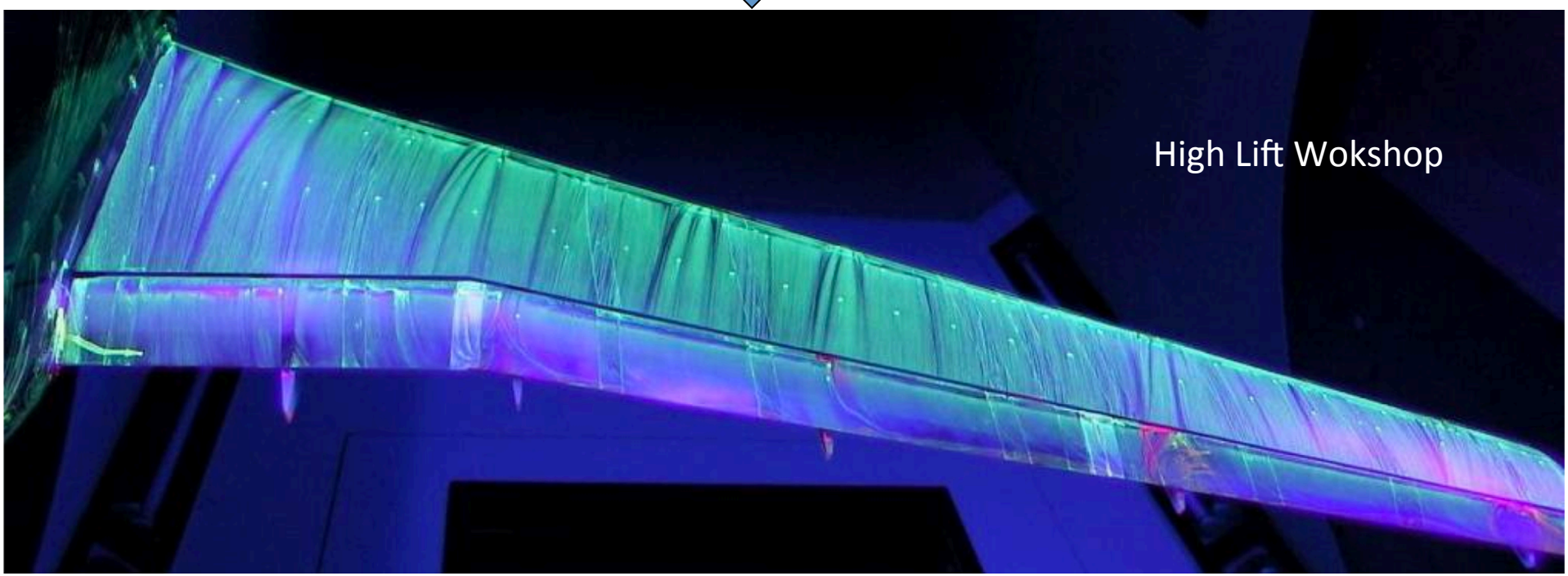
Hybrid RANS-LES / WMLES

RANS

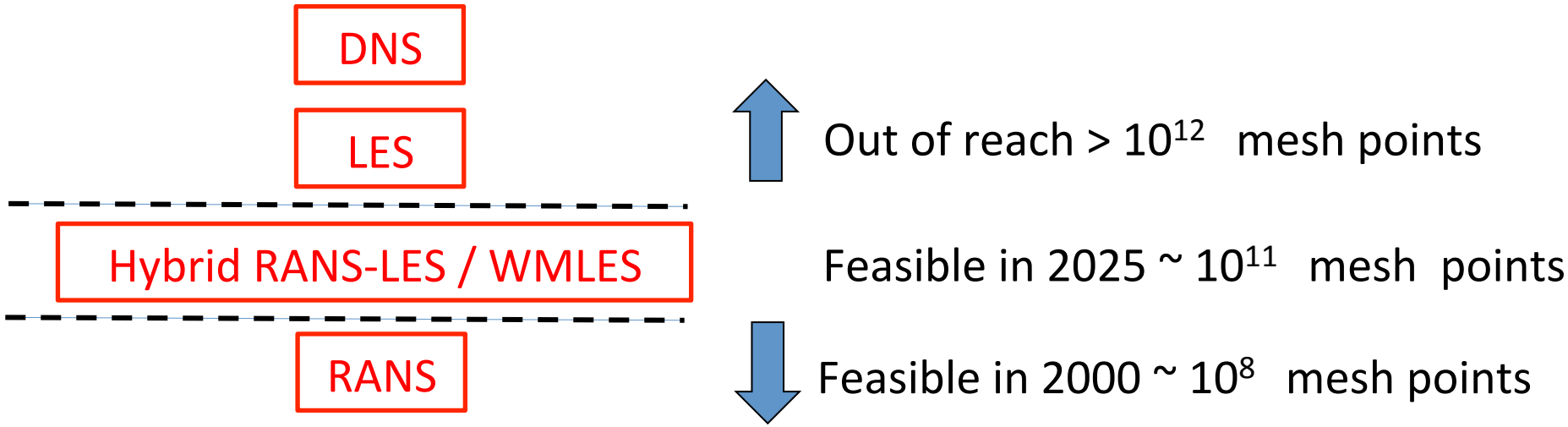
Out of reach > 10^{12} mesh points

Feasible in 2025 ~ 10^{11} mesh points

Feasible in 2000 ~ 10^8 mesh points



Resolution requirements for aircraft wing ($Re_c = 2x10^7$)



Near-wall modeling is here to stay for the next 20 years for analysis and much longer for design

Thoughts on RANS Modeling

- Limited to between one and seven variables, and up to 30 adjustable constants.

- Their creation is unpredictable and appears to rest on large amounts of intuition and luck, in spite of using a “rigorous” approach

- It has become clear in the CFD community that we have reached a ”idea saturation” stage

- Recent work has seen a shift toward somewhat greater complication in RANS systems, with mixed degree of success

➔Need to determine a large number of free parameters from a small set of test cases

➔Developed using canonical theories in idealized cases

➔Damage to convergence, uncertainty over boundary conditions, etc

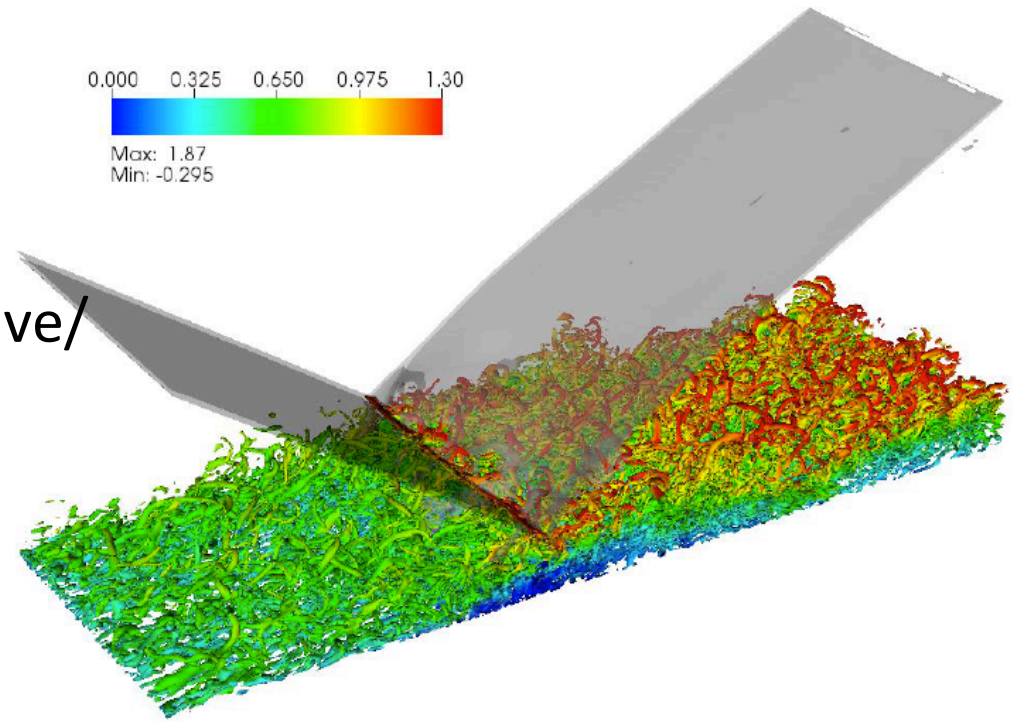
On the other hand..

- DNS and LES have been produced in quantity over the last decade.

- Experimental PIV and MRV data sets

- Data sets have not had a substantial impact on closure modeling

  ➔ DNS is more comprehensive/ accurate than earlier lab experiments but the impact on modeling has been additive, not transformative.



Why is it hard ?

Data

?

Predictive capability

Data doesn't exist in a form that is directly useful

Our approach

Recognize that traditional approach to model development does not leverage availability of massive amounts of data.

➔ Informing closure terms with a broader set of simulations can yield more universally-accurate simulations of turbulence

We propose large-scale data-driven techniques (based on inverse modeling and machine learning) to enable the construction of accurate models of turbulence

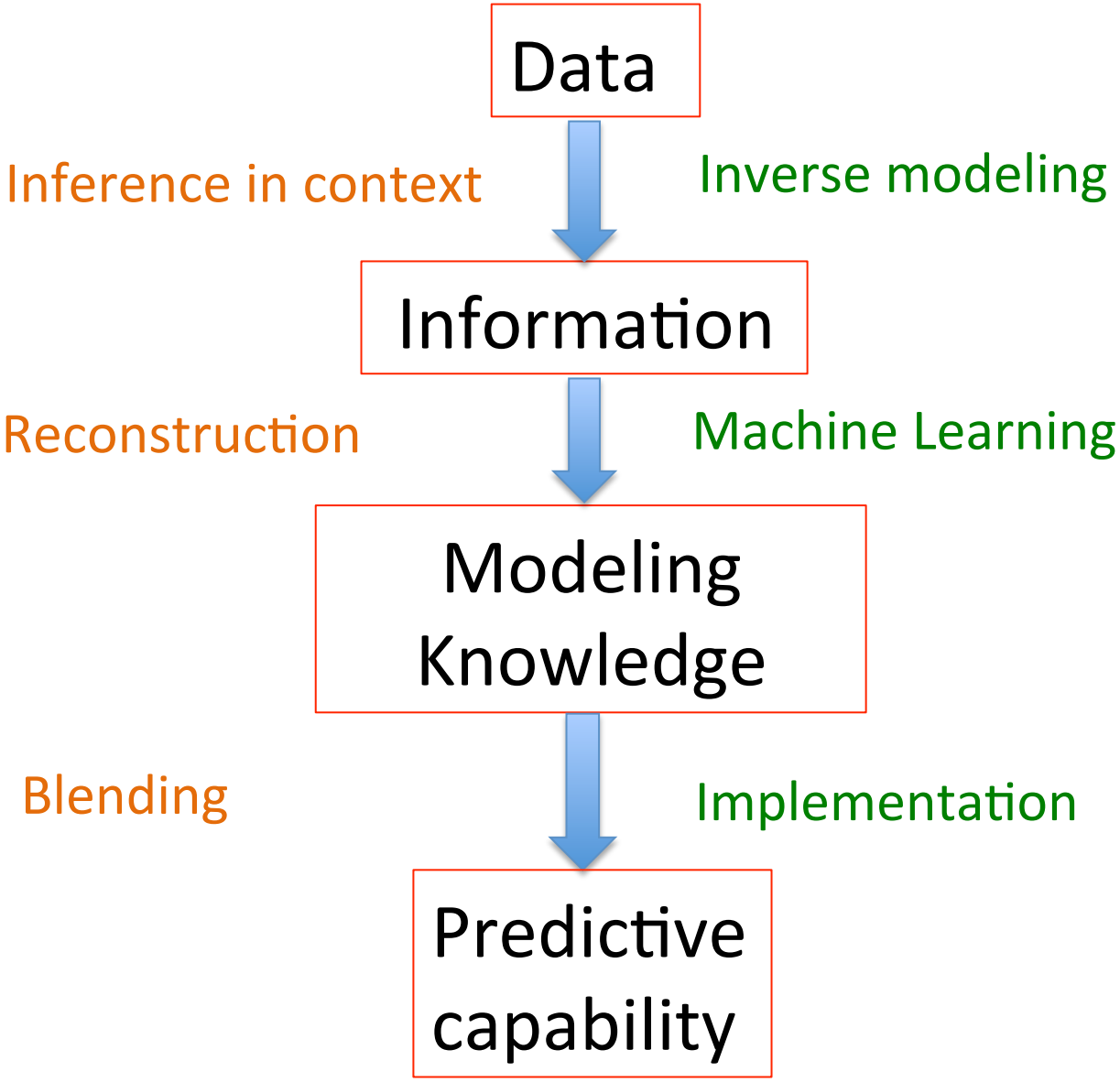
➔ Not replacing existing modeling knowledge, but just building on it

This technique enables

➔ The ability to "infer" what's missing in the closure

➔ The ability to convert that inference into modeling knowledge

Our approach



Questions to be answered

- Is there merit in the general idea?
- How to setup a properly-posed data-driven-turbulence-modeling problem ?
- What are the most effective ways to use Machine Learning approaches?
- What data (and how much data) is needed to improve the predictive capabilities?
- What are the new modeling techniques and algorithms that must be developed to make these approaches a reality?
- What improvements can be shown in a number of flows of interest?
- Once a model has been learned, how is it best embedded in an existing RANS solver?

Providing some fundamental answers to these questions is precisely what our Phase I NASA LEARN Project is about.

Key Innovations in the field of turbulence modeling

Quantitative information on model deficiencies

Functional forms of what the model is missing and not just parameters

A formalism to use data directly in model development

A framework to embed machine learned models in a predictive environment

Related Work

UQ Perspective:

Kennedy & O'Hagan (2000)
Dow & Wang (2010)
Tracey, Duraisamy, Alonso (2012)
Emory, Iaccarino & Larsson (2013)

Prediction Perspective:

Richard Dwight (2014)

All of these works are, however, focused on inferring model coefficients in one form or the other

Outline

- Motivation
- Is it even possible?
- Passive Learning from DNS and LES
- Active Learning from Improved RANS
- Demonstration
- Technology / Tools & Implementation
- Summary, Implications, Vision

Can we recreate the Spalart-Allmaras model from data?

$$\mu_t = \rho \hat{\nu} f_{v1}$$

$$\underbrace{\frac{\partial \hat{\nu}}{\partial t} + u_j \frac{\partial \hat{\nu}}{\partial x_j}}_{\text{Convection}} = \underbrace{c_{b1}(1-f_{t2})\hat{S}\hat{\nu}}_{\text{Production}} - \underbrace{\left(c_{w1} f_w - \frac{c_{b1}}{\kappa^2} f_{t2}\right)\left(\frac{\hat{\nu}}{d}\right)^2}_{\text{Destruction}} + \frac{1}{\sigma}\left(\underbrace{\frac{\partial}{\partial x_j}\left((\nu+\hat{\nu})\frac{\partial \hat{\nu}}{\partial x_j}\right)}_{\text{Diffusion}} + \underbrace{c_{b2}\frac{\partial \hat{\nu}}{\partial x_i}\frac{\partial \hat{\nu}}{\partial x_i}}_{\text{Cross Production}}\right)$$

Source = Production – Destruction + Cross Production

- Convection and diffusion from transport equations
- Source term is an input-output mapping:

Inputs:

