RESEARCH

Gradient Information and Regularization for Gene Expression Programming to Develop Data-Driven Physics Closure Models

Fabian Waschkowski[1] · Haochen Li[2] · Abhishek Deshmukh[3] · Temistocle Grenga[3] · Yaomin Zhao[2] · Heinz Pitsch[3] · Joseph Klewicki[1] · Richard D. Sandberg[1]

Abstract

Learning accurate numerical constants when developing algebraic models is a known challenge for evolutionary algorithms, such as Gene Expression Programming (GEP). This paper introduces the concept of adaptive symbols to the GEP framework by Weatheritt and Sandberg (J Comput Phys 325:22–37, 2016a) to develop advanced physics closure models. Adaptive symbols utilize gradient information to learn locally optimal numerical constants during model training, for which we investigate two types of nonlinear optimization algorithms. The second contribution of this work is implementing two regularization techniques to incentivize the development of implementable and interpretable closure models. We apply L_2 regularization to ensure small magnitude numerical constants and devise a novel complexity metric that supports the development of low complexity models via custom symbol complexities and multi-objective optimization. This extended framework is employed to four use cases, namely rediscovering Sutherland's viscosity law, developing laminar flame speed combustion models and training two types of fluid dynamics turbulence models. The model prediction accuracy and the convergence speed of training are improved significantly across all of the more and less complex use cases, respectively. The two regularization methods are essential for developing implementable closure models and we demonstrate that the developed turbulence models substantially improve simulations over state-of-the-art models.

Keywords Gene Expression Programming · Nonlinear optimization · Regularization · Model complexity

✉ Fabian Waschkowski
   fwaschkowski@student.unimelb.edu.au

[1] Department of Mechanical Engineering, University of Melbourne, Melbourne, VIC 3010, Australia

[2] Center for Applied Physics and Technology, Peking University, Beijing 100871, China

[3] Institute for Combustion Technology, RWTH Aachen University, 52062 Aachen, Germany

1 Introduction

The development of physics models is at the core of most science and engineering disciplines. With the goal of understanding physical systems, predicting their future states and discovering governing laws, researchers have been developing physics models for centuries. The ability to predict the dynamics of a system-of-interest allows the design of advanced technologies, from microprocessors to spacecraft engines. Traditionally, such predictive models have been conceived based on physical, mathematical and empirical insights (Durbin 2018). Due to high costs of physical and numerical experiments in the past, data was limited and primarily used to verify and calibrate models (Montáns et al. 2019).

The rapid increase in computational resources in the last decades, however, enabled the generation of large quantities of high-fidelity data. Additional advances of machine learning (ML) algorithms, which are able to utilize big datasets, motivated a paradigm shift from traditional model development to data-driven modeling in various scientific fields (Montáns et al. 2019). The most frequently applied ML models are deep neural networks (DNN) (Goodfellow et al. 2016), which demonstrated remarkable success on previously intractable problems, from computer vision initially to protein folding more recently (Krizhevsky et al. 2012; Jumper et al. 2021). While DNNs posses excellent predictive capabilities on high-dimensional datasets, the trained models are difficult to interpret and have been shown to generalize poorly to data outside of the training distribution for nonlinear target functions (Xu et al. 2021). Furthermore, the models resulting from DNN training or decision tree-based methods, another often applied type of ML algorithm, are highly complex equations or algorithmic models. For the application as closure models, i.e. the implementation in underdetermined systems of equations, which are the focus of this work, these models are difficult to use due to stability issues during the numerical solution of equation systems.

An alternative to such highly complex models are algebraic models, which are typically the result of traditional model development. However, Schmidt and Lipson (2009) used genetic programming (GP) (Koza 1992), which is an evolutionary algorithm, to symbolically regress algebraic models from experimental data. This data-driven approach yields interpretable and implementable models. Cranmer et al. (2020) showed that algebraic models derived via symbolic regression even generalize better to out-of-distribution data than DNNs for different physical problems.

Ferreira (2001) developed with Gene Expression Programming (GEP) an algorithm that improves over GP by introducing a genotype-phenotype distinction (see Sect. 2.1 for details), as inspired by natural evolution. While symbolic regression via GP or GEP is popular due to a high flexibility in the resulting model structure, the problem of not converging to accurate numerical constants in the model equation is well known (Ryan and Keijzer 2003; Zhong et al. 2017). Typically, a finite number of numerical constants are created before the training and other constants in the model equation are only achieved by combining existing constants via mathematical operators. Thus, converging to specific constants is challenging, especially since evolutionary algorithms are fundamentally stochastic.

A different approach for the development of algebraic models was devised by Brunton et al. (2016), which performs linear regression on a library of equation snippets with sparsity-enforcing regularization. This approach, which is termed sparse regression, computes numerical constants accurately using a deterministic linear least squares

solver. A considerable downside is, however, that the possible model structures are limited to linear combinations of the equation snippets in the library.

The goal of this paper is combining the advantages of symbolic and sparse regression to develop data-driven closure models with flexible structures and accurate numerical constants. We extend the GEP framework developed by Weatheritt and Sandberg (2016a) to incorporate gradient information to optimize model constants. In the literature, both gradient-free and gradient-based methods for constant optimization in GEP exist. Li et al. (2004) investigated random and creep mutation operators that modify numerical constants during training. Other gradient-free methods applied to this problem are hill climbing (Lopes and Weinert 2004) and differential evolution (Zhang et al. 2007). To improve over these stochastic methods, Zarnegar et al. (2009) employed a linear least squares solver to calculate the values of constants pre-multiplied to trained basis functions. Dominique et al. (2021) defined a special power operator, of which the exponent was determined using a gradient-based nonlinear least squares solver.

A novelty of this paper is the introduction of so-called adaptive symbols, which allow gradient-informed numerical constants in GEP in a general form, i.e. at any position in the trained model equation. To determine the values of adaptive symbols p during training, both a general optimizer to solve

$$p^* = \operatorname{argmin}_{p \in \mathbb{R}^n} J(p), \quad \|p - p^*\| < \delta, \quad \delta > 0 \tag{1}$$

for any objective function $J : \mathbb{R}^n \to \mathbb{R}$ and a nonlinear least squares optimizer to solve Eq. (1) for

$$J(p) = \frac{1}{2} \sum_{i=1}^{m} r_i(p)^2, \quad r : \mathbb{R}^n \to \mathbb{R}^m, \quad m \geq n \tag{2}$$

are investigated (Nocedal and Wright 1999). Furthermore, we apply two regularization methods to avoid overfitting the training data and incentivize the development of implementable models. Small magnitudes of numerical constants and a low complexity of the evolved model equation are considered indicators for the numerical stability of closure models. Thus, we implement L_2 regularization (Goodfellow et al. 2016) and define a model complexity metric, which is set as an additional objective function similar to the approach for GP of Schmidt and Lipson (2009).

This new gradient-informed and regularized GEP framework is applied to four different use cases. First, we rediscover Sutherland's viscosity law from generic data as a proof-of-concept. Next, laminar flame speed models are trained on data of unstretched premixed flames. Finally, we develop two types of models for turbulent flow. A subgrid-scale (SGS) model for large eddy simulations (LES) of homogenous isotropic turbulence and a nonlinear eddy viscosity model (NLEVM), including a turbulence production correction model, for Reynolds-averaged Navier–Stokes (RANS) calculations of a three-dimensional flow around a wall-mounted square cylinder are trained.

The structure of this paper is as follows. We briefly discuss the standard GEP framework in Sect. 2.1 and then introduce the concept of adaptive symbols and the optimization algorithms to determine their values in Sect. 2.2. In Sect. 2.3, the regularization techniques that support generalizability and implementability of the developed models are described. Section 3 is organized such that each use case is introduced in detail with its modeling and training strategies before the respective training results are analyzed. Conclusions are drawn in Sect. 4.
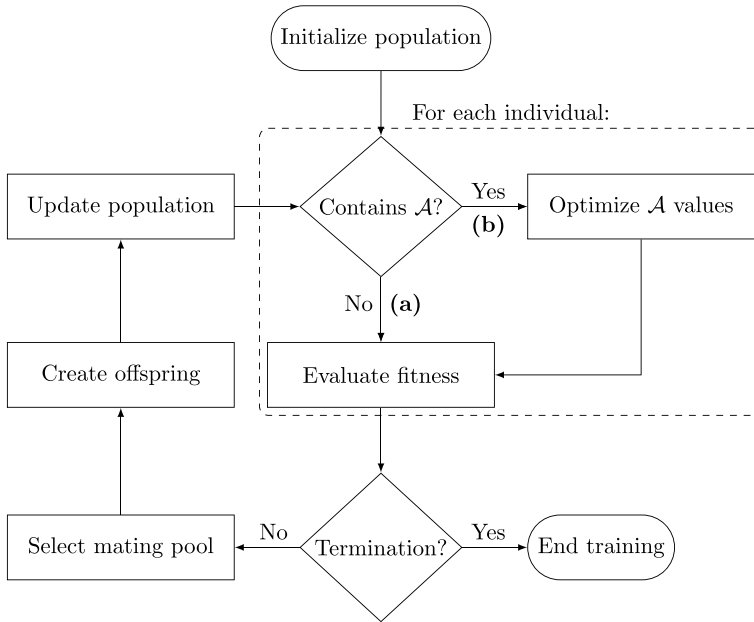
Fig. 1 Flowchart of the GEP framework without (**a**) and with (**b**) the extension to handle adaptive symbols ($\mathcal{A}$)

2 Methodology

The novel contributions of this paper are the introduction of adaptive symbols and the application of L_2 regularization and a model complexity objective function to GEP. The former is discussed in Sect. 2.2, while the latter is described in Sect. 2.3. We start by outlining the utilized GEP framework.

2.1 Standard GEP Framework

Weatheritt and Sandberg (2016a) developed the GEP framework employed in this paper to enable tensor regression, i.a. for turbulence closure modeling. The framework implements the GEP algorithm by Ferreira (2001), which is a type of evolutionary algorithm. As such, a population of candidate models, so-called individuals, is evolved over numerous generations to minimize a specific training objective. Figure 1a illustrates the corresponding flowchart. Initially, a population of random individuals is created. In each generation, the fitness of each individual according to the training objective is evaluated. If the set termination criterion is not fulfilled, individuals compete based on their fitness for selection to the mating pool. Possible termination criteria are a minimum fitness in the population, a specified number of generations or a maximum training runtime. Next, offspring are created by applying genetic operators, such as mutation or crossover, to individuals in the mating pool. Finally, the offspring update the population by replacing unfit individuals.

Structurally, individuals in GEP consist of multiple genes and each gene is encoded as a linear string of symbols, which is referred to as its genotype. Typical symbols are input
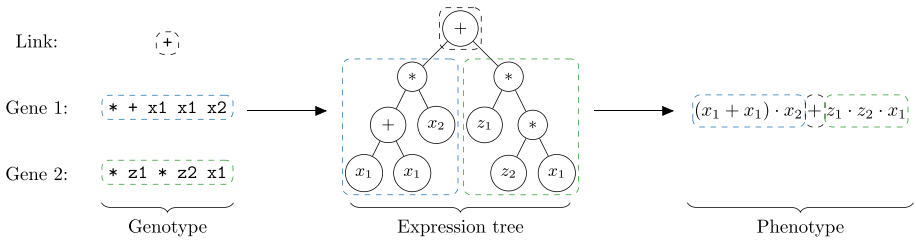
Fig. 2 Translation from genotype to phenotype of exemplary individual consisting of two genes (blue, green) and one link (black)

variables, mathematical operators and numerical constants. To evaluate the fitness of an individual, the genotype of each gene is translated to a nonlinear expression tree, which can be interpreted as an algebraic equation. This equation is referred to as the gene's phenotype. Then, these phenotypes are linked to yield the complete algebraic model, i.e. the individual's phenotype, and the training objective is calculated. Figure 2 shows the translation of an individual with two genes for the input variables x_1 and x_2, addition and multiplication operators and the numerical constants z_1 and z_2.

