Recent evolution of Gene-Expression Programming for developing turbulence models

Richard Sandberg
richard.sandberg@unimelb.edu.au

THE UNIVERSITY OF MELBOURNE

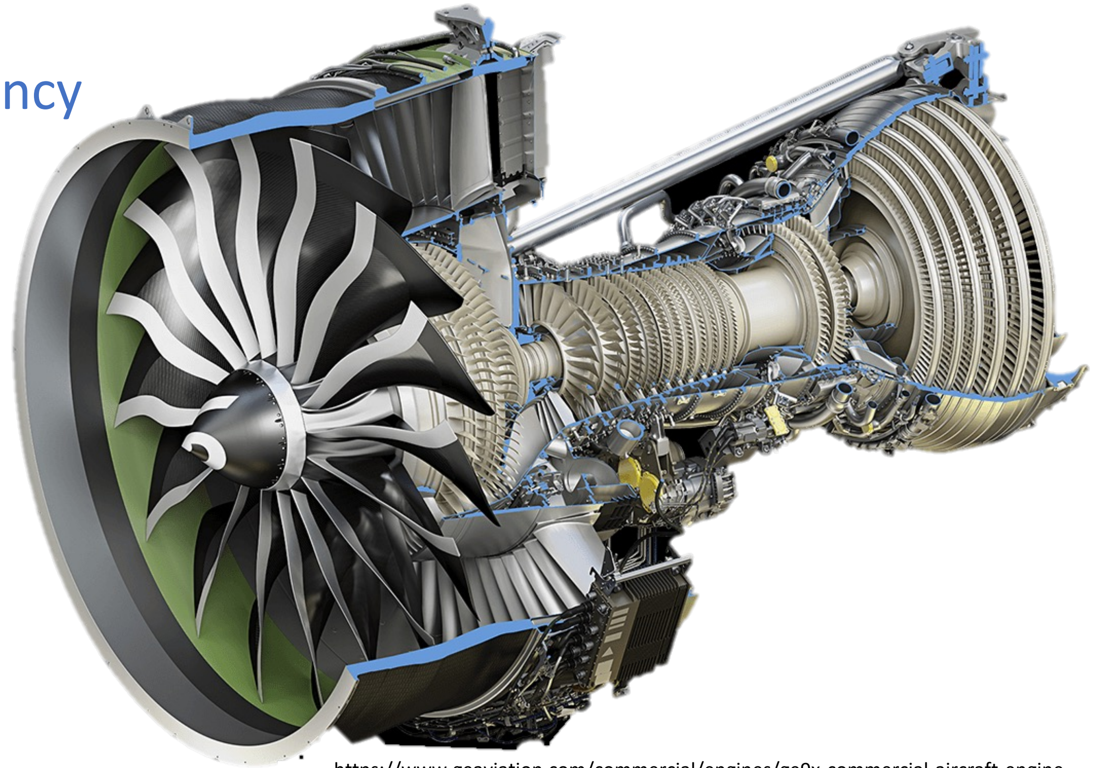
Acknowledgements: Dr J Weatheritt, Dr Y Zhao, Dr H Akolekar, Dr C Lav, Dr A Haghiri, F Waschkowski, M Schoepplein, M Reissmann, Prof M Klein

Background

- Why do we need better turbulence models?

  Example of gas turbines

  - Correlation-based approaches unable to improve efficiency
  - Experiments expensive
  - High-fidelity simulations too expensive for design ($> 10^{15}$ DOF)



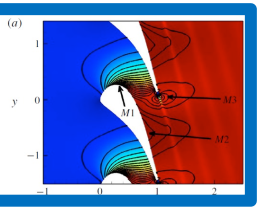
https://www.geaviation.com/commercial/engines/ge9x-commercial-aircraft-engine

- Why accuracy important?
  - Inaccurate prediction can erode operability
    → unexpected compressor stall at off-design
    → 2% error in predicted metal temperature can halve blade life (Han et al. 2012)
  - Inaccurate prediction of complex flows can cost 0.5% efficiency
    → small gains have significant impact on fuel use (>300 billion litres!), emissions and potential uptake of new fuels

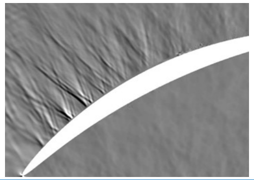
Key Challenges for RANS in Turbomachinery

Blade-to-blade effects

Pressure gradients

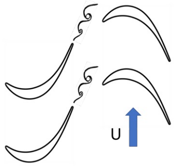


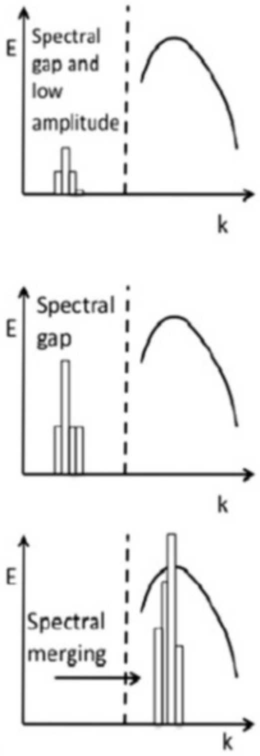
Surface curvature, transonic flow

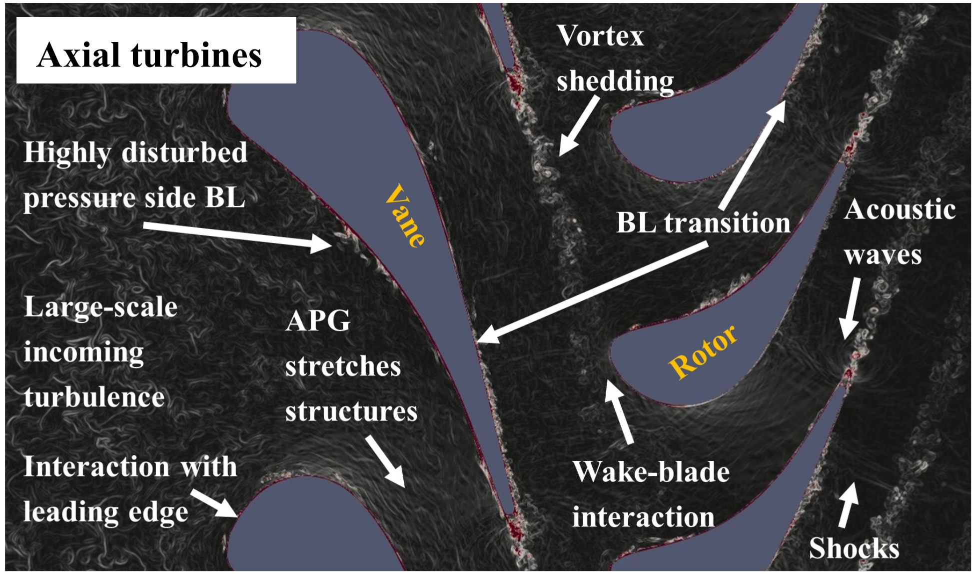


Blade-row to blade-row effects

Deterministic unsteadiness
Blade-to-blade interaction, vortex shedding, intermittent wakes







Challenges that lead to inaccuracies:

- Enthalpy and thermal mixing not correct
  → recalibrate coefficients (NOT GENERAL)
- Deterministic vs stochastic unsteadiness
  - Vortex shedding
  - Wakes
  - SBLI

  → URANS (spectral gap!)

- Transition
  - Natural
  - Bypass
  - Separated flow transition

  → Transition models (MORE MODELING)
- Misalignment of τ_{ij} and S_{ij}
  - Non-equilibrium BL, Wake distortion

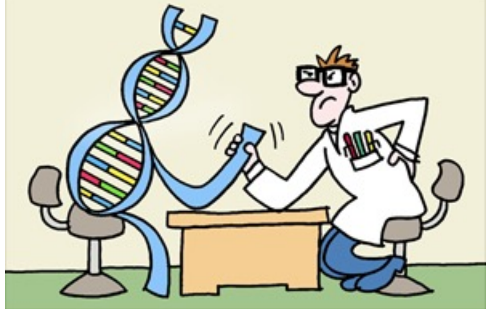
  → Fix inherent model error (ML?)

Background – Gene Expression Programming

We want a ***symbolic*** regression approach to develop turbulence models from hi-fi data

- Get robust CFD-ready models (plug and play)
- Interpretable
- Training on simulation or experimental data
- Train models for *unsteady flows*

Evolutionary Algorithm



Evolutionary concepts borrowed from biology (evolve suitable functions)

- Survival of fittest idea
- Incremental improvements via genetic operations (cloning, mutation, crossover)

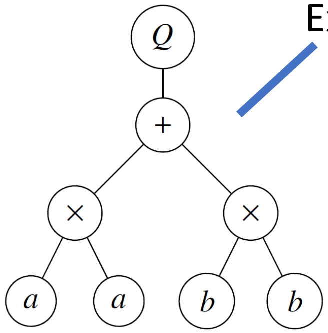
How do we evolve symbolic expressions that are syntactically correct?

- Gene Expression Programming (GEP) transforms symbols to equations (Ferreira, 2001) :

Head – function set | Tail – terminal set

0	1	2	3	4	5	6	7	8	9	10	11	12
Q	$+$	$\times$	$\times$	a	a	b	b	a	a	b	c	1

Chromosome - list of symbols (exists in code)



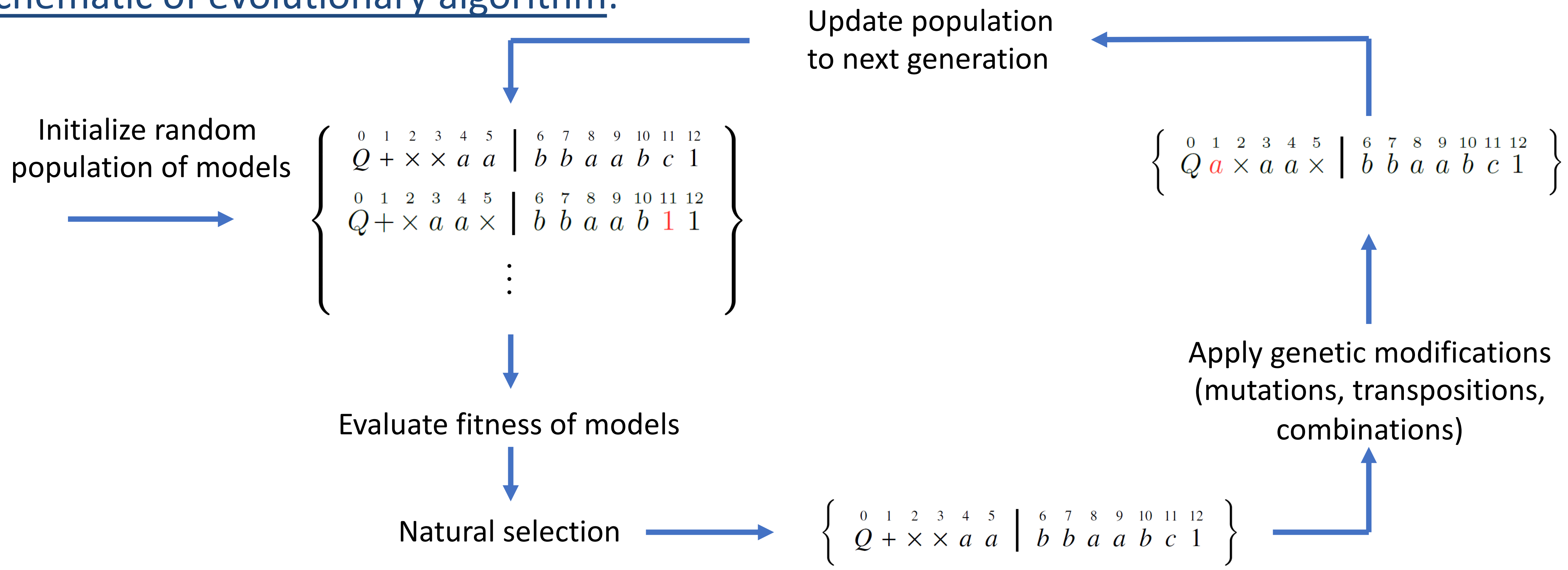
Expression tree

$$\sqrt{a^2 + b^2}$$

Predictive model (valid expression - can be nonlinear)

Background – Gene Expression Programming

Schematic of evolutionary algorithm:



- Set of predictive models (*population*) is developed over multiple generations to fit the available training data
- The fittest model of the last generation is the training outcome
- Can do that with tensors and vectors as well (Weatheritt & Sandberg, JCP 2016)

Gene Expression Programming for turbulence modelling

Development of improved anisotropy model (Weatheritt & Sandberg, 2016)

Extend the linear model to include higher order gradients

$$\tau_{ij} - \frac{2}{3}\rho k \delta_{ij} = -2\mu_t S'_{ij} + 2\rho k \sum_{k=1}^{10} \zeta_k (I_1, I_2, I_3, I_4, I_5) V^k_{ij}$$

Unknown coefficients, functions of independent variables

$$I_1 = s_{mn}s_{nm}, I_2 = \omega_{mn}\omega_{nm}$$

Independent scalar variables

Independent tensor variables

$$V^1_{ij} = s_{ij}, V^2_{ij} = s_{ik}\omega_{kj} - \omega_{ik}s_{kj},$$
$$V^3_{ij} = s_{ik}s_{kj} - \frac{1}{3}\delta_{ij}s_{mn}s_{nm},$$

Basis Functions

Pope (1975)

Strain rate tensor: $s_{ij} = \tau S_{ij}$

Vorticity rate tensor: $w_{ij} = \tau \Omega_{ij}$

turbulent time scale: $\tau = 1/\omega$

Extension of approach by introducing correction to production in k-ω equations (Schmelzer et al., 2020)

$$\frac{\partial k}{\partial t} + \bar{u}_j \frac{\partial k}{\partial x_j} = (P_k + R) - \beta^* k\omega + \frac{\partial}{\partial x_j}\left((\nu + \sigma_k \nu_t) \frac{\partial k}{\partial x_j} \right)$$

$$\frac{\partial \omega}{\partial t} + \bar{u}_j \frac{\partial \omega}{\partial x_j} = \frac{\gamma}{\nu_t} (P_k + R) - \beta\omega^2 + \frac{\partial}{\partial x_j}\left((\nu + \sigma_\omega \nu_t) \frac{\partial \omega}{\partial x_j} \right) + 2(1 - F_1)\sigma_{\omega 2} \frac{1}{\omega} \frac{\partial k}{\partial x_j} \frac{\partial \omega}{\partial x_j}$$

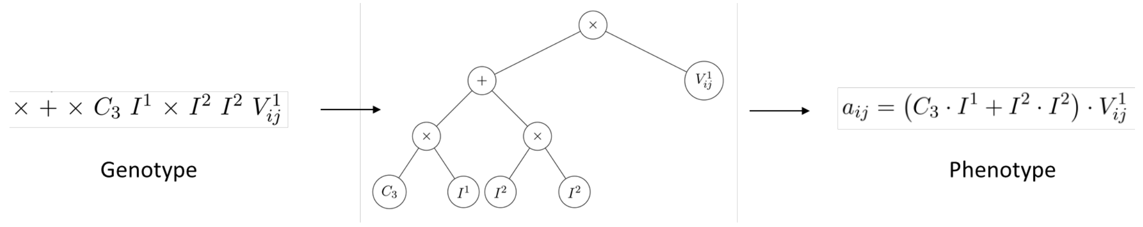
$$R = k a^R_{ij} \frac{\partial \bar{u}_i}{\partial x_j}$$

Unknown coefficients, functions of independent variables

$$a^R_{ij} = \sum_{k=1}^{10} \xi_k (I_1, I_2, I_3, I_4, I_5) V^k_{ij}$$

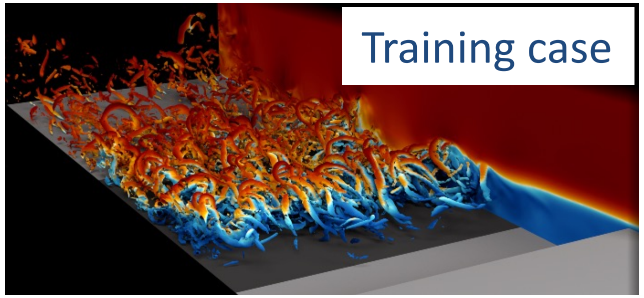
With **high-fidelity data** try to **find** ζ_k and ξ_k as functions of independent variables I_k

Pool of symbols: $S = \left\{ V^k_{ij}, I^l, C_m, +, -, \times \right\}$

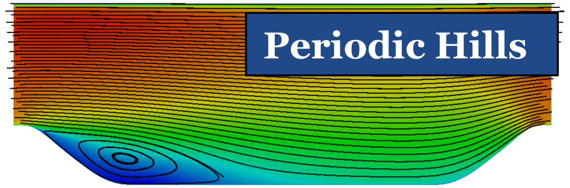


Gene Expression Programming – statistically 2D flows

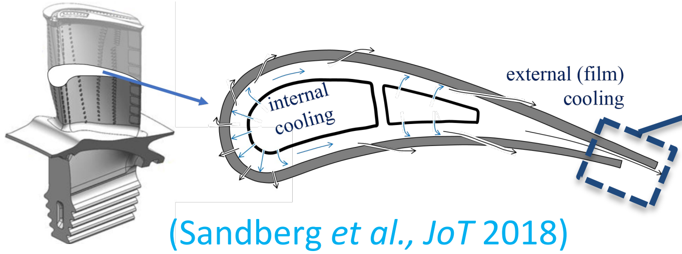
1) Reynolds stress:

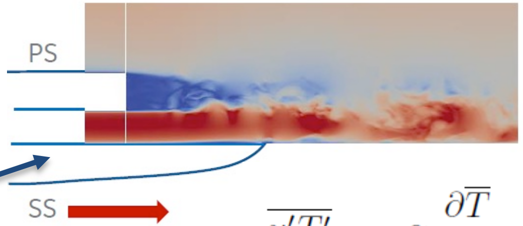


Apply trained model to different flow



2) Heat flux vector:



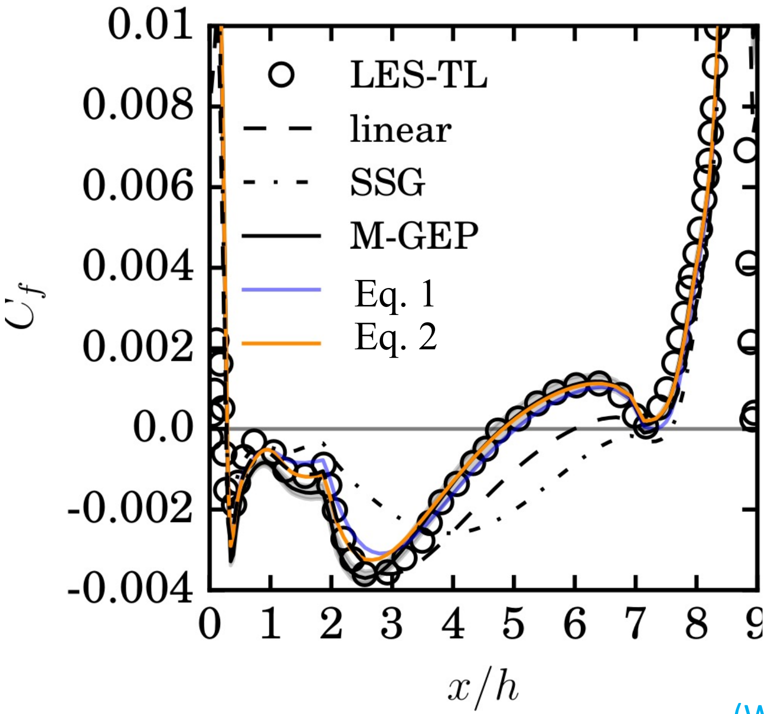


(Sandberg *et al.*, *JoT* 2018)

$$\overline{u_i'T'} = -\alpha_t \frac{\partial \overline{T}}{\partial x_i}$$

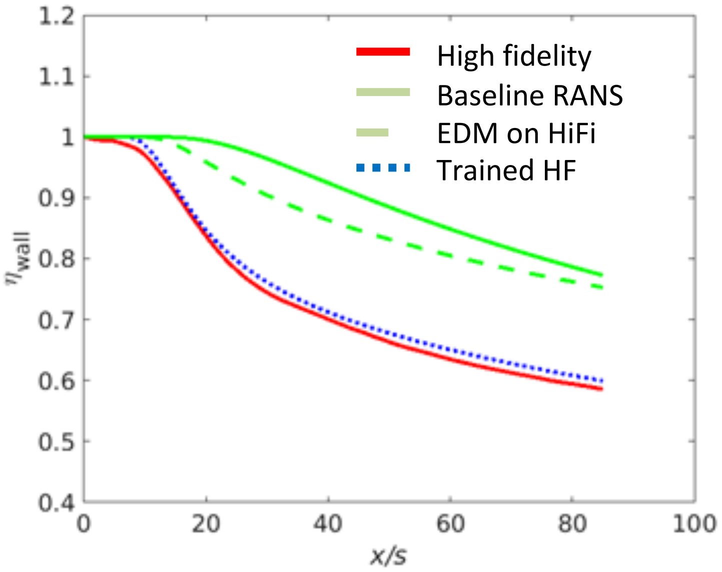
EDM: $\alpha_t = \frac{\nu_t}{Pr_t}$

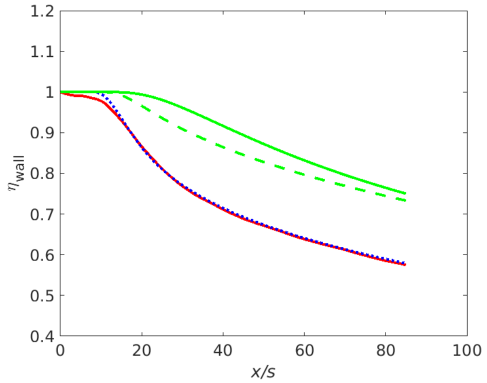
GEP model: $\alpha_t^{mod,1} = \{6.806 I_2 - 109.407 J_1 + 2.0 J_2 + 2.368\}\, \nu_t$

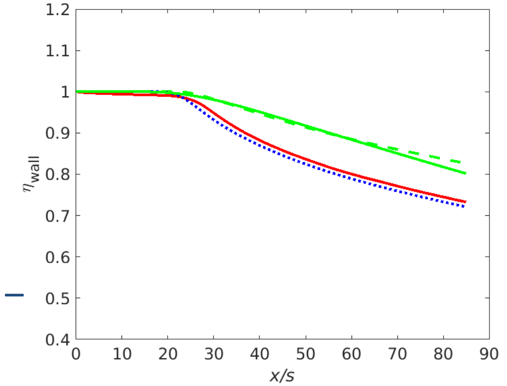


LES-TL: reference LES
Linear : linear RANS model (SST)
SSG : non-linear EASM model
M-GEP: ensemble average of trained models
Eq 1/2 : individual new models

(Weatheritt & Sandberg, 2016)

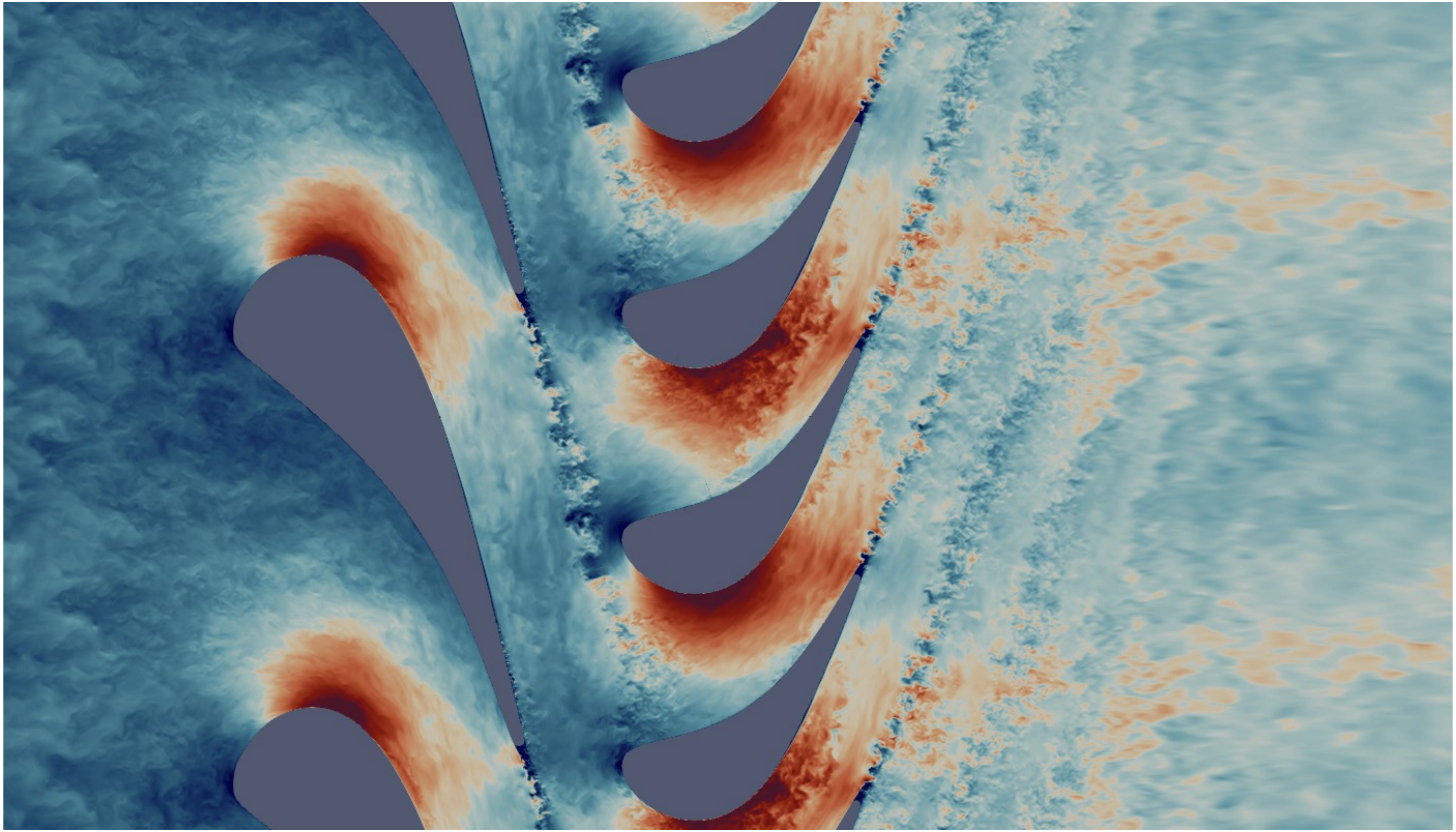






New models tested on 9 other cases with different slot geometries and Blowing Ratios – 2 examples

Gene Expression Programming – unsteady flows

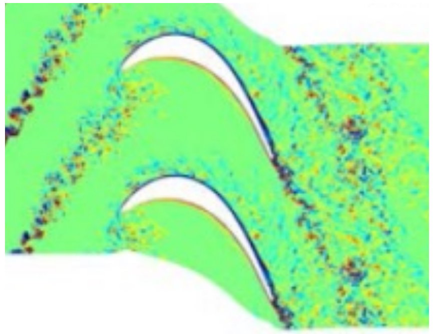


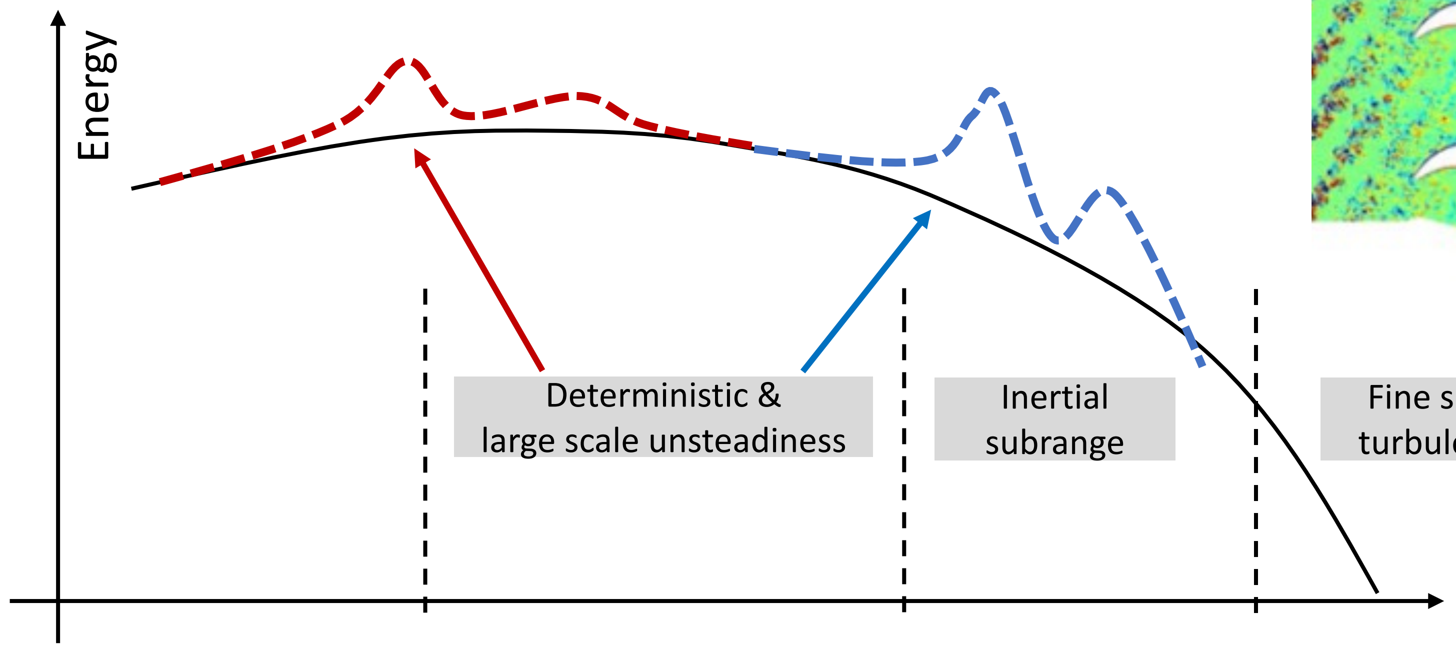
Gene Expression Programming – unsteady flows

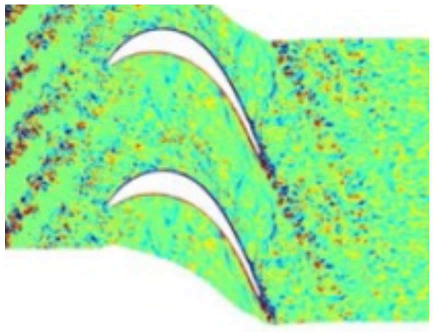
How to develop models for unsteady flows?

- Underpinning feature of turbomachinery is interaction of stochastic (turbulence-driven) and deterministic (stator-rotor interaction / vortex shedding) flow unsteadiness

- Drives mixing processes of momentum & enthalpy → determines level of irreversibility and thus efficiency

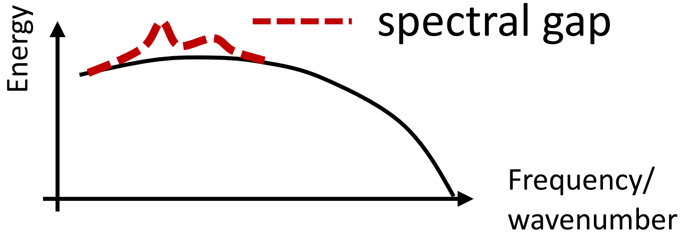


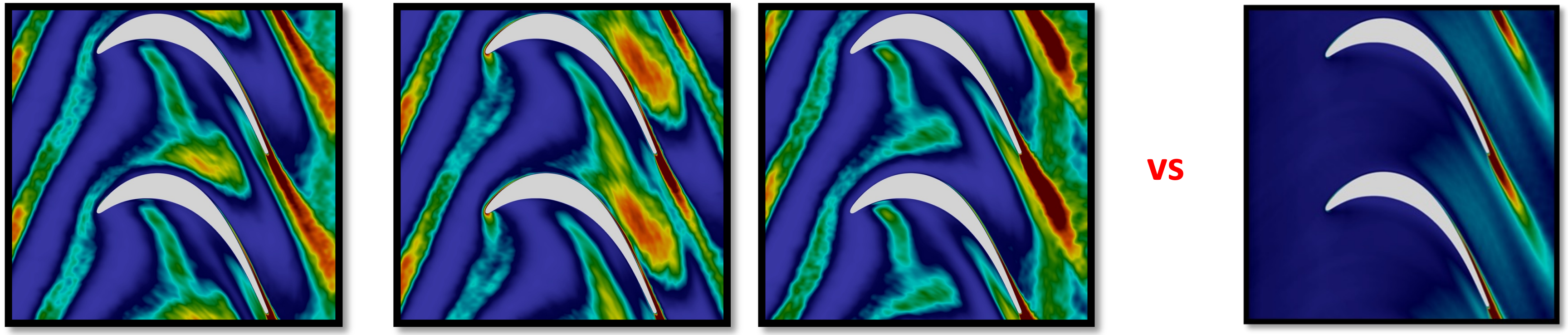




Heat-flux & Reynolds stress modelling – unsteady flows

Approach 1: Use phase-lock averaged DNS data to train models

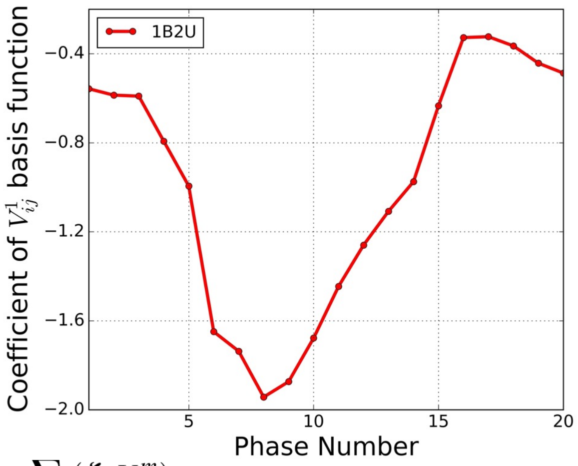


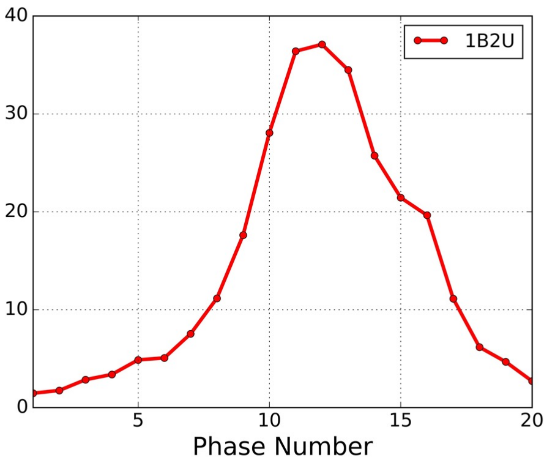


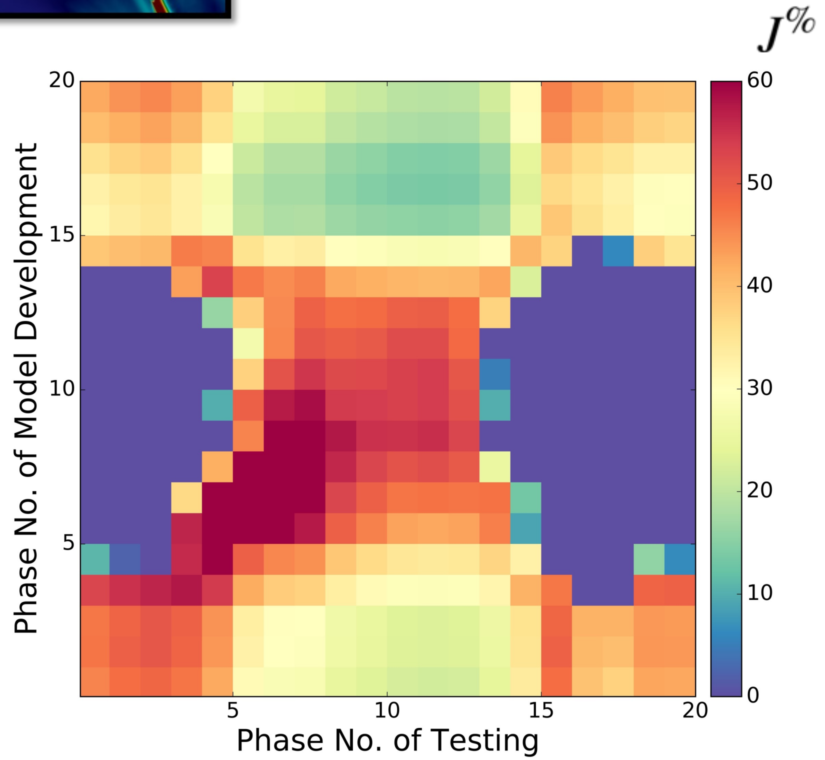
Phase 1 Phase 7 Phase 16 vs Time average