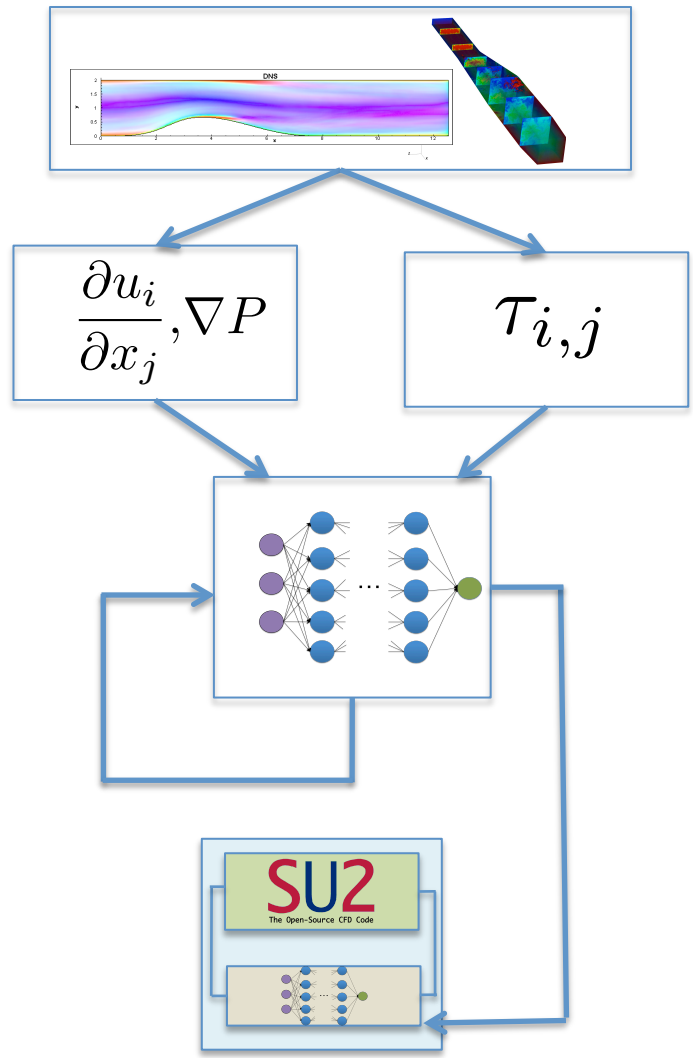
$$\nu,\ \hat{\nu},\ \Omega,\ d,\ N = \frac{\partial \hat{\nu}}{\partial x_i}\frac{\partial \hat{\nu}}{\partial x_i}$$

Output:

Source

Procedure

1) Select representative datasets
   - Flat plates, pressure-driven channels, airfoils
2) Choose and extract input and output features
   - Spalart-Allmaras quantities
3) Select learning algorithm
   - Neural Network
4) Train learning algorithm
   - BFGS optimizer
5) Embed learned model within flow solver
   - SU2



Challenges

A number of things can go wrong

- Errors in learning the training data
- Poor prediction on new locations
- Instabilities in PDE solver introduced by ML

Results

- We used this machinery to run >450 cases.
- Flat plate boundary layers, pressure-driven channels, Airfoils, ONERA Wing, etc

Features: Learn full source term: $\bar{s} = f(\bar{\Omega}, \chi, \bar{N})$

$$\underbrace{\frac{\partial \hat{\nu}}{\partial t} + u_j \frac{\partial \hat{\nu}}{\partial x_j}}_{\text{Convection}} = \underbrace{c_{b1}(1-f_{t2})\hat{S}\hat{\nu}}_{\text{Production}} - \underbrace{\left(c_{w1} f_w - \frac{c_{b1}}{\kappa^2} f_{t2}\right)\left(\frac{\hat{\nu}}{d}\right)^2}_{\text{Destruction}} + \frac{1}{\sigma}\left(\underbrace{\frac{\partial}{\partial x_j}\left((\nu+\hat{\nu})\frac{\partial \hat{\nu}}{\partial x_j}\right)}_{\text{Diffusion}} + \underbrace{c_{b2}\frac{\partial \hat{\nu}}{\partial x_i}\frac{\partial \hat{\nu}}{\partial x_i}}_{\text{Cross Production}}\right)$$

Results

- We used this machinery to run >450 cases.
- Flat plate boundary layers, pressure-driven channels, Airfoils, ONERA Wing, etc

Features: Learn full source term: $\bar{s} = f(\bar{\Omega}, \chi, \bar{N})$

$$\underbrace{\frac{\partial \hat{\nu}}{\partial t} + u_j \frac{\partial \hat{\nu}}{\partial x_j}}_{\text{Convection}} = \underbrace{\text{[redacted]}}_{\text{Production}} - \underbrace{\text{[redacted]}}_{\text{Destruction}} + \frac{1}{\sigma}\left(\underbrace{\frac{\partial}{\partial x_j}\left((\nu + \hat{\nu}) \frac{\partial \hat{\nu}}{\partial x_j} \right)}_{\text{Diffusion}} - \underbrace{\text{[redacted]}}_{\text{Cross Production}} \right)$$

Input features

Dimensional Features

$$\nu,\ \hat{\nu},\ \Omega,\ d,\ N = \frac{\partial \hat{\nu}}{\partial x_i}\frac{\partial \hat{\nu}}{\partial x_i}$$

Locally Non-dim

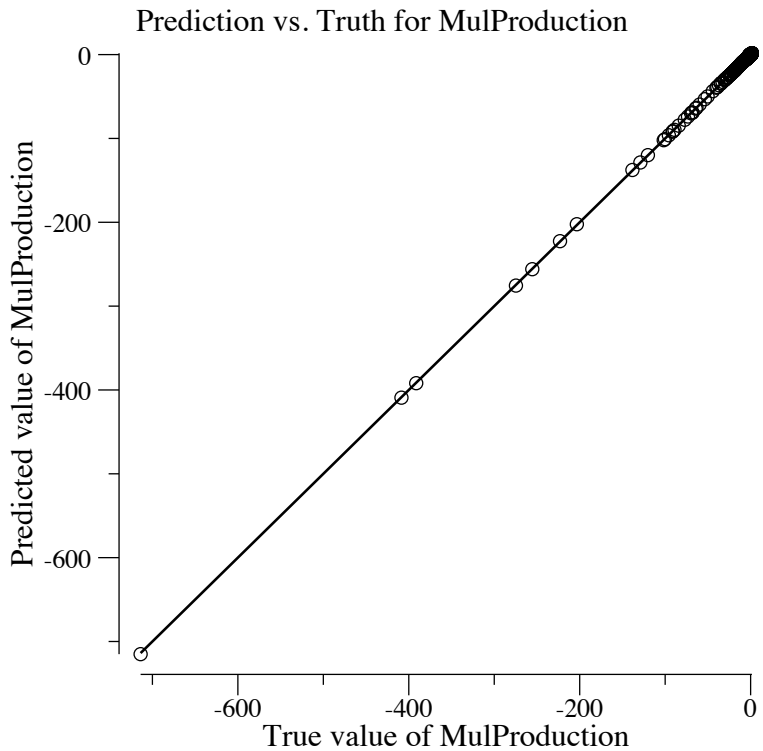
$$\chi = \hat{\nu}/\nu$$

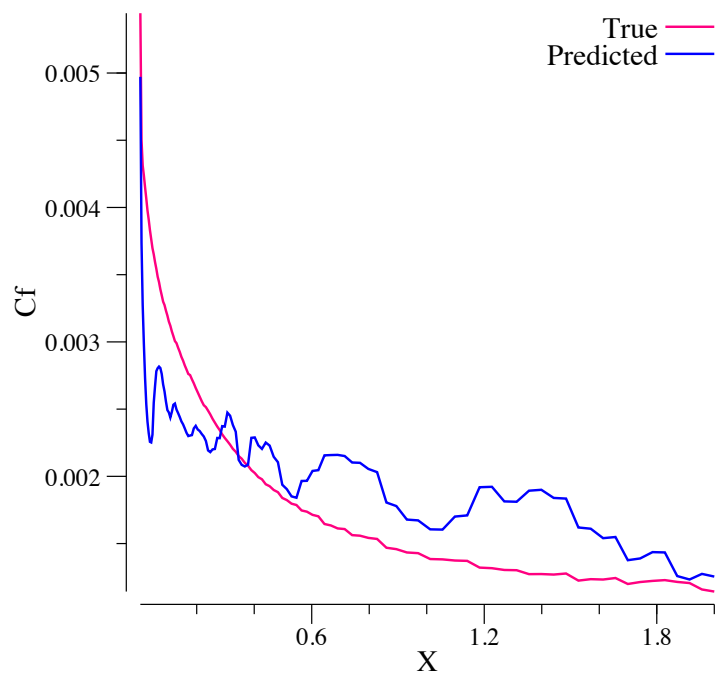
$$\bar{\Omega} = \frac{d^2}{\hat{\nu} + \nu}\Omega$$

$$\bar{N} = \frac{d^2}{(\hat{\nu} + \nu)^2}N$$

Misleading results

Favorable pressure gradient channel flow





Must test within a CFD solver

Loss function during training

- Loss function describes the quality of a prediction during the training
- Squared-error

$$L = \sum_{i=1}^{k} (p_i - t_i)^2$$

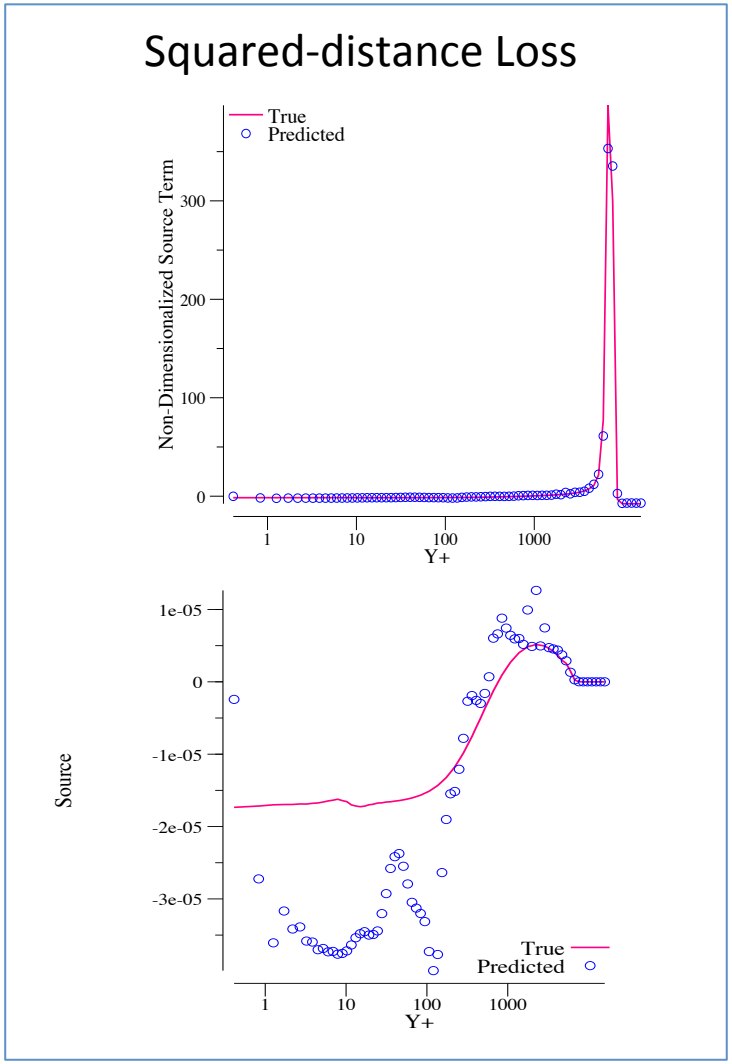
Does not appropriately Reflect the cost of misprediction.

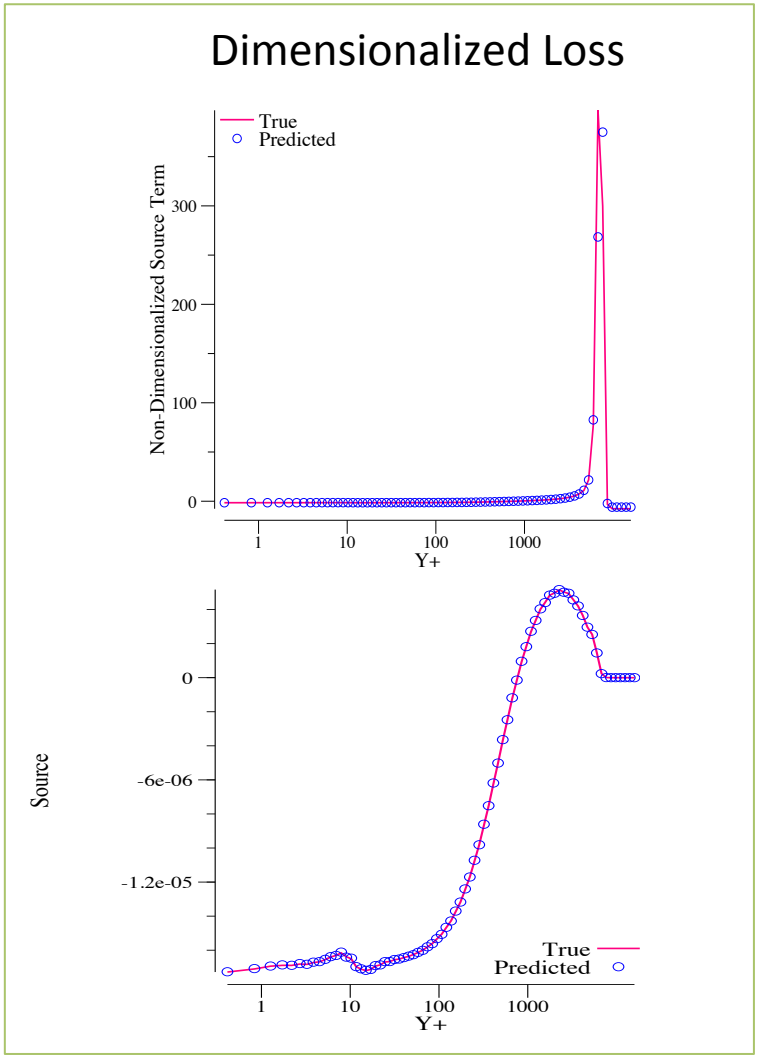
- Dimensional squared-error

$$L_2 = \sum_{i=1}^{k} \left(\left(\frac{d_i^2}{(\hat{\nu}_i + \nu_i)^2} \right) p_{\bar{s},i} - t_{s,i} \right)^2$$

penalizes differences in the quantity that is relevant to the PDE solver

Results





Loss function needs to align with CFD

Summary of feasibility study

- Demonstrated procedure for creating a machine-learned turbulence model
- Successfully replicated Spalart-Allmaras turbulence model
- Good agreement seen on flows not in the training set
- Loss function selection is critical for prediction quality
- Must test model in PDE solver to evaluate performance

More results in:
A Machine Learning Strategy to Assist Turbulence Model Development (AIAA 2015-1287)

Outline

- Motivation
- Is it even possible? (Definitely not impossible)
- Passive Learning from DNS and LES
- Active Learning from Improved RANS
- Demonstration
- Summary, Implications, Vision

Passive Learning

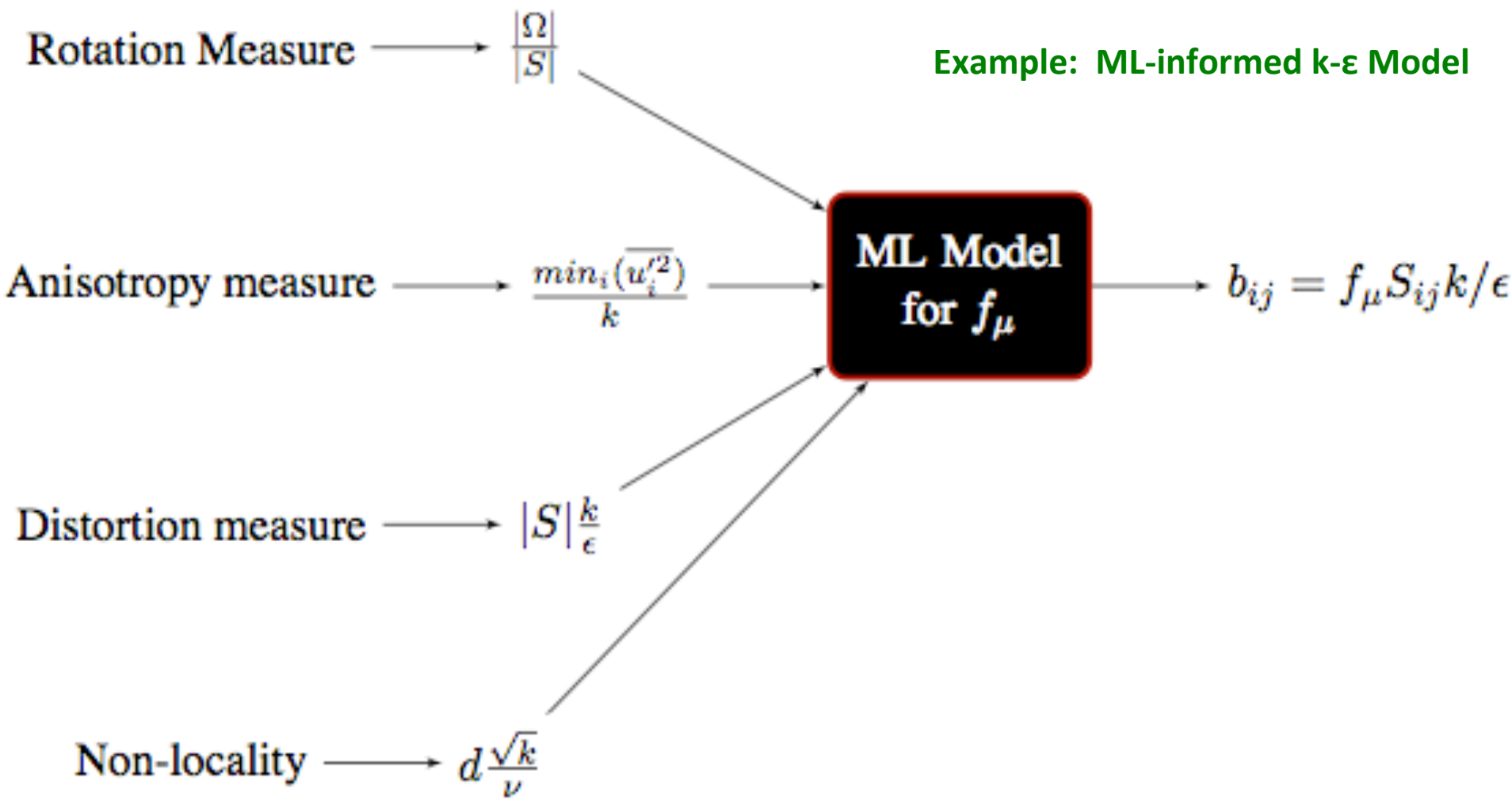
Can we directly learn the functional form of modeling terms from DNS/ LES Data?