In comparison to genetic algorithms (GA) (Holland 1975) and GP, the advantage of GEP results from the distinction between the genotype and the phenotype. While genetic operators are applied to the genotype, the phenotype of an individual determines its fitness. This distinction allows easy genetic manipulations and complex functional expressions. In contrast, genes in GA are implemented as linear strings only, which limits their functional expressivity, and genes in GP are represented as expression trees only, which complicates genetic manipulation.

2.2 Adaptive Symbols and Numerical Optimizers

The concept of adaptive symbols is introduced in GEP to generate algebraic models with accurate numerical constants and to determine these constants efficiently, i.e. based on gradient information instead of stochastic processes. Adaptive symbols are devised as a new type of symbol that are, in addition to standard symbols like input variables, mathematical operators or fixed numerical constants, available to the GEP algorithm to build and mutate the genotype of individuals in the population.

In the genotype, adaptive symbols behave precisely like standard symbols, so that the same genetic operators can be applied. In the phenotype, however, adaptive symbols act as placeholders. When one or more of these placeholders are detected prior to an individual's fitness evaluation, a process is started to determine the optimal numerical constants at the placeholder positions (see Fig. 1b). In this process, an iterative gradient-based optimizer is employed with the goal of minimizing the specified training objective. The resulting constants, i.e. the adaptive symbol values, are inserted at the placeholder positions and the fitness of the individual is calculated. Finally, each adaptive symbol stores its value as an initial value for next generation's fitness evaluation. The following aspects further define the concept of adaptive symbols:

1. Adaptive symbols can occur at any position in the phenotype.
2. Values of adaptive symbols are unique for each individual, not across the population.

3. The number of adaptive symbols per individual is user-defined.
4. One adaptive symbol can be selected to the genotype multiple times.
5. All instances of one adaptive symbol share the same value in the phenotype.

As an example, data generated from the canonical function $f(x_1, x_2) = 0.196\, x_1^2 + 0.616\, x_2 + 3.142$ could be approximated using the input symbols x_1 and x_2, the operators $+$ and $*$ and two adaptive symbols p_1 and p_2 with the genotype

$$+ * \mathrm{p1} * \mathrm{x1x1} + ** \mathrm{p1p2x2p2}$$

that translates to the phenotype

$$p_1 \cdot x_1^2 + p_1 \cdot p_2 \cdot x_2 + p_2\,,$$

where the numerical optimizer would calculate $p_1 = 0.196$ and $p_2 = 3.142$.

2.2.1 Numerical Optimizers

Determining the locally optimal adaptive symbol values p^* is generally a nonlinear optimization problem, as any training objective J depends on the candidate model and adaptive symbols can occur at any position in the candidate model. Such a problem requires an iterative solution of Eq. (1) and we investigate two types of numerical optimizers.

First, a general optimizer is applied to solve Eq. (1) for any objective function $J : \mathbb{R}^n \to \mathbb{R}$, which provides a high flexibility for defining the training objective. Specifically, we select the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm (Broyden 1970; Fletcher 1970; Goldfarb 1970; Shanno 1970) based on the results of a preliminary study. The algorithm improves over the standard gradient descent algorithm by utilizing curvature information and is computationally efficient by approximating instead of calculating the inverse Hessian matrix of J. This approximation is updated at every iteration based on the observed change of gradients (Nocedal and Wright 1999).

The second type of investigated optimizers are nonlinear least squares optimizers, as most ML training objectives are formulated to minimize the square errors between the model predictions and the training data. Equation (2) defines such a training objective J, where the residual function r describes the dependency of these prediction errors on the adaptive symbol values p. The mathematical structure of least squares problems can be exploited to approximate the Hessian matrix of J based on the Jacobian matrix of r, which is considered to be an often accurate approximation (Nocedal and Wright 1999). The Levenberg-Marquardt (LM) algorithm (Levenberg 1944; Marquardt 1963), which is applied in the studies in Sect. 3, combines this approximation of the Hessian with additional regularization to control the iteration step size. The BFGS and LM algorithms are implemented in the GEP framework via the SciPy package for Python (Virtanen et al. 2020).

2.3 Regularization and Model Complexity

Regularization is applied in ML in general to prevent a trained model from overfitting the training data. In other words, the model is incentivized to not memorize the training data points but approximate the underlying data generating function, so that the error on testing data from the same data distribution is close to the training error. For the development of closure models in particular, the experience of previous studies is that implementing

complex models with large numerical constants to close underdetermined systems of equations can lead to instabilities when solving these systems numerically (Weatheritt et al. 2017). To control the magnitudes of adaptive symbol values and the complexity of the evolved algebraic models, L_2 regularization and a model complexity objective function are added to the GEP framework.

L_2 regularization is the most frequently used regularization technique and drives model parameters towards small magnitudes (Goodfellow et al. 2016). Therefore, the square L_2 norm of the parameter vector $p \in \mathbb{R}^n$ is multiplied by a scalar regularization parameter λ and added to the training objective J to yield the extended training objective

$$\hat{J} = J + \lambda \cdot \|p\|_2^2 = J + \lambda \cdot \sum_{i=1}^{n} p_i^2 \,, \tag{3}$$

where the number of parameters n corresponds to the number of unique adaptive symbols in the phenotype of the evaluated individual. The regularization parameter λ is user-defined and balances fitting the training data with reducing adaptive symbol magnitudes. Due to the nature of L_2 regularization, determining p^* that minimizes $\hat{J}$ remains a least squares problem to which both investigated optimizers can be applied.

2.3.1 Model Complexity

The complexity of a model describes the capability of its model class F to approximate a wide range of functions by adapting its parameters p (Goodfellow et al. 2016). One example of a model class are univariate polynomials of degree d

$$F_d(x) = \sum_{i=0}^{d} p_i x^i \,. \tag{4}$$

If we assume that exemplary training data is generated by a quadratic function, the complexity of any linear model in F_1 is too low to accurately fit the training data. On the other hand, the complexity of a polynomial of degree five is considered too high, as training a model in F_5 will likely overfit the training data. Thus, controlling the model complexity is another regularization technique to improve generalizability. Additionally, a low complexity naturally increases a model's interpretability and, as discussed above, is beneficial for the implementability of closure models.

Quantifying model complexity, however, is not a straightforward task, as no universally accepted metric exists. In GP, published metrics can be classified as calculating either the structural complexity or the functional complexity of a model. Structural complexity metrics analyze the genotype of an individual, which is an expression tree in GP, and measure, for example, the number of tree nodes or the sum of subtree nodes (Schmidt and Lipson 2009; Smits and Kotanchek 2005). In contrast, the functional complexity of a model depends on its phenotype, i.e. the resulting algebraic equation. Vladislavleva et al. (2008) estimated the order of nonlinearity of models to measure complexity and Vanneschi et al. (2010) devised a curvature-based metric. Furthermore, metrics from statistical learning theory were investigated, such as the Vapnik-Cervonenkis dimension or the Rademacher complexity (Chen et al. 2018, 2020). Complexity metrics in GP are either used to extend the objective function to a weighted composite function (Soule and Foster 1998) or set as an additional objective function (Smits and Kotanchek 2005).

Table 1 Overview of studied use cases

Use case	Optimizers	Regularization
Sutherland's law	BFGS, LM	–
Laminar flame speed modeling	BFGS	J_c
Subgrid-scale modeling	BFGS, LM	L_2
Nonlinear eddy viscosity modeling	BFGS, LM	L_2, J_c

In GEP, to the authors' knowledge, only Ferreira (2006) studied model complexity and applied a metric based on the count of expression tree nodes. As structural complexity metrics are computationally inexpensive and successful at preventing ineffective symbols in the model equation, we also define a structural model complexity metric, but one that is based on the symbol complexity of expression tree nodes. Therefore, each symbol s in the set of symbols $\mathcal{S}$ available to the GEP algorithm is assigned a user-defined complexity value c_s. The complexity of an evolved model is then calculated as

$$J_c = \sum_{s \in L} c_s, \tag{5}$$

where L is the list of symbols in the individual's expression tree. The ability to define custom symbol complexities, in contrast to counting the number of symbols, which is equivalent to $c_s = 1 \ \forall \ s \in \mathcal{S}$, allows incentivizing specific model structures. For example, setting $c_+ = 1$ and $c_{\exp} = 3$ for the symbols $s_+(x_1, x_2) = x_1 + x_2$ and $s_{\exp}(x) = \exp(x)$ supports the usage of linear over nonlinear operators. While Ferreira (2006) extended the training objective via a weighted sum approach to account for model complexity, we apply Eq. (5) as a second objective function. In comparison, our multi-objective approach does not require weighting the training error and the model complexity metric, which is difficult to estimate before the training, and enables the analysis of models of different complexity selected from the Pareto front (see Figs. 10 and 11) after the training. We utilize the multi-objective optimization extension to the GEP framework by Waschkowski et al. (2022).

3 Results

The application of the presented adaptive symbol concept (see Sect. 2.2) and regularization methods (see Sect. 2.3) to four different use cases is discussed in the following. While all cases employ adaptive symbols, we investigate different optimizers and regularization techniques in each use case.

First, we compare the performance of the BFGS and LM optimizers when rediscovering Sutherland's law in Sect. 3.1. Next, laminar flame speed models are developed for unstretched premixed flames and the advantages of the model complexity objective function J_c are explored (see Sect. 3.2). In Sect. 3.3, the impact of L_2 regularization on the development of SGS models for LES of homogenous isotropic turbulence is demonstrated. Lastly, we combine the two regularization techniques and compare the two optimizers when training NLEVMs for RANS calculations of a wall-mounted square cylinder flow (see Sect. 3.4). Table 1 presents an overview of the use cases.

All results are compared to the standard GEP framework and a maximum training runtime is set as the termination criterion, unless otherwise stated. As optimizing the values of

adaptive symbols increases the computational training costs, we aim to ensure an unbiased comparison between training runs with varying numbers of adaptive symbols by providing equal computational resources to all runs. In other words, training runs with no or few adaptive symbols are able to evolve their population for more generations, while runs with higher numbers of adaptive symbols can benefit from more gradient-informed numerical constants. The novel hyperparameters in this work, i.e. the number of adaptive symbols and the L_2 regularization parameter, are analyzed via hyperparameter studies in Sects. 3.2 and Appendix A. All remaining parameters of the GEP framework are determined based on the authors' experience from previous projects (Weatheritt and Sandberg 2016a; Sandberg et al. 2018; Zhao et al. 2020; Waschkowski et al. 2022) and selective preliminary studies. An overview of all hyperparameters for the different use cases is included in Table 7. In the following, the details of each case and its modeling and training strategies are presented before the respective results are discussed.

3.1 Sutherland's Law

Sutherland's law (Sutherland 1893) models the dynamic viscosity μ of dilute gases as

$$\mu = \mu_0 \left(\frac{T}{T_0} \right)^{\frac{3}{2}} \frac{T_0 + C}{T + C} \,, \tag{6}$$

where μ is solely a function of the temperature T. The Sutherland temperature C is a gas-specific constant and μ_0 and T_0 are reference values. Sutherland's law is derived from kinetic gas theory and assumes an idealized intermolecular-force potential (White and Majdalani 2006). In computational fluid dynamics (CFD) simulations of compressible flows, Eq. (6) is commonly applied as a closure model to describe the linear dependency of the viscous stresses of Newtonian fluids on the strain rate tensor.

We utilize Sutherland's law as a canonical example to discuss the benefits of adaptive symbols and analyze the two selected optimizers. We aim to rediscover first Eq. (6) and then its normalized version $\hat{\mu} = \mu/\mu_0$ for air with $C = 110.4\text{K}$ and $\mu_0 = 1.716 \times 10^{-5}\,\text{kg/(m s)}$ at $T_0 = 273.15\text{K}$. The training data for μ and $\hat{\mu}$ is generated according to Eq. (6) at 100 uniformly spaced data points in a range from $T = 250\text{K}$ to $T = 1750\text{K}$.

3.1.1 Modeling and Training Strategies

Three strategies to rediscover Sutherland's law are investigated. We compare the standard training approach of using five random numerical constants (RNC) drawn from a uniform distribution $U(-1, 1)$ to using five adaptive symbols with values determined by either the BFGS or LM optimizer.