(Akolekar et al., *JoT* 2018)

Machine-learned models can provide information on physics



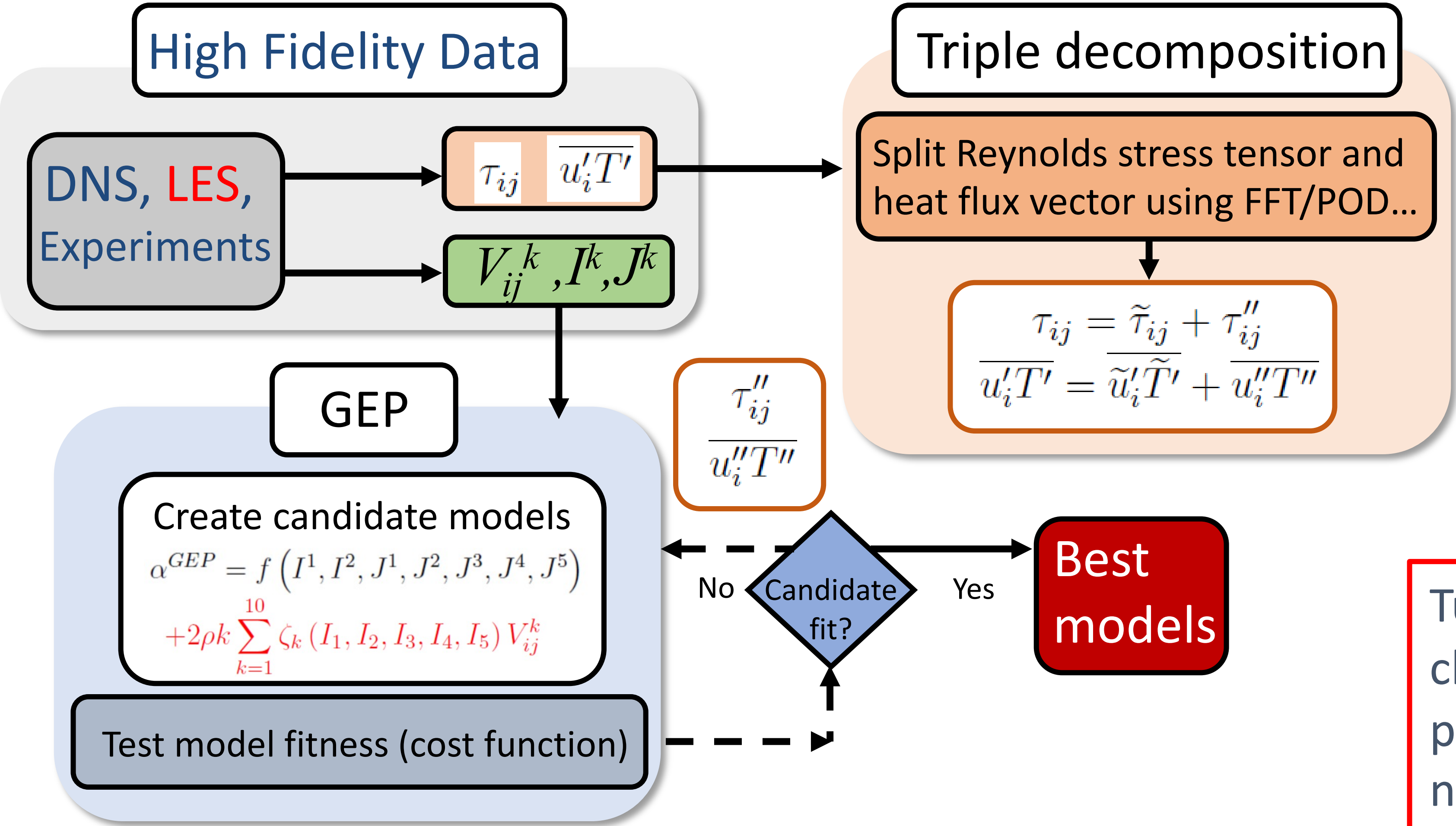


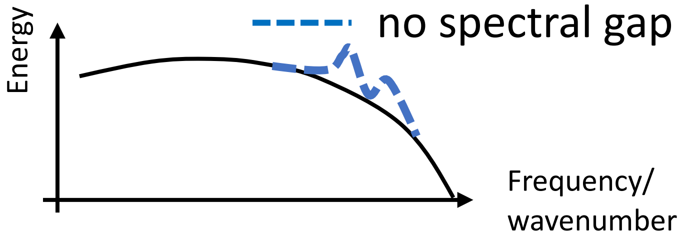


$$a_{ij}^{EARSM} = -\frac{\nu_t}{k} S'_{ij} + \zeta_1 V^1_{ij} + \sum_{m=2} (\zeta_m V^m_{ij})$$

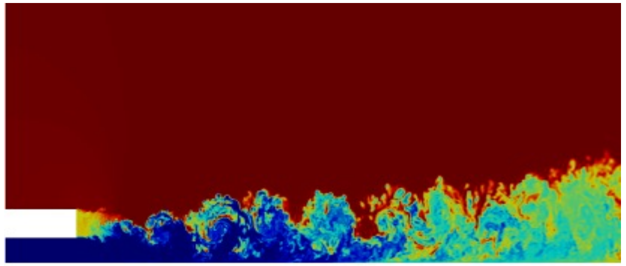
Machine-Learning (GEP) for unsteady flows

Approach 2: Develop bespoke model for unsteady RANS





Lav, Sandberg & Philip (JCP 2019)

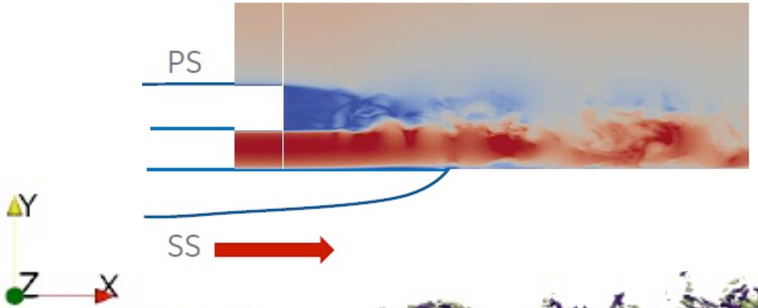


Bespoke GEP models for URANS

Turbulence stress and heat flux closures model only stochastic part of fluctuations, other scales need to be resolved

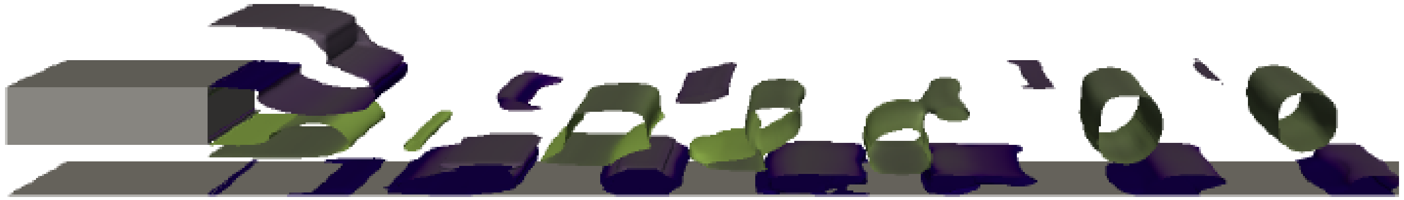
Machine-Learning (GEP) for unsteady flows

Approach 2: Develop bespoke model for unsteady RANS

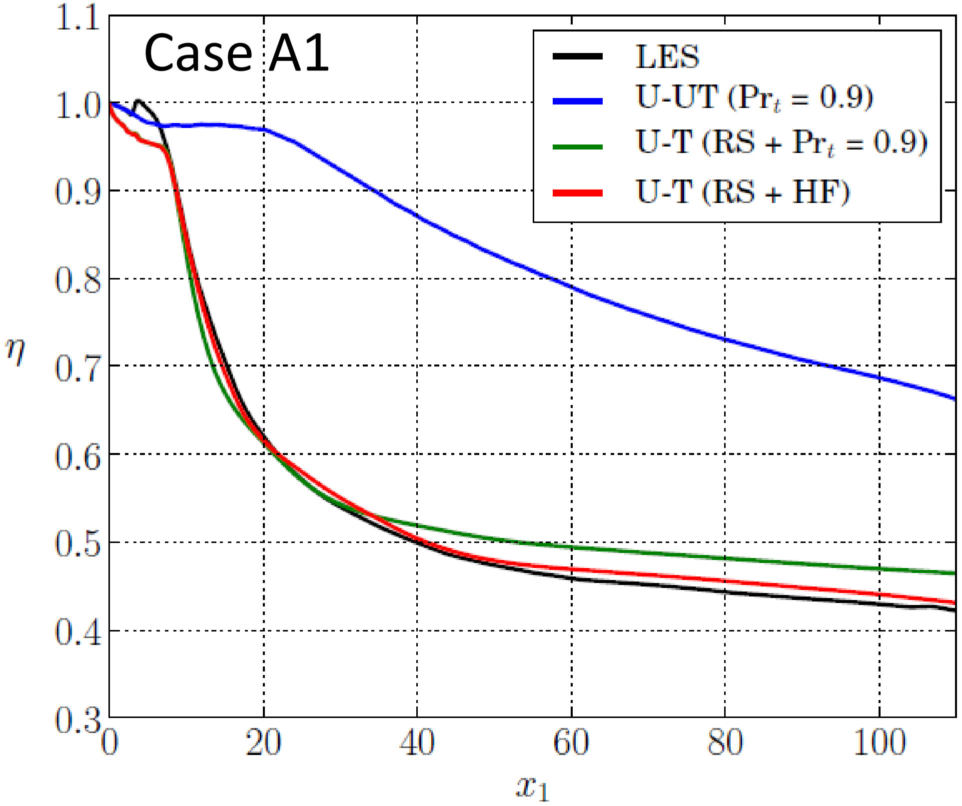


LES: Case A1

Untrained URANS



GEP-trained URANS

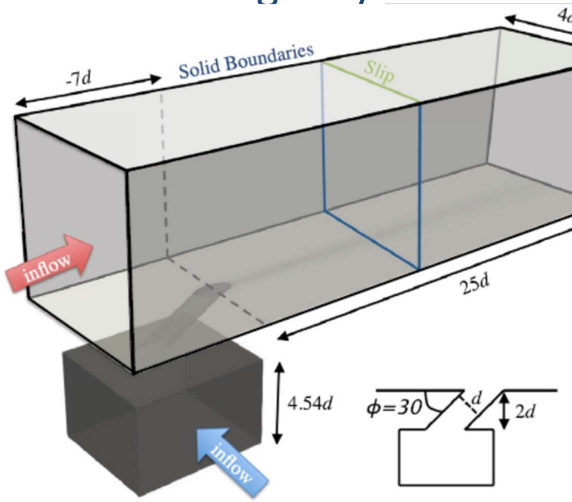


- Greatest improvement from RS model
- HF model provides most improvement downstream

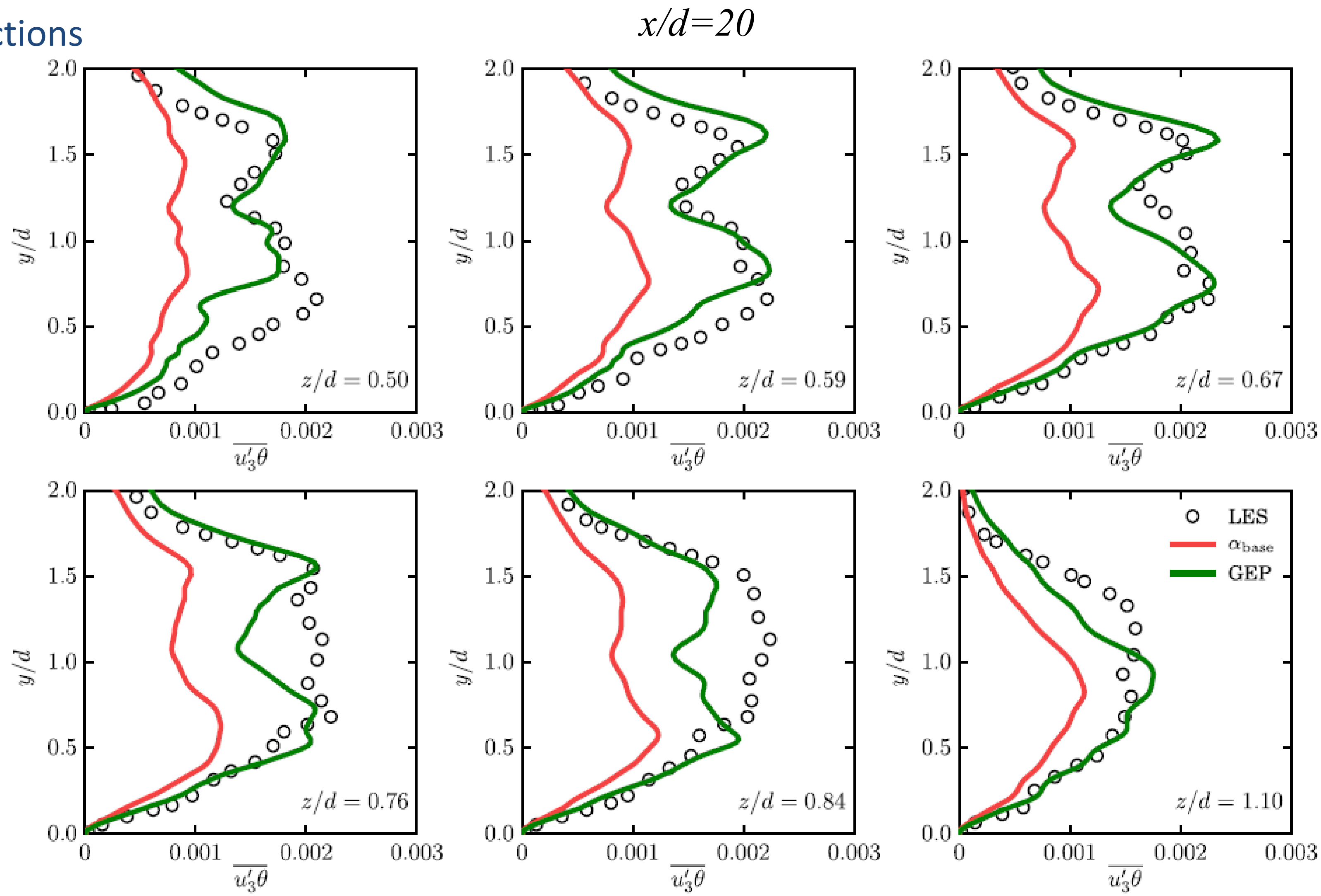
Gene Expression Programming – statistically 3D flows

Perform recursive feature elimination: $V_{ij}^4 > V_{ij}^9 > V_{ij}^2 > V_{ij}^5 > V_{ij}^1 > V_{ij}^7 > V_{ij}^{10} > V_{ij}^6 > V_{ij}^8$ (Weatheritt et al, *IJHMT* 2020)