Example: ML-informed k-ε Model

$$\frac{\overline{u_i' u_j'}}{2k} - \frac{1}{3}\delta_{ij} = b_{ij} = f_\mu S_{ij} k/\epsilon$$

Passive Learning

Can we directly learn the functional form of modeling terms from DNS/ LES Data?



Example: ML-informed k-ε Model

$$|b_{ij,data} - f_\mu S_{ij} k/\epsilon|$$

- ε will be highly erroneous in a RANS calculation (and to an extent, k as well)

- ε cannot be ”measured” accurately even in an LES calculation

- Need RANS inputs

Need dependent variables that are more intimately related to data within modeling context

Outline

- Motivation
- Passive Learning from DNS and LES (Not very effective)
- Active Learning from Improved RANS
- Demonstration
- Technology / Tools & Implementation
- Summary, Implications, Vision

A model that interacts with data

$$R(Q) = \nabla.\tau$$

RANS Equations

$$\tau = E(Q, k, \omega, \gamma, ...)$$

Turbulence model

$$R(Q) = \nabla.\tau$$

$$\tau = E(Q, k, \omega, \gamma, \alpha, ..)$$

Introduce field variable α(x) (Not a parameter)

$$G(Q)$$

Choose Qty of Interest

$$\min_{\alpha} J = ||G(\alpha) - G_{exp}||_2$$

Solve inverse problem for field variable

Application to SA Model

Original Model

$$\frac{D\tilde{\nu}}{Dt} = P - D + T$$

Introduce correction

$$\frac{D\tilde{\nu}}{Dt} = P - \alpha D + T$$

Objective function

$$G = \int_s [\tau_w^{data} - \tau_w^{model}]^2 ds$$

Application to SA Model

Original Model

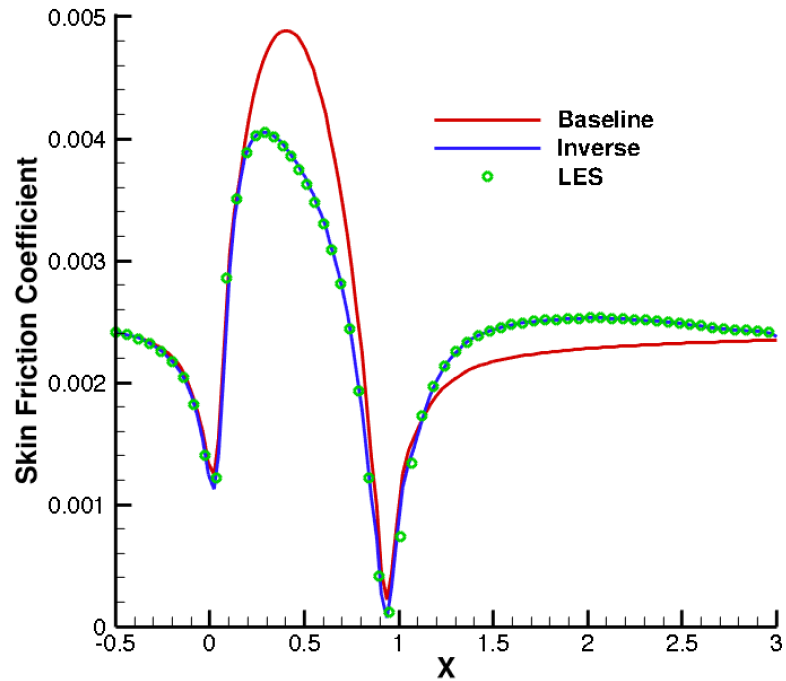
$$\frac{D\tilde{\nu}}{Dt} = P - D + T$$

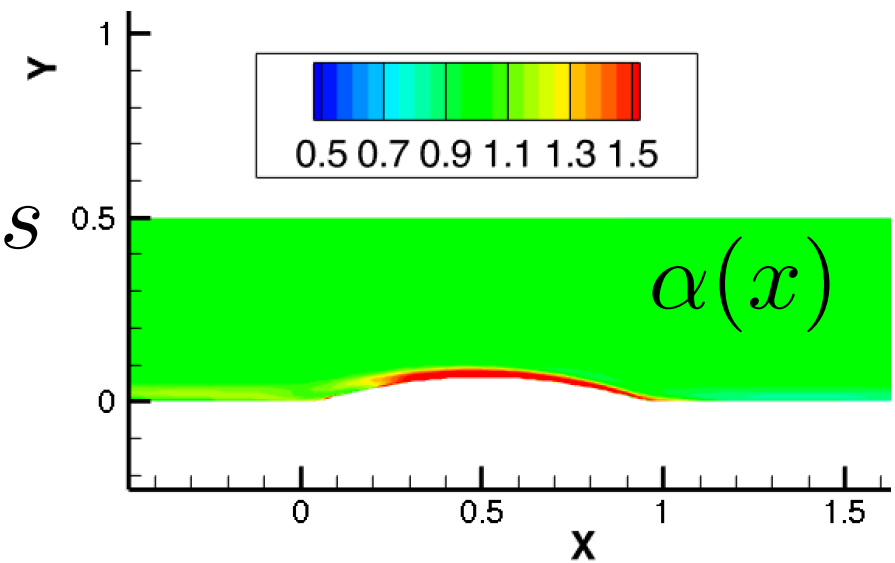
Introduce correction

$$\frac{D\tilde{\nu}}{Dt} = P - \alpha D + T$$

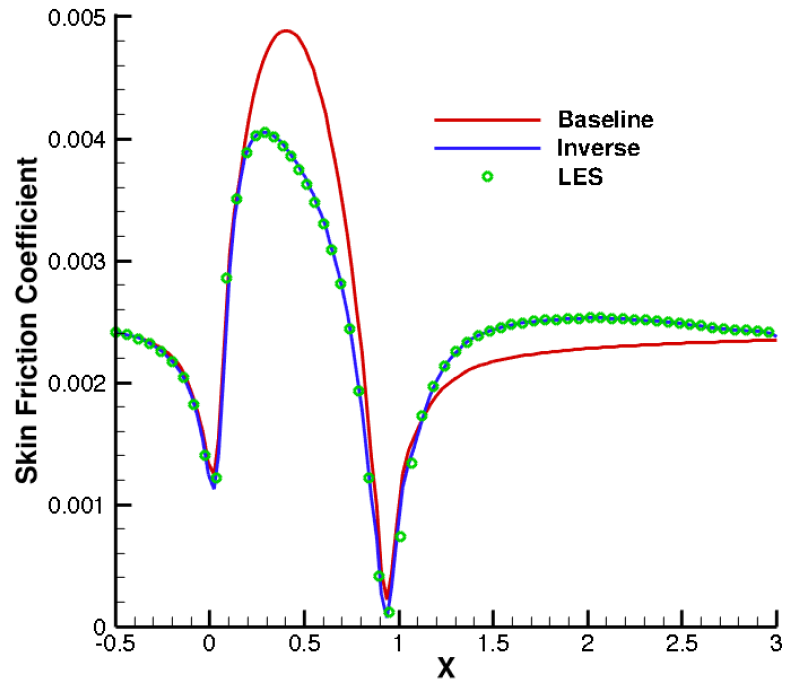
Objective function

$$G = \int_s [\tau_w^{data} - \tau_w^{model}]^2 ds$$



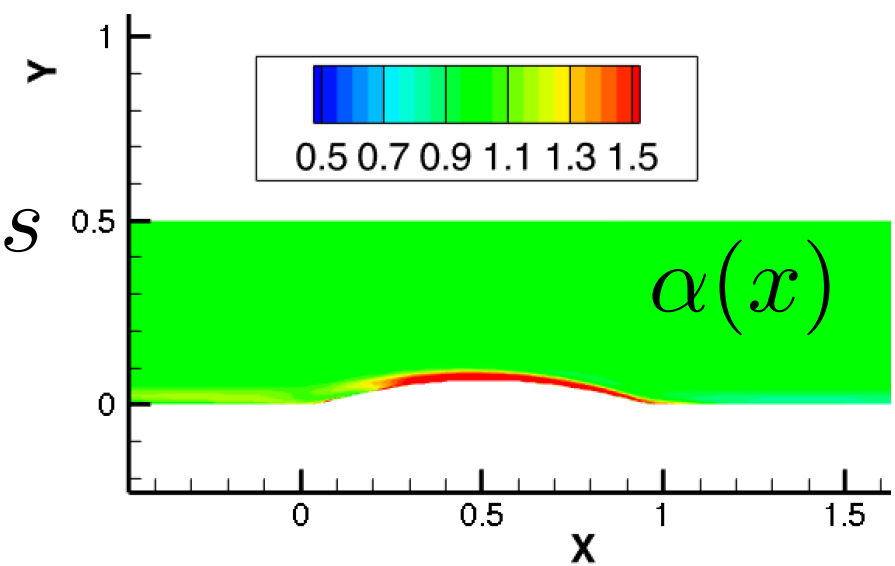


Application to SA Model



$$\frac{D\tilde{\nu}}{Dt} = P - \alpha D + T$$

$$G = \int_s [\tau_w^{data} - \tau_w^{model}]^2 ds$$



Now we know what was missing in the model. Again, α(x) is not just a parameter

Machine Learning

- $\alpha(x)$ field by itself is of limited use in quantitative modeling
- Machine learning techniques used to extract the functional relationship $\alpha(x) \approx \alpha(\eta)$
  - ➔ Gaussian Process Regression
  - ➔ Neural Networks
- 10-fold cross-validation used to narrow down hyper-parameters
- More details on ML algorithms in the paper

Active Learning

Passive approach: Learn from DNS/LES

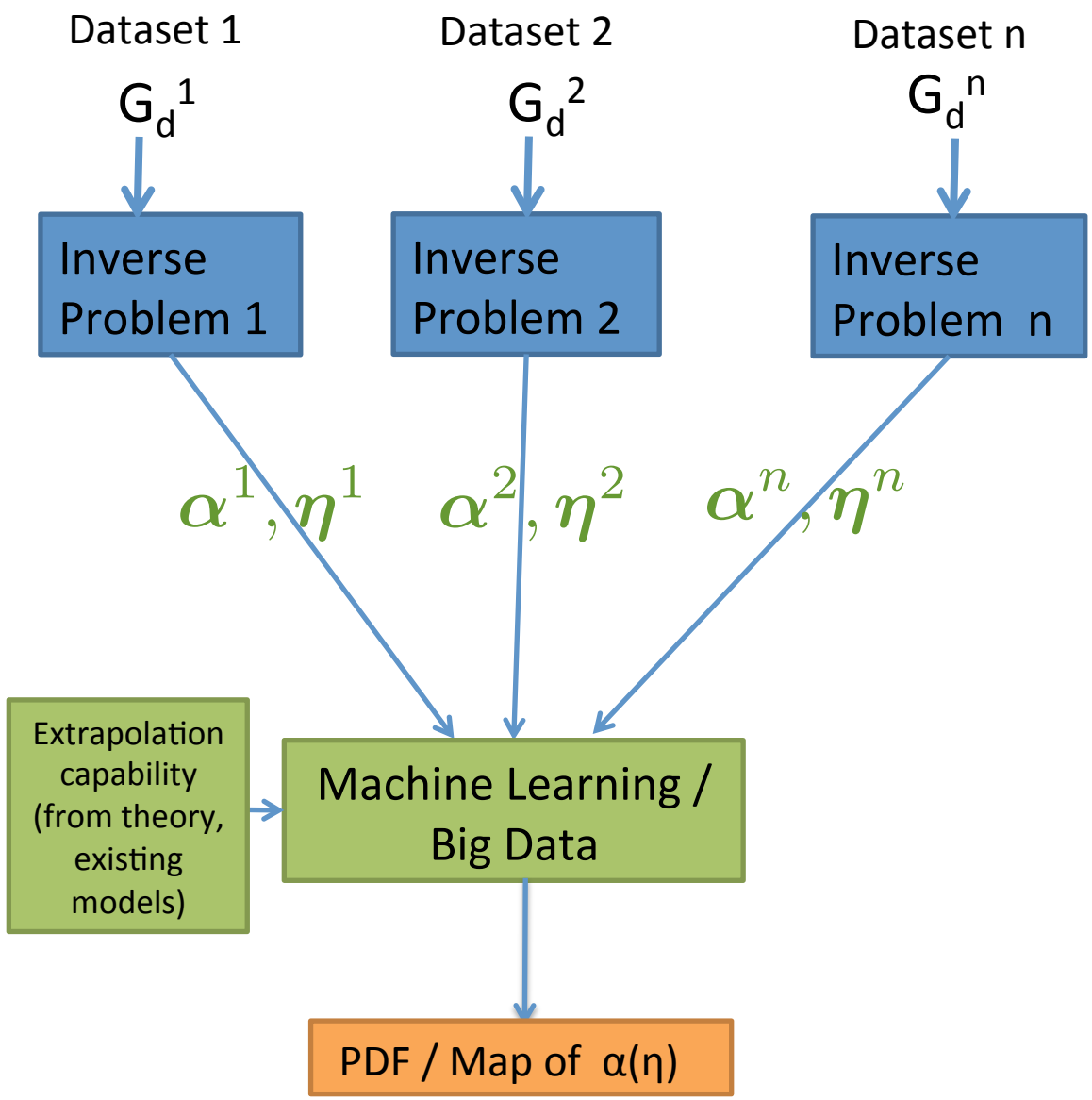
Active approach: Improve RANS; then learn from RANS

Consistent, Realizable !