The remaining symbols and hyperparameters are unchanged across the different training approaches. The single input symbol is T and the mathematical operators are $+$, $-$, $\times$, $\div$ and $(\cdot)^{\frac{3}{2}}$.[1] The integer values 0, 1 and 2 are provided as additional numerical constants. In every training run, a population of 1000 individuals consisting of two genes each is evolved to minimize the mean squared error (MSE) between the predicted (normalized)

---

[1] Note that the choice of $(\cdot)^{\frac{3}{2}}$ over the more generic $\sqrt{\cdot}$ operator reduces the required computational budget but does not impact the comparison between RNC and the two optimizers.

Table 2 Accuracy and training time (*mean ± SD*) for rediscovering Sutherland's law in its standard (μ) and normalized form ($\hat{\mu}$) using the standard training strategy (RNC) or adaptive symbols with different optimizers (BFGS, LM)

Strategy	μ		$\hat{\mu}$	
	Acc	Time [CPU$_s$]	Acc	Time [CPU$_s$]
RNC	0/5	–	0/5	–
BFGS	0/5	–	4/5	355.7 ± 209.4
LM	5/5	217.4 ± 185.0	5/5	113.1 ± 73.0

viscosities and their training data values. For each of the three strategies and both μ and $\hat{\mu}$, we perform the training with five different random initializations of the population. The maximum runtime per training run is set to 0.25 CPU hours. An overview of all training settings is provided in Table 7.

3.1.2 Analysis and Discussion

The results of training models for μ and $\hat{\mu}$ to rediscover Sutherland's law using the three introduced training strategies are presented in Table 2. The accuracy describes the ratio of rediscoveries to randomly initialized training runs and the reported time is the mean training time for successful rediscoveries and its standard deviation (SD).

Training with five adaptive symbols and employing the LM optimizer allows to identify the correct μ and $\hat{\mu}$ equations in all training runs. In contrast, using the BFGS optimizer, the GEP framework is only able to learn Sutherland's law in its normalized form.

Sutherland's law can be simplified to $\mu = C_1 T^{\frac{3}{2}}/(T + C)$, where the combined constant is $C_1 = 1.458 \times 10^{-6} \text{kg/m/s/K}^{\frac{1}{2}}$. For the normalized version $\hat{\mu}$, C_1 changes to $8.496 \times 10^{-2} \text{K}^{\frac{1}{2}}$. Thus, the difference in the order of magnitude between C_1 in the numerator and $C = 110.4\text{K}$ in the denominator is reduced significantly for $\hat{\mu}$, which enables rediscoveries using the BFGS optimizer and speeds up the training with the LM optimizer.

The standard training approach utilizing RNCs is not capable of rediscovering Sutherland's law in any training run, despite evolving populations for up to 798 generations (in contrast to a maximum of 114 generations with the LM optimizer). One issue that we identified is a conflict between model structure, constants and fitness. The fittest model resulting from the training with RNCs to rediscover $\hat{\mu}$ is

$$\hat{\mu} = 1.469 \times 10^{-3} \, \text{K}^{-1} \cdot T + 0.86 \tag{7}$$

with a fitness value of $J = 0.82 \times 10^{-2}$. This model has a structure and constants that are less similar to the simplified version of Sutherland's law than

$$\hat{\mu} = 7.925 \times 10^{-2} \, \text{K}^{-0.5} \cdot T^{\frac{3}{2}}/T \, , \tag{8}$$

which is the least fit model resulting from the five training runs ($J = 2.1 \times 10^{-2}$).

Adaptive symbols are capable of resolving this conflict and demonstrate significant performance advantages on this canonical example. In particular, the least

squares-specific LM optimizer robustly determines accurate numerical constants across varying orders of magnitude. While the results of the standard training approach could potentially be improved by modifying the RNC sampling distribution, adapting GEP hyperparameters or normalizing the training data differently, the concept of adaptive symbols simplifies the training procedure and adds flexibility to the GEP framework.

3.2 Laminar Flame Speed Modeling

The laminar flame speed S_L is an important characteristic of a given fuel-air mixture in combustion engines. S_L describes the rate of propagation of the flame surface in premixed combustion processes as a result of chemical reactions, mass diffusion and heat conduction. Numerical simulations of premixed combustion engines require an accurate description of the combustion processes and typically use empirical or analytical laminar flame speed models. The accuracy of these models affects the numerical prediction of the combustion behavior and the emission of pollutants significantly, which in turn influences the potential for improving the combustion processes. Therefore, obtaining accurate laminar flame speed models that apply to the wide range of operating conditions observed in industrial premixed combustion engines is a key challenge.

Among the most widely used S_L models is the Gülder model (Gülder 1984), which is an empirical model fitted to experimental data. Metghalchi and Keck (1982) introduced an empirical power-law expression that accounts for the temperature and pressure dependency of S_L and captures dilution effects. In contrast, Göttgens et al. (1992) proposed a physics-based approach to derive an analytical expression for S_L based on rate-ratio asymptotics. The constants in this analytical expression have been fitted to a wide range of fuels, such as hydrogen, methane, ethylene, ethane, acetylene, propane, ethanol, n-heptane, iso-octane and primary reference fuel (Göttgens et al. 1992; Röhl et al. 2009; Ewald and Peters 2005; Beeckmann et al. 2017). Due to the physics-based approach, the asymptotic model extrapolates reasonably outside the temperature and pressure ranges of the calibration data and thus can be readily implemented in combustion engine simulations (Ewald and Peters 2005; Hesse et al. 2018; Hann et al. 2020). One general limitation of the asymptotic model is its applicability to lean equivalence ratios only, although more complex extensions have been proposed to include rich mixtures (Hann et al. 2020; Seshadri and Göttgens 1991).

In this section, we utilize numerical data of a well-validated chemical mechanism for methane-air mixtures to develop advanced data-driven laminar flame speed models. This mechanism consists of 79 species and 1055 reactions and we perform one-dimensional unstretched premixed flame simulations using the FlameMaster software package (Pitsch 1998) to generate a dataset of S_L values that covers a wide range of temperatures T (300 to 1200 K) and pressures p (1 to 40 bar) at lean-to-stoichiometric equivalence ratios ϕ (0.5–1.0). These training data ranges are selected to ensure a fair comparison to the asymptotic model by Göttgens et al. (1992), which only applies to lean equivalence ratios, and due to the relevance for industrial applications, as lean mixtures are important for containing NOx, soot and pollutant emissions. Extra data points at $\phi = 0.0$ are added to prevent unphysical non-zero flame speeds. The complete dataset of size $m = 9945$ is non-dimensionalized using quantities from the asymptotic analysis
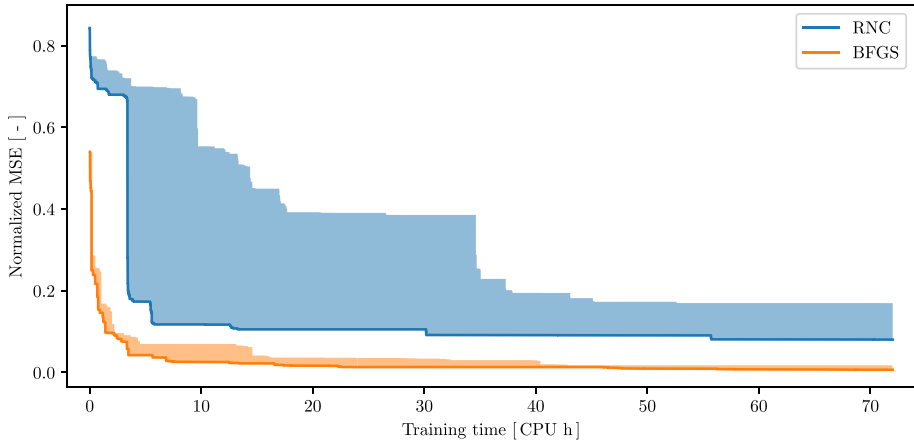
Fig. 3 Convergence of normalized MSE of S_L models from standard (RNC) and gradient-informed (BFGS) training without regularization (shaded areas represent variation due to random initialization)

of the flame structure, which is essential for developing models that generalize outside the training data distribution.

3.2.1 Modeling and Training Strategies

The gradient-informed GEP framework is applied to develop analytical models $\hat{S}_L = f(\hat{T}, \hat{p}, \hat{\phi})$, where $\hat{\cdot}$ denotes non-dimensionalized quantities. We utilize ten adaptive symbols, which are optimized with the BFGS algorithm,[2] and compare the performance to the standard training approach with random numerical constants (RNC). The available mathematical operators are $+$, $-$, $\times$, exp, log and $\sqrt{\cdot}$. The evolved models are evaluated using the normalized MSE between the model predictions and the simulation data

$$J = \frac{1}{m} \sum_{i=1}^{m} \left(\frac{\hat{S}_{L,i,\text{GEP}} - \hat{S}_{L,i,\text{data}}}{\hat{S}_{L,i,\text{data}}} \right)^2.$$ (9)

Additionally, in the second part of the S_L modeling analysis, we investigate setting the model complexity metric J_c (see Eq. (5)) as a second objective function. All others details on the modeling and training settings are listed in Table 7.

3.2.2 Analysis and Discussion

Initially, the S_L models are trained without regularization. Figure 3 shows the convergence of the normalized MSE objective function for training with adaptive symbols (BFGS) and RNCs. Note that the solid lines represent the minimum error across three randomly initialized training runs and the shaded areas indicate the respective variation. Utilizing adaptive

---

[2] The LM optimizer without L_2 regularization tends to converge to very high numerical constants for regression problems (see Sect. 3.4) and this section's goal is investigating the impact of the model complexity metric independent from L_2 regularization.
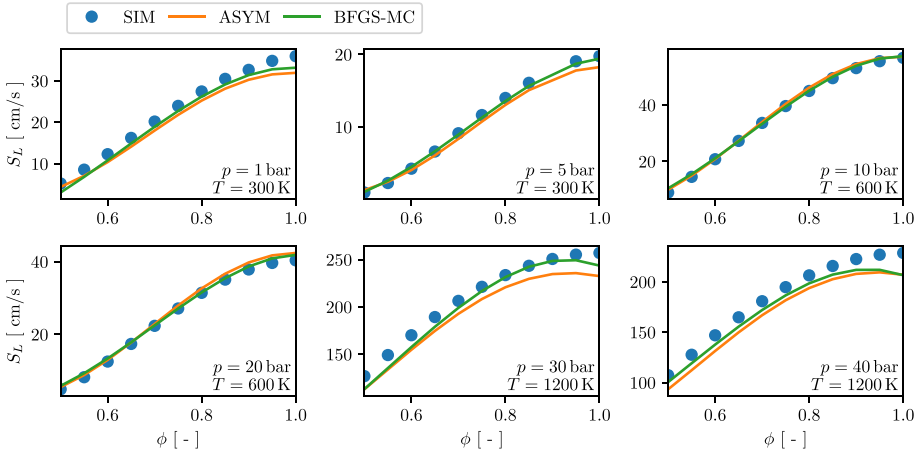
Fig. 4 Overview of S_L predictions over equivalence ratio ϕ of gradient-informed and regularized model (BFGS-MC) and asymptotic model (ASYM) with target simulation data (SIM) as reference at various pressure and temperature values

symbols is clearly beneficial compared to the standard training approach, as the normalized MSE is reduced by an entire order of magnitude. Additionally, the variance in the training performance across different random initializations is reduced substantially. The fittest S_L models resulting from the two training approaches are the following:

$$
\begin{aligned}
\hat{S}_L^{\text{RNC}} &= 0.91 \cdot \hat{\phi}^2 \cdot (1.10 \cdot \hat{p} + 2.10) \cdot \exp(\hat{p} \cdot \hat{\phi} - \hat{T} + \sqrt{1618.18 \cdot \hat{T}}) \\
&\quad \cdot (\hat{p} + \hat{\phi} + \sqrt{\hat{T}} - \log(\hat{p}) + 0.43 \cdot \log(|\hat{\phi} + \sqrt{\hat{T}} - 0.09|) + 0.33) ,
\end{aligned}
\tag{10}
$$

$$
\begin{aligned}
\hat{S}_L^{\text{BFGS}} &= \hat{T} \cdot \hat{\phi} \cdot (486.95 \cdot (\hat{p} - 1.30) \cdot (\hat{T} - 0.035) \cdot \sqrt{\hat{p}} - 10.13) \\
&\quad \cdot (-12.49 \cdot \hat{p} + 221923.13 \cdot \hat{T}^2 + 22.94 \cdot (\log(|\hat{\phi} - 0.029|) - \hat{T}) \\
&\quad + \log(\hat{\phi}) + 153.58 \cdot (\log(|(\hat{p} + \hat{\phi}) \cdot \log(\hat{\phi})|) - 2.07) .
\end{aligned}
\tag{11}
$$

We observe that both S_L models contain nested terms of the nonlinear operators exp, log and $\sqrt{\cdot}$, which is considered unphysical and could impair the prediction accuracy when applying the models outside the training data.