Train models using only first three basis functions



(Bodard et al., CTR 2013)



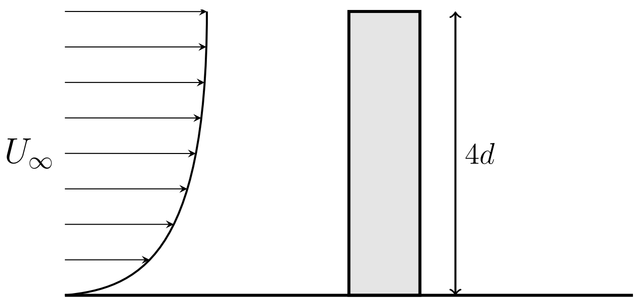
- Overall good improvement with GEP model

Gene Expression Programming – statistically 3D flows

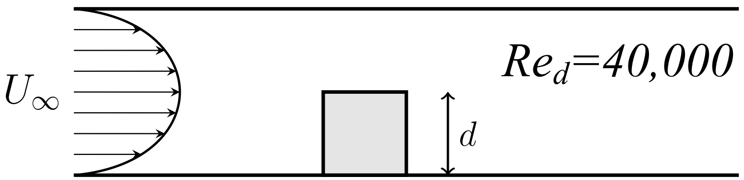
Perform recursive feature elimination: $V_{ij}^1 > V_{ij}^4 > V_{ij}^3 > V_{ij}^2 > V_{ij}^6 > V_{ij}^5 > V_{ij}^7 > V_{ij}^{10} > V_{ij}^8 > V_{ij}^9$

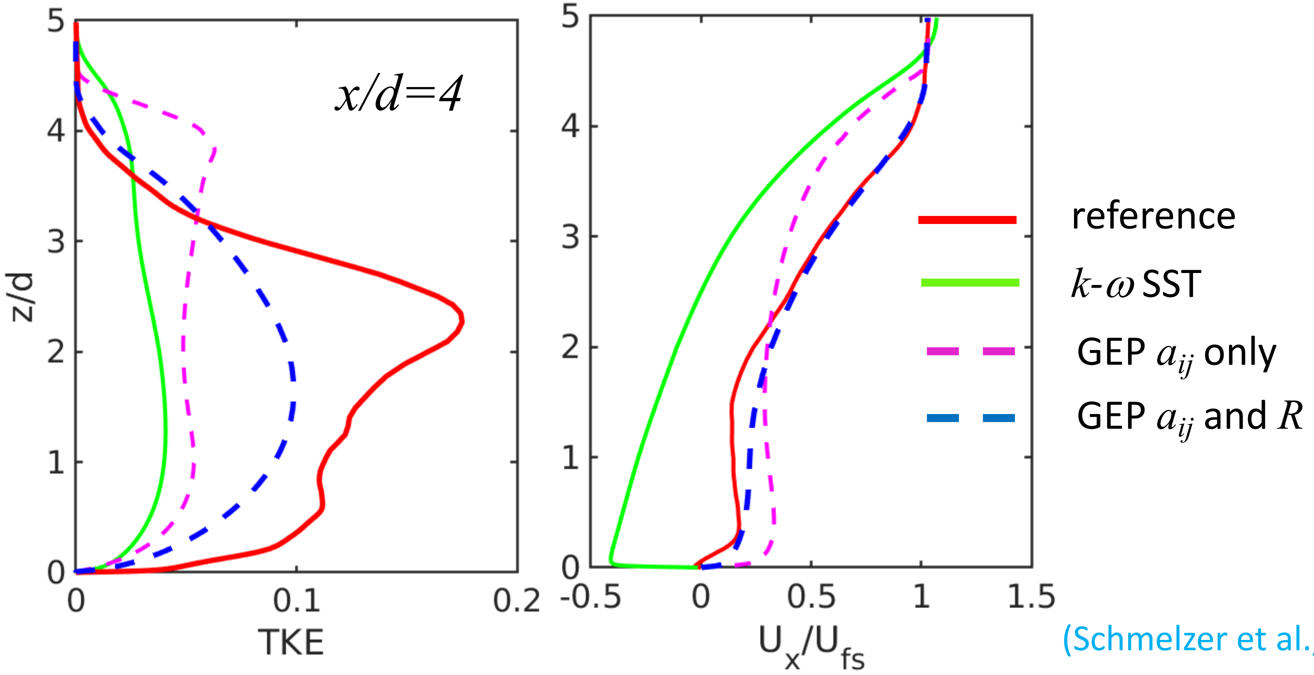
Train models using only first four basis functions

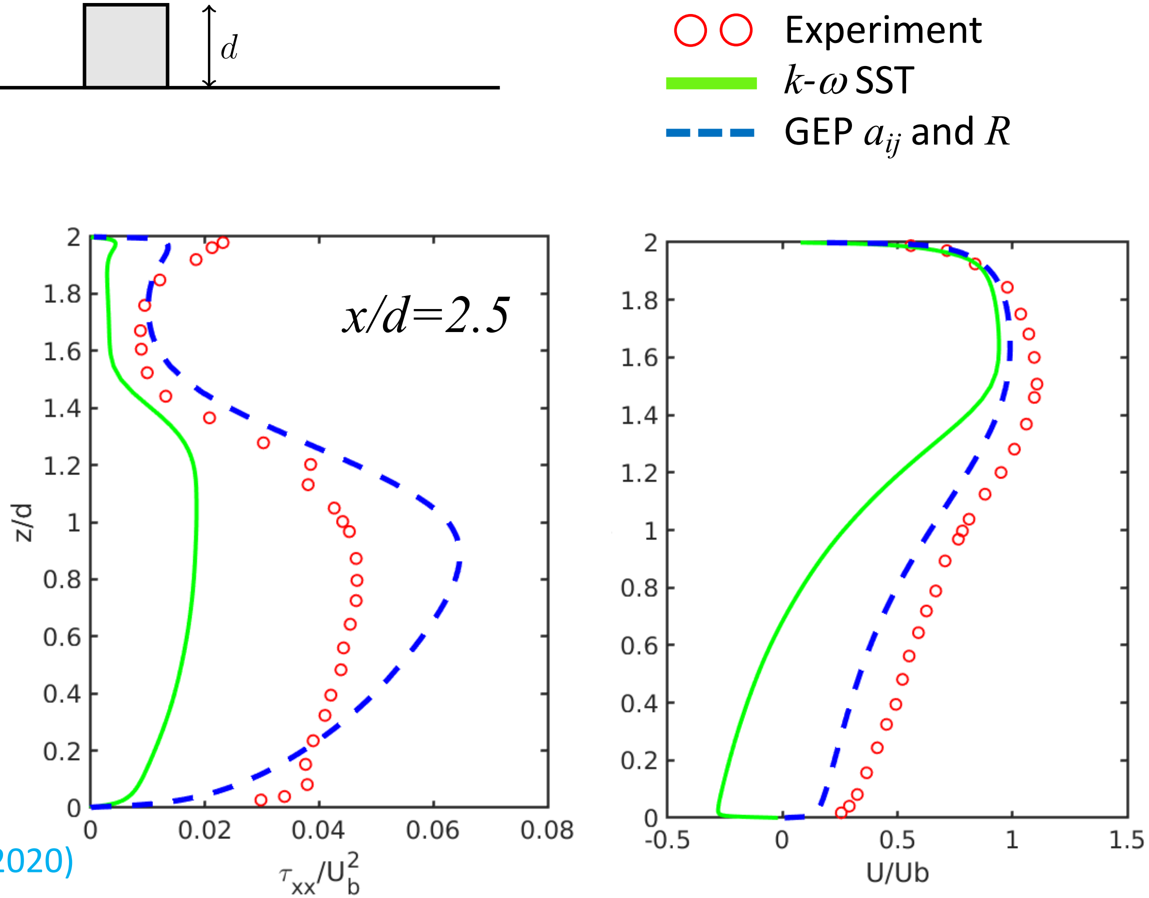
Training case



Testing case



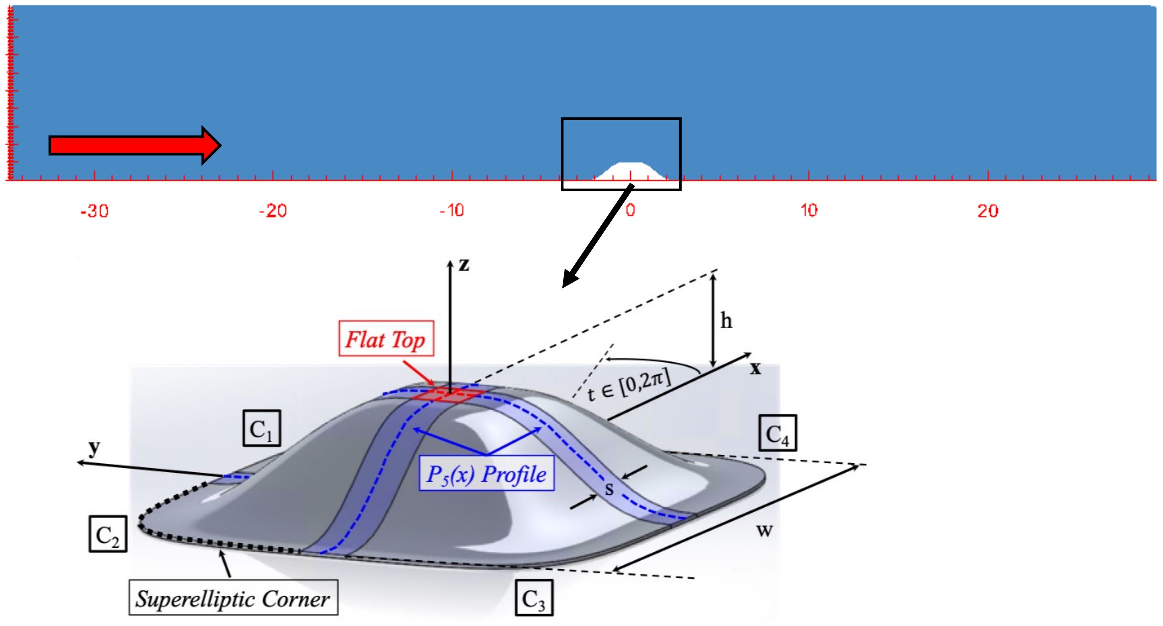




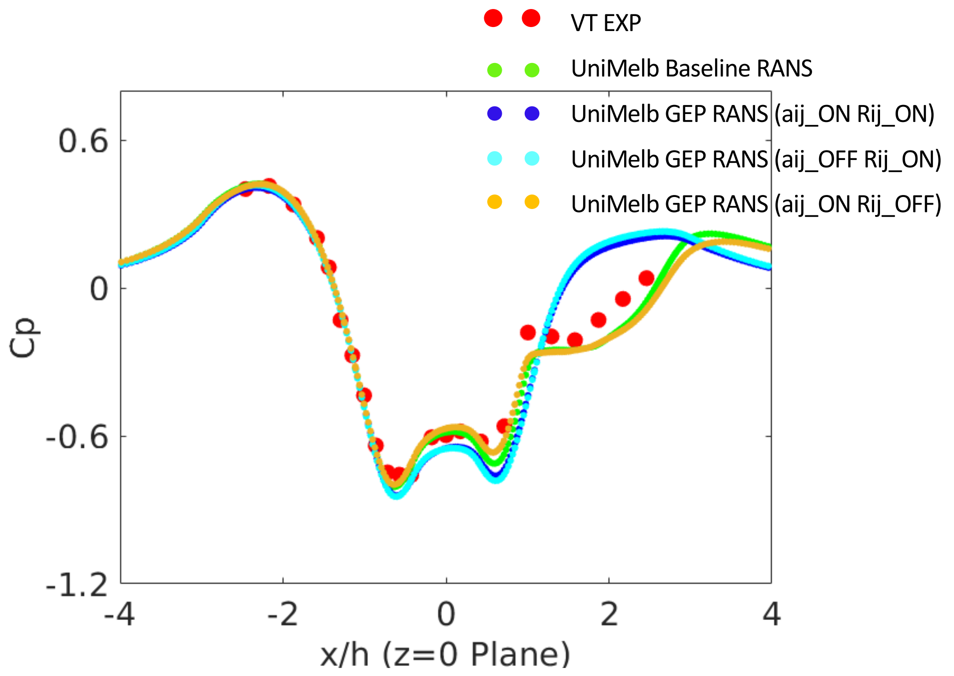
(Schmelzer et al., 2020)

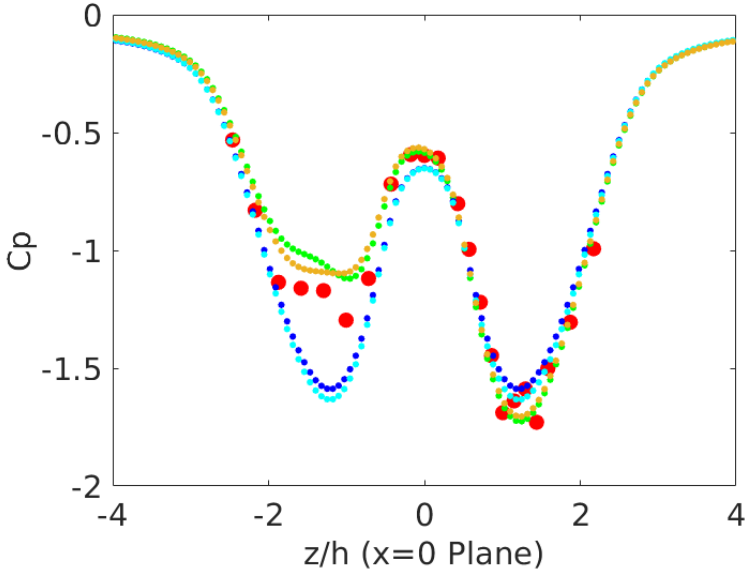
Gene Expression Programming – statistically 3D flows

BUT: 3D bump case (BeVERLI Hill) from NATO AVT-349



- For this case, production correction term results in too high TKE levels over bump, changing or even preventing separation entirely
- Ideally, should train new model on more similar problem (smooth surface)?
- Other ways to improve model consistency?





Gene Expression Programming – CFD-driven training

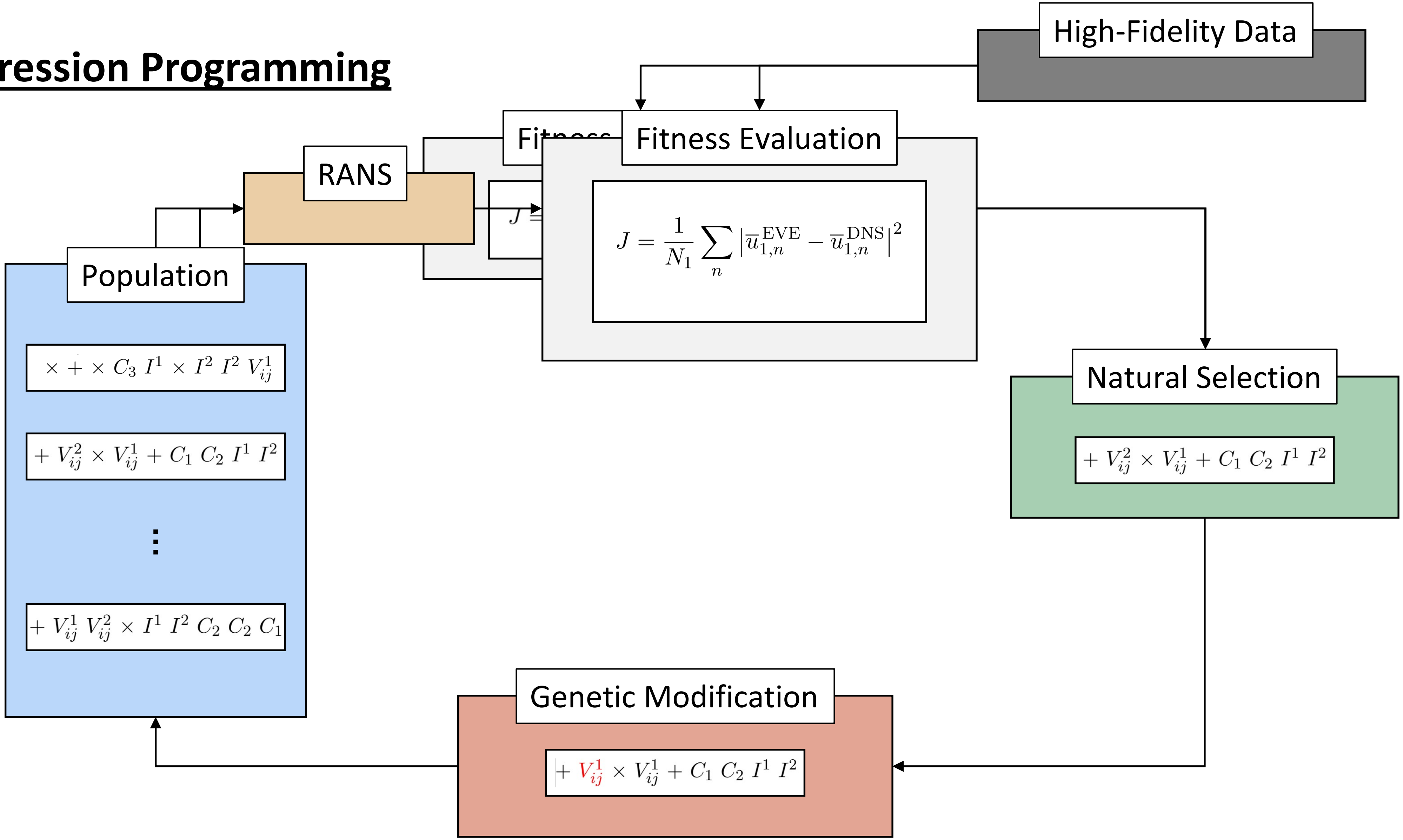
‘Frozen’ Gene-Expression Programming

‘CFD-driven’ GEP

A-posteriori cost fcn – external code

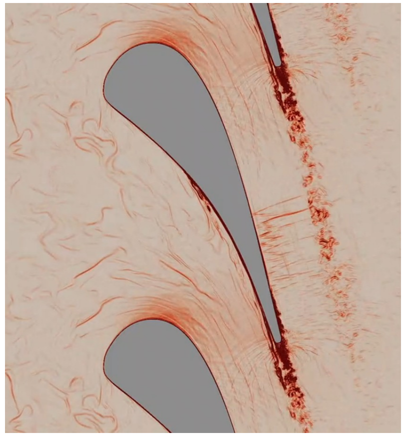
Benefits:

- Built-in model consistency
- Flexible choice of variables in objective function
- Reduced amount of required high-fidelity data



Gene Expression Programming – CFD-driven training

Model trained on HPT data at Re=570,000



$$\tau_{ij} - \frac{2}{3}\rho k \delta_{ij} = -2\mu_t S'_{ij}$$

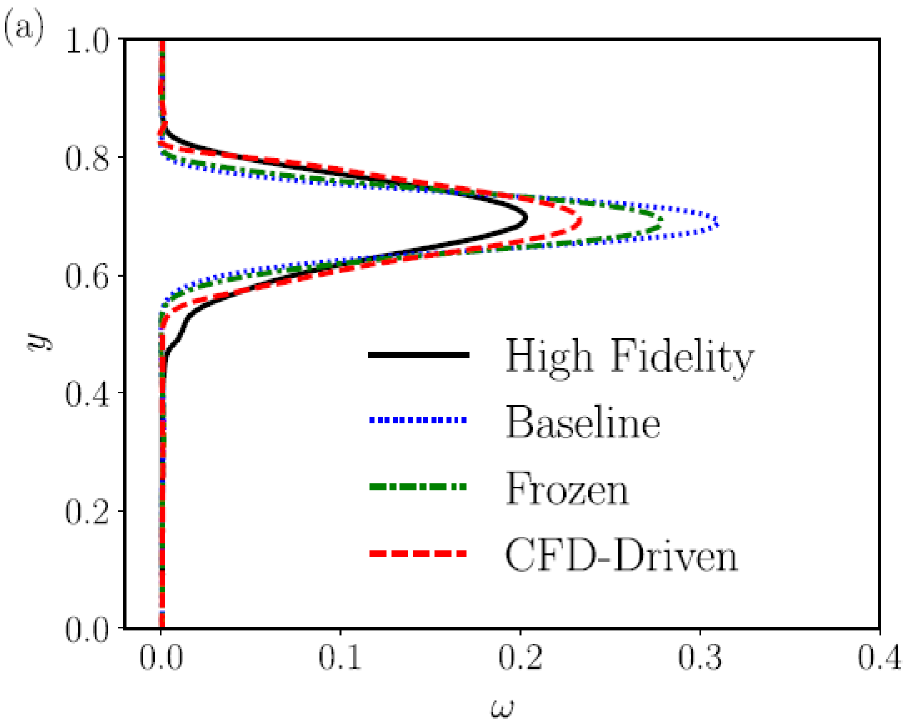
Standard linear model (baseline)

$$+2\rho k\left[(-3.57 + I_1)\,V^1_{ij} + 4.0V^2_{ij} + \left(-0.11 + 0.09 I_1 I_2 + I_1 I_2^2\right)V^3_{ij}\right]$$

Machine-learnt model extension

Much simpler expression than from 'frozen' training

$$\tau^{fro}_{ij} = \frac{2}{3}\rho k \delta_{ij} - 2\mu_t S'_{ij} + 2\rho k[$$
$$(-1.334 + 0.438 I_1 + 2.653 I_2 + 0.0102 I_1^2 - 1.021 I_2^2 + 12.280 I_1 I_2)V^1_{ij}$$
$$+ (0.573 - 1.096 I_1 + 8.985 I_2 - 0.1102 I_1^2 + 2.876 I_2^2 + 90.633 I_1 I_2)V^2_{ij}$$
$$+ (12.861 - 25.094 I_1 + 6.449 I_2 + 1.020 I_1^2 - 304.979 I_1 I_2 - 184.519 I_2^2)V^3_{ij}]$$

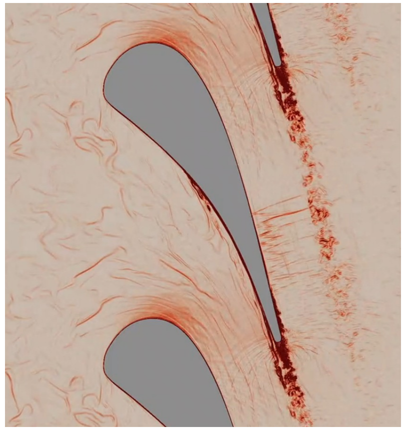


Error reduced by factor > 5

(Zhao et al., JCP 2020)

Gene Expression Programming – CFD-driven training

Model trained on HPT data at Re=570,000



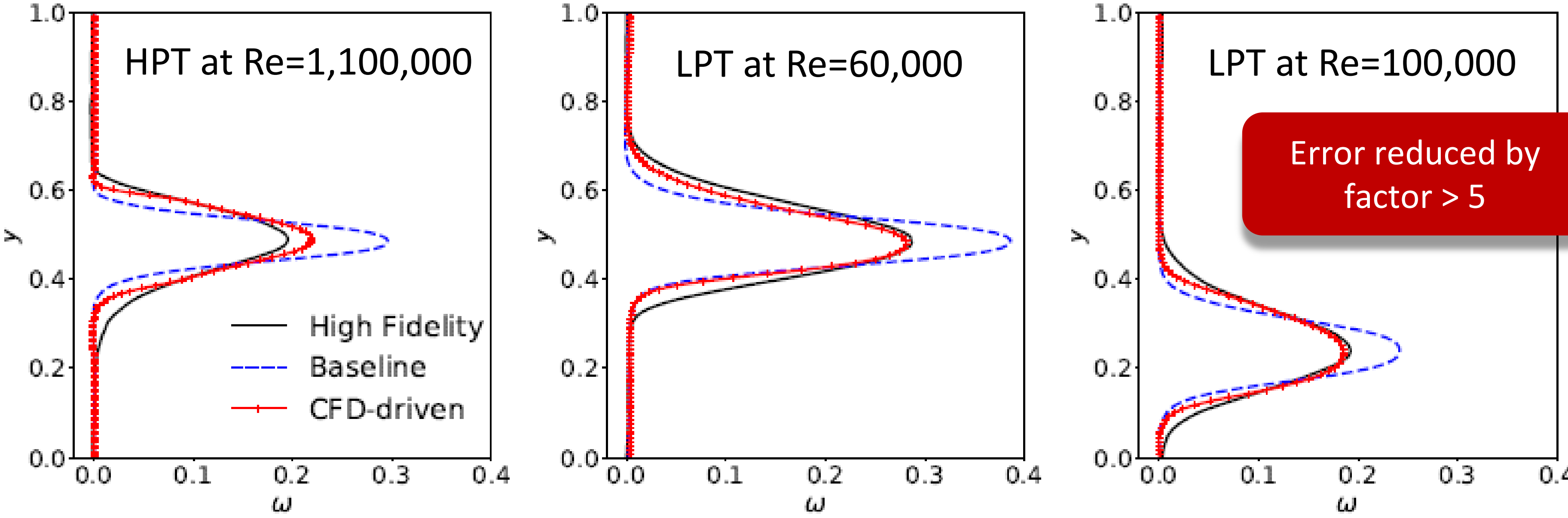
Standard linear model (baseline)

$$\tau_{ij} - \frac{2}{3}\rho k \delta_{ij} = -2\mu_t S'_{ij}$$

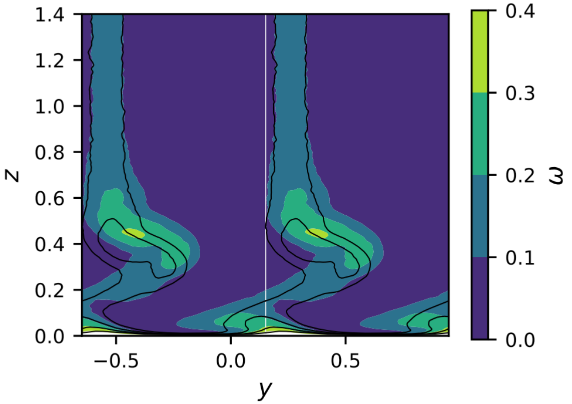
$$+2\rho k \left[\left(-3.57 + I_1\right) V_{ij}^1 + 4.0 V_{ij}^2 + \left(-0.11 + 0.09 I_1 I_2 + I_1 I_2^2\right) V_{ij}^3 \right]$$

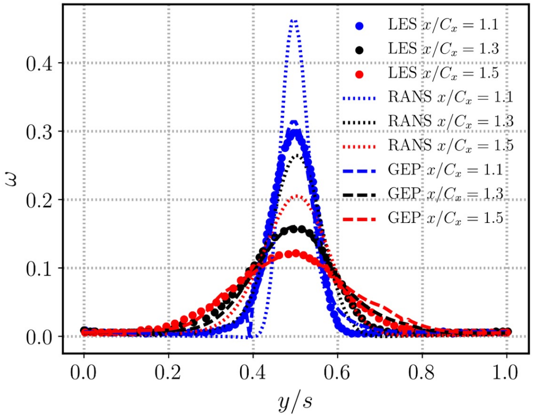
Machine-learnt model extension

Tested on:



Error reduced by factor > 5



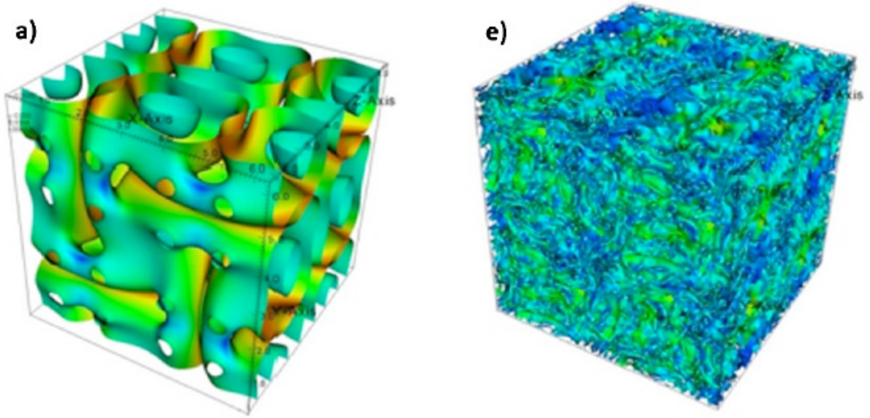


New model trained on one data set performs well on all test cases, at different flow conditions and for different geometries

Gene Expression Programming – CFD-driven training

CFD-driven LES (Reissmann et al., *JCP* 2020)

- Need to run 1,000s or 10,000s of LES – need to be ‘affordable’
- Pick Taylor-Green-Vortex as test problem
- Demanding for SGS-models as it features laminar-turbulent transition



$$\tau_{ij}^{GEP} = -2\Delta^2 \left|\overline{S}\right|^2 \sum_{k=1}^{n} \xi_k \left(I_1, \ldots, I_n\right) V_{ij}^k$$

With inverse time scale $\left|\overline{S}\right| = \sqrt{\overline{S}_{mn}\overline{S}_{nm}}$

LES setup

- Incompressible solver (PARIS (Ling et al, Int J Multiph Fl. 2015)) – 0.75core h/run (10,000)
- Re=1,600
- Grid with 32^3 grid points
- Reference DNS with 256^3 grid points

Cost function

$$J(\varphi) = \frac{1}{T}\int_0^T \frac{\left|\varphi_{DNS}(t) - \varphi_{LES}(t)\right|}{\left|\varphi_{DNS}(t)\right|}$$

$$J^{tot} = \frac{3}{5} J(TKE) + \frac{2}{5} J(\epsilon)$$

Machine-Learning (GEP)

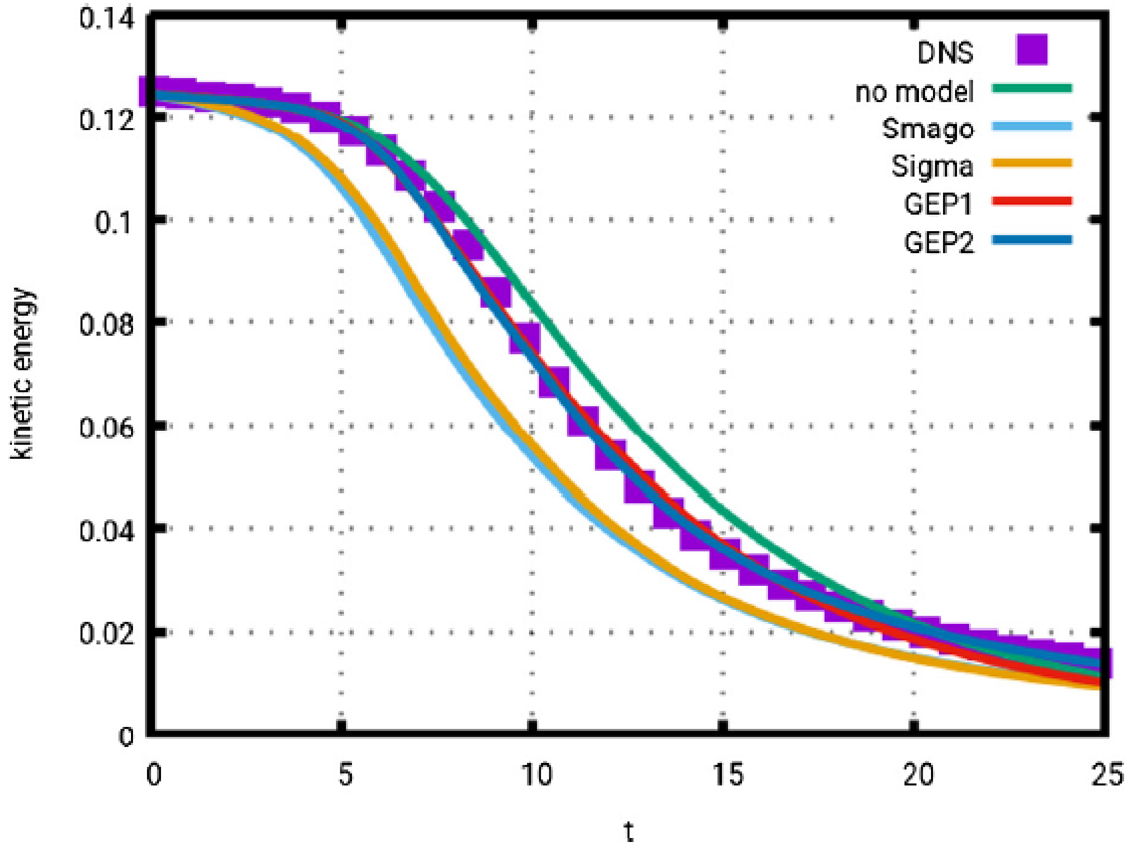
CFD-driven LES

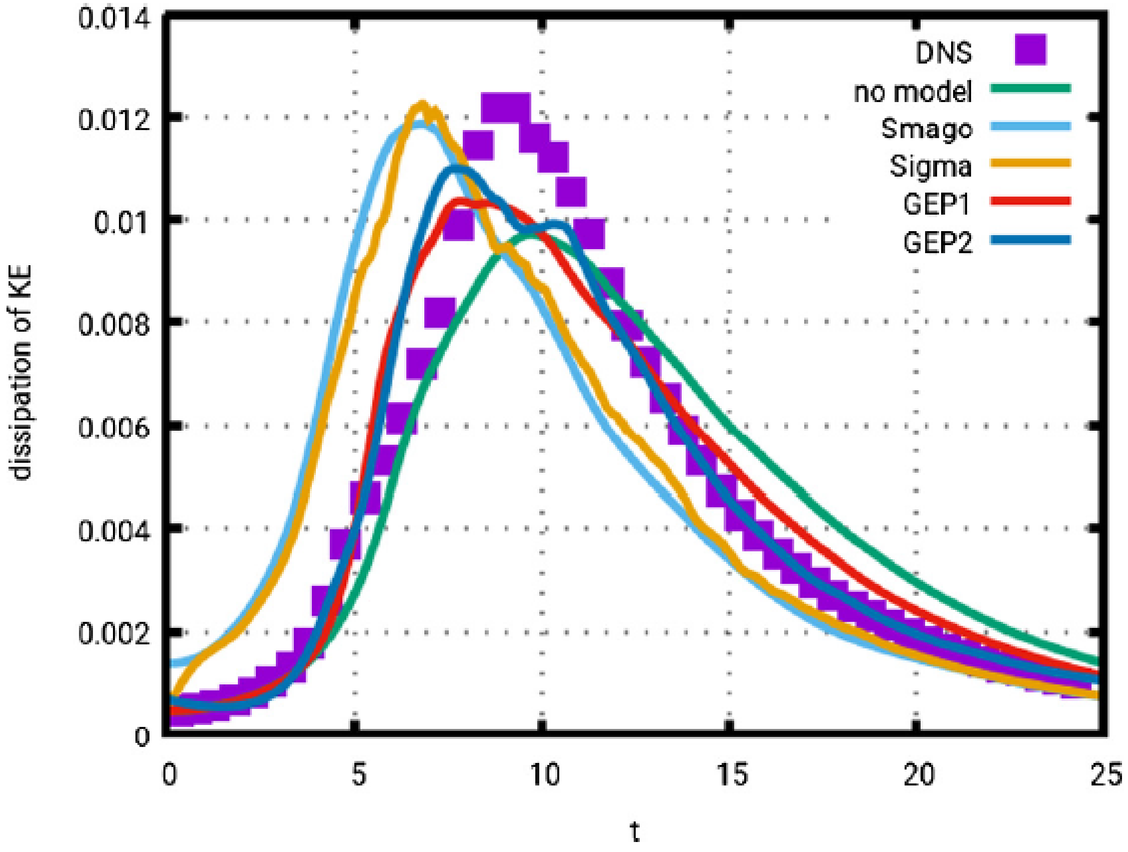
(Reissmann et al., *JCP* 2020)

$$\tau_{ij}^{GEP1} = -2\Delta^2 \left\{ -\left(I_3 + 0.04\right) V_{ij}^2 \right\}$$

$$\tau_{ij}^{GEP2} = -2\Delta^2 \left\{ 0.01 \left|\overline{S}\right| V_{ij}^1 - 0.146 V_{ij}^2 + 0.01 V_{ij}^3 - 0.011 V_{ij}^4 \right\}$$