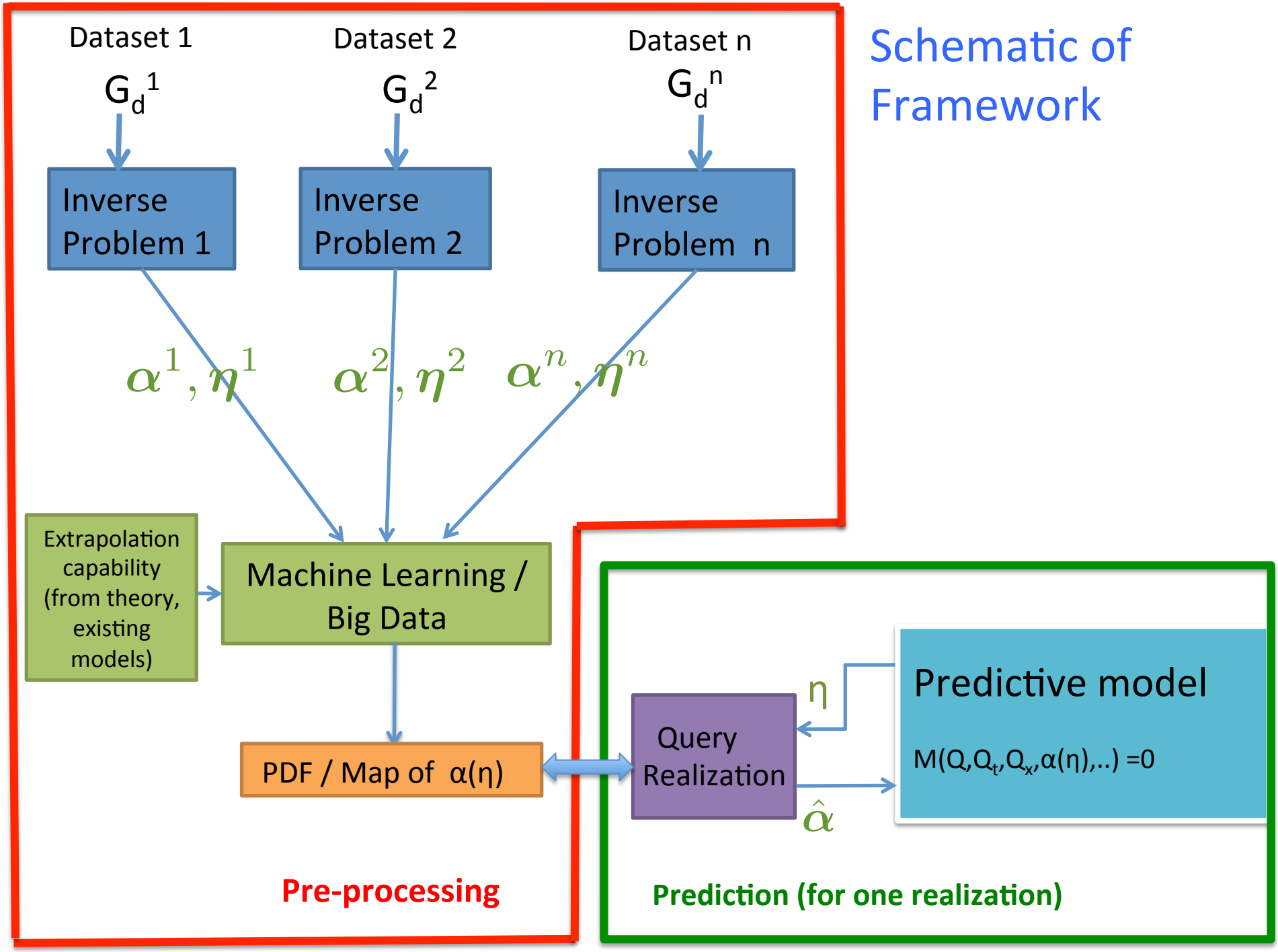
No need for full field of data!

Can work with experimental data!

Schematic of Framework



Schematic of Framework



Outline

- Motivation
- Is it even possible ?
- Passive Learning from DNS and LES
- Active Learning from Improved RANS
- Demonstration
- Technology / Tools & Implementation
- Summary, Implications, Vision

Application to Bypass transition

- Models of bypass transition in general CFD codes are a relatively recent development (Menter 2006)
- Bypass process occurs as turbulence diffuses into the laminar boundary layer, and generates disturbances known as Klebanoff modes

Intermittency transport-based model [Ge et al. (2014)]

$$\frac{Dk}{Dt} = 2\nu_T |S|^2 \gamma - C_\mu k\omega + \partial_j \left[\left(\nu + \frac{\nu_T}{\sigma_k} \right) \partial_j k \right]$$

$$\frac{D\omega}{Dt} = 2C_{\omega 1} |S|^2 - C_{\omega 2}\omega^2 + \partial_j \left[\left(\nu + \frac{\nu_T}{\sigma_\omega} \right) \partial_j \omega \right]$$

$$\frac{D\gamma}{Dt} = \partial_j \left[\left(\frac{\nu}{\sigma_l} + \frac{\nu_T}{\sigma_\gamma} \right) \partial_j \gamma \right] + P_\gamma - E_\gamma$$

Application to Bypass transition

- Models of bypass transition in general CFD codes are a relatively recent development (Menter 2006)
- Bypass process occurs as turbulence diffuses into the laminar boundary layer, and generates disturbances known as Klebanoff modes

Intermittency transport-based model [Ge et al. (2014)]

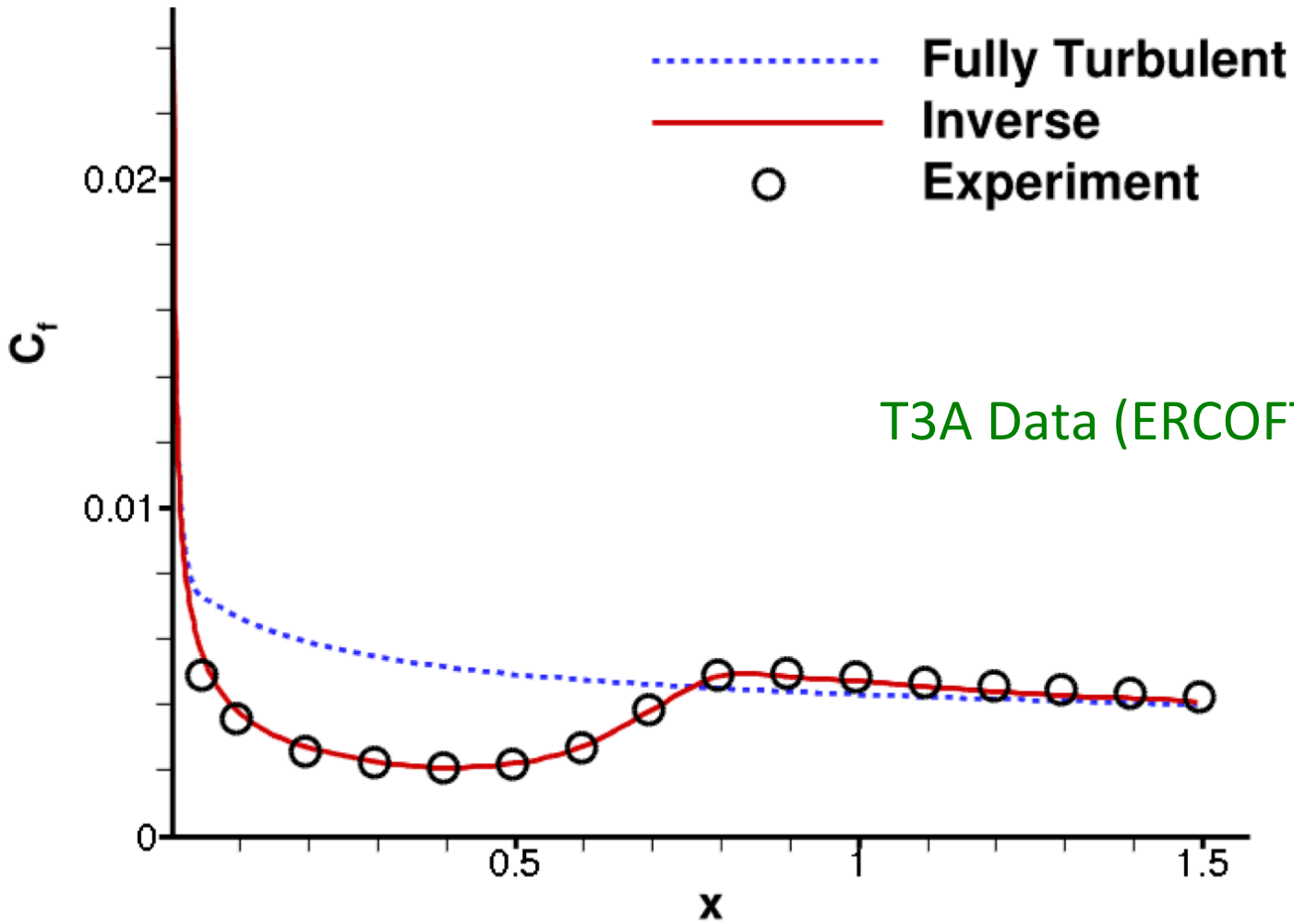
$$\frac{Dk}{Dt} = 2\nu_T |S|^2 \gamma - C_\mu k\omega + \partial_j \left[\left(\nu + \frac{\nu_T}{\sigma_k} \right) \partial_j k \right]$$

$$\frac{D\omega}{Dt} = 2C_{\omega 1} |S|^2 - C_{\omega 2}\omega^2 + \partial_j \left[\left(\nu + \frac{\nu_T}{\sigma_\omega} \right) \partial_j \omega \right]$$

$$\cancel{\frac{D\gamma}{Dt} = \partial_j \left[\left(\frac{\nu}{\sigma_l} + \frac{\nu_T}{\sigma_\gamma} \right) \partial_j \gamma \right] + P_\gamma - E_\gamma}$$

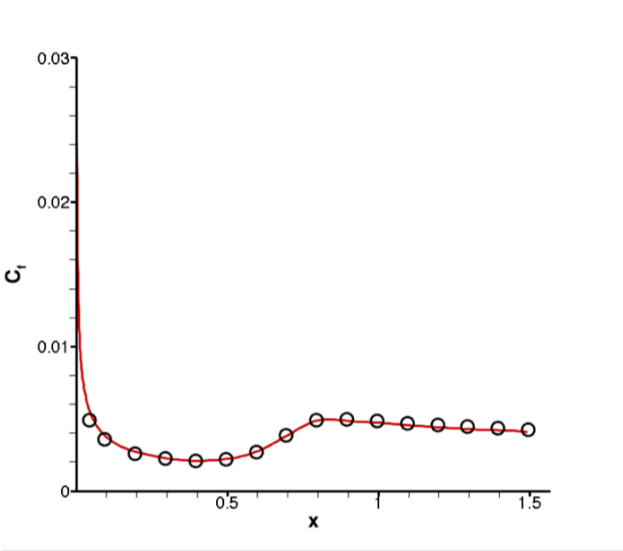
How about inferring the intermittency field from data?

$$\mathbf{G}(\gamma) = \int_w \left[\tau_w^{data}(s) - \tau_w^{model}(s)\right]^2 dS,$$

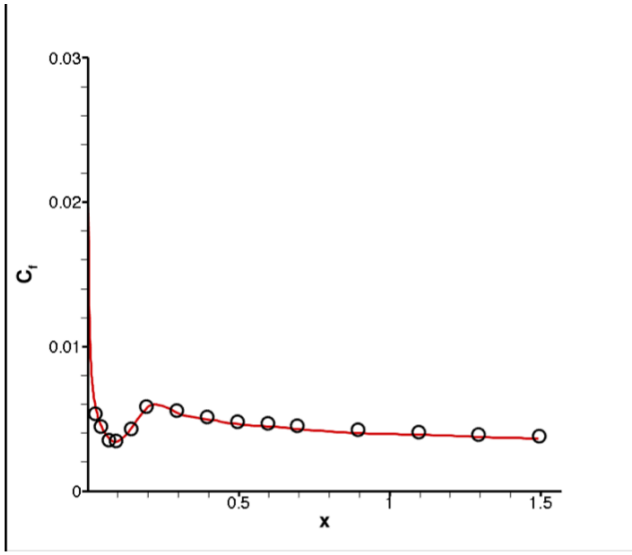


T3A Data (ERCOFTAC)