Thus, we add the model complexity metric J_c as a second objective function and specifically set the symbol complexity values for $s \in \{\exp, \log, \sqrt{\cdot}\}$ from the unity default value to $c_s = 5$, which disincentivizes the usage of these operators. To further reduce the normalized MSE of the developed models, the maximum training runtime is increased from 72 to 120 CPU hours. The resulting S_L model, which is selected as the optimal trade-off between the two objective functions (see Sect. 3.4 for details on the selection process), yields a normalized MSE value of 4.80×10^{-3} compared to 6.63×10^{-3} for Eq. (11) and reads as follows:

$$
\begin{aligned}
\hat{S}_L^{\text{BFGS-MC}} &= (\hat{T} + 0.008) \cdot (\hat{\phi} - 0.071) \cdot (1.27 \cdot \hat{p} - 27.48 \cdot \hat{T} - 1.18) \cdot 10^6 \\
&\quad \cdot (2.16 \cdot \hat{p} + 55.94 \cdot \hat{T} + 61.26 \cdot \hat{\phi} - \log(\hat{p}) - 3.89) \cdot \sqrt{\hat{T}} \cdot \sqrt{\hat{\phi}} \\
&\quad \cdot (\hat{T} + \hat{\phi} - 0.034) .
\end{aligned}
\tag{12}
$$

While the overall expression length is similar compared to Eqs. (10) and (11), no nested nonlinear operators occur in Eq. (12). An overview of the S_L predictions of the developed model (BFGS-MC) at representative p and T values in comparison to the asymptotic model (ASYM) by Göttgens et al. (1992) is presented in Fig. 4. The prediction accuracy of $\hat{S}_L^{\text{BFGS-MC}}$ improves noticeably over the asymptotic model, in particular towards high equivalence ratios. Therefore, we summarize that the gradient-informed GEP framework distinctly outperforms the standard training approach for this use case and derives a model exceeding the popular asymptotic model. Additionally, model complexity regularization with custom symbol complexities allows steering the model development towards desired model structures.

3.3 Subgrid-Scale Modeling

Large eddy simulations are one of the most important methods for turbulence simulation and have been widely applied in various areas, e.g. in aerospace engineering. LES obtain large-scale flow structures by solving filtered Navier–Stokes equations, while the effects of subgrid-scale structures on the large-scale structures are approximated by SGS models (Smagorinsky 1963; Lilly 1967; Pope 2001; Sagaut 2006). As established SGS models have inherent limitations, e.g. an excessive dissipation predicted by the Smagorinsky model (Vreman et al. 1997), and no overall satisfactory model has emerged from multiple decades of research (Gamahara and Hattori 2017), developing closure models for LES remains an active area of study.

For this use case, we consider incompressible turbulence for which the filtered continuity and momentum equations (Sagaut 2006; Meneveau and Katz 2000) are

$$\frac{\partial \widetilde{u}_i}{\partial x_i} = 0 \,, \tag{13}$$

$$\frac{\partial \widetilde{u}_i}{\partial t} + \frac{\partial \widetilde{u}_i \widetilde{u}_j}{\partial x_j} = -\frac{\partial \widetilde{p}}{\partial x_i} - \frac{\partial \tau_{ij}}{\partial x_j} + \nu \frac{\partial^2 \widetilde{u}_i}{\partial x_j x_j} + \widetilde{\mathcal{F}}_i \,, \tag{14}$$

where $\widetilde{\cdot}$ denotes filtered variables and is defined for the velocity u as

$$\widetilde{u}(x,t) = \int_D G(r,x)u(x-r,t)dr \,, \tag{15}$$

where D is the entire flow domain and G is the filter function which satisfies the normalization condition $\int G(r,x)dr = 1$. The SGS stress tensor τ_{ij} in Eq. (14) is defined as $\tau_{ij} = \widetilde{u_i u_j} - \widetilde{u}_i \widetilde{u}_j$.

In this section, we apply the gradient-informed GEP framework to develop improved SGS models. The high-fidelity dataset for training and testing is obtained from a direct numerical simulation (DNS) of three-dimensional forced incompressible isotropic turbulence (Xie et al. 2020), which is a canonical use case that is frequently employed for the comparison of SGS models, as many of the basic features of turbulence are present while its computational costs are inexpensive (Fureby et al. 1997). The DNS velocity field is calculated on a uniform grid of size 1024^3 and the Taylor Reynolds number is

$Re_\lambda \approx 260$. To extract the filtered velocity field, we apply a top-hat filter in three dimensions. The top-hat filter in one dimension is

$$\widetilde{f}_i = \frac{1}{2n_f}\left(f_{i-n_f/2} + 2\sum_{j=i-\frac{n_f}{2}+1}^{i+\frac{n_f}{2}-1} f_j + f_{i+n_f/2}\right), \tag{16}$$

where the filter width can be expressed as $\Delta = n_f \Delta x$. We choose $n_f = 16$ in this case such that about 5% of the turbulence kinetic energy is filtered, as discussed by Xie et al. (2020). Furthermore, coarse-graining that yields 16^3 different datasets with 64^3 coarse grid points each is applied to the filtered DNS data. We select a total dataset of 8×64^3 grid points and 70% of points are randomly selected for training and the remaining 30% for testing.

3.3.1 Modeling and Training Strategies

For this use case, we compare the BFGS and LM optimizers in combination with L_2 regularization for modeling the anisotropic SGS stress tensor τ_{ij}^A. Following Li et al. (2021), we model τ_{ij}^A as a function of the filter width Δ and the local filtered strain and rotation rate tensors $\widetilde{S}_{ij}$ and $\widetilde{\Omega}_{ij}$. We assume a linear dependency on Δ and utilize the integrity basis proposed by Pope (1975) to model τ_{ij}^A as

$$\tau_{ij}^A = (\Delta|\widetilde{S}|)^2 \cdot \sum_{k=1}^{4} g^k\left(I^1, \ldots, I^4\right) V_{ij}^k, \tag{17}$$

where $|\widetilde{S}| = \sqrt{\widetilde{S}_{mn}\widetilde{S}_{mn}}$ is an inverse time scale. The basis tensors V_{ij}^k and the invariants I^l are only functions of $\widetilde{S}_{ij}$ and $\widetilde{\Omega}_{ij}$ non-dimensionalized by $|\widetilde{S}|$. The definitions of V_{ij}^k and I^l are available in Li et al. (2021). The scalar functions g^k are the modeling target of the GEP framework.

During training, the MSE between the predicted anisotropic SGS stress tensor $\tau_{ij}^{A,\text{GEP}}$ and the DNS training data value $\tau_{ij}^{A,\text{data}}$ is minimized. Therefore, a population of 100 individuals with four adaptive symbols per scalar function is evolved. Table 7 lists the remaining modeling and training settings.

3.3.2 Analysis and Discussion

First, we perform a hyperparameter study to identify an appropriate order of magnitude for the L_2 regularization parameter λ in Eq. (3). We utilize the BFGS optimizer and explore $\lambda \in \{10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$. The maximum training runtime is set to 13.89 CPU hours.

Figure 5 shows the MSE of SGS models regularized with different λ values during training. There is approximately a constant factor between the training and testing MSE across all λ values, displayed as solid and dashed lines, respectively. Improvements on the training dataset result in improvements on the testing dataset, which suggests that no overfitting occurs. Furthermore, we observe that starting with $\lambda = 10^{-5}$ the MSE increases with increasing λ value. L_2 regularization incentivizes parameters of small magnitude and thus, restricts the search space for adaptive symbol values. From $\lambda = 10^{-4}$, the search space gets too restricted and the optimal numerical constants cannot be learned, which results in
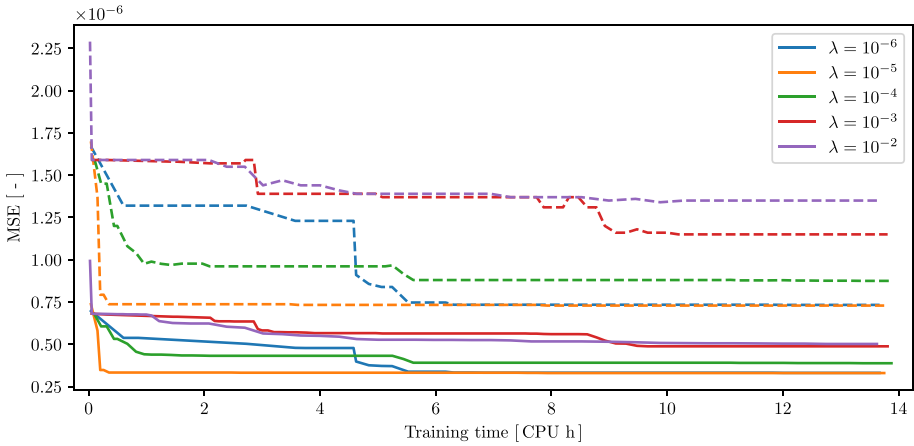
Fig. 5 Convergence of MSE of SGS models in λ hyperparameter study evaluated on training (solid) and testing (dashed) data
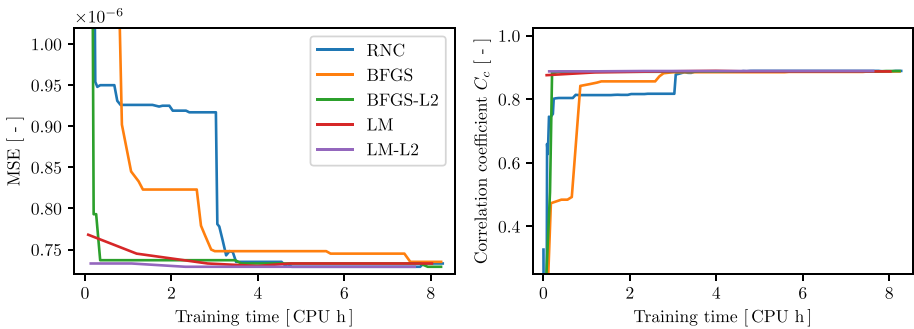

Fig. 6 Convergence of MSE (left) and C_c (right) of different SGS models evaluated on testing data

deteriorating model performances. However, comparing the results based on $\lambda = 10^{-5}$ and $\lambda = 10^{-6}$ indicates that a limited amount of L_2 regularization, i.e. a small restriction to the parameter search space, improves the convergence speed significantly.