Model	$J^{tot}\ (K) \times 100$
No Model	9.67
Clark	11.78
Smago	21.45
Sigma	20.54
Mixed	18.64
GEP1	5.59
GEP2	1.51





Machine-Learning (GEP)

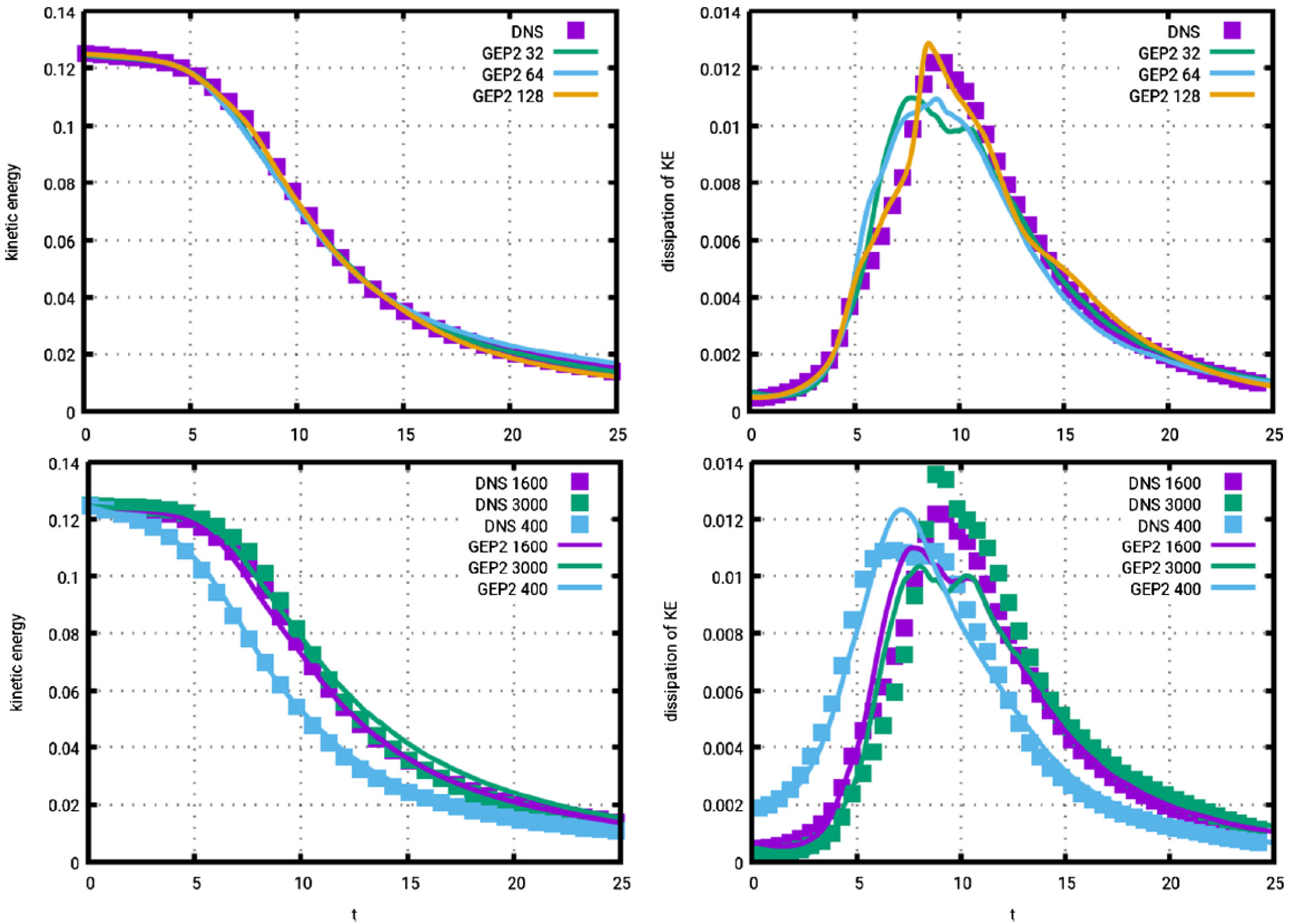
CFD-driven LES

Robustness of GEP2 model

$$\tau_{ij}^{GEP2} = -2\Delta^2 \left\{0.01 \left|\overline{S}\right| V_{ij}^1 - 0.146 V_{ij}^2 + 0.01 V_{ij}^3 - 0.011 V_{ij}^4\right\}$$

GEP2 model produces good results for different LES resolutions

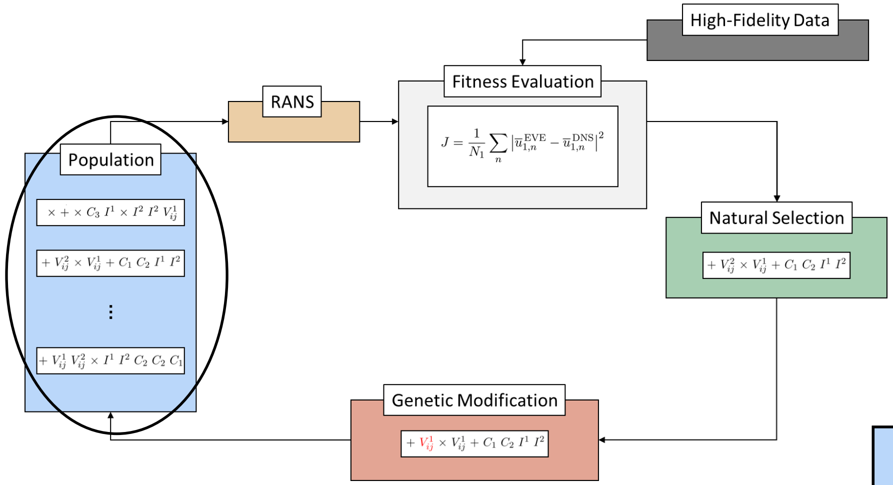
GEP2 model works well for different Reynolds numbers



Gene Expression Programming – Multi-expression

(Waschkowski et al., 2021)

Multi-expression GEP training



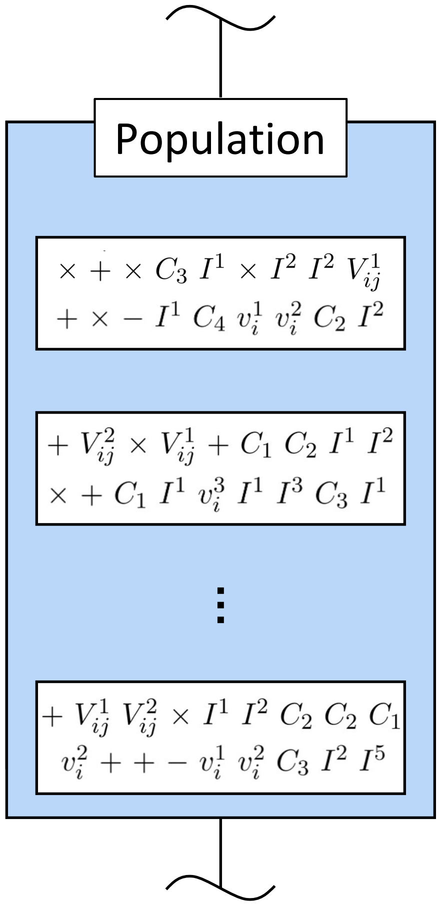
Motivation: Capturing coupling effects when training multiple closure models

Idea:

Extension of candidate solutions from one expression to multiple expressions

Assignment of shared fitness value to each set of expressions

Exchange of genetic material only between alike expressions



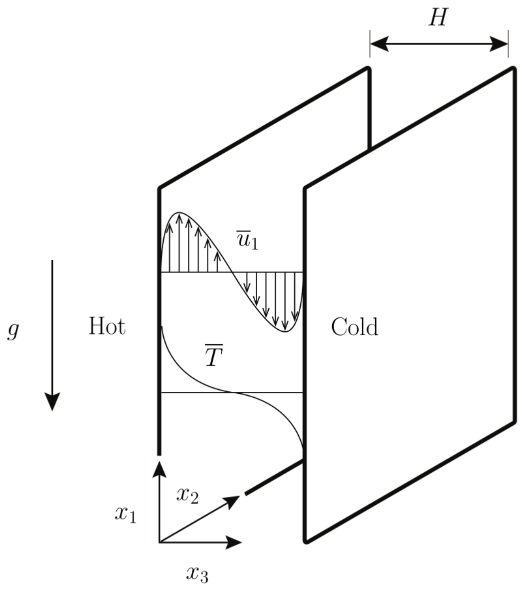
Pope (1975): $$a_{ij} = \sum_{k=1}^{10} g^k \left(I^1, I^2, ..., I^5\right) V_{ij}^k$$

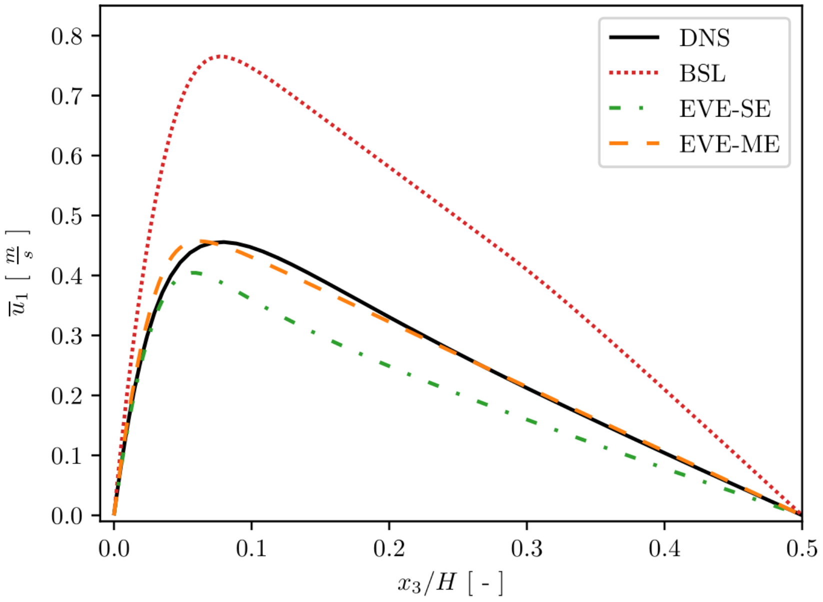
Zheng (1994): $$\overline{u_i' T'} = \sum_{k=1}^{6} h^k \left(I^1, I^2, ..., I^{13}\right) v_i^k$$

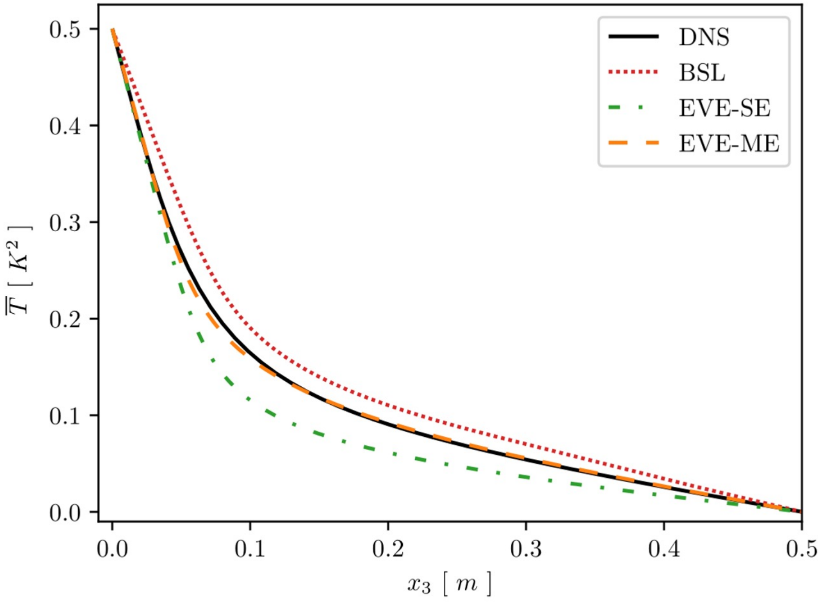
Multi-expression GEP training

Example: Vertical natural convection

$$J = \frac{1}{N_1} \sum_n \left| \overline{u}_{1,n}^{\mathrm{EVE}} - \overline{u}_{1,n}^{\mathrm{DNS}} \right|^2 + \frac{1}{N_2} \sum_n \left| \overline{T}_n^{\mathrm{EVE}} - \overline{T}_n^{\mathrm{DNS}} \right|^2$$





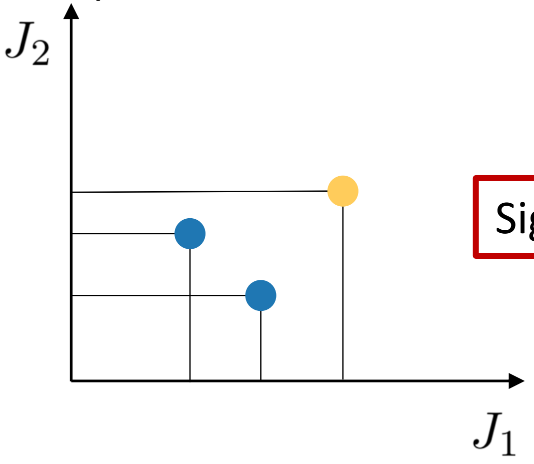


How to know beforehand whether we need weights for cost function?