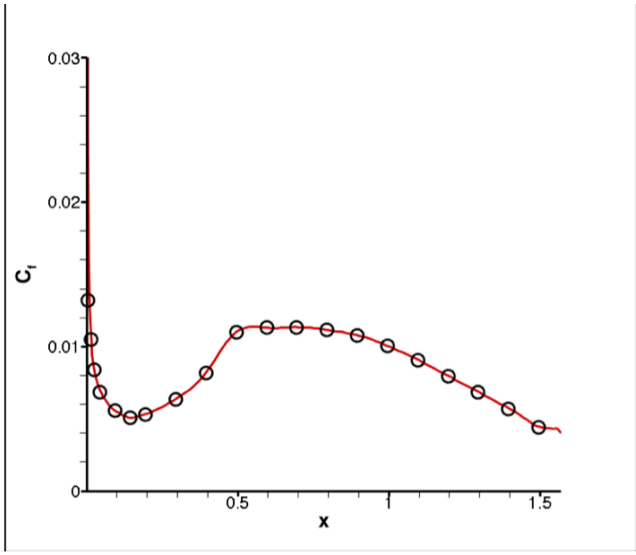
Full set of test cases



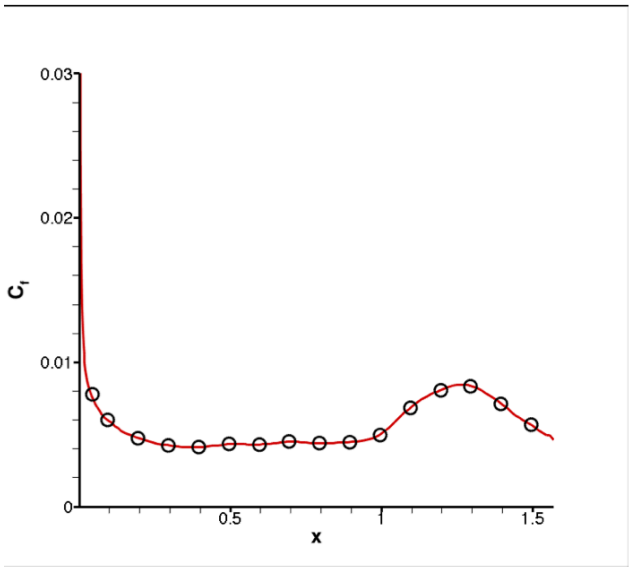
(a) T3A



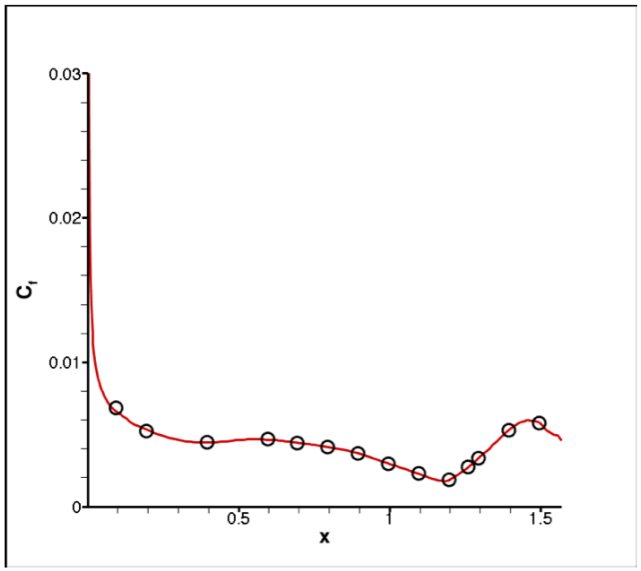
(b) T3B



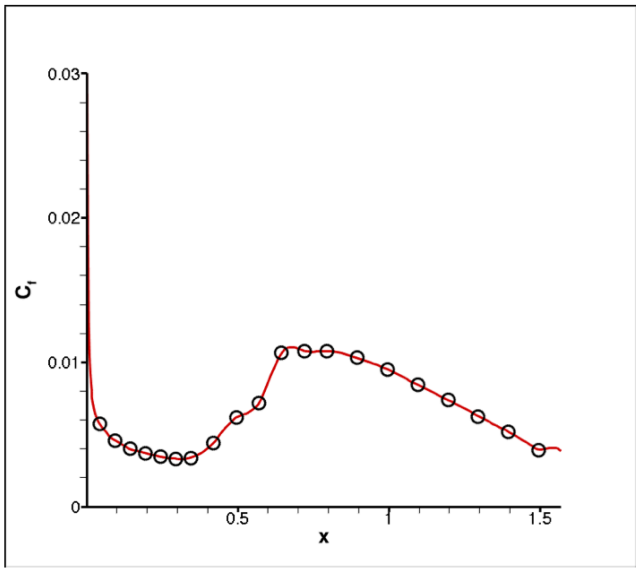
(c) T3C1



(d) T3C2

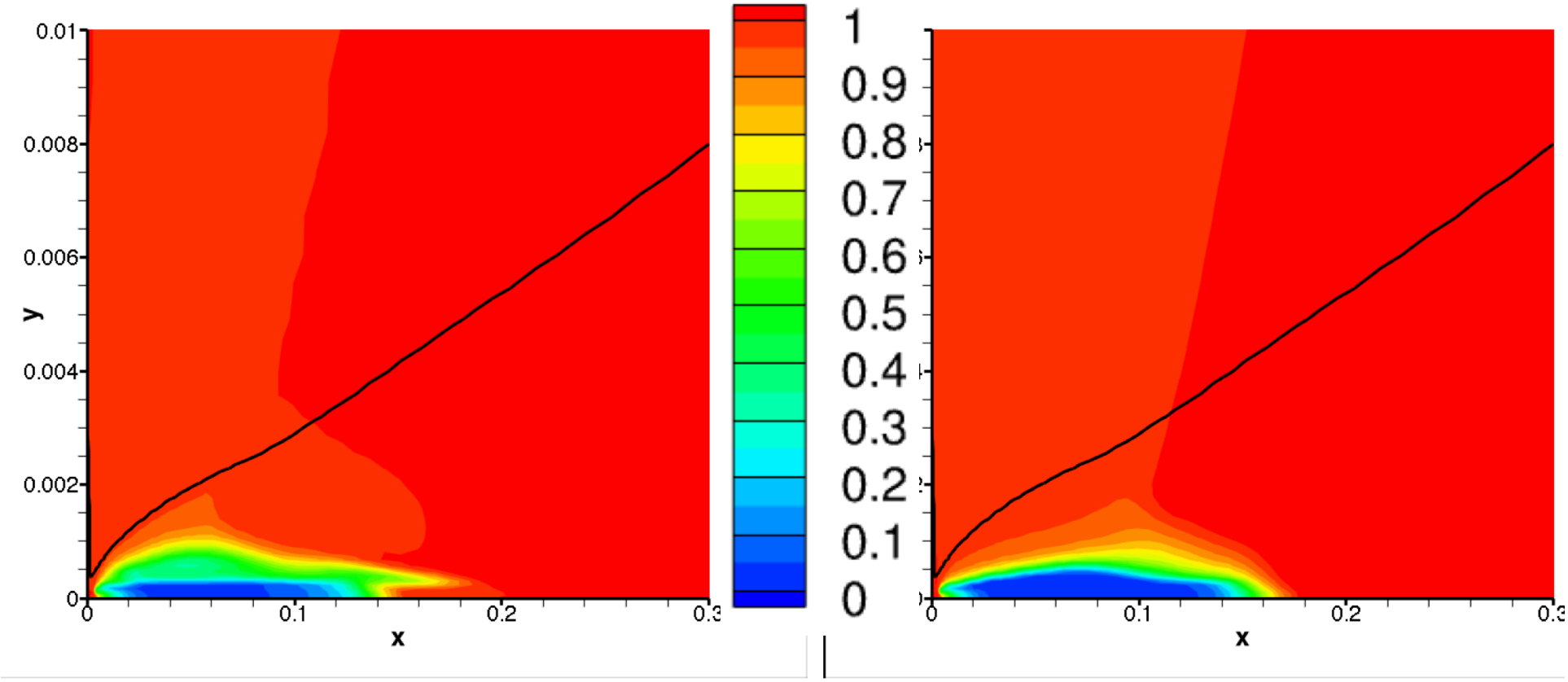


(e) T3C3



(f) T3C5

Inferred intermittency vs Ge et al. model



(a) T3B: Optimal γ (b) T3B: Model γ

Very useful info for the modeler

OK, but how to build a model ?

Assume that budget holds

$$\frac{D\gamma}{Dt} = \partial_j \left[\left(\frac{\nu}{\sigma_l} + \frac{\nu_T}{\sigma_\gamma} \right) \partial_j \gamma \right] + S_\gamma$$

Extract source term

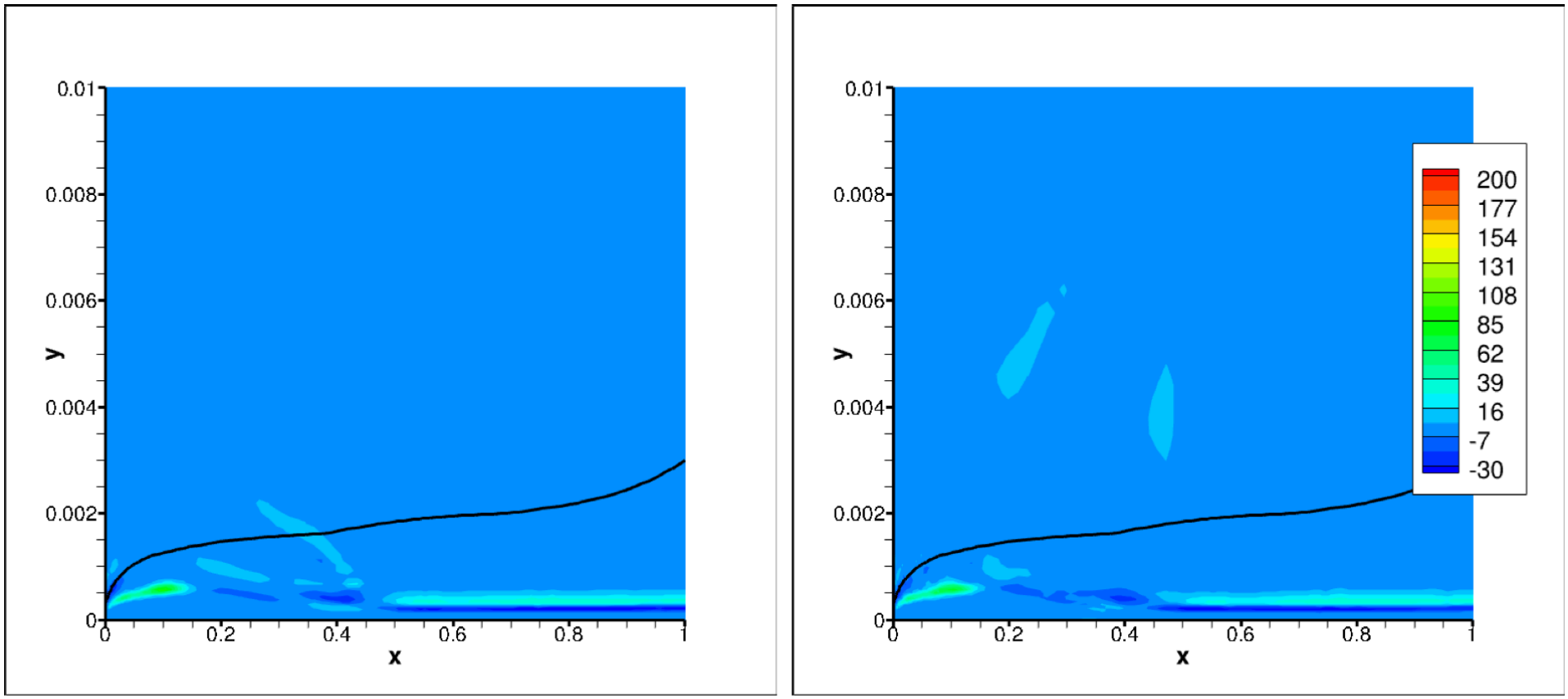
$$S_\gamma = \frac{D\gamma}{Dt} - \partial_j \left[\left(\frac{\nu}{\sigma_l} + \frac{\nu_T}{\sigma_\gamma} \right) \partial_j \gamma \right]$$

Reconstruct

$S_\gamma(\mathbf{x})$ → Machine Learning → $\boldsymbol{\eta}$

Machine Learning → $S_\gamma(\boldsymbol{\eta})$

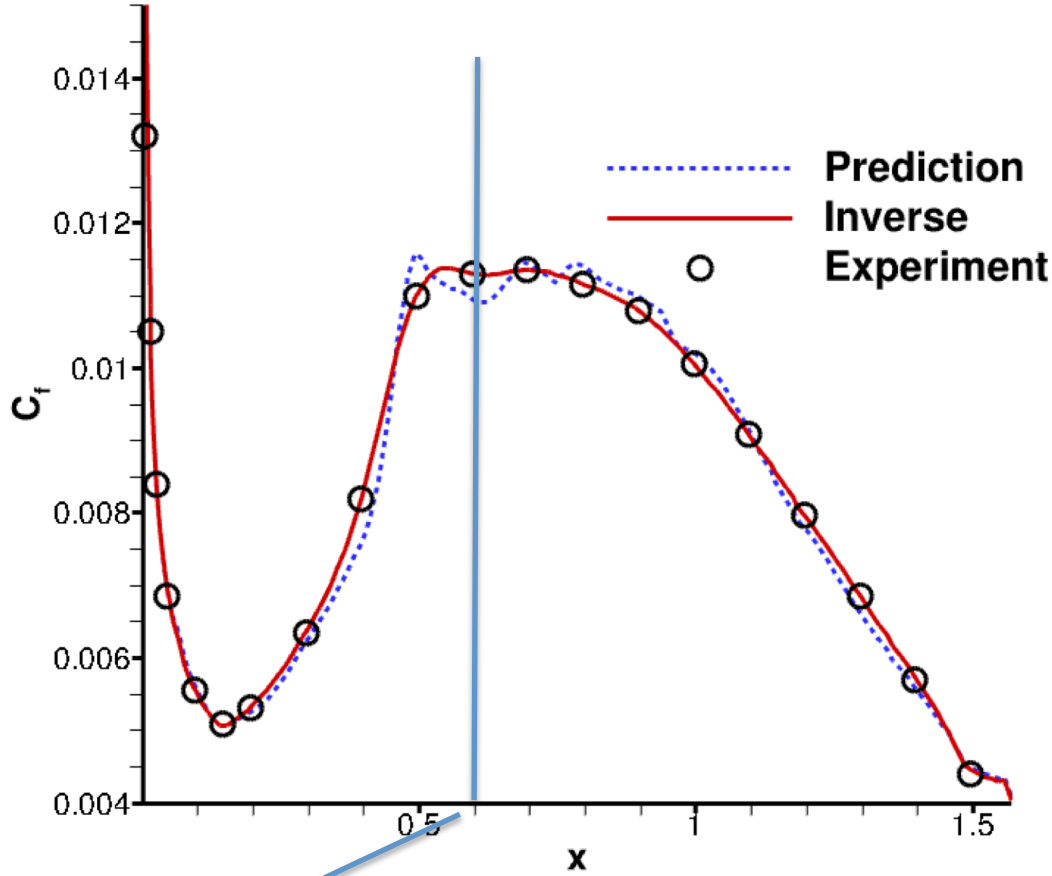
Machine Learning in action : T3C1

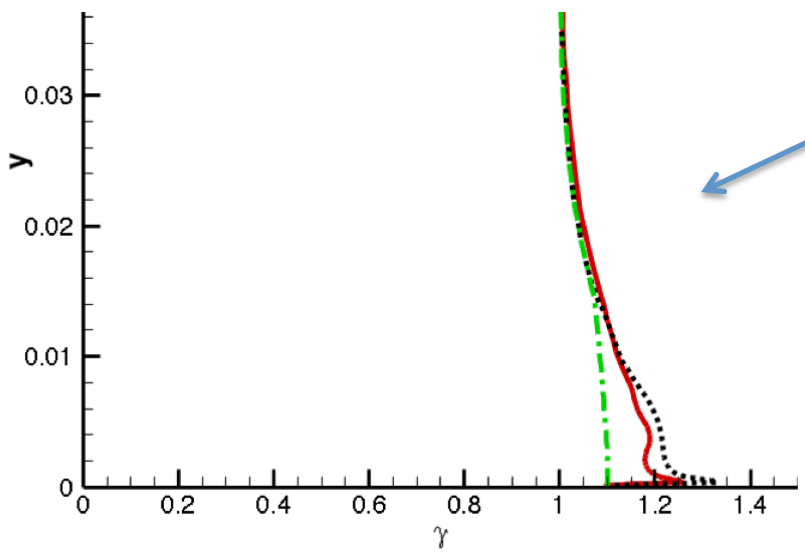


(a) Inferred Source term (b) Predicted Source term

$$\boldsymbol{\eta} = \{k, \omega, \gamma, \frac{\partial u_i}{\partial x_j}, \nu\}$$

Data Infusion : T3C1

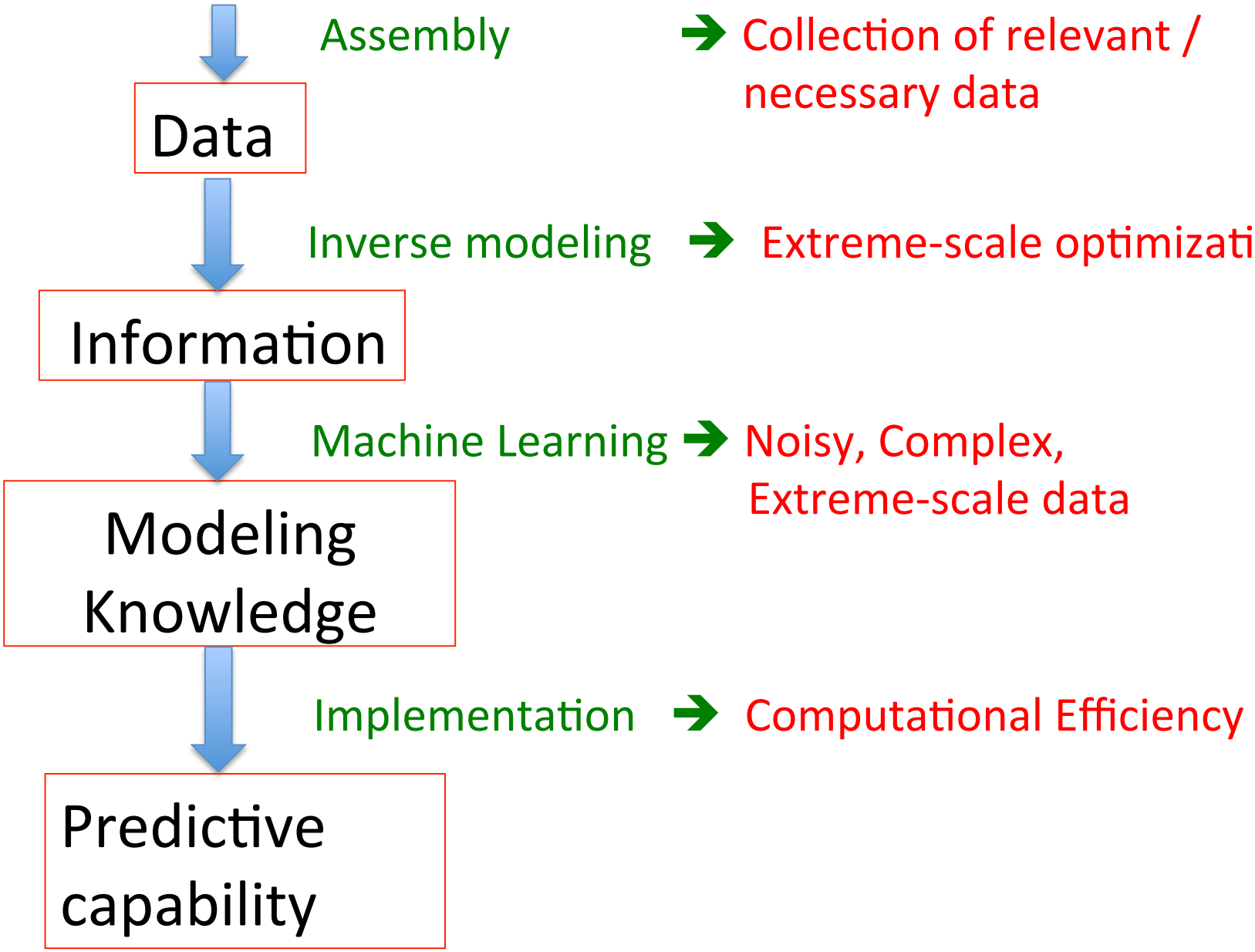




Outline

- Motivation (We need it!)
- Is it even possible ? (At least not impossible)
- Passive Learning from DNS and LES (Not very effective)
- Active Learning from Improved RANS (Very promising)
- Demonstration (It works reasonably well)
- Technology / Tools & Implementation (Requires a lot of work)
- Summary, Implications, Vision

Challenges


Assembly
Collection of relevant / necessary data
Data
Inverse modeling
Extreme-scale optimization
Information
Machine Learning
Noisy, Complex, Extreme-scale data
Modeling Knowledge
Implementation
Computational Efficiency
Predictive capability

Inverse modeling

$$\underset{\alpha}{min} \; J = ||G(\alpha) - G_{data}|| + \epsilon ||\nabla \alpha||$$

