OPEN ACCESS
IOP Publishing
Journal of Physics G: Nuclear and Particle Physics
J. Phys. G: Nucl. Part. Phys. **52** (2025) 102001 (30pp)
https://doi.org/10.1088/1361-6471/ae0b98

Guide

Genetic programming for the nuclear many-body problem: a guide

Illya Bakurov[1], Pablo Giuliani[2], Kyle Godbey[2], Nathaniel Haut[3], Wolfgang Banzhaf[1,4] and Witold Nazarewicz[2,5,*]

[1] Department of Computer Science and Engineering, Michigan State University, East Lansing, MI, United States of America
[2] Facility for Rare Isotope Beams, Michigan State University, East Lansing, MI 48824, United States of America
[3] Department of Computational Mathematics, Science, and Engineering, Michigan State University, East Lansing, MI, United States of America
[4] BEACON Center for the Study of Evolution in Action, Michigan State University, East Lansing, MI, United States of America
[5] Department of Physics and Astronomy, Michigan State University, East Lansing, MI 48824, United States of America

E-mail: witek@frib.msu.edu

Received 10 June 2025, revised 8 September 2025
Accepted for publication 25 September 2025
Published 14 October 2025

Abstract

Genetic Programming (GP) is an evolutionary algorithm that generates computer programs, or mathematical expressions, to solve complex problems. In this Guide, we demonstrate how to use GP to develop surrogate models to mitigate the computational costs of modeling atomic nuclei with ever increasing complexity. The computational burden escalates when uncertainty quantification is pursued, or when observables must be globally computed for thousands of nuclei. By studying three models in which the mean field depends on the total particle density self-consistently, we show that by constructing reduced order models supported by GP one can speed up many-body computations by several orders of magnitude with a negligible loss in accuracy.

* Authors to whom any correspondence should be addressed.

Keywords: nuclear many-body problem:, genetic programming, reduced order models, dimensionality reduction

1. Introduction

Computational models for many-body quantum systems, such as the atomic nucleus, have quickly grown in complexity as theories are refined, more experimental data became available, and computers grew more powerful. In research areas with practical applications, the demand for more accurate descriptions of physical phenomena, together with a push for predictions with well-quantified uncertainties, has driven various efforts to create surrogate models that can learn low-dimensional representations of the underlying system equations.

To overcome the dimensional barrier of nuclear models with many degrees of freedom, we explore the application of Genetic Programming (GP) [1] for obtaining reduced equations from data. GP is a type of Artificial Intelligence (AI) that falls under the broader category of evolutionary computation, which is a subset of AI inspired by biological evolution mechanisms such as natural selection and stochastic variation [2, 3].

Here we use GP in the context of the reduced basis method (RBM) [4–6] that works by identifying reduced coordinates as the amplitudes of a reduced basis informed by a small number of high-fidelity (or full order model, FOM) evaluations of the system. Some of these developments follow a supervised machine learning (SML) [7] approach whereby various algorithms are trained to reproduce the dynamics of the reduced coordinates as control parameters change.

In this work, we explore GP to obtain the reduced equations from data. GP begins with a randomly generated initial population of individuals—each representing a model/equation of the control parameters in the reduced coordinates. The algorithm then simulates an evolutionary process over thousands of generations. During this process, individuals undergo selection based on their fitness, which evaluates their effectiveness in capturing the underlying reduced dynamics of the system. Concurrently, stochastic modifications are made to their structure, similar to genetic mutations and crossover found in biological DNA. These modifications help to explore and exploit the search space by creating increasingly effective models that can discover simple yet accurate equations for the dynamics of the system in reduced coordinates as a function of the controlling parameters. With this approach we are able to create reduced-order models in a straightforward manner for describing many-body quantum systems that would have otherwise posed appreciable challenges with traditional approaches.

This Guide is organized as follows. The principles of GP are outlined in section 2. Three illustrative many-body case studies are introduced in section 3. Section 4 describes the dimensionality reduction techniques employed. In section 5 we present the results of our numerical tests varying in functional complexity, regularization strength, and the inclusion of feature selection strategies. This comparative analysis helps to highlight the strengths and limitations of GP relative to conventional data-driven approaches. Finally, in section 6, we close by presenting our conclusions and outlook for future directions.