In the following, we select $\lambda = 10^{-5}$ and compare the BFGS and LM optimizers, both with and without L_2 regularization, to the standard GEP training approach. We analyze the convergence of the MSE and the correlation coefficient C_c in Fig. 6 for the different approaches evaluated on the testing dataset. The correlation coefficient measures the average componentwise correlation between the predicted and the high-fidelity aniso-tropic SGS stress tensors $\tau_{ij}^{A,\text{GEP}}$ and $\tau_{ij}^{A,\text{data}}$ and is calculated as

$$C_c = \frac{1}{6} \sum_{i=1}^{3} \sum_{j=i}^{3} \frac{\left\langle \left(\tau^{\text{GEP}} - \left\langle \tau^{\text{GEP}} \right\rangle \right) \odot \left(\tau^{\text{data}} - \left\langle \tau^{\text{data}} \right\rangle \right) \right\rangle}{\sqrt{\left\langle \left(\tau^{\text{GEP}} - \left\langle \tau^{\text{GEP}} \right\rangle \right)^2 \right\rangle \odot \left\langle \left(\tau^{\text{data}} - \left\langle \tau^{\text{data}} \right\rangle \right)^2 \right\rangle}}, \tag{18}$$
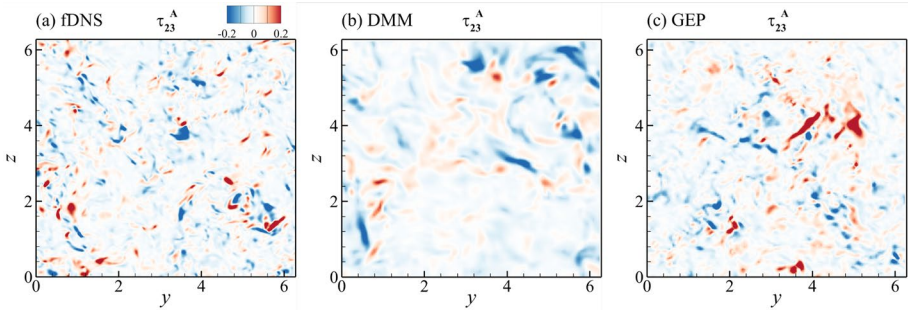
Fig. 7 Contours of SGS stress component τ_{23}^A at arbitrary domain slice: **a** fDNS, **b** DMM, **c** GEP

where $\langle \cdot \rangle$ indicates averaging over the dataset, $\odot$ represents componentwise multiplication and τ is a short notation for τ_{ij}^A.

Figure 6 (left) shows that all training approaches converge to a similar MSE value within the training runtime of 8.3 CPU hours. This can be explained by the simple model structure of the fittest SGS models. Using the LM optimizer with L_2 regularization, the resulting model for $\tau_{ij}^{A,GEP}$ is

$$\tau_{ij}^{A,GEP} = (\Delta |\widetilde{S}|)^2 \cdot \left(-0.01 V_{ij}^1 - 0.1 V_{ij}^2 + 0.07 V_{ij}^3 - 0.11 V_{ij}^4 \right), \tag{19}$$

where the scalar functions g^k are constant. Thus, determining accurate numerical constants is the key challenge for this use case. Consequently, utilizing adaptive symbols and numerical optimizers significantly increases the convergence speed, although the BFGS optimizer without L_2 regularization appears to converge to suboptimal constants until close to the maximum training time. The general observations are that the least squares-specific LM optimizer outperforms the BFGS optimizer and that L_2 regularization further speeds up training convergence, which is in line with the results of Sect. 3.1 and the λ hyperparameter study. Furthermore, C_c values of close to 0.9 in Fig. 6 (right) support that the resulting models are highly correlated with the high-fidelity data and that, for example, not only the error in one component of $\tau_{ij}^{A,GEP}$ is reduced, which might cause unphysical predictions.

In order to investigate the suitability of Eq. (19) as a closure model, i.e. ensuring accurate predictions and numerical stability, we perform an LES of homogeneous isotropic turbulence with the developed SGS model. The LES grid is of size 128^3 with a filter width of $\Delta = 16\Delta x$ and the numerical methods are described in (Li et al. 2021). We compare the LES results based on $\tau_{ij}^{A,GEP}$ with the filtered high-fidelity DNS data (fDNS) and the predictions of an LES applying a standard SGS model, the dynamic mixed model (DMM) (Liu et al. 1994).

Figure 7 displays the τ_{23}^A contours of the different simulations at an arbitrary slice in the flow domain. The SGS model developed with the gradient-informed GEP framework captures fine-scale structures similar to the fDNS data and predicts stress intensity levels accurately. In contrast, the DMM models more dissipation which leads to larger τ_{23}^A structures of lower intensity. The energy spectrum $E(\kappa)$, premultiplied by the wave number κ, in Fig. 8 (left) shows reduced energy content of the small-scale structures, i.e. at high κ values, due to this overpredicted dissipation of the DMM. The $\tau_{ij}^{A,GEP}$ LES results approximate the spectrum of the fDNS data more accurately and match the target τ_{12}^A distribution in
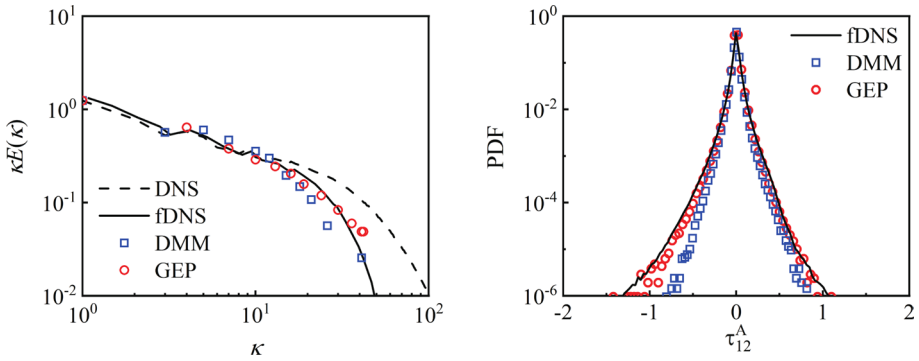
Fig. 8 Premultiplied energy spectrum $\kappa E(\kappa)$ (left) and probability density function (PDF) of SGS stress component τ_{12}^A (right)

Fig. 8 (right) closely. The width of the probability density function (PDF) of τ_{12}^A resulting from the DMM is not sufficiently large.

In summary, adaptive symbols, especially when employing the LM optimizer, and a moderate level of L_2 regularization significantly increase the convergence speed when developing models with a simple structure such as in Eq. (19). The developed SGS model is easily implementable in an LES solver and was demonstrated to substantially improve τ_{ij}^A predictions over a standard SGS model.

3.4 Nonlinear Eddy Viscosity Modeling

RANS calculations remain the primary tool to perform CFD simulations of turbulent flows of industrial interest. In contrast to spatial filtering in LES, Reynolds averaging allows calculating time-averaged flow fields, which is computationally more efficient and often sufficient for industrial applications. For an incompressible flow with constant density, the Reynolds-averaged continuity and momentum equations are

$$\frac{\partial \overline{u}_i}{\partial x_i} = 0, \tag{20}$$

$$\overline{u}_j \frac{\partial \overline{u}_i}{\partial x_j} = -\frac{\partial \overline{p}}{\partial x_i} + \frac{\partial}{\partial x_j}\left(\nu \frac{\partial \overline{u}_i}{\partial x_j} - \overline{u'_i u'_j} \right), \tag{21}$$

with the mean velocity $\overline{u}_i$, the density-corrected pressure $\overline{p}$ and the kinematic viscosity ν. The Reynolds stress tensor $\overline{u'_i u'_j}$ describes the impact of turbulent fluctuations on the averaged flow field and can be deconstructed into an isotropic ($\frac{2}{3}k\delta_{ij}$) and an anisotropic term ($2ka_{ij}$), where k is the turbulence kinetic energy and δ_{ij} is the Kronecker delta.

The anisotropy tensor a_{ij} is modeled as $a_{ij} = -\frac{\nu_t}{k}S_{ij}$ by linear eddy viscosity models (LEVM), such as the popular k-ω SST model (Menter 1994), which solves two additional transport equations for k and ω, the specific dissipation rate, to calculate the eddy viscosity ν_t. The mean strain rate S_{ij} is calculated from $S_{ij} = \frac{1}{2}\left(\frac{\partial \overline{u}_i}{\partial x_j} + \frac{\partial \overline{u}_j}{\partial x_i} \right)$. To improve the well-known shortcomings of LEVMs, e.g. predicting flows with separation or curvature

inaccurately (Leschziner 2015), NLEVMs assume a nonlinear dependency of a_{ij} on S_{ij} and the mean rotation rate $\Omega_{ij} = \frac{1}{2}\left(\frac{\partial \overline{u}_i}{\partial x_j} - \frac{\partial \overline{u}_j}{\partial x_i}\right)$.

In this section, we develop NLEVMs for a wall-mounted square cylinder flow using the GEP framework with adaptive symbols and the introduced regularization techniques. In addition to modeling a_{ij}, we evolve a turbulence production correction model R, as proposed by Schmelzer et al. (2019). This second model extends the production term of k in the turbulence transport equations and ensures the consistency of k between its transport equation and its high-fidelity data values.

This section's use case is a complex three-dimensional flow around a square cylinder with a height-to-width ratio of $h/d = 4$ at a Reynolds number of $Re_d = 11,000$. The resulting flow features include horseshoe vortices around the cylinder, upwash from the boundary layer, downwash over the cylinder tip and a von Kármán vortex street (Wang and Zhou 2009). These secondary features are inherently difficult to predict via RANS calculations, which makes this case a challenging hurdle for data-driven turbulence modeling. Additionally, its computational costs are too expensive to perform simulation-driven training, where the numerical solver is invoked to evaluate candidate models and the models' implementability is guaranteed (Zhao et al. 2020). Hence, this case particularly relies on regularization techniques to incentivize the development of implementable models. As high-fidelity data for developing the a_{ij} and R models, we utilize a well-validated hybrid RANS/LES dataset generated by Weatheritt and Sandberg (2016b). Following the results of Haghiri et al. (2020), who split the flow domain of this case into a near-body region and a downstream region and showed that the NLEVM developed for the downstream region is responsible for nearly all prediction improvements, we extract the training and testing data from the downstream region starting at $x/d = 2$. As a compromise between low computational cost and avoiding overfitting, 10^5 data points are selected for training while the remaining data points are set aside for testing. In order to calculate ω and R values from the high-fidelity data, we employ the k-corrective frozen RANS approach (Schmelzer et al. 2019), which solves the turbulence transport equations with frozen $\overline{u}_i$, k and a_{ij} high-fidelity data values.

3.4.1 Modeling and Training Strategies

For developing a_{ij} and R models, we utilize adaptive symbols and both L_2 and model complexity regularization. We analyze the performance of the BFGS and LM optimizers and make a comparison to unregularized training and the standard GEP training approach. The modeling approach for the turbulence production correction R is

$$R = 2k a_{ij}^R \frac{\partial \overline{u}_i}{\partial x_j}, \qquad (22)$$

where a_{ij}^R is modeled similar to a_{ij} based on the integrity basis derived by Pope (1975), which is defined for the anisotropy tensor as

$$a_{ij} = \sum_{k=1}^{10} g^k(I^1, I^2, ..., I^5) V_{ij}^k. \qquad (23)$$

In contrast to Sect. 3.3, all five scalar invariants I^l and ten basis tensors V_{ij}^k are included to capture the complexity of this use case. The mean strain and rotation rates S_{ij} and Ω_{ij} are
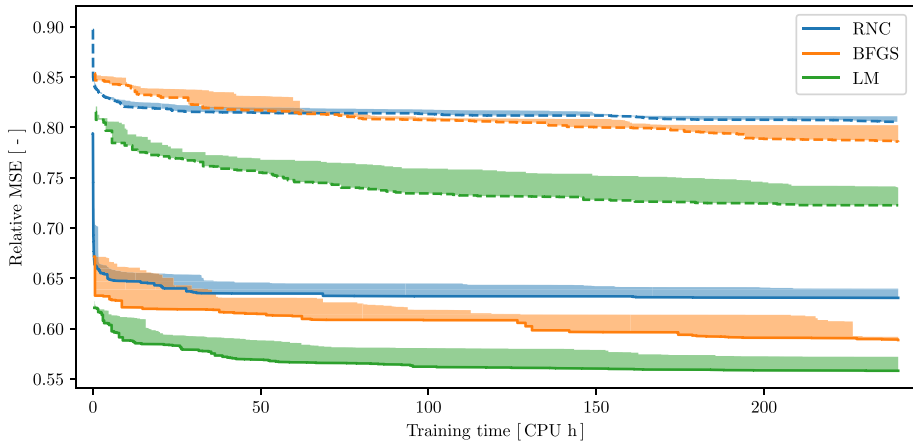
Fig. 9 Convergence of MSE relative to LEVM of a_{ij} (dashed) and R (solid) models trained without regularization for NLEVM case (shaded areas represent variation due to random initialization)

non-dimensionalized by ω before calculating I^l and V_{ij}^k according to the definitions in Pope (1975). The scalar functions g^k are learned by the GEP framework.