Very high dimensional minimization problem

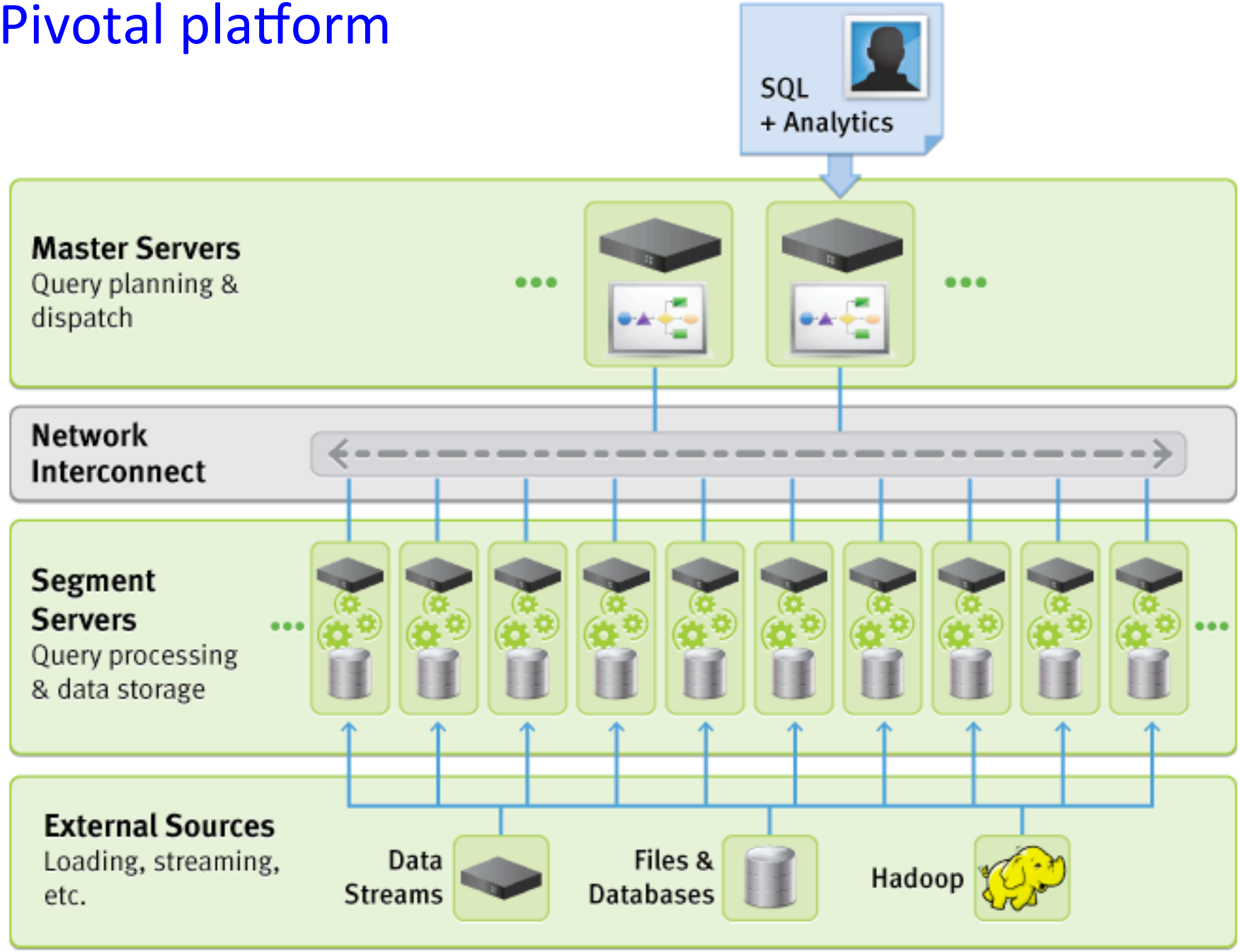
L-BFGS Quasi-Newton optimizer

Gradients from discrete adjoint

Machine Learning Requirements

- Highly multidimensional
- Since learning is in feature space, very highly multi-scale (coarse & rich)
- Multiscale learning is an active research area
- The training stage requires solution of a large ill-posed linear system of algebraic equations
- Regularization and speedups of solution can be achieved via employment of methods for efficient complexity reduction, including
  - Construction of compact bases via data structures
  - Nystrom methods (low-rank approximations)
  - Preconditioned iterative procedures
  - Specially designed Krylov subspace methods
- The test stage requires fast procedures for large matrix-vector products
- Computation of predictive variance can be also done efficiently using low-rank decompositions

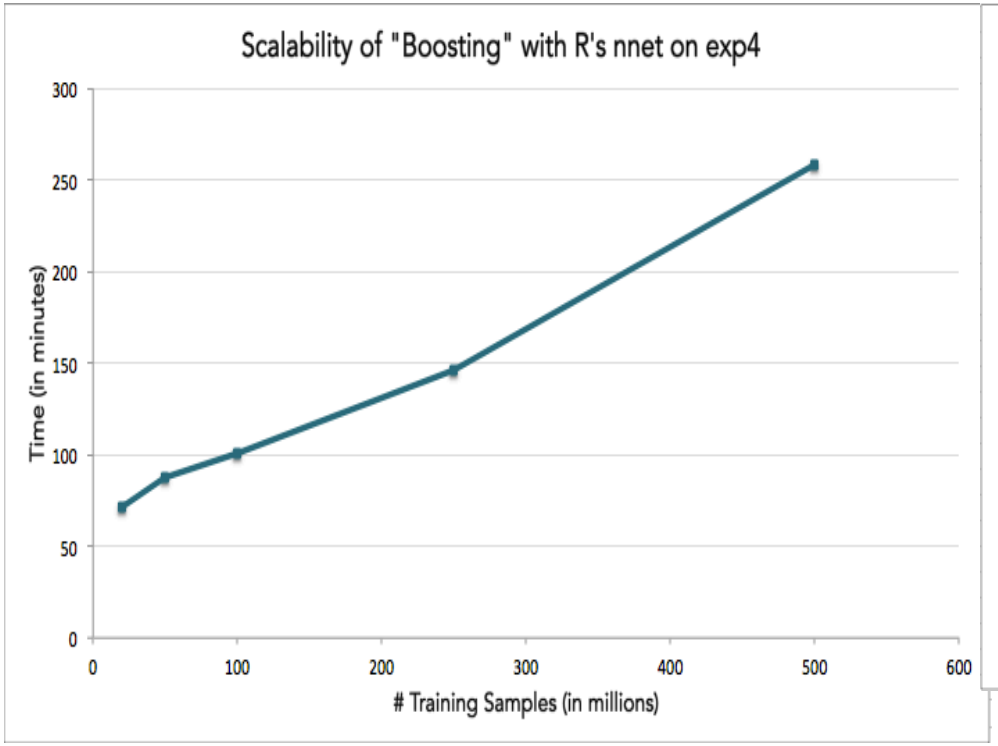
Pivotal platform



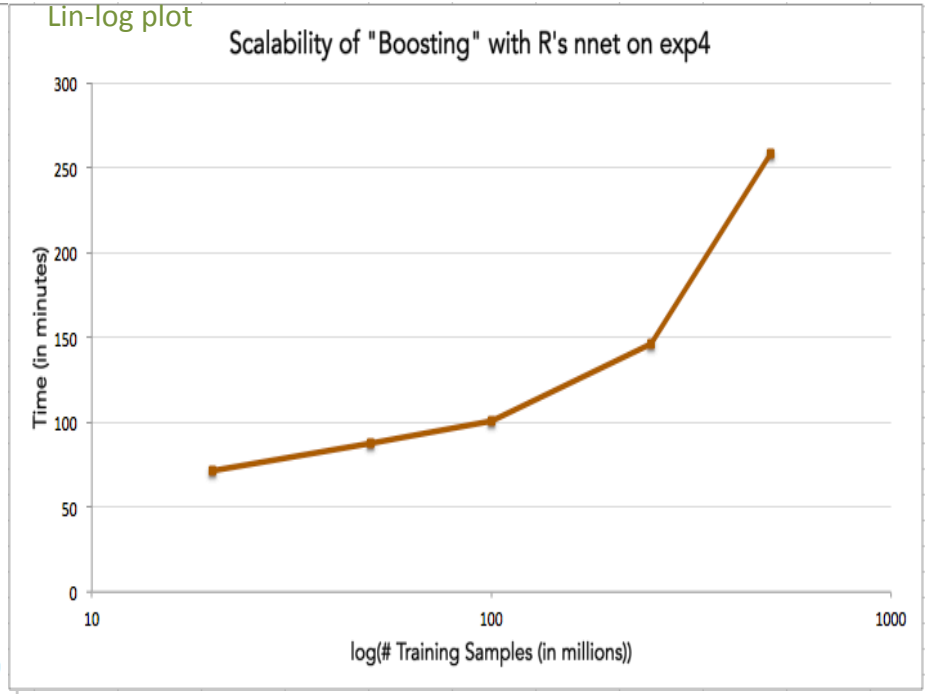
Scalability of Parallelized NN Implementation

Linearly scalable as data set scales up

- Boosting implementation proved to be linearly scalable up to at least 500 million rows of data
- Tests done on duplicated LES dataset
  - Model fits stayed relatively the same (no improvement due to no new data)



Lin-log plot



Multiscale GP : Model

input noise weights basis functions

output

$$y = F(\mathbf{x}) + \epsilon, \quad F(\mathbf{x}) = \boldsymbol{\phi}(\mathbf{x})^T \mathbf{w} = \sum_{i=1}^{D} w_i \phi_i(\mathbf{x}), \quad \epsilon \sim \mathcal{N}(0, \sigma^2),$$

feature space design matrix noise variance

$$\boldsymbol{\phi}(\mathbf{x}) : \quad \mathbb{R}^d \to \mathbb{R}^D, \quad \Phi = \{\Phi_{ij}\}, \quad \Phi_{ij} = \phi_i(\mathbf{x}_j), \quad i = 1, \ldots, D, \quad j = 1, \ldots, n,$$

extended feature space training point

$$\phi_i(\mathbf{x}) = \exp\left(-\frac{|\mathbf{x} - \mathbf{x}_i|^2}{h_i^2} \right), \quad i = 1, \ldots, D.$$

scale of each basis function

Multiscale GP : Solution

maximum a posteriori (MAP) estimate of **w**

covariance matrix

identity matrix

$$\overline{\mathbf{w}} = \frac{1}{\sigma^2}\Sigma\Phi\mathbf{y}, \quad \Sigma^{-1} = \frac{1}{\sigma^2}\Phi\Phi^T + I_D,$$

log marginal likelihood

$$LML = -\frac{1}{2\sigma^2}(\mathbf{y} - \Phi^T\overline{\mathbf{w}})^T\mathbf{y} - \frac{1}{2}\log\left|\frac{1}{\sigma^2}\Phi\Phi^T + I\right| - \frac{n}{2}\log(2\pi\sigma^2).$$

predictive mean

$$m_* = \boldsymbol{\phi}_*^T\overline{\mathbf{w}}, \quad \sigma_*^2 = \sigma^2 + \boldsymbol{\phi}_*^T\Sigma\boldsymbol{\phi}_*, \quad \boldsymbol{\phi}_* = \boldsymbol{\phi}(\mathbf{x}_*).$$

test point

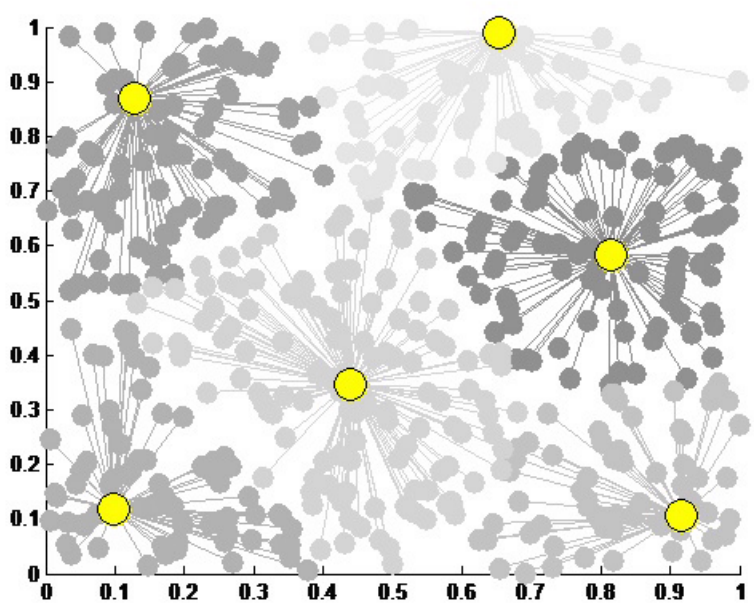
predictive variance

Construction of reduced bases

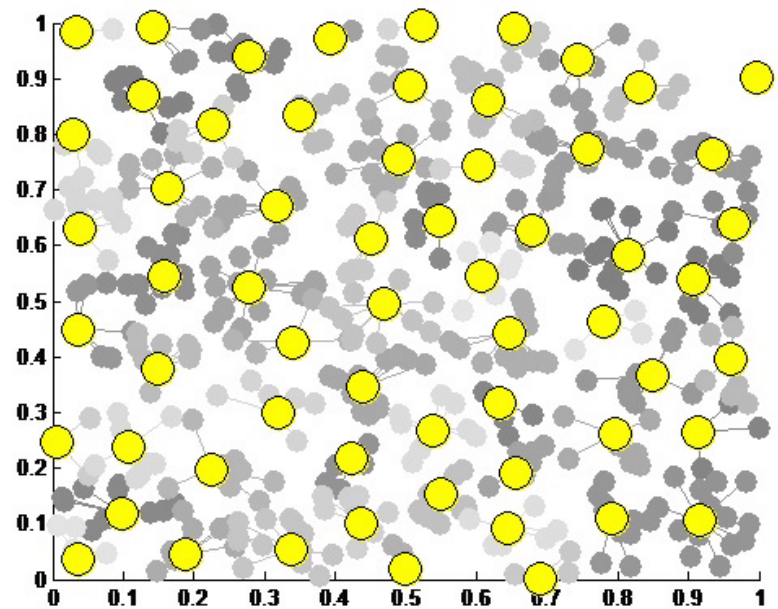
- Given training set, X, usually there exists its subset, B, such that for a required accuracy and noise level, basis function centered at some x can be expanded over the centers from B
- The centers of B should be distributed in a way that for a given scale, h, the distance between them is not less than some a, $a/h < 1$
- The distance from any point x to the closest center is not more than a
- Parameter a should be chosen based on the tolerance and is a subject for optimization

d-Dimensional *k*-Center Type Space Partitioning (Clustering)

n = 500 uniform random distribution in [0,1] x [0,1]



a = 0.4 (k = 6)

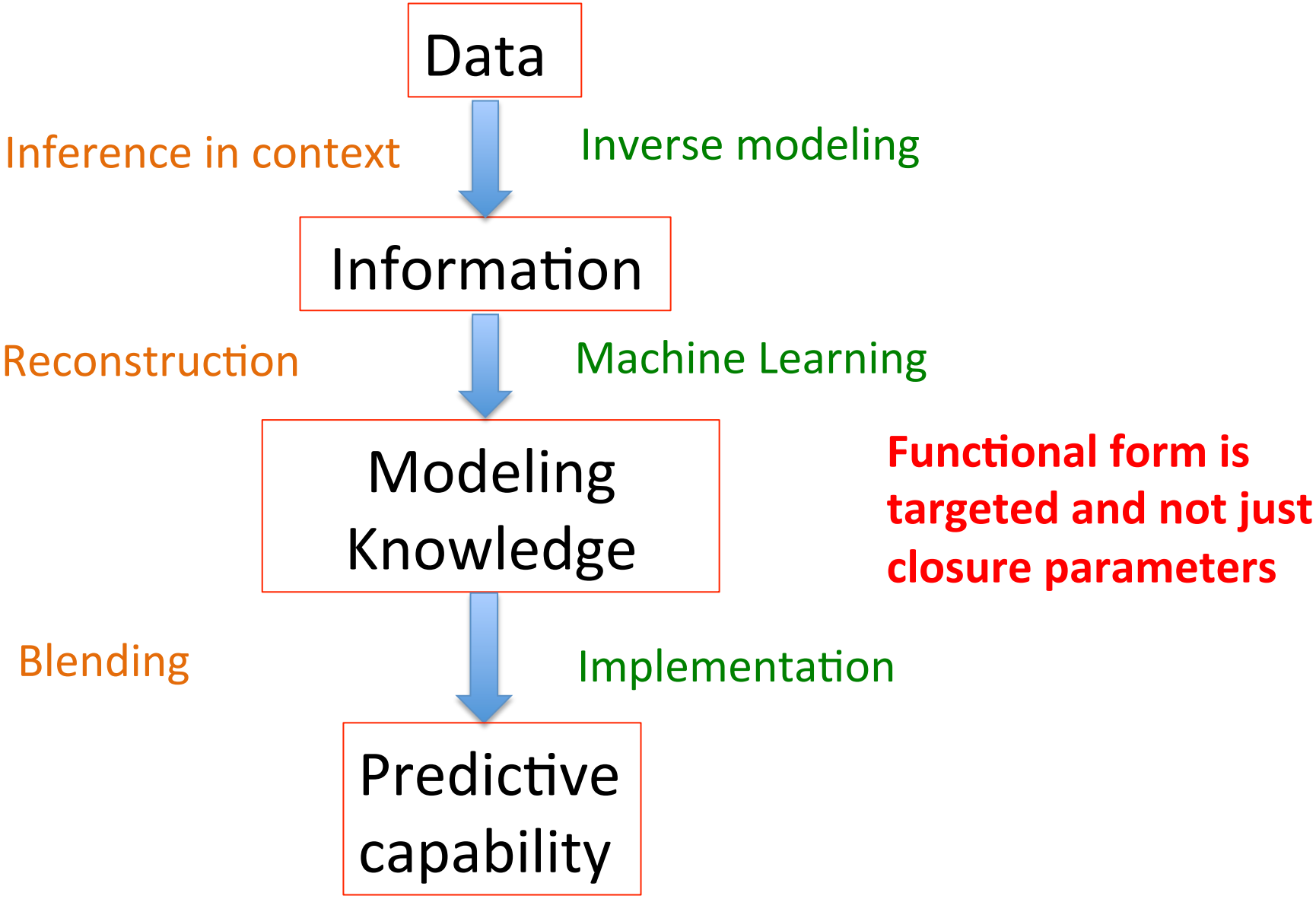


a = 0.1 (k = 61)