2. Genetic programming

Evolutionary Computation is a subset of AI inspired by biological evolution mechanisms. A prominent technique within this domain is the Genetic Algorithm, a search heuristic that mimics the process of natural selection to find optimal solutions. Genetic algorithms operate on a population of potential solutions, often represented as fixed-length strings, and apply bio-inspired operators like selection, crossover, and mutation to iteratively evolve better solutions. These algorithms are well-suited for optimization and search problems where the solution space is large and complex. Building upon the foundational principles of Genetic Algorithms, GP emerges as a specialized extension. In GP, the individuals undergoing the evolutionary process are not simple parameter strings but are entire computer programs. While Genetic algorithms are typically employed to find an optimal set of parameters for a problem with a known structure, GP tackles the more complex challenge of evolving the structure of the program itself.

Due to the high degree of flexibility of representations and modularity, GP has been extensively used in numerous machine learning (ML) domains. It has been demonstrated to be among the most effective approaches for solving binary [8] and multi-label classification problems [9]. A common application of GP is Symbolic Regression [10, 11], which is the learning of mathematical expressions to approximate some observed training data. In [12] contemporary GP-based Symbolic Regression methods are shown to achieve competitive or better generalization to state-of-the-art ML methods in this domain, while producing simpler models. Other relevant applications of GP include feature engineering for regression [13] and classification tasks [14, 15], learning lower-dimensional representations (embeddings) of high-dimensional spaces through nonlinear transformations [16], ensemble learning with stacked generalization [17] and boosting methods [18], active learning [19], design of image enhancement pipelines [20, 21], biomedical image classification [22, 23] and segmentation [24, 25], and even automating certain ML workflows (aka AutoML) [26, 27]. Recent works show that GP can be used for enhancing code-generation in Large Language Models (LLMs) [28].

Examples of applications of Genetic Algorithms and GP to various nuclear-physics problems, ranging from data analysis to theoretical modeling, can be found in, e.g. [29–37].

There are many possible representations employed in GP but it is common to use expression trees in tree GP [1], linear sequences of instructions in linear GP [38], and circuit-type graphs in Cartesian GP [39]. The choice of representation can have a significant impact on solution reachability since each representation requires different genetic operators (functions that modify the genotype) that impact how solutions transfer learned knowledge across generations [40]. In this work, we use tree GP and will refer to it as simply GP.

In GP, the evolving programs are constructed by composing elements belonging to two specific, predefined, sets: a set of primitive functions F, which appear as the internal nodes of the trees, and a set of terminal input features T, and constant values C, which represent the leaves of the trees. In the context of SML problems, the trees encode symbolic expressions mapping inputs $T \cup C$ to outputs Y. Figure 1 provides a visualization of how GP-models can represent a mathematical function as a tree of functions, input features and constant values. Note that some functions are unary, while other are binary, which results in subtrees with one or two leaf nodes. In general, GP trees can use functions of any arity, although it is common to see just unary and binary operators.

Typically, GP is used with the so-called subtree mutation and swap crossover [1]. The former randomly selects a subtree in the structure of the parent individual and replaces it with a new, randomly generated tree. The goal of mutation is to develop new traits/skills in

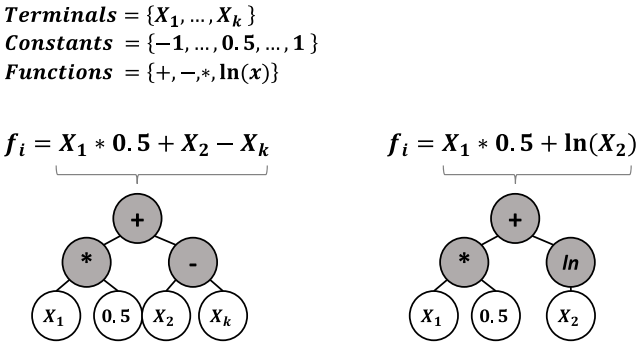

Figure 1. A diagram showing how mathematical functions can be represented as trees in Tree Genetic Programming. The gray nodes represent the mathematical functions of different arities, whereas the white nodes represent the input features and constants (aka terminal nodes).

the population (exploration). The swap crossover exchanges two randomly selected subtrees between two different parent individuals. The goal here is to transfer learned skills to new individuals to hopefully find improved combinations of existing skills in the population (exploitation). The quality of an individual is assessed using a fitness function. The fitness function is problem specific, but should capture how well an individual accomplished the task. For example, in symbolic regression, root-mean-squared error and R^2 are often used as fitness functions. Algorithm 1 provides pseudo-code for a typical GP algorithm. In the algorithm, elitism refers to the process of preserving the best individual(s) with respect to the fitness function across generations. This ensures that the algorithm does not lose progress, and retains a good-performing individual.