The training objective is minimizing the MSE between the predictions of the a_{ij} and R models and the corresponding training data values. For L_2 regularization, the MSE objective function is extended according to Eq. (3). The regularization parameter is determined via a hyperparameter study to $\lambda = 10^{-7}$ (see Appendix A). For model complexity regularization, Eq. (5) is set as a second objective function. To further support the development of low complexity models, the numerical constant 0 is assigned a symbol complexity of $c_0 = 0$, as, for example, multiplication with 0 reduces complexity beyond the calculated J_c value. All other symbols $\mathcal{S} \setminus \{0\}$ have a symbol complexity of unity.

During training, a population of 100 individuals is evolved for a maximum training runtime of 240 CPU hours. The optimal number of adaptive symbols is a trade-off between functional expressivity and computational cost, as one adaptive symbol is insufficient to approximate a function with multiple distinct numerical constants, but more adaptive symbols complicate the optimization problem. In Appendix A, we identify five adaptive symbols per scalar function g^k to be an optimal trade-off. The remaining modeling and training settings are listed in Table 7.

3.4.2 Analysis and Discussion

First, we develop a_{ij} and R models without regularization. Figure 9 shows the convergence of the MSE, relative to the linear k-ω SST model, of the different training approaches, where shaded areas represent variation resulting from different random initializations. All training approaches improve both models over the LEVM and the relative improvement is larger for the turbulence production correction model R. Utilizing adaptive symbols is clearly advantageous in comparison to the standard training approach (RNC) and, in line with the results of Sects. 3.1 and 3.3, the LM optimizer yields lower MSE values than the BFGS optimizer.

Table 3 Training and testing MSE relative to LEVM, model complexity J_c and maximum absolute adaptive symbol value $\max_i(|p_i|)$ of a_{ij} and R models trained without regularization for NLEVM case

Model	Strategy	Train. MSE	Test. MSE	J_c	$\max_i(|p_i|)$
a_{ij}	RNC	0.8056	0.8049	330	–
	BFGS	0.7858	0.7849	140	1.09×10^2
	LM	0.7226	0.7224	180	3.71×10^{14}
R	RNC	0.6305	0.6189	320	–
	BFGS	0.5886	0.5789	166	4.24×10^2
	LM	0.5581	0.5497	210	3.06×10^{13}

Fig. 10 MSE relative to LEVM and model complexity J_c of a_{ij} models developed with regularized training approaches for NLEVM case (markers represent different random initialization)
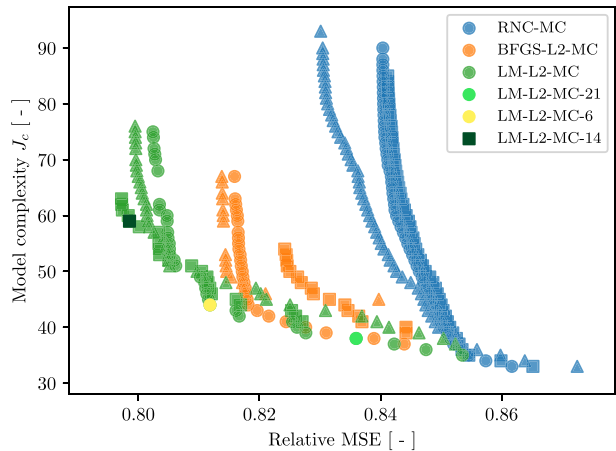
Table 3 presents the performance of the resulting a_{ij} and R models on the training and testing datasets. Additionally, the respective model complexity J_c and the maximum adaptive symbol magnitude $\max_i(|p_i|)$ are listed, which are both not regularized in this first study. Comparing the errors on the training and testing datasets shows that no overfitting occurs. However, the model complexity values of the developed models are too high to allow interpretation. For example, the following is the a_{ij} model developed using the BFGS optimizer, which is the least complex trained model and has additionally been simplified using the SymPy package (Meurer et al. 2017):

$$
\begin{aligned}
a_{ij}^{\text{BFGS}} = {} & (306.79 \cdot I^3 + (0.88 - 175.64 \cdot I^2) \cdot (-I^1 + I^3 + 0.12) - 1.86) \cdot V_{ij}^1 \\
& + (I^2 - 0.02) \cdot (I^5 + 109.19) \cdot V_{ij}^2 - (-I^1 + I^2 + I^3 + 2.54) \cdot V_{ij}^3 \\
& + (I^1 - 15.42) \cdot (I^1 + I^2 + 0.08) \cdot (84.03 \cdot I^2 - 84.03 \cdot I^4 + 6.74) \cdot V_{ij}^4 \\
& + (I^3 \cdot (I^1 + I^3) + 12.91) \cdot V_{ij}^5 + I^3 \cdot (I^2 + 263.97) \cdot (I^4 + 41.48) \cdot V_{ij}^6 \\
& - 191.44 \cdot (I^1 - 1) \cdot V_{ij}^7 + (5.15 \cdot I^1 + I^2 + 36.44) \cdot V_{ij}^8 \\
& + (I^3 + I^5 - 3.07) \cdot V_{ij}^9 + (I^4 + 2.13) \cdot V_{ij}^{10} .
\end{aligned}
$$

The standard training approach (RNC) yields the models with the highest complexity values J_c. In order to approximate a certain numerical constant, the combination of multiple random numerical constants and mathematical operators is generally required. This

Fig. 11 MSE relative to LEVM and model complexity J_c of R models developed with regularized training approaches for NLEVM case (markers represent different random initialization)
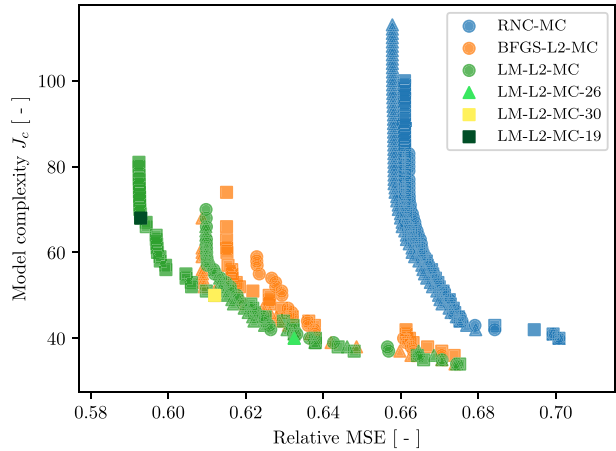

Table 4 Training and testing MSE relative to LEVM, model complexity J_c and maximum absolute adaptive symbol value $\max_i(|p_i|)$ of a_{ij} and R models trained with regularization for NLEVM case

Model	Strategy	Train. MSE	Test. MSE	J_c	$\max_i(|p_i|)$
a_{ij}	LM-L2-MC-21	0.8360	0.8352	38	1.15×10^1
	LM-L2-MC-6	0.8118	0.8109	44	1.27×10^1
	LM-L2-MC-14	0.7985	0.7978	59	1.30×10^1
R	LM-L2-MC-26	0.6325	0.6212	40	1.37×10^1
	LM-L2-MC-30	0.6120	0.6012	50	1.86×10^1
	LM-L2-MC-19	0.5929	0.5828	68	1.64×10^1

increases the search space of possible expressions and complicates symbolic regression. The gradient-informed GEP framework derives an accurate numerical constant with a single adaptive symbol. However, we notice that the LM optimizer converges to extraordinarily high p_i values. Table 3 shows that the LM-optimized models do not overfit the training data, but numerical constants on the order of $\mathcal{O}(10^{13})$ will likely cause stability issues of the numerical solver that applies the trained models.

To develop interpretable and implementable closure models, we apply L_2 regularization when optimizing adaptive symbol values and add the model complexity metric defined in Eq. (5) as a second objective function. Figures 10 and 11 show the performance of the developed a_{ij} and R models on the two objective functions. Note that the marker symbols represent training runs with different random initializations and that only the Pareto front is plotted, i.e. models that are not outperformed by any other model in the training run population. The advantages of utilizing adaptive symbols compared to the standard training approach (RNC-MC) are significant, as lower prediction errors and J_c values are achieved for both models. We notice that the LM optimizer improves over the BFGS optimizer especially towards higher J_c values, where more adaptive symbols in the expression tree are likely and thus, the optimization problem is more complex. In comparison to unregularized training, the complexity values of the gradient-informed models reduce from a minimum of 140 to a maximum of 81, while the relative MSE values are slightly increased (by 0.076 for a_{ij} and 0.035 for R).

Fig. 12 Wall-normal profiles of mean streamwise velocity (left) and turbulence kinetic energy (right) on centerline at $x/d = 3$ of NLEVM case
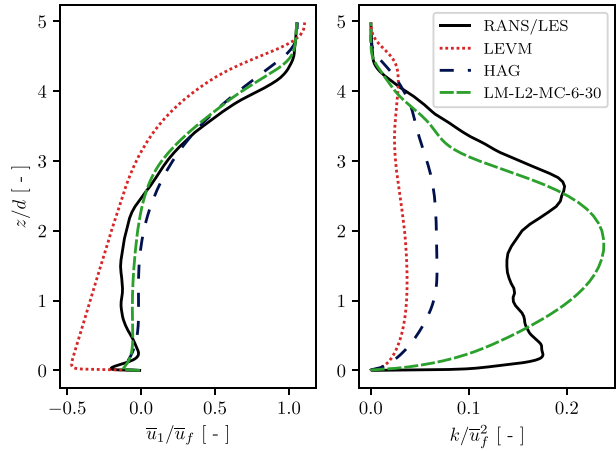
One benefit of multi-objective optimization is that the trade-off between the objective functions can be done after the training. We select three candidate models of increasing complexity for a_{ij} and R, which are highlighted in light green, yellow and dark green in Figs. 10 and 11. Table 4 demonstrates that, in addition to avoiding overfitting and reducing model complexity, the adaptive symbol values are on the order of $\mathcal{O}(10^1)$ across the varying J_c values. Therefore, we conclude that the two regularization techniques are effective.

For a detailed analysis, the models LM-L2-MC-6 (a_{ij}^6) and LM-L2-MC-30 (R^{30}) are selected as a compromise between prediction accuracy and model complexity. The respective model expressions are the following:

$$a_{ij}^6 = (160.48 \cdot I^3 - 1.41) \cdot V_{ij}^1 + (89.48 \cdot I^2 - 1.41) \cdot V_{ij}^2 \\ - 6.95 \cdot V_{ij}^4 + 159.33 \cdot V_{ij}^7 + 111.25 \cdot V_{ij}^9 , \tag{24}$$

$$R^{30} = 2k \cdot \left(\left(-101.10 \cdot I^1 + 236.20 \cdot I^2 + 7.19 \right) \cdot V_{ij}^1 \\ - \left(5.86 \times 10^4 \cdot I^1 I^2 - 1.09 \times 10^4 \cdot I^2 \right) \cdot V_{ij}^6 \right) \frac{\partial \overline{u}_i}{\partial x_j} , \tag{25}$$

which reduce the number of basis tensors V_{ij}^k from ten in unregularized training to merely five and two, respectively, and contain mostly constant and linear terms. Interestingly, the basis tensors in Eq. (24) are a subset of the tensors that Haghiri et al. (2020) identified based on their high alignment with the high-fidelity a_{ij} data of the downwash over the cylinder tip ($V_{ij}^1, V_{ij}^4, V_{ij}^9$) and the upwash from the boundary layer and the near-wall region ($V_{ij}^2, V_{ij}^7, V_{ij}^8$) in the downstream region. Furthermore, the LM-L2-MC-30 model achieves constants on the order of $\mathcal{O}(10^4)$ by combining multiple adaptive symbols, despite a $\max_i(|p_i|)$ value of 18.6 and applying model complexity regularization, which signals the importance of the multiplied tensor V_{ij}^6 for modeling R.