Multiscale Space Partitioning and Basis Construction

- Assign scales. For a space scaled to a unit box, by default one can set $h_1 = 1$, $h_2 = 0.5h_1, \ldots, h_k = 0.5h_{k-1}$.
- Get characteristic inter-cluster distances, $a_j = \alpha h_j$, $j=1,\ldots,k$.
- Apply partitioning algorithm for each scale.
- Get basis functions $\phi_i(x)$ (of scale h_j, centered at the j-th cluster centers, $j=1,\ldots,k$)

Summary


Data
Inference in context
Inverse modeling
Information
Reconstruction
Machine Learning
Modeling Knowledge
Functional form is targeted and not just closure parameters
Blending
Implementation
Predictive capability

Implications & Impact

Modeling insight : Modeler can understand what the model lacks to match data. This is done within the context of the model

Improved models : Can learn the missing components of model and generate improved models

Uncertainty quantification: Can obtain modeling error bounds

Generality : Can apply to any problem with data & a low fidelity model that is asymptotically equivalent.

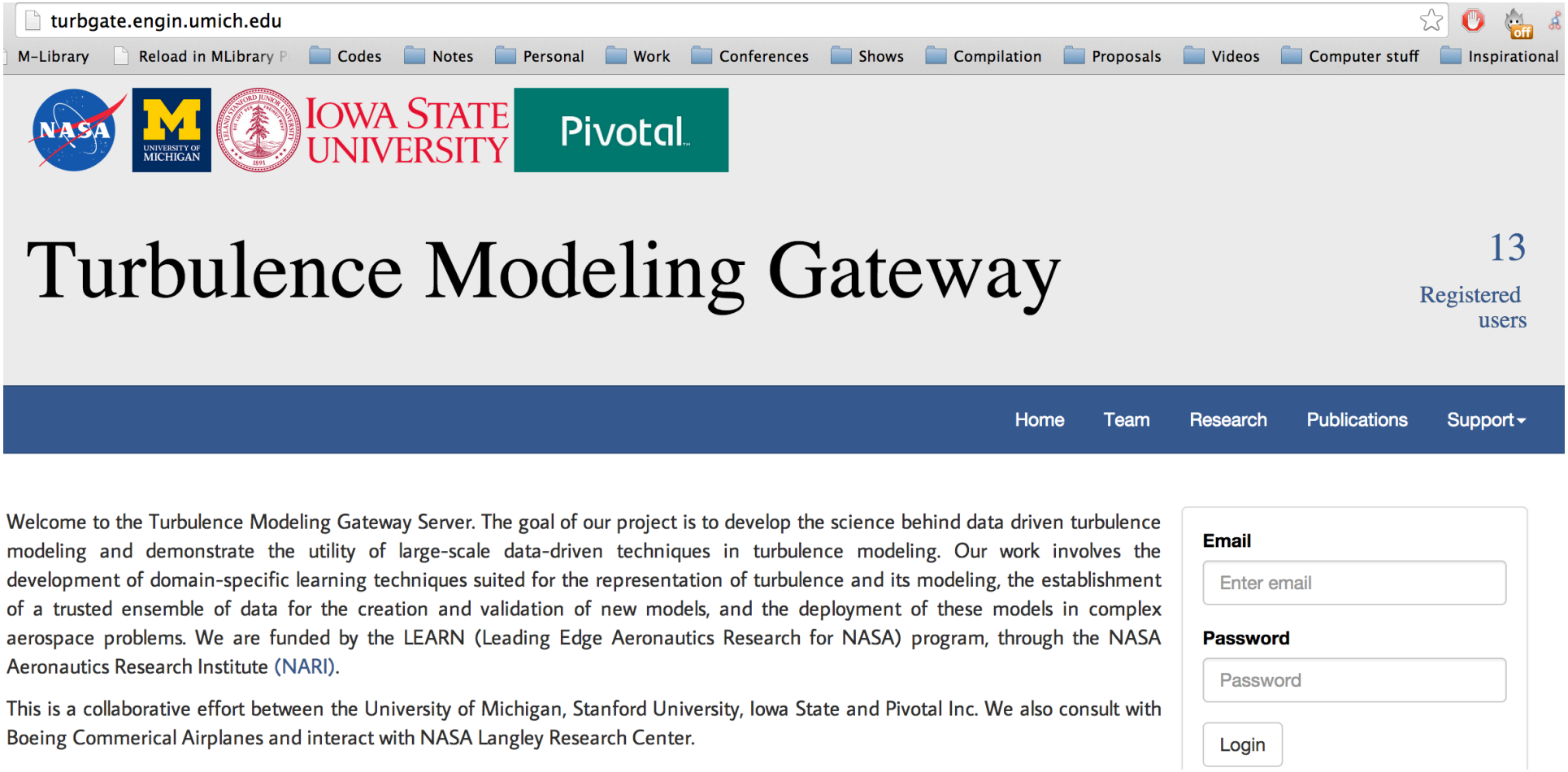
Turbulence, Transition, Combustion, High Enthalpy flows, etc

Much improved techniques for analysis and design of Aerospace vehicles

Vision for the future

A continuously augmented curated database / website of high-fidelity CFD solutions (and experimental data!) that are input to the machine learning process

Users download and plug map into their code.



Turbulence models of the future

- Traditionally, RANS models have been provided through:
  - A complete set of PDEs with modeled closure terms
  - A number of “fitting” coefficients
  - A printed article to deliver all this content
- In the future, we expect to provide:
  - A set of PDEs comprising all that is well known about the behavior of the turbulence
  - An auxiliary piece of software that can be embedded into a RANS solver and that contains the machine-learned closure terms (appropriately version controlled)
  - A software repository with clear explanations of the datasets that were used to create the “model”
  - Possibly multiple versions of the “models” that have been trained with different datasets and are more appropriate for different flow conditions.

Continuing work & Focus of Phase II

More comprehensive (but selective) data

- ➔ Targeted test cases
- ➔ NASA's benchmark cases

Work on ML algorithms (not very well developed for things other than clicking ads)

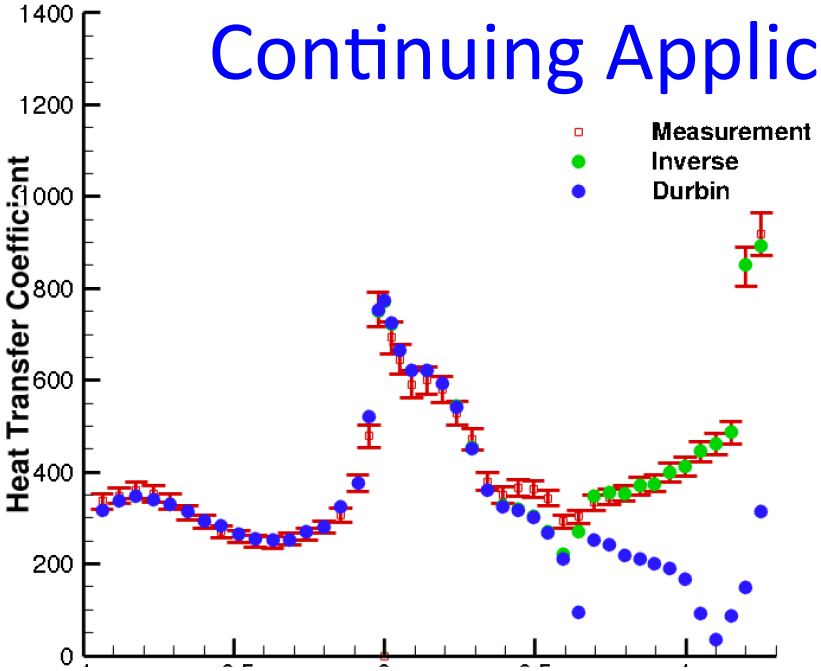
Optimizer needs further development

Build statistical framework (path to UQ)

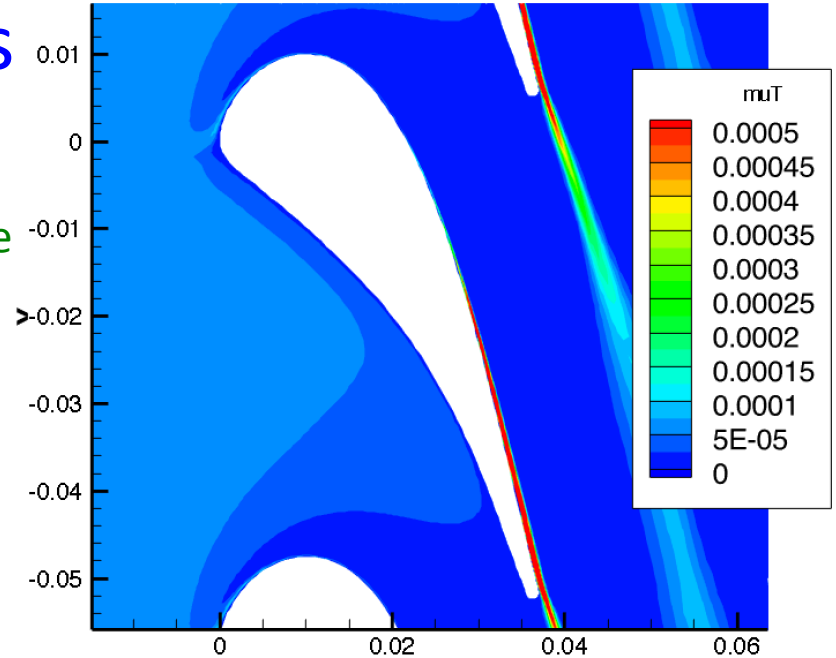
Fill in theory where data is missing & for asympotics

Full integrated inversion / learning with Billions of data points

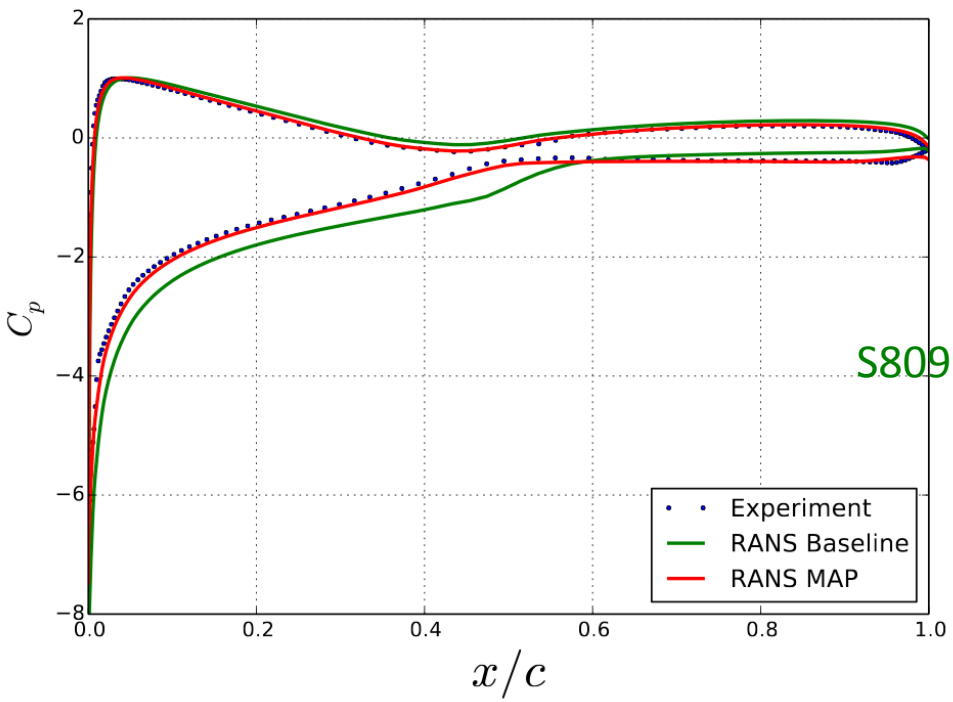
Continuing Applications

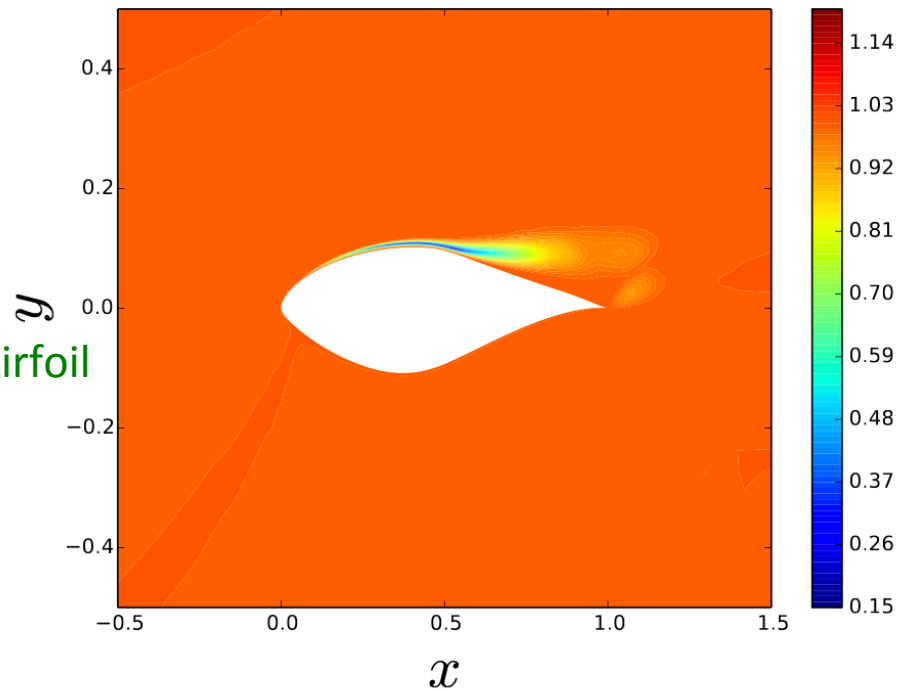


VKI blade



S809 Airfoil





Dissemination

- Participation in NASA/Stanford Summer Turbulence Research Program (June 2014)
- Presentation at NASA Langley (September 2014)
- Visit by NASA Tech. Monitor to Michigan (Dec 2014)
- Collaboration with Big data analytics group at NASA Langley
- Discussions with several NASA researchers (Ames, Langley)

- Interest from GE Aviation to fund pilot project
- Discussions with AFRL, Dayton
- 3 Conference proceedings papers
- Journal papers under preparation
- Project website

- Planning a symposium on New Directions in Turbulence Modeling

Acknowledgements

NASA LEARN program (Koushik Datta, Michael Dudley, Tech. Monitor: Gary Coleman)