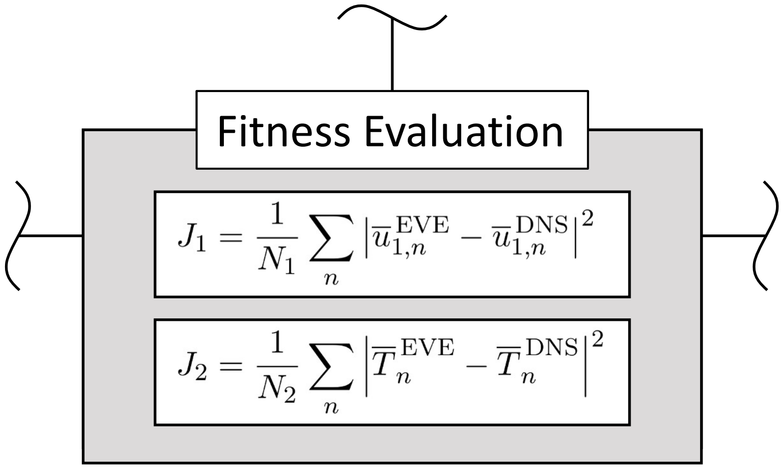
Gene Expression Programming – Multi-objective

Multi-objective GEP training

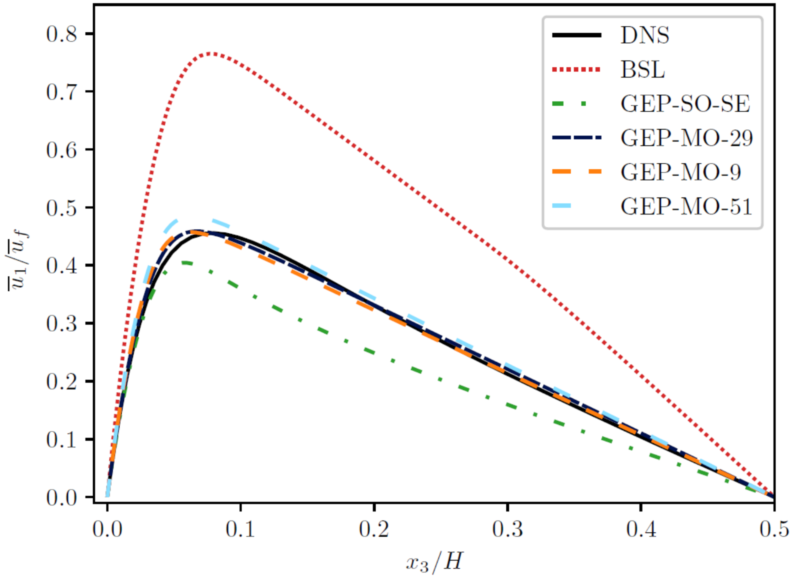
Example: Vertical natural convection

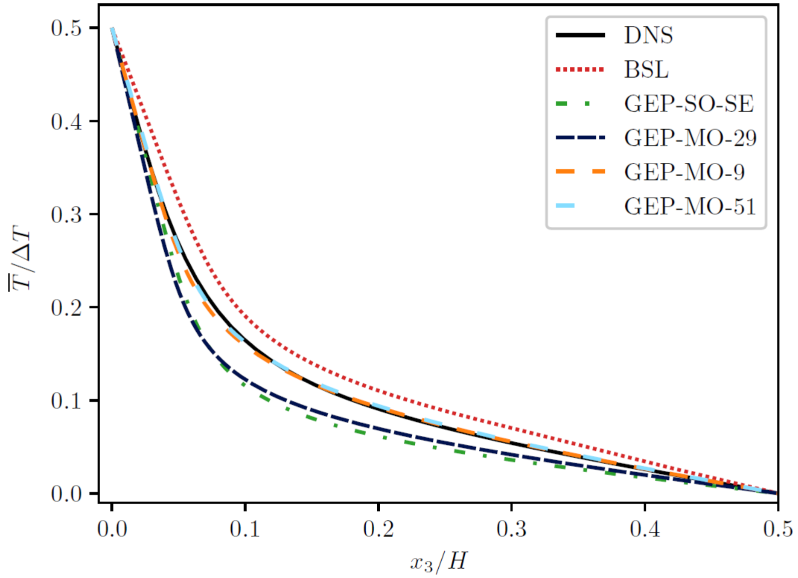


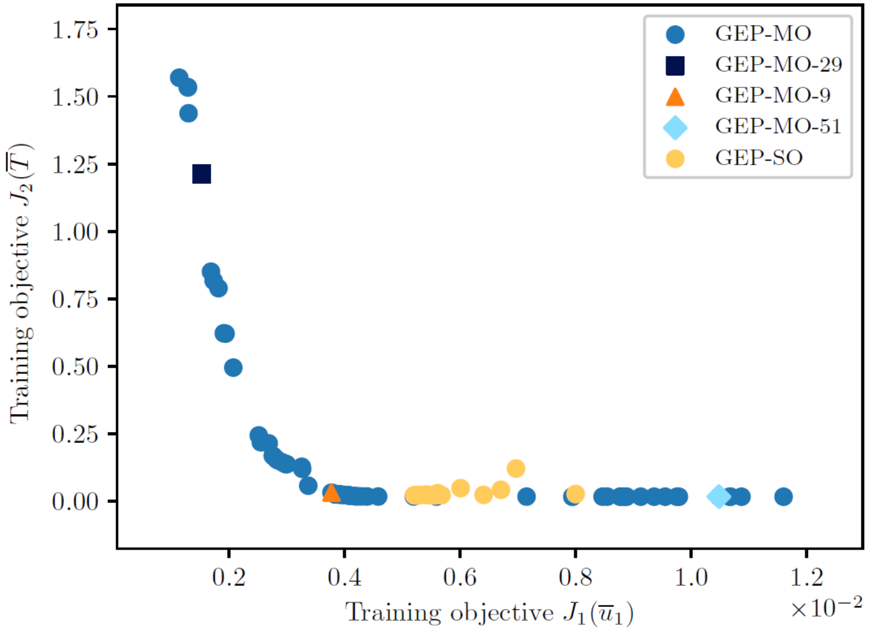
Idea: NSGA-II algorithm (Deb, 2002)
Pareto domination to minimize separate training objectives



Significant benefits when expressions strongly coupled!







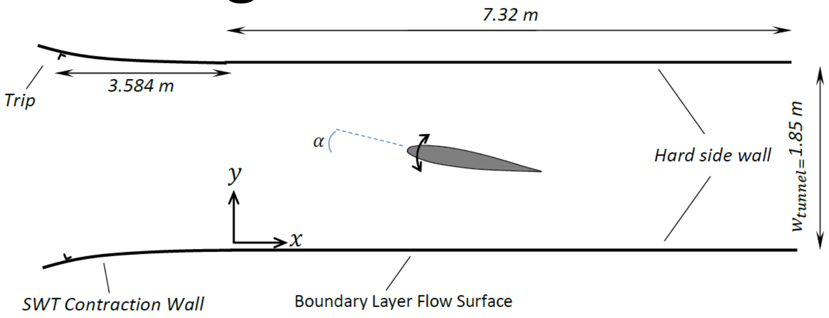
(Waschkowski et al., JCP 2021)

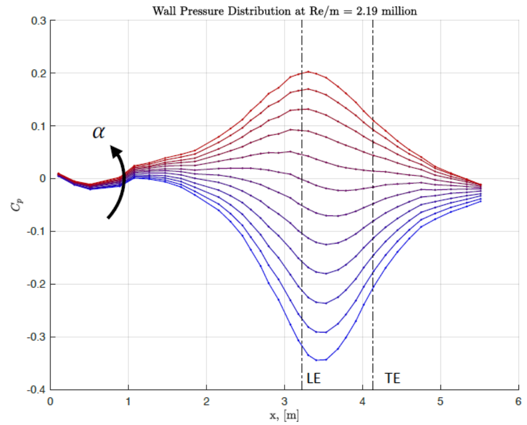
Gene Expression Programming – Multi-objective

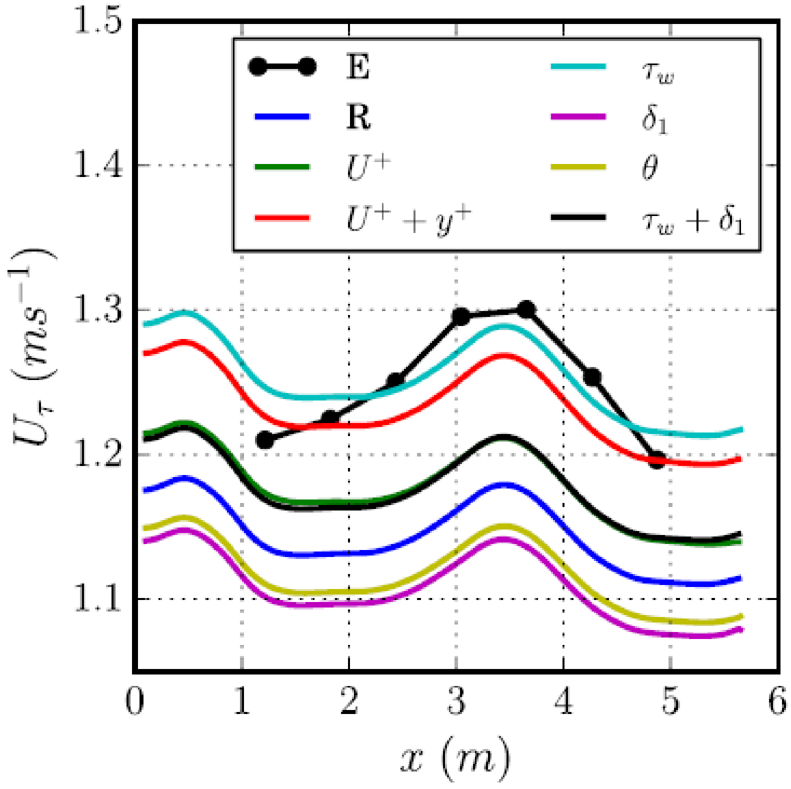
Turbulent boundary layers in pressure gradients – data from VT experiments

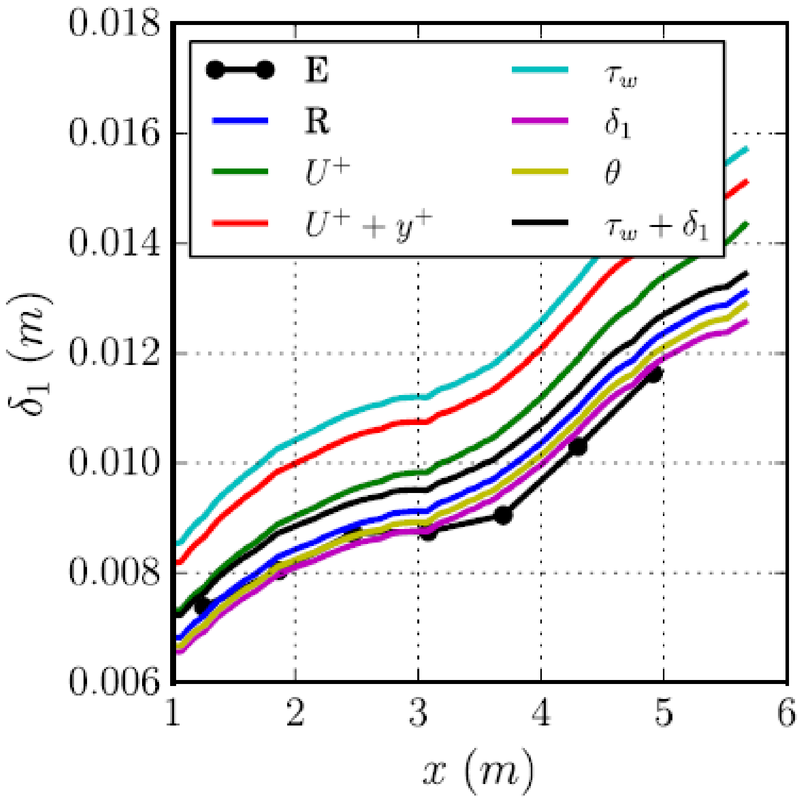
Importance of cost functions

- Have used novel multi-objective optimization (e.g. U and τ_w)









Legends:

- E: Experiment
- R: Baseline RANS
- U^+ : CFD-driven using U^+ as cost function
- $\tau_w + \delta_1$: CFD-driven using τ_w and δ_1 as cost function

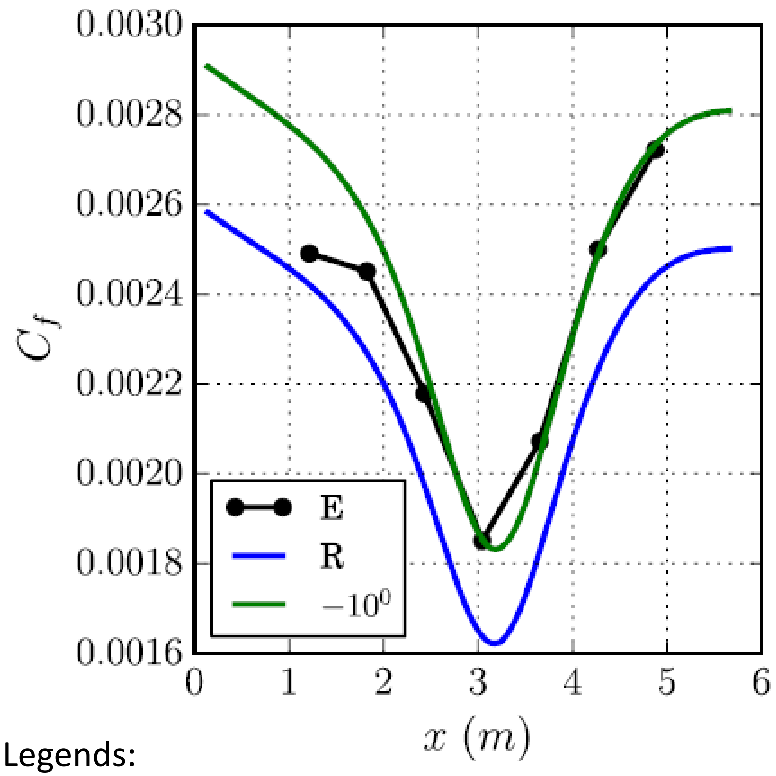
Decided to use τ_w and δ_1 as cost functions

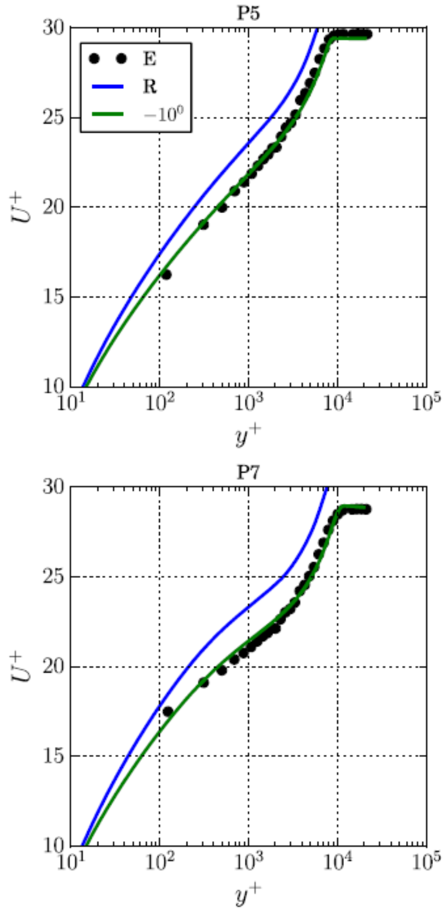
Gene Expression Programming – Multi-objective

Multi-objective model training, using -10° at Re=2.5 x 10^6 data, yields:

$$a_{ij}^{x} = -0.15\, V_{ij}^{1} + 0.43\, V_{ij}^{2} + (I_2 + 1)V_{ij}^{3}$$

Testing on 12° case at Re=3.6 x 10^6





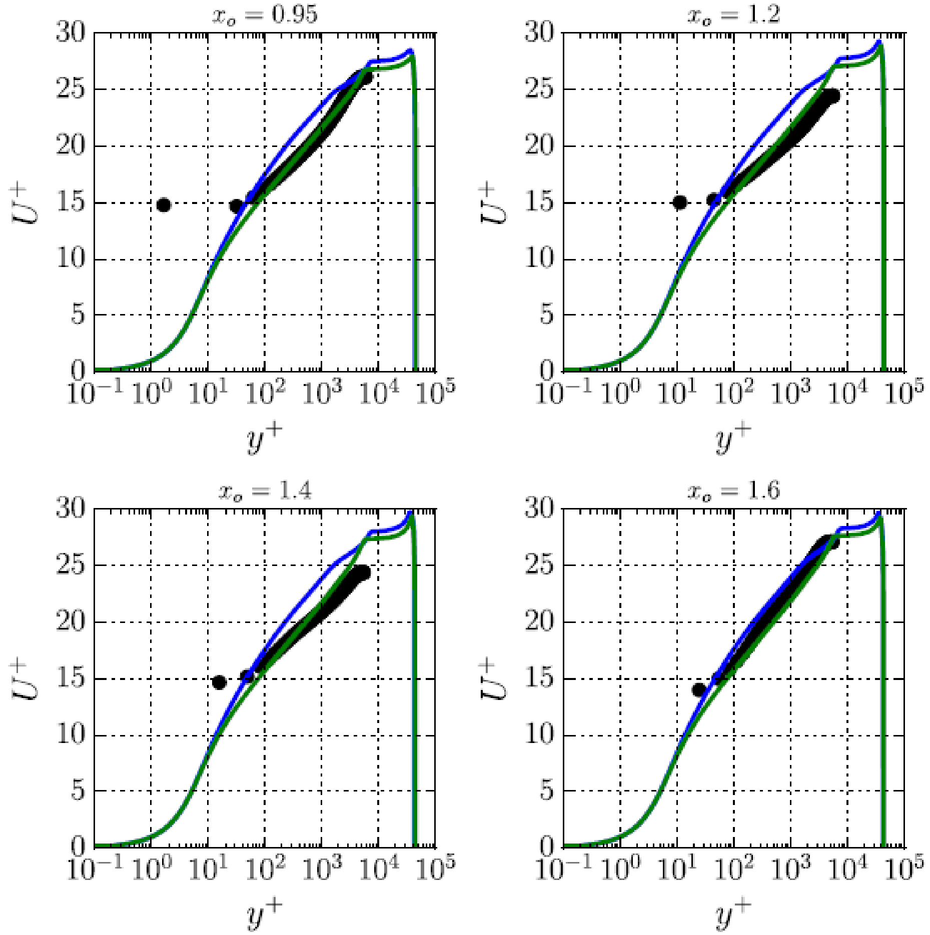
Legends:

- E: Experiment
- R: Baseline RANS
- -10^o : Model trained on case at -10°

(Lav & Sandberg, SNH 2022)

Can we generalize to a completely different case?