Algorithm 1. Pseudo-code for the typical GP algorithm

1: Create an n-sized initial population of random individuals P
2: Repeat until matching a termination condition (e.g. number of generations):
(**a**)**:** Calculate fitness $\forall$ individuals i in P
(**b**)**:** Initialize child population P'
(**c**)**:** Repeat until P' contains n individuals:
 (**i**)**:** With probability P_c, choose crossover (transfer)
 operator; else choose mutation $(1 - P_c)$
 (**ii**)**:** Select 2 individuals using a selection algorithm
 (**iii**)**:** Apply the operator to the selected individuals
 (**iv**)**:** Insert the resulting individuals into P'
(**d**)**:** Replace P with P' (copy elites from P if elitism is used)
3: Return the best individual(s) in P

In a ML context, GP provides users a different experience in terms of transparency and interpretability of the final models when compared to more 'black-box' models of, e.g. deep artificial neural networks. This is because GP typically outputs symbolic expressions or structures representing programs, which are generally interpretable and can be analyzed by the researcher. By manipulating discrete structures, the inner workings of a GP-based model are clearer, and the relationships between features and predictions are explicitly defined. As a

result, evolved solutions can offer valuable insights into how the model arrives at its predictions, allowing users to understand the algorithmic decisions and relate this to their domain-knowledge.

Another major advantage of GP is that the evolutionary process tends to promote individuals (i.e. programs or models) that manipulate relevant input variables in informative ways, while those relying on redundant or uninformative features are less likely to be selected for reproduction. As a result, GP naturally performs feature selection during the search, often eliminating the need for an explicit feature selection step commonly required in other ML pipelines [41]. Moreover, GP's ability to evolve modular substructures allows it to abstract and reuse useful building blocks, further enhancing its capacity to identify important relationships in the data and construct meaningful features [42]. These properties make GP particularly well-suited for modeling tasks involving high-dimensional input spaces or requiring interpretable and low-latency inference, like in this paper.

Beyond the models just being symbolic, GP has the benefit that it evolves a population of models rather than just a single solution. Having access to a final population of models unlocks a lot of opportunities that are unavailable to other ML workflows. For example, while it is customary to just pick the most accurate model from the population on the training set, it is possible to explore many of the candidate models and use human expertize to select from those models the one, or multiple, that best align with domain knowledge. Rather than picking one model from the evolved population, there are methods used in GP to select a model ensemble to be used in multi-modeling approaches [43]. GP ensembles can leverage the diversity of patterns to improve stability in deployment and also mitigate single-model risks.

The populations of solutions can also be studied to gain insight into the system of study. For example, it is possible to detect patterns regarding which features are common in the population, or which sets of features commonly appear together in models. This can provide researchers with key insights into which features provide predictive synergies.

Although GP is not the only 'white-box' approach in the spectrum of ML methods, it allows for unprecedented flexibility of representations and modularity, making it suitable for numerous tasks. The area of explainable AI receives consistently more attention from both practitioners and researchers [44] and GP has gained popularity where human-interpretable solutions are paramount. Real-world examples include medical image segmentation [24], prediction of human oral bioavailability of drugs [45], skin cancer classification from lesion images [46], and even conception of models of human visual perception [47]. Besides being able to provide interpretable models, there is evidence that GP can also help to unlock the behavior of black-box models [48].