Backup slides

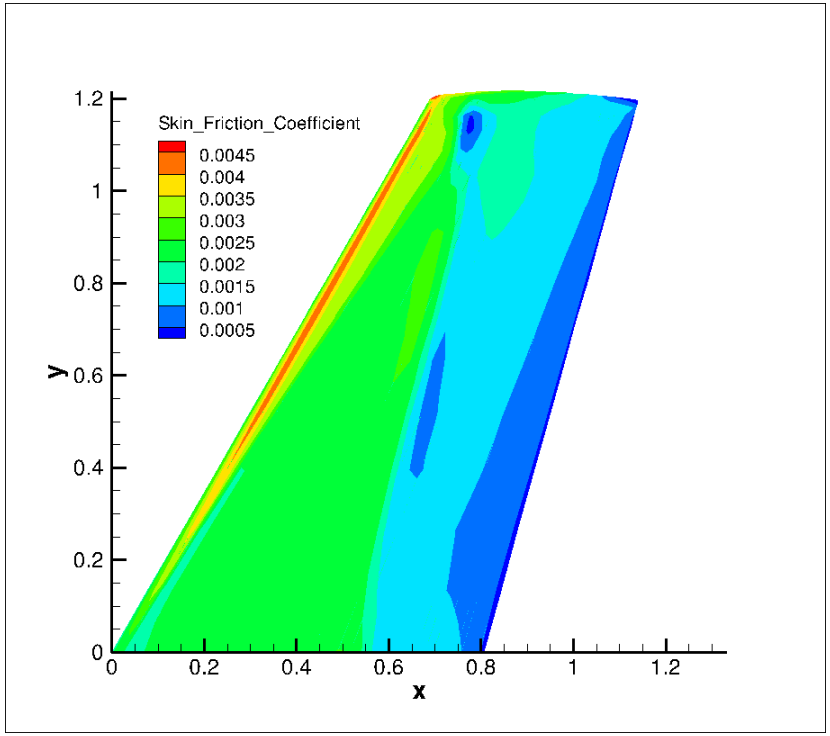
Disclaimer on RANS models

- Single point closures based on local `well-behaved' quantities
  - ➔ Miss out on spectral and structural information
  - ➔ Do not process disparity of turbulence scales
  - ➔ Cannot distinguish inactive motions and low frequency unsteadiness

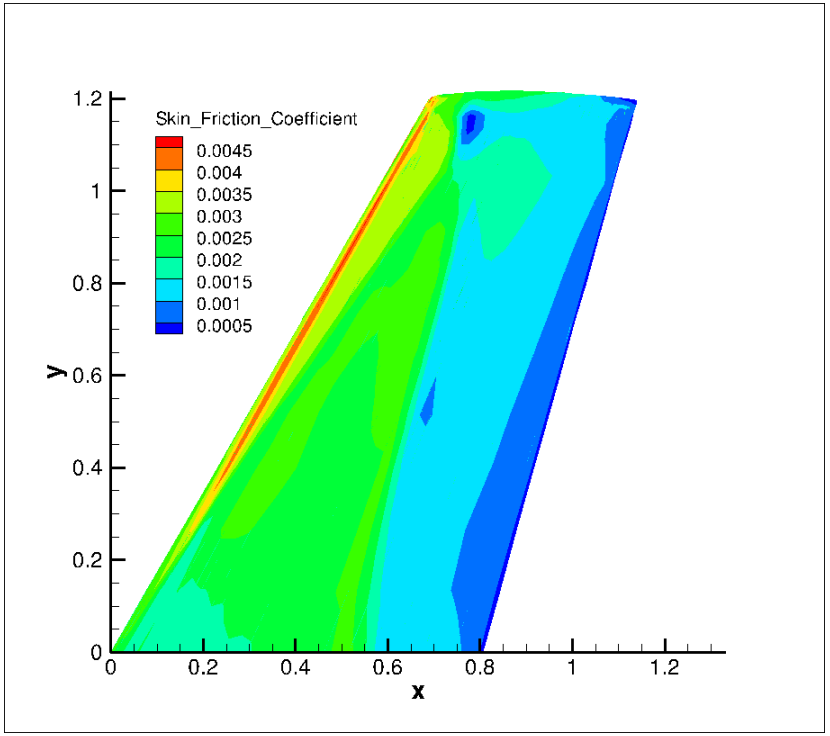
- But, room for improvement is vast

Onera M6

- Lift and drag coefficient match to within 2%

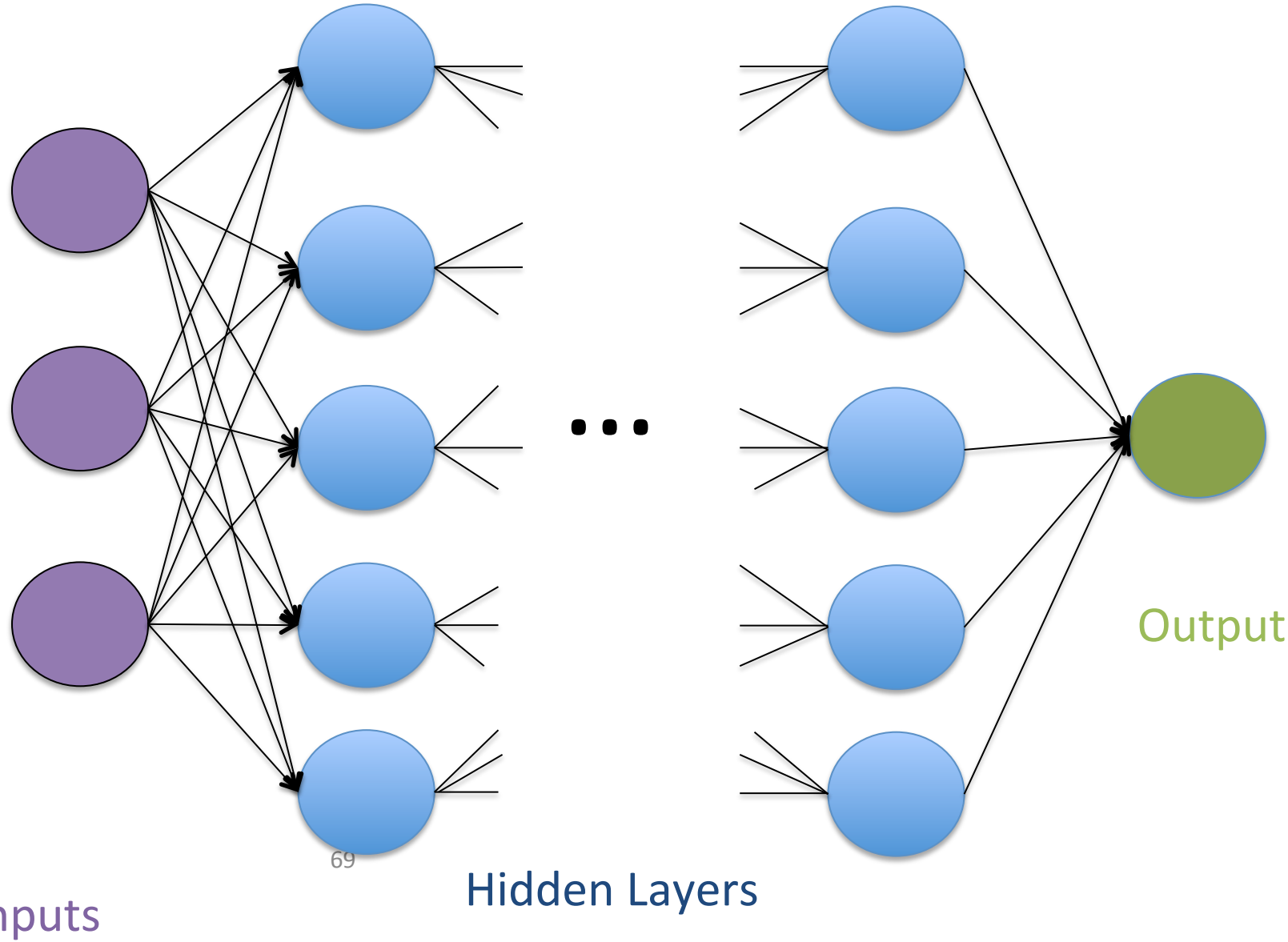


True



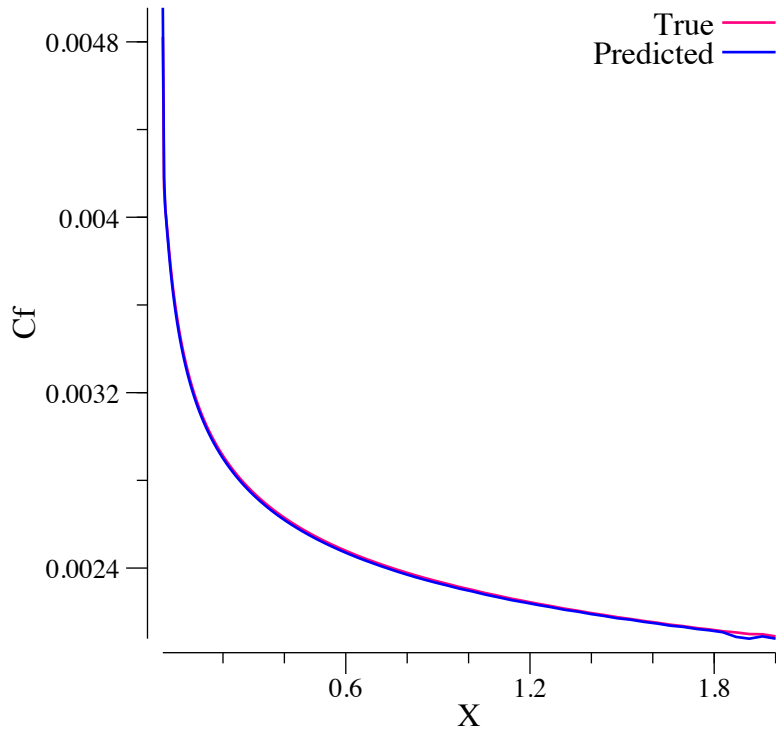
ML

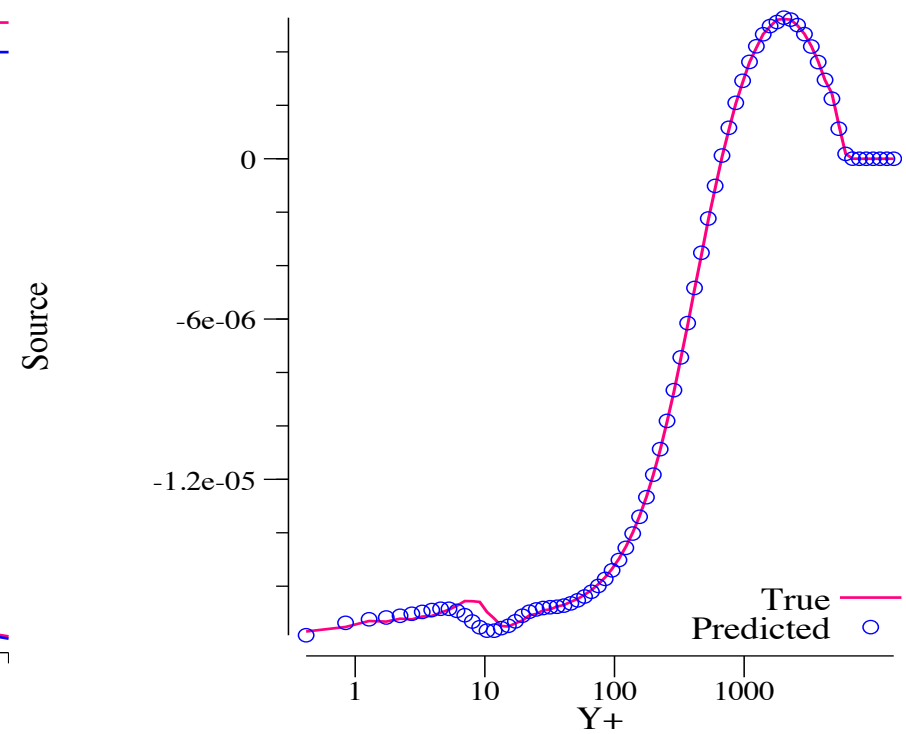
Neural Networks



Results – Budget

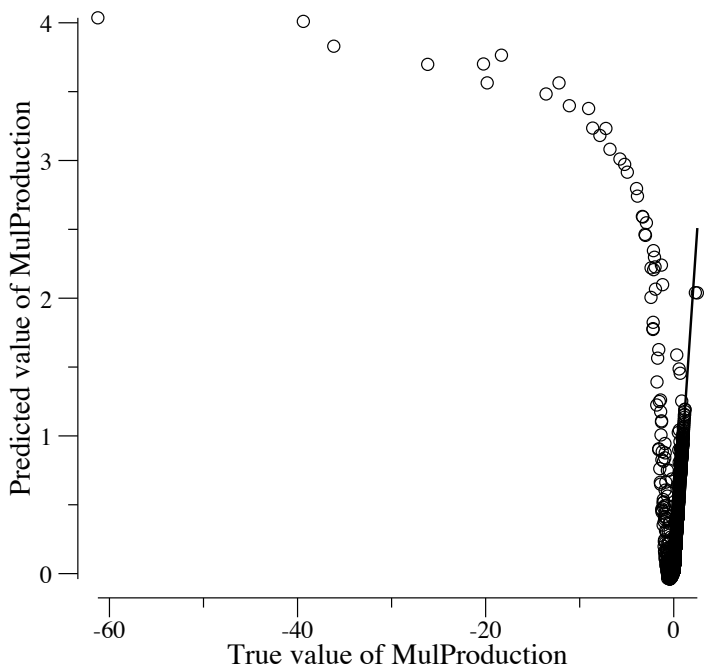
- Source term computed from budget balance

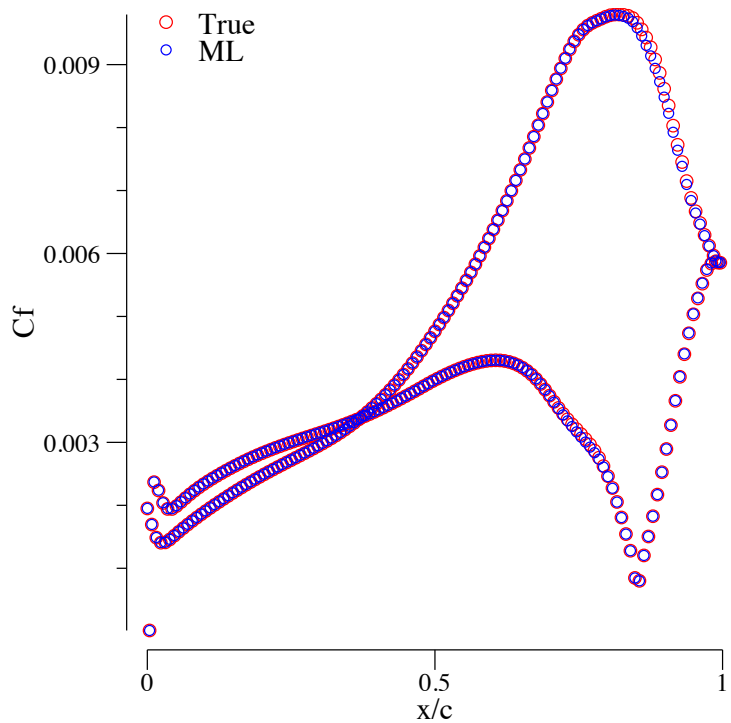




Great match even with noisy data

Misleading results





Model Problem

- Use Spalart-Allmaras as a model problem
  - Easy to generate flow data
  - Analytic functional form
  - Known feature set
- Learn (parts of) the source term
- Separate flaws in training procedure vs. errors in data fit.

Implementation

```
 for i = 1, Niter
     call Mean_RHS
     call Turb_RHS
     call Compute_Jacobians
     call Solve
     call Update
End

Subroutine Turb_RHS
     call ComputeEddyViscosity
     call Turb_Convection
     call Turb_Diffusion
     call Turb_Source
End  Subroutine
```

Implementation in Turb_Source

```
double* fann_inp = (double*)malloc(size_inp);        // Inputs to Map
double* fann_out = (double*)malloc(size_out);        // Outputs of Map

// Create input deck
int counter_inp = 0; int counter_out = 0;
for (int icv = 0; icv < ncv; icv++) {
  fann_inp[counter_inp] = Rt[inp];  counter_inp ++ ;
  fann_inp[counter_inp] = kine[icv]; counter_inp ++;
  fann_inp[counter_inp] = omega[icv];  counter_inp ++;
 }
  // Obtain output
  int ierr = fann_model(ninput_query, ncv, fann_inp, fann_out);

  // Inject
  for (int icv=0; icv<ncv; icv++) {
    Source[icv] = fann_out[counter_out]; counter_out++;
  }
```