Lastly, we investigate the suitability of the models in Eqs. (24) and (25) as closure models. The models are implemented in the CFD software OpenFOAM (Weller et al. 1998) to extend the baseline k-ω SST model. We run a steady-state RANS calculation on the numerical grid of the hybrid RANS/LES simulation that generated the high-fidelity data

Table 5 Mean (MAE) and maximum (ME) absolute errors relative to LEVM of mean streamwise velocity $\overline{u}_1$ and turbulence kinetic energy k for NLEVM case

Model	MAE $\overline{u}_1$	ME $\overline{u}_1$	MAE k	ME k
HAG	0.257	0.367	0.822	0.870
LM-L2-MC-6-30	0.208	0.272	0.491	0.572

(Weatheritt and Sandberg 2016b). Figure 12 shows the wall-normal profiles of the mean streamwise velocity $\overline{u}_1$ and the turbulence kinetic energy k, non-dimensionalized by the freestream velocity $\overline{u}_f$, on the centerline at $x/d = 3$. We compare the two developed models (LM-L2-MC-6-30) to the high-fidelity hybrid RANS/LES data, the baseline LEVM and the NLEVM developed by Haghiri et al. (2020) for the downstream region (HAG). In addition to clear improvements over the LEVM, the LM-L2-MC-6-30 models predict the $\overline{u}_1$ profile slightly more accurately than the HAG model. However, the most noticeable advancement of the LM-L2-MC-6-30 models is predicting the turbulence kinetic energy on level with the high-fidelity data. While some inaccuracies in the k profile remain due to the unsteady nature of the flow features in this use case, the improvements compared to the HAG model are substantial. Table 5 quantifies these observations by calculating the mean (MAE) and maximum (ME) absolute errors between the RANS predictions of the NLEVMs and the high-fidelity data relative to the respective LEVM errors. Both NLEVMs reduce the MAE of the $\overline{u}_1$ predictions by around 75% compared to the baseline model. The significant improvement of the LM-L2-MC-6-30 models is a reduction of the relative k MAE by 50.9%, while the HAG model only achieves a 17.8% improvement. Furthermore, the small deviations between the mean and maximum of the absolute error distributions suggest that both NLEVMs improve consistently over the baseline LEVM.

We summarize that adaptive symbols lead to significant improvements in unregularized and regularized training with the GEP framework for this complex three-dimensional flow. The two introduced regularization techniques allow the development of implementable and more interpretable a_{ij} and R models, which, for the first time, yield steady-state RANS predictions for k on level with the high-fidelity data. Notably, we show that model complexity regularization automatically selects basis tensors for a_{ij} that are associated with characteristic flow features, which previously required a manual alignment analysis.

4 Conclusion

The concept of adaptive symbols is introduced in this paper to advance the development of physics closure models from high-fidelity data. The fundamentally stochastic GEP framework by Weatheritt and Sandberg (2016a) is extended via adaptive symbols to incorporate gradient information in order to learn accurate numerical constants. A general, i.e. objective function independent, optimizer (BFGS) and a specific nonlinear least squares optimizer (LM) are compared to determine locally optimal adaptive symbol values. Furthermore, two regularization methods are implemented to support the development of interpretable and implementable closure models, which is typically associated with numerical constants of small magnitude and model expressions of low complexity. We add L_2 regularization to the objective function for the gradient-based optimization and define a novel structural model complexity metric, which allows assigning custom symbol

complexities in order to incentivize the usage of certain symbols. The model complexity metric is set as an additional objective function, so that candidate models of varying fitness and complexity can be compared after the training.

The gradient-informed GEP framework is applied to four use cases to rediscover Sutherland's law (see Sect. 3.1), develop laminar flame speed models (see Sect. 3.2) and train two types of turbulence models for LES (see Sect. 3.3) and RANS calculations (see Sect. 3.4). All use cases demonstrate significant improvements in prediction accuracy and training convergence speed for more and less complex optimization problems, respectively. While the BFGS optimizer provides more flexibility for defining the objective function, the proof-of-concept use case of rediscovering Sutherland's law already demonstrates the advantages of the LM optimizer for least squares objective functions. However, developing NLEVM for RANS calculations shows that the LM optimizer can converge to extraordinarily high numerical constants without regularization. L_2 regularization is effective at maintaining small magnitude constants and even speeds up training convergence for developing SGS models for LES. The model complexity objective function reduces the expression length of the developed NLEVM and improves interpretability by selecting model components that are associated with the use case's physical features. Furthermore, setting custom symbol complexity values is shown to be useful to prevent an excessive usage of nonlinear mathematical operators when developing laminar flame speed models. Finally, the implementability of the developed turbulence models, enabled by the introduced regularization methods, is demonstrated by running LES and RANS calculations, which yield promising predictions that outperform state-of-the-art turbulence models.

We consider the concept of adaptive symbols and the two regularization techniques important extensions to the GEP framework to progress towards developing more accurate and generalizing closure models while ensuring their implementability and ideally achieving interpretability. Although algebraic models are not inherently interpretable, mathematical expressions are the backbone of the natural sciences and many scientists have formed strong intuitions for their interpretation, which is not the case for the extremely large matrices that define parametric models, such as DNNs. Incentivizing the development of low complexity algebraic models further increases this advantage. Furthermore, developing closure models that continue to predict the learned statistical patterns outside the training data is necessary for generalization, but not sufficient. Once the underlying physical mechanisms of a use case change, accurate predictions can no longer be expected from the developed model. We think that including multiple, diverse use cases in the training dataset is necessary to advance towards generalizing closure models and Fang et al. (2023) demonstrated first promising results for training turbulence models on sets of diverse free-shear and wall-bounded flows.

Future research on adaptive symbols will focus on their utilization not only for supervised training on high-fidelity data but also in simulation-driven training. This approach ensures consistency between the training and prediction environments of the developed models, which is a frequent issue in the practial application of data-driven closure models (Duraisamy 2021). However, simulation-driven training is generally expensive and gradient-based optimization requires even more objective function evaluations than the gradient-free standard training approach. Identifying low-order approximations to the numerical solver could be one path to drastically reduce computational costs and enable the development of consistent models with accurate numerical constants.
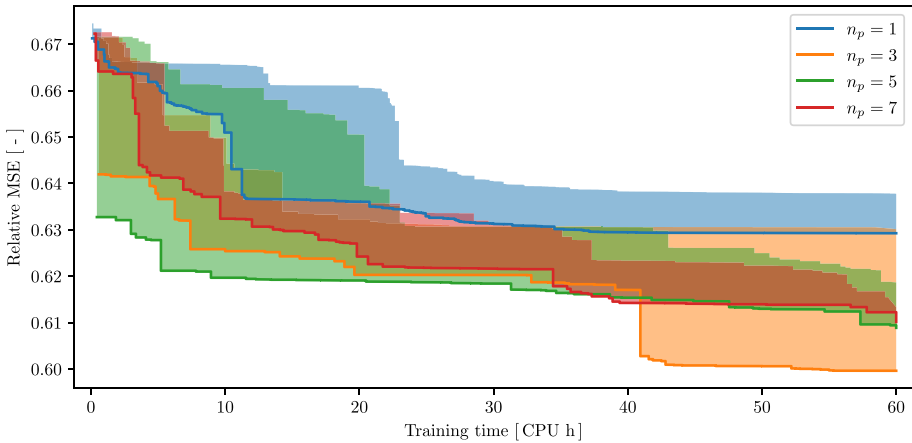
Fig. 13 Convergence of MSE relative to LEVM of R models trained with varying number of adaptive symbols per scalar function for NLEVM case (shaded areas represent variation due to random initialization)

Table 6 Training and testing MSE relative to LEVM, model complexity J_c and maximum absolute adaptive symbol value $\max_i(|p_i|)$ of R models trained with varying L_2 regularization parameter λ for NLEVM case

λ	Train. MSE	Test. MSE	J_c	$\max_i(|p_i|)$
0.0	0.5644	0.5558	186	8.55×10^{17}
10^{-9}	0.5790	0.5700	170	8.72×10^{1}
10^{-7}	0.5874	0.5770	164	1.12×10^{1}
10^{-5}	0.6166	0.6052	194	6.97×10^{0}
10^{-3}	0.6162	0.6048	226	2.18×10^{-1}

Hyperparameter Studies for NLEVM Use Case

Two hyperparameter studies are performed to determine the optimal number of adaptive symbols per scalar function n_p and the L_2 regularization parameter λ for the NLEVM use case in Sect. 3.4. We focus on the development of R models, which present a higher potential for improvement compared to a_{ij} models (see Fig. 9), and limit the maximum training runtime to 60 CPU hours in order to reduce the computational costs. The other modeling and training settings are unchanged from Table 7.

To identify the optimal n_p value, we employ the BFGS optimizer, which is expected to require more adaptive symbols than the LM optimizer due to the performance disadvantages observed in Sects. 3.1 and 3.3. Figure 13 shows the convergence of the MSE relative to the baseline LEVM, which applies no turbulence production correction, for training with $n_p \in \{1, 3, 5, 7\}$ adaptive symbols per scalar function. While $n_p = 1$ is clearly not sufficient to reduce the error to the level of the other approaches, utilizing three adaptive symbols per scalar function achieves the minimum MSE value. However, the orange shaded area indicates a high dependency on the random initialization for the $n_p = 3$ training runs. This dependency is significantly reduced for $n_p = 5$ and $n_p = 7$. Since $n_p = 7$ does not further improve the MSE value compared to $n_p = 5$, we select five adaptive symbols per scalar function as the optimal trade-off between training performance and reliability.

For the L_2 regularization parameter study, the LM optimizer is selected, as unregularized NLEVM training with this optimizer leads to very high adaptive symbol values (see Table 3). We explore $\lambda \in \{0.0, 10^{-9}, 10^{-7}, 10^{-5}, 10^{-3}\}$ and Table 6 lists the training and testing MSE values, the model complexities J_c and the maximum adaptive symbol magnitudes $\max_i(|p_i|)$ of the resulting R models. Interestingly, a small amount of L_2 regularization, realized by setting $\lambda = 10^{-9}$, is sufficient to yield p values below 10^2. Increasing the regularization parameter to $\lambda = 10^{-7}$ results in a similar testing error and reduces the $\max_i(|p_i|)$ value by nearly one order of magnitude. While the even larger values of $\lambda = 10^{-5}$ and $\lambda = 10^{-3}$ reduce the $\max_i(|p_i|)$ value further, both MSE values and the model complexity increase. Thus, we choose $\lambda = 10^{-7}$ for the regularized training in Sect. 3.4.

Modeling and Training Strategies

See Table 7.

Table 7 Overview of modeling and training strategies for investigated use cases

Use case	Sutherland's law	Laminar flame speed	SGS modeling	NLEVM modeling
Models	$\mu, \hat{\mu}$	$\hat{S}_L$	$\tau_{ij}^{A,GEP}$	a_{ij}/a_{ij}^R
Training objective	MSE	Normalized MSE	MSE	MSE
Input symbols	T	$\hat{p}, \hat{T}, \hat{\phi}$	$I^1, \dots, I^4$	$I^1, \dots, I^5$
Mathematical operators	$+, -, \times, \div, (\cdot)^{\frac{3}{2}}$	$+, -, \times, \exp, \log, \sqrt{\cdot}$	$+, -, \times$	$+, -, \times$
Numerical constants	0, 1, 2	$-1, 0, 1, 2$	$-1, 0, 1, 2$	$-1, 0, 1, 2$
# RNCs	5/0	5/0	5/0	5/0
# Adaptive symbols	0/5	0/10	0/16	0/50
Optimizers	–/BFGS/LM	–/BFGS	–/BFGS/LM	–/BFGS/LM
Regularization	–	$-/J_c$	$-/\lambda = 10^{-5}$	$-/\lambda = 10^{-7}, J_c$
Population size	1000	200	100	100
# Genes	2	5	4	10
# Initializations	5	3	5	3
Max. runtime [CPU h]	0.25	72/120	8.3	240

92152102) for financial support. Richard D. Sandberg acknowledges financial support from the Australian Research Council.