A distinctive advantage of GP, is the flexibility to use any optimization objective or to perform multi-objective optimization (MOO). For example, ParetoGP algorithm adopts a MOO approach to balancing expression complexity and accuracy while performing selection [49], the authors of [9] use parent selection and survival techniques that strive towards a higher accuracy and a lower age of the individuals in the a steady-state evolution, while in [50] a two-step parent selection is used by alternating between error and sharpness (i.e. smoothness of the GP-induced hyperplanes). Whereas many deep learning approaches require gradient information making them unsuitable for problems where derivatives are difficult to compute or do not exist [51]. In contrast, GP can use any objective function as long as there is some way to distinguish between more and less fit candidate-solutions (individuals) in a population.

ML methods for regression, such as Linear Regression, artificial neural networks [51], and support vector machines [52], typically rely on error-based metrics (penalty function)

such as the mean squared error to guide the model optimization. However, in the context of GP, such penalty functions have been found to be less effective at guiding the search toward solutions that both (i) exhibit the desired geometric structure of the response surface in the input space, and (ii) closely fit the target data [53, 54]. To mitigate this issue, [53] proposed to estimate the optimal linear transformation (slope and intercept) of the GP-generated expression with respect to the target outputs, allowing the evolutionary process to focus primarily on discovering the correct functional form or shape of the model. Although effective, this approach requires recomputing the scaling parameters at every generation for each individual. Later it was understood that using the coefficient of determination (R^2, i.e. squared Pearson's correlation) removes the need for explicit linear scaling and translation, as it directly identifies models that match the shape or form of the target function [49, 55].

Recently [56], emphasized that using R^2 as the fitness function in GP for SR can lead to significantly better generalization than standard Root Mean Squared Error (RMSE) alone (i.e. without scaling), particularly when training on small datasets. Importantly, this improvement is achieved without the need for computationally expensive linear scaling at each generation. Instead, scaling is applied only once as a post-processing step after the evolution. However, their experiments also revealed that while R^2 performs well under low-noise conditions, its advantage diminishes as noise levels increase, eventually degrading to the performance level of RMSE or worse. This suggests that R^2 is less robust to noise, limiting its effectiveness in high-noise regimes. Later, more evidence on the utility of R^2 to relieve GP from learning coefficients in SR was provided in [57, 58]. In particular, in [58] it has been found that R^2 improves generalization of evolved GP models regardless of the selection scheme and pressure.

2.1. GP software

Several open-source GP libraries are available, each addressing different audiences, needs, and perspectives with their GP implementations. GPLearn offers a scikit-learn-compatible API in Python for applying tree-based GP in SML problem-solving [59]. Distributed Evolutionary algorithms in Python (DEAP) is a Python-based framework for evolutionary algorithms, including tree-based GP [60]. Moreover, it offers parallel processing capabilities and its modular implementation is suitable for rapid prototyping. PushGP evolves programs in the stack-based Push language, designed to be used as the programming language within which evolving programs are expressed [61]. General Purpose Optimization Library provides a unified Python interface for a wide range of algorithms (including tree- and linear-based GP), supports CPU/GPU computations and batch processing [62]. KarooGP [63] and TensorGP [64] utilize TensorFlow to accelerate GP through vectorized operations on GPU, enhancing performance on large-scale problems. Waikato Environment for Knowledge Analysis (WEKA) is a Java-based software that provides a collection of several ML algorithms, including GP, for data mining and knowledge discovery [65]. HeuristicLab is a C# framework offering a wide range of algorithms for solving optimization problems, including GP, with a user-friendly GUI for experiment design and analysis [66]. In this paper, we rely on a commercially available system, called DataModeler [67]. It offers a user-friendly GUI and, most importantly, implements several state-of-the-art GP techniques such as Pareto tournament selection [49], correlation as a fitness function [40, 56], continued fraction building blocks [40], OrdinalGP [68], model curvature control [69], among others. The collection of many state-of-the-art features packed into a single tool, as a result of 25 years of dedicated development, is one its biggest attractions, as open-source implementations frequently use standard methods or only implement a few state-of-the-art methods, which might

result in under-performance. Moreover, the software provides discounted licenses for universities, facilitating its adoption in academic research. The developers at Evolved Analytics also regularly publish their techniques, so while the software itself is not open-source, the techniques developed by them and used within their tool are.