Used UniBW and DLR smooth wall setup (10m/s)



→ downstream not so good

Needed: stronger PG datasets

Gene Expression Programming – Multi-objective

(Akolekar et al., 2022)

Development of improved transition modelling <u>and</u> wake mixing modelling for LPT

2 expressions: modify/extend terms in Laminar Kinetic Energy Transition model

3 objectives :

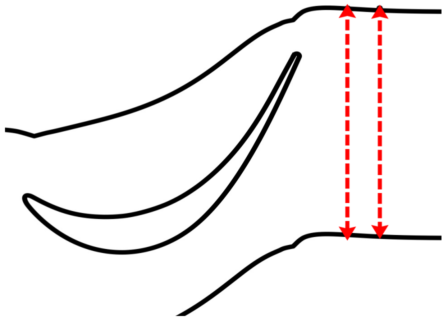
$$J^{MO1} = \sum_{x/C_{ax}=0.6}^{1} \left(\tau_w^{DNS} - \tau_w^{RANS}\right)^2_{SS},$$

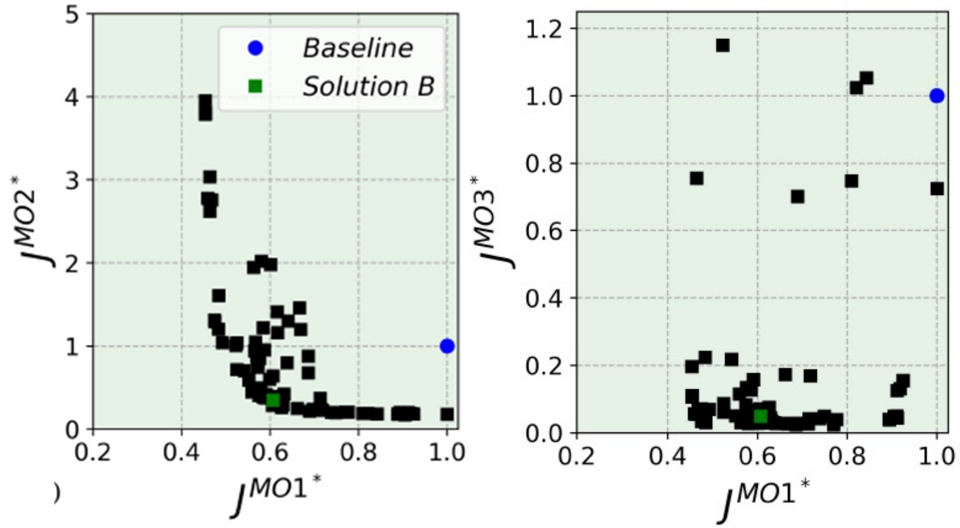
$$J^{MO2} = \sum_{x/C_{ax}=0.6}^{1} \left(C_p^{DNS} - C_p^{RANS}\right)^2_{SS}.$$

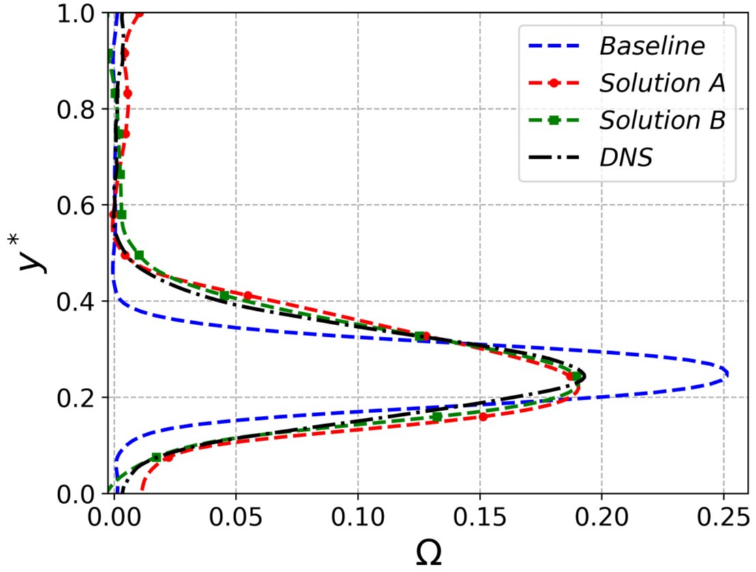
$$\Omega^*(y) = \frac{p_{01} - p_{02}(y)}{p_{01} - p_2},$$

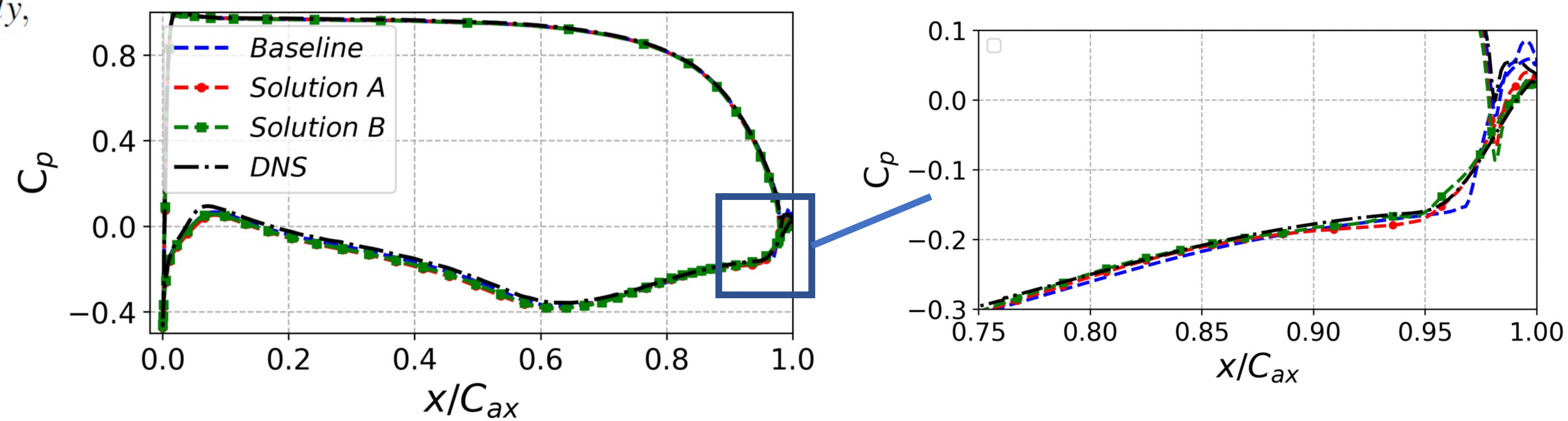
$$\Delta_C = \frac{1}{w} \int_0^w \left(\frac{\Omega^*_{DNS} - \Omega^*_{RANS}}{max(\Omega^*_{DNS})}\right)^2 dy,$$

$$J^{MO3} = \Delta_{C1} + \Delta_{C2},$$









Gene Expression Programming – Adaptive Symbols

Introduction of additional, adaptive symbols

Motivation: Challenge for GEP to learn accurate numerical constants (Zhong et al., 2017)

$$S = \left\{V_{ij}^{k}, I^{l}, C_{m}, +, -, \times\right\} \qquad \mathrm{C} = \{-1, 1, 2\} \cup \{-0.67, -0.32, 0.11, 0.35, 0.82\}$$

Idea: / **Learn:** $\mathrm{C}^{*} = 9.99$ **adaptive symbol** values during training via gradient-based numerical optimizers

$$\times\ 2\ 2$$

$$\mathrm{C} = \left\{p_1^*, p_2^*, \ldots, p_n^* \mid p^* = \mathrm{argmin}_{p \in \mathbb{R}^n} J(p)\right\}$$

$$+\ \times\ \times\ 2\ 2\ 2\ 2 \longrightarrow 2 \cdot 2 \cdot 2 + 2 = 10$$

$$-\ +\ \times\ \times\ 2\ 2\ 2\ 2\ \times\ 0.11\ 0.11 \longrightarrow 2 \cdot 2 \cdot 2 + 2 - 0.11 \cdot 0.11 = 9.9879$$

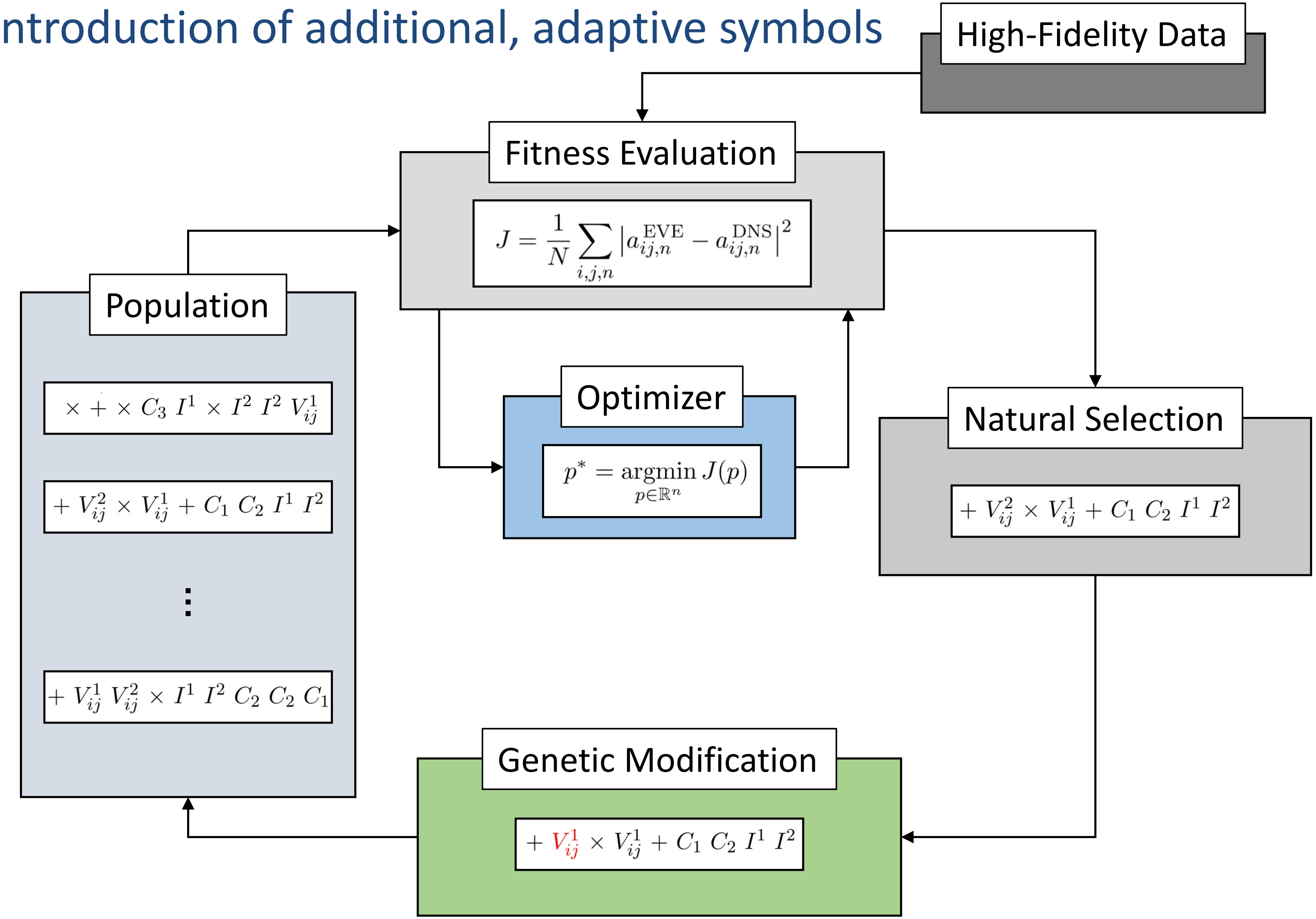
Optimizers:

Broyde… algorithm

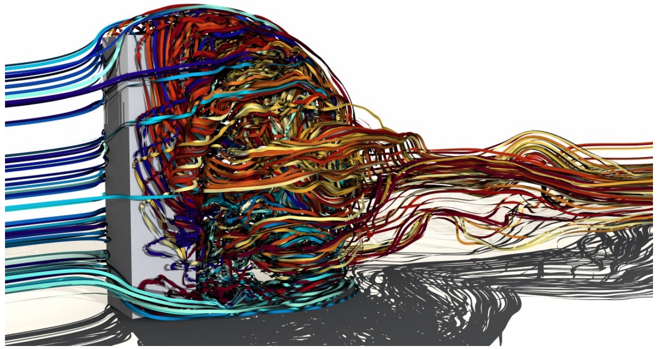
Levenberg-Marquardt (LM) algorithm

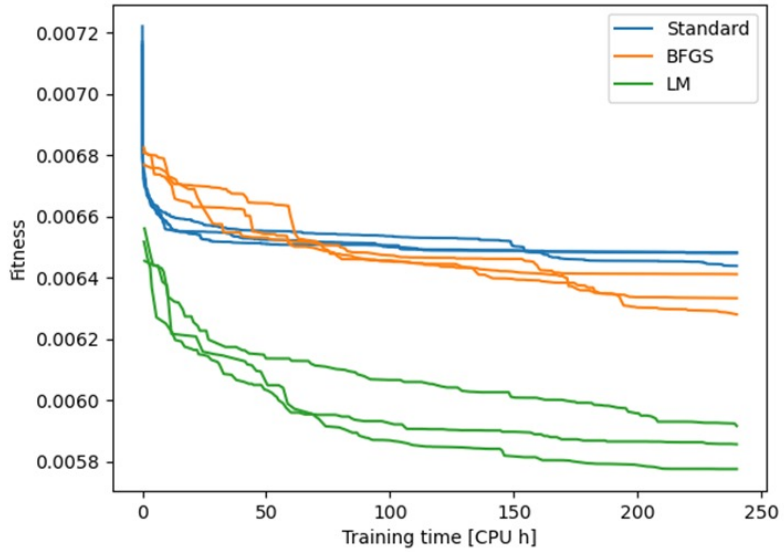
Gene Expression Programming – Adaptive Symbols

Introduction of additional, adaptive symbols



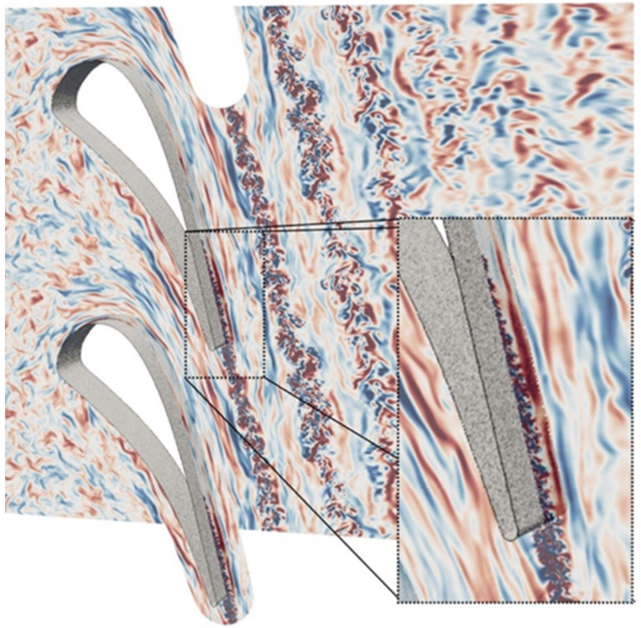
Wall-mounted square cylinder





Summary & Outstanding Issues

- GEP produces CFD-ready and interpretable turbulence closures
- CFD-driven GEP produces model-consistent closures
  - Only requires limited data
  - Allows for multi-expression training in which model interactions are considered
  - Enables multiple objectives (any quantity) to be met



Issues

- CFD-driven too costly for complex (3D) geometries and high Re
  - What is needed to make its use more practical?
- Are we using correct input features and basis functions?
- How do we ensure better generalizability of models?
  Will we have enough data?
- GEP not that good in finding ‘hidden features’, patterns in data → can leverage NNs?

Thank you for your attention

Questions?

THE UNIVERSITY OF MELBOURNE

OAK RIDGE National Laboratory

NCI AUSTRALIA

PAWSEY supercomputing centre

Backup

Machine-Learning (GEP) for unsteady flows

Approach 1: Use phase-lock averaged DNS data to train models

Not practical to apply a different model for each phase

→ chose two models, one each for which additional diffusion and non-linear terms dominate

- Mod-PA (one model for each phase) best performance
- Mod-TA (model from time-average) worst performance
- Mod-Phy (2 models based on flow physics) quite good overall

Transition modeling

Development of improved transition modeling

(Akolekar et al., 2021)

Modify/Extend Laminar Kinetic Energy Transition model (Pacciani et al., 2011)

$$\frac{Dk_l}{Dt} = P_l - 2\nu\frac{k_l}{y^2} + \nu\nabla^2 k_l - R$$

$$P_l = \nu_l S^2,$$

$$\nu_l = C_1 f_1 \sqrt{k_l}\delta_\Omega,$$

$$f_1(Tu) = max\left[0.8, 2.0\cdot tanh\left(\left(\frac{Tu}{4.5}\right)\right)\right],$$

$$\delta_\Omega = max_y\left(\frac{U}{\Omega}\right),$$

$$R = C_2\beta^* f_2\omega k_l,$$

$$f_2 = 1 - e^{-\psi/C_3},$$

$$\psi = max(0, R_y - C_4),$$

$$R_y = \frac{\sqrt{k}y}{\nu},$$

Laminar eddy viscosity

$$\nu_l = f_{1a} y\sqrt{k_l},$$

$$f_{1a} = f(\Pi_i)$$

Transition parameter

$$\psi = max(f_{2a}(\Pi_i), 0)$$

Non-dimensional Pi groups

$$\Pi_1 = \frac{k_l}{\nu\Omega};\ \Pi_2 = \frac{\Omega y}{U};\ \Pi_3 = \frac{y}{l_t};\ \Pi_4 = \frac{\sqrt{k}y}{\nu};$$

$$\Pi_5 = \frac{k}{\nu\Omega};\ \Pi_6 = \frac{Sy}{U};\ \Pi_7 = \frac{\omega}{\Omega},$$

Multi-objective modelling (multi-expression)

$$J^{MO1} = \sum_{x/C_{ax}=0.6}^{1}\left(\tau_w^{DNS} - \tau_w^{RANS}\right)^2,$$

$$J^{MO2} = \sum_{x/C_{ax}=0.6}^{1}\left(C_p^{DNS} - C_p^{RANS}\right)^2.$$