Declarations

References

Beeckmann, J., Hesse, R., Bejot, F., Xu, N., Pitsch, H.: Assessment of the Approximation Formula for the Calculation of Methane/Air Laminar Burning Velocities Used in Engine Combustion Models. Technical Report, SAE Technical Paper (2017)

Broyden, C.G.: The convergence of a class of double-rank minimization algorithms: 2. The new algorithm. IMA J. Appl. Math. **6**, 222–231 (1970)

Brunton, S.L., Proctor, J.L., Kutz, J.N.: Discovering governing equations from data by sparse identification of nonlinear dynamical systems. Proc. Natl. Acad. Sci. **113**, 3932–3937 (2016)

Chen, Q., Zhang, M., Xue, B.: Structural risk minimization-driven genetic programming for enhancing generalization in symbolic regression. IEEE Trans. Evol. Comput. **23**, 703–717 (2018)

Chen, Q., Xue, B., Zhang, M.: Rademacher complexity for enhancing the generalization of genetic programming for symbolic regression. IEEE Trans. Cybern. (2020)

Cranmer, M., Sanchez Gonzalez, A., Battaglia, P., Xu, R., Cranmer, K., Spergel, D., Ho, S.: Discovering symbolic models from deep learning with inductive biases. Adv. Neural. Inf. Process. Syst. **33**, 17429–17442 (2020)

Dominique, J., Christophe, J., Schram, C., Sandberg, R.D.: Inferring empirical wall pressure spectral models with Gene Expression Programming. J. Sound Vib. **506**, 116162 (2021)

Duraisamy, K.: Perspectives on machine learning-augmented Reynolds-averaged and large eddy simulation models of turbulence. Phys. Rev. Fluids **6**, 050504 (2021)

Durbin, P.A.: Some recent developments in turbulence closure modeling. Annu. Rev. Fluid Mech. **50**, 77–103 (2018)

Ewald, J., Peters, N.: A level set based flamelet model for the prediction of combustion in spark ignition engines. In: 15th International Multidimensional Engine Modeling User's Group Meeting, Detroit, MI (2005)

Fang, Y., Zhao, Y., Waschkowski, F., Ooi, A.S.H., Sandberg, R.D.: Toward more general turbulence models via multicase computational-fluid-dynamics-driven training. AIAA J. **61**, 2100–2115 (2023)

Ferreira, C.: Gene Expression Programming: mathematical modeling by an artificial intelligence, volume 21 of Studies in Computational Intelligence. Springer (2006)

Ferreira, C.: Gene Expression Programming: a new adaptive algorithm for solving problems. Complex Syst. **13**, 87–129 (2001)

Fletcher, R.: A new approach to variable metric algorithms. Comput. J. **13**, 317–322 (1970)

Fureby, C., Tabor, G., Weller, H.G., Gosman, A.D.: A comparative study of subgrid scale models in homogeneous isotropic turbulence. Phys. Fluids **9**, 1416–1429 (1997)

Gamahara, M., Hattori, Y.: Searching for turbulence models by artificial neural network. Phys. Rev. Fluids **2**, 054604 (2017)

Goldfarb, D.: A family of variable-metric methods derived by variational means. Math. Comput. **24**, 23–26 (1970)

Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press, Cambridge (2016)

Göttgens, J., Mauss, F., Peters, N.: Analytic approximations of burning velocities and flame thicknesses of lean hydrogen, methane, ethylene, ethane, acetylene, and propane flames, In: Symposium (International) on Combustion, Elsevier. pp. 129–135 (1992)

Gülder, Ö.L.: Correlations of Laminar Combustion Data for Alternative SI Engine Fuels. Technical Report. Society of Automotive Engineers, Inc., Warrendale, PA (1984)

Haghiri, A., Lav, C., Sandberg, R.: Data-driven turbulence modelling for improved prediction of ship air-wakes. In: 33rd Symposium on Naval Hydrodynamics, Osaka, Japan (2020)

Hann, S., Grill, M., Bargende, M., Altenschmidt, F.: A Quasi-Dimensional SI Burn Rate Model for Predicting the Effects of Changing Fuel, Air-Fuel-Ratio. EGR and Water Injection, Technical Report (2020)

Hesse, R., Beeckmann, J., Wantz, K., Pitsch, H.: Laminar Burning Velocity of Market Type Gasoline Surrogates as a Performance Indicator in Internal Combustion Engines. Technical Report (2018)

Holland, J.H.: Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. MIT Press, Cambridge (1975)

Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A.: Highly accurate protein structure prediction with AlphaFold. Nature **596**, 583–589 (2021)

Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection, vol. 1. MIT Press, Cambridge (1992)

Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Adv. Neural. Inf. Process. Syst. **25**, 1097–1105 (2012)

Leschziner, M.: Statistical Turbulence Modelling for Fluid Dynamics-Demystified: An Introductory Text for Graduate Engineering Students. World Scientific (2015)

Levenberg, K.: A method for the solution of certain non-linear problems in least squares. Q. Appl. Math. **2**, 164–168 (1944)

Li, X., Zhou, C., Nelson, P.C., Tirpak, T.M.: Investigation of constant creation techniques in the context of gene expression programming. LNCS 3103 (2004)

Li, H., Zhao, Y., Wang, J., Sandberg, R.D.: Data-driven model development for large-eddy simulation of turbulence using gene-expression programing. Phys. Fluids **33**, 125127 (2021)

Lilly, D.K.: The representation of small-scale turbulence in numerical simulation experiments. IBM Form , 195–210 (1967)

Liu, S., Meneveau, C., Katz, J.: On the properties of similarity subgrid-scale models as deduced from measurements in a turbulent jet. J. Fluid Mech. **275**, 83–119 (1994)

Lopes, H.S., Weinert, W.R.: EGIPSYS: An enhanced gene expression programming approach for symbolic regression problems. Int. J. Appl. Math. Comput. Sci. **14**, 375–384 (2004)

Marquardt, D.W.: An algorithm for least-squares estimation of nonlinear parameters. J. Soc. Ind. Appl. Math. **11**, 431–441 (1963)

Meneveau, C., Katz, J.: Scale-invariance and turbulence models for large-eddy simulation. Annu. Rev. Fluid Mech. **32**, 1–32 (2000)

Menter, F.R.: Two-equation eddy-viscosity turbulence models for engineering applications. AIAA J. **32**, 1598–1605 (1994)

Metghalchi, M., Keck, J.C.: Burning velocities of mixtures of air with methanol, isooctane, and indolene at high pressure and temperature. Combust. Flame **48**, 191–210 (1982)

Meurer, A., Smith, C.P., Paprocki, M., Čertík, O., Kirpichev, S.B., Rocklin, M., Kumar, A., Ivanov, S., Moore, J.K., Singh, S.: SymPy: symbolic computing in Python. PeerJ Comput. Sci. **3**, e103 (2017)

Montáns, F.J., Chinesta, F., Gómez-Bombarelli, R., Kutz, J.N.: Data-driven modeling and learning in science and engineering. Comptes Rendus Mécanique **347**, 845–855 (2019)

Nocedal, J., Wright, S.J.: Numerical Optimization. Springer, New York (1999)

Pitsch, H.: FlameMaster: A C++ computer program for 0D combustion and 1D laminar flame calculations. Cited in 81 (1998)

Pope, S.B.: A more general effective-viscosity hypothesis. J. Fluid Mech. **72**, 331–340 (1975)

Pope, S.B.: Turbulent Flows. IOP Publishing (2001)

Röhl, O., Jerzembeck, S., Beeckmann, J., Peters, N.: Numerical Investigation of Laminar Burning Velocities of High Octane Fuel Blends Containing Ethanol. Technical Report, SAE Technical Paper (2009)

Ryan, C., Keijzer, M.: An analysis of diversity of constants of genetic programming. In: European Conference on Genetic Programming, pp. 404–413. Springer (2003)

Sagaut, P.: Large Eddy Simulation for Incompressible Flows: An Introduction. Springer, New York (2006)

Sandberg, R.D., Tan, R., Weatheritt, J., Ooi, A., Haghiri, A., Michelassi, V., Laskowski, G.: Applying machine learnt explicit algebraic stress and scalar flux models to a fundamental trailing edge slot. J. Turbomach. **140**, 101008 (2018)

Schmelzer, M., Dwight, R.P., Cinnella, P.: Discovery of Algebraic Reynolds-Stress Models Using Sparse Symbolic Regression. Flow, Turbulence and Combustion (2019)

Schmidt, M., Lipson, H.: Distilling free-form natural laws from experimental data. Science **324**, 81–85 (2009)

Seshadri, K., Göttgens, J.: Structure of the oxidation layer for stoichiometric and lean methane-air flames. In: Reduced Kinetic Mechanisms and Asymptotic Approximations for Methane-Air Flames, pp. 111–136 (1991)

Shanno, D.F.: Conditioning of quasi-Newton methods for function minimization. Math. Comput. **24**, 647–656 (1970)

Smagorinsky, J.: General circulation experiments with the primitive equations: I. The basic experiment. Month. Weather Rev. **91**, 99–164 (1963)

Smits, G.F., Kotanchek, M.: Pareto-front exploitation in symbolic regression. In: Genetic Programming Theory and Practice II, pp. 283–299. Springer (2005)

Soule, T., Foster, J.A.: Effects of code growth and parsimony pressure on populations in genetic programming. Evol. Comput. **6**, 293–309 (1998)

Sutherland, W.: LII. The viscosity of gases and molecular force. Lond. Edinb. Dublin Philos. Mag. J. Sci. **36**, 507–531 (1893)

Vanneschi, L., Castelli, M., Silva, S.: Measuring bloat, overfitting and functional complexity in genetic programming. In: Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation, pp. 877–884 (2010)

Virtanen, P., Gommers, R., Oliphant, T.E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J.: SciPy 1.0: fundamental algorithms for scientific computing in python. Nat. Methods **17**, 261–272 (2020)

Vladislavleva, E.J., Smits, G.F., Den Hertog, D.: Order of nonlinearity as a complexity measure for models generated by symbolic regression via pareto genetic programming. IEEE Trans. Evol. Comput. **13**, 333–349 (2008)

Vreman, B., Geurts, B., Kuerten, H.: Large-eddy simulation of the turbulent mixing layer. J. Fluid Mech. **339**, 357–390 (1997)

Wang, H.F., Zhou, Y.: The finite-length square cylinder near wake. J. Fluid Mech. **638**, 453–490 (2009)

Waschkowski, F., Zhao, Y., Sandberg, R., Klewicki, J.: Multi-objective CFD-driven development of coupled turbulence closure models. J. Comput. Phys. **452**, 110922 (2022)

Weatheritt, J., Sandberg, R.: A novel evolutionary algorithm applied to algebraic modifications of the RANS stress–strain relationship. J. Comput. Phys. **325**, 22–37 (2016a)

Weatheritt, J., Sandberg, R.: Hybrid simulation of the surface mounted square cylinder. In: Proceedings, pp. 5–8 (2016b)

Weatheritt, J., Pichler, R., Sandberg, R.D., Laskowski, G., Michelassi, V.: Machine learning for turbulence model development using a high-fidelity HPT cascade simulation, In: ASME Turbo Expo 2017: Turbomachinery Technical Conference and Exposition, American Society of Mechanical Engineers, pp. V02BT41A015–V02BT41A015 (2017)

Weller, H.G., Tabor, G., Jasak, H., Fureby, C.: A tensorial approach to computational continuum mechanics using object-oriented techniques. Comput. Phys. **12**, 620–631 (1998)

White, F.M., Majdalani, J.: Viscous Fluid Flow, vol. 3. McGraw-Hill, New York (2006)

Xie, C., Yuan, Z., Wang, J.: Artificial neural network-based nonlinear algebraic models for large eddy simulation of turbulence. Phys. Fluids **32**, 115101 (2020)

Xu, K., Zhang, M., Li, J., Du, S.S., Kawarabayashi, K.i., Jegelka, S.: How neural networks extrapolate: from feedforward to graph neural networks. In: International Conference on Learning Representations (ICLR) (2021)

Zarnegar, A., Vamplew, P., Stranieri, A.: Inference of gene expression networks using memetic gene expression programming. In: Proceedings of the Thirty-Second Australasian Conference on Computer Science-vol. 91, pp. 29–36 (2009)

Zhang, Q., Zhou, C., Xiao, W., Nelson, P.C.: Improving gene expression programming performance by using differential evolution. In: Sixth International Conference on Machine Learning and Applications (ICMLA 2007), pp. 31–37. IEEE (2007)

Zhao, Y., Akolekar, H.D., Weatheritt, J., Michelassi, V., Sandberg, R.D.: RANS turbulence model development using CFD-driven machine learning. J. Comput. Phys. **411**, 109413 (2020)

Zhong, J., Feng, L., Ong, Y.S.: Gene expression programming: a survey. IEEE Comput. Intell. Mag. **12**, 54–72 (2017)