2.2. GP illustration

To illustrate the application of GP in a Symbolic Regression task, let's take equation II.3.24 from the Feynman Symbolic Regression Database [70] that relates flux heat (Flux) to the power source (Pwr):

$$\text{Flux} = \text{Pwr}/(4 * \pi * r^2). \tag{1}$$

Taking into account this equation, we built a data set with 6 variables and 100 data records. Each data record was pulled from a uniform distribution. Two of the variables are the real inputs, Pwr and r, two of the variables, rand1 and rand2, are randomly generated, and two of the features are noise-corrupted versions of Pwr and r, which we will call PwrNoise and rNoise. The first four features (Pwr, r, rand1, and rand2) are limited to values between 1 and 3. Noise-corrupted versions are generated with 50% noise where NoiseData = Data + ϵ with ϵ being uniformly distributed $\pm 25\%$ the range of Data. Random and noisy features were included here to illustrate the ability of GP to perform automatic feature selection and focus on only the most informative (noiseless) features. Once the data set is setup, we can proceed with applying GP to discover the functional relationship.

The first step in using GP is to choose the operator set (choice of mathematical operators). When setting up the operator set, the goal is to include enough operators so that it is useful, but not too many such that the search space becomes excessively large. In this case, we include the following operators, which happen to be the default set in DataModeler [67]: +, −, *, /, negate, x^2, sqrt, inverse, x^n, and continuedFraction.

Once the operator set is chosen, we can set the termination condition, typically either run-time or number of generations. In this case, we chose run-time, as it is typically more useful in application, with generation count more interesting for GP theory. In this example, we set run-time to 10 s, after which the search terminates. We can also set the number of parallel searches that we want to use. Typically, we set this to the number of cores available on the machine. In this case, that means 16 cores, i.e. 16 parallel searches. The search trace on this example problem is shown in figure 2. We can see that all of the parallel searches found a highly-fit solution within the time limit. Additionally, we can observe that there is a small variability between the best-of-the-search solutions which indicates a consistency between different instances of stochastic search in GP.

Once the GP search is complete, we can extract the evolved models to select a model, or a model ensemble, as well as explore properties of the evolved populations. Figure 3 shows the Pareto front plot of the evolved population along with the best model found:

$$\text{Flux} = 0.07957747154594767\frac{\text{Pwr}}{r^2} + 9.34 \cdot 10^{-18}, \tag{2}$$

which is the correct model within machine-precision errors. The red points represent the Pareto front [71] (non-dominated models with respect to complexity and accuracy), while the blue points represent the rest of the models returned by the search. As discussed, the population can also be studied directly to reveal important information. For example, we can look at the percentage of models that include each of the variables in the data set, as shown in figure 4. This shows that the GP search was able to correctly identify the useful variables and mostly ignored the noise-corrupted and non-informative variables. It is important to note that

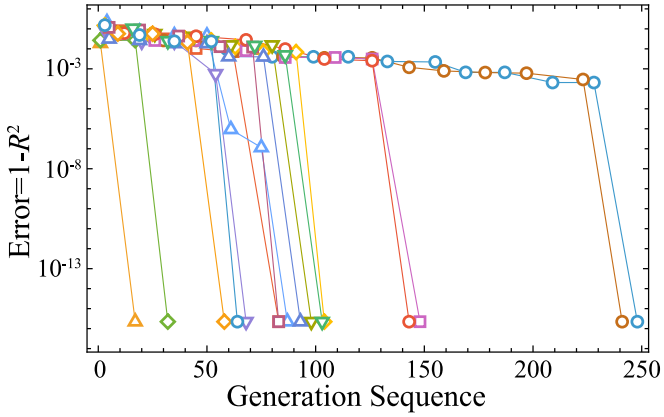

Figure 2. GP search progress over generations. Each color corresponds to an independent parallel search (run), with 16 total searches executed on 16 cores. The symbols indicate the fitness of the best-performing individual in the population at each generation, within each respective parallel search.

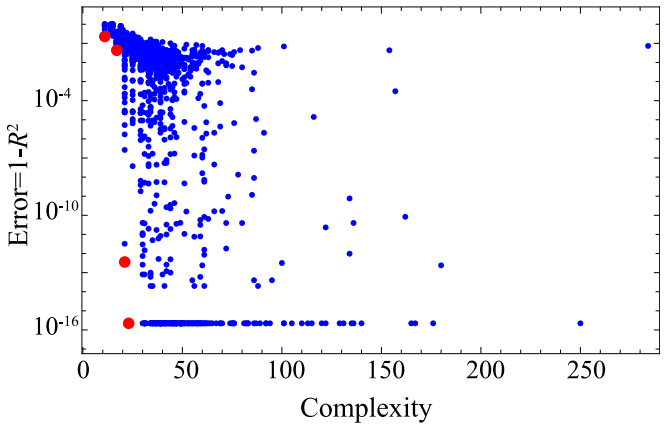

Figure 3. The Pareto front of GP models with respect to complexity and accuracy for illustration. The best model found is displayed and shows that the correct model was found by search. The red points represent the Pareto front of models.

the best model found contains only noise-free features. Since models in the population are randomly mutated, it is expected that some models may contain uninformative features in each generation, but the fraction of models with those uninformative features is small compared to the fraction of models using the informative features. If exploring a problem with an unknown form, this information could reveal which variables are key to understanding the system of study.

Although there is much more flexibility and features of GP, this illustration provides a simple and direct overview of how GP can be applied to a sample data set while providing some context for how it is applied in the rest of this work.

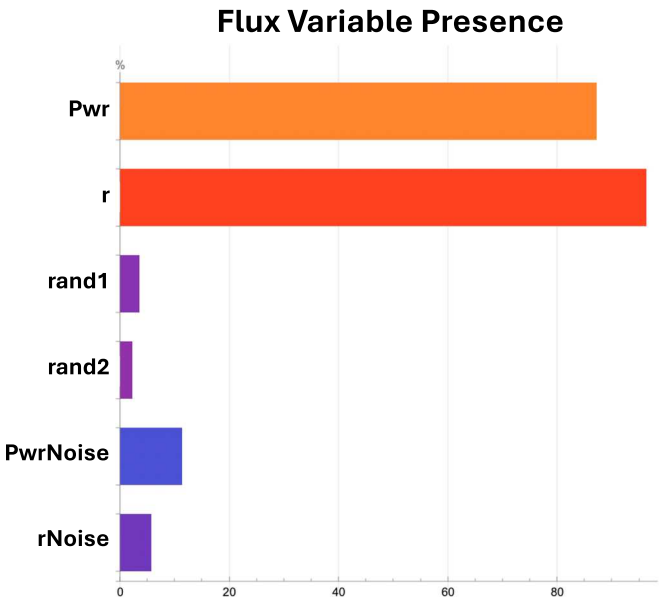

Figure 4. The percentage of models that include each variable from the sample data set. Showcasing GP's ability to focus on most informative features.

2.3. Linear regression

We benchmark the performance of GP as an alternative to build the reduced expressions for surrogate models with that of Linear Regression (LR), one of the most widely used SML methods for regression, known for its simplicity, interpretability, and efficiency. LR assumes a linear relationship between input variables/features (x) and the output target variable (y), expressed as a weighted sum of terms: $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n$. Here, the coefficients β_k are learnable parameters associated with the intercept and k^{th} input feature. Learning of parameters occurs by minimizing the sum of squared residuals (i.e. difference between the observed target values in the training dataset, and the values predicted by the linear model).

However, LR exhibits a fundamental limitation: it relies upon a fixed additive structure where the functional form of the input features and their interactions must be pre-defined by the user. This constrains LR's ability to accurately model complex nonlinear relationships or discover concise and appropriate functional representations of the data. GP, in contrast, explores and even composes diverse mathematical transformations and interactions without rigid assumptions.

To overcome the limitation of LR and make it more expressive, we designed a custom feature transformation pipeline that augments the original input space with various functional forms (e.g. powers, roots, logarithms, exponentials) and interaction terms (products and quotients). Recognizing the risk of overfitting due to the inflated feature space and fixed data size, we incorporate regularization techniques (such as L1 [72], L2 [73], ElasticNet [74], and up to two consecutive feature selection steps. The method is based on the Extremely Randomized Trees ensemble [75]—a method that combines decision-tree regression models similar to Random Forests [76], but with stronger randomization of the split point when inferring trees. Specifically, instead of exhaustively enumerating all possible splits to choose the one that minimizes a given loss function, it randomly chooses a set of splitting points and

then selects the best from those. This strategy was shown to report a better generalization ability and faster run-times when compared to Random Forests. Moreover, it was found to be particularly useful when dealing with high-dimensional and noisy data [75]—the reason why we employed Extremely Randomized Trees in the first step of our feature selection pipeline. The second feature selection step uses LR itself for refinement as it keeps the input features with above-median absolute coefficients. This allows us to focus on strong predictors based on linear influence.

Both feature transformations, regularization and feature selection steps are treated as hyperparameters of the pipeline, which are tuned by means of cross-validation. This strategy not only enhances LR's ability to model non-linearities but also ensures a fairer comparison with GP. Full implementation details and reference to the source code are provided in Appendix.

3. Many-body case studies

3.1. Modified Gross–Ppitaevskii equation

The first case study considered is a modified version of the one-dimensional modified Gross–Pitaevskii equation (MGPE), which is a nonlinear Schrödinger equation originally introduced to describe the ground state of identical bosons (Bose–Einstein condensate). The MGPE used in this study can be written as:

$$\left[-\frac{\mathrm{d}^2}{\mathrm{d}x^2} + \kappa x^2 + q\rho(x)^\sigma\right]\phi_i(x) = E_i\phi_i(x). \tag{3}$$

The mean-field Hamiltonian for a single-particle wave function ϕ_i consists of a trapping term proportional to x^2, as well as an interaction term depending on the total particle density ρ. In its original form [77], the particle density appears linearly, i.e. $\sigma = 1$. In this work, we modified the original equation to include more general, even fractional, powers of the density, resulting in three total control parameters $\alpha = \{\kappa, q, \sigma\}$. We also extended the MGPE to fermions. In this case, the particle density can be written as:

$$\rho(x) = \sum_i^N |\phi_i(x)|^2, \tag{4}$$

where N is the number of fermions.

These extensions stem from our desire to construct a simple model with a parametric dependence that mimics the fractional power density terms present in some nuclear models. As such, equation (3) should be viewed as an illustrative case for the non-affine and non-linear parametric dependence that challenges the application of some reduced-order model techniques [4, 6].

3.2. Skyrme density functional theory

The second case study is the description of the complex nucleus ^{48}Ca using nuclear Density Functional Theory (DFT). DFT is widely used in nuclear physics to describe complex atomic nuclei in a self-consistent framework. In nuclear DFT, the nucleons (protons and neutrons) interact through an effective interaction in the medium described by the energy density functional (EDF) of densities and currents.

The first DFT framework used in this study is a non-relativistic Skyrme-DFT (SDFT). In the simplest time-even variant, one considers local particle (ρ), kinetic (τ), spin-current

tensor ($\mathbb{J}$), and spin-orbit current ($\boldsymbol{J}$) densities (see [78, 79] for more details). Each of these densities is a function of the spatial coordinate $\boldsymbol{r}$, e.g. $\rho = \rho(\mathbf{r})$. In the following, this set of densities is denoted as $\{\rho\} \equiv \{\rho, \tau, \mathbb{J}, \boldsymbol{J}\}$. The total energy of the nucleus can be written as a functional of $\{\rho\}$:

$$E = E[\{\rho\}] = \int \mathcal{H}(\boldsymbol{r}) \mathrm{d}^3\boldsymbol{r}, \tag{5}$$

where $\mathcal{H}(\boldsymbol{r})$ is the Hamiltonian density.

The Skyrme EDF [78, 80, 81] can be expressed through densities $\{\rho\}$ as

$$\mathcal{H}(\boldsymbol{r}) = \mathcal{H}_0(\boldsymbol{r}) + \mathcal{H}_1(\boldsymbol{r}), \tag{6}$$

with

$$\begin{aligned}\mathcal{H}_t(\boldsymbol{r}) = {} & C_t^{\rho}\rho_t^2 + C_t^{\rho\Delta\rho}\rho_t\Delta\rho_t + C_t^{\tau}\rho_t\tau_t + C_t^{J}\mathbb{J}_t^2 \\ & + C_t^{\rho\nabla J}\rho_t\boldsymbol{\nabla}\cdot\boldsymbol{J}_t,\end{aligned} \tag{7}$$

where the isospin index t labels isoscalar ($t = 0$) and isovector ($t = 1$) densities. The Skyrme EDF coupling constants (here: controlling model parameters) are denoted as $\{C_t^{\rho}, C_t^{\rho\Delta\rho}, C_t^{\tau}, C_t^{J}, C_t^{\rho\nabla J}\}$. This functional is supplemented by the Coulomb term that describes the electromagnetic interaction, for which the coupling constant is fixed.

By varying the energy functional $E[\{\rho\}]$ with respect to the densities $\{\rho\}$ one obtains the mean-field, or Hartree–Fock, Hamiltonian $\hat{h}$. In the absence of nucleonic pairing, the single-nucleonic wave functions are obtained by solving the self-consistent Hartree–Fock eigen-problem:

$$\hat{h}[\{\rho\}]\phi_i = e_i\phi_i, \tag{8}$$

which is a coupled nonlinear integro-differential equation as the densities $\{\rho\}$ are built from single-particle wave functions ϕ_i as in equation (4).

3.3. Covariant nuclear density functional theory

The second DFT framework we consider is the covariant DFT (CDFT) approach, also known as relativistic mean-field theory. The basic assumptions of CDFT are similar to those of the Skyrme-DFT. Since the underlying formalism is relativistic, the Hamiltonian density is replaced by the Lagrangian density whose effective degrees of freedom are nucleons and mesons [78, 82–84]. This Lagrangian density consists of a nucleon-nucleon interaction mediated by various mesons alongside nonlinear meson interactions [85, 86]:

$$\begin{aligned}\mathscr{L}_{\mathrm{int}} = {} & \bar{\psi}\left[g_{\mathrm{s}}\phi - \left(g_{\mathrm{v}}V_{\mu} + \frac{g_{\rho}}{2}\tau\cdot\mathbf{b}_{\mu} + \frac{e}{2}(1+\tau_3)A_{\mu}\right)\gamma^{\mu}\right]\psi \\ & - \frac{\kappa}{3!}(g_{\mathrm{s}}\phi)^3 - \frac{\lambda}{4!}(g_{\mathrm{s}}\phi)^4 + \frac{\zeta}{4!}g_{\mathrm{v}}^4(V_{\mu}V^{\mu})^2 + \\ & - \Lambda_{\mathrm{v}}(g_{\rho}^2\,\mathbf{b}_{\mu}\cdot\mathbf{b}^{\mu})(g_{\mathrm{v}}^2V_{\nu}V^{\nu}),\end{aligned} \tag{9}$$

where the nucleon fields are denoted by ψ, meson fields by ϕ, V, and $\mathbf{b}$, and the Lagrangian coupling constants (controlling parameters) are $\{g_{\mathrm{s}}, g_{\mathrm{v}}, g_{\rho}, \kappa, \lambda, \Lambda_{\mathrm{v}}, \zeta\}$. From this Lagrangian density, coupled equations can be derived for the meson fields (the Klein–Gordon equations) and the nucleonic wave functions ϕ_i (the Dirac equation). For links between CDFT and SDFT, see [78].

Table 1. Ranges of the parameters used to train and test the reduced order model in the three case studies considered. The MGPE parameters are dimensionless. The DFT parameters J and L are in MeV. 492 evaluations of the MGPE were used for training the reduced models, and 212 evaluations were sampled across the same parameter range for testing. 49 evaluations of the SDFT and CDFT models were used for training the reduced models, and 49 for testing them in both cases, sampled from the ranges specified in the table under the subscripts 'train' and 'test', respectively.

Case study	Parameter	Range
MGPE	κ	$[0.5, 3.0]_{\text{train and test}}$
	q	$[-2.0, 2.0]_{\text{train and test}}$
	σ	$[0.5, 3.0]_{\text{train and test}}$
SDFT	J	$[25, 35]_{\text{train}}$, $[20, 43]_{\text{test}}$
	L	$[20, 60]_{\text{train}}$, $[10, 90]_{\text{test}}$
CDFT	J	$[30, 40]_{\text{train}}$ $[28, 50]_{\text{test}}$
	L	$[50, 130]_{\text{train}}$,$[40, 160]_{\text{test}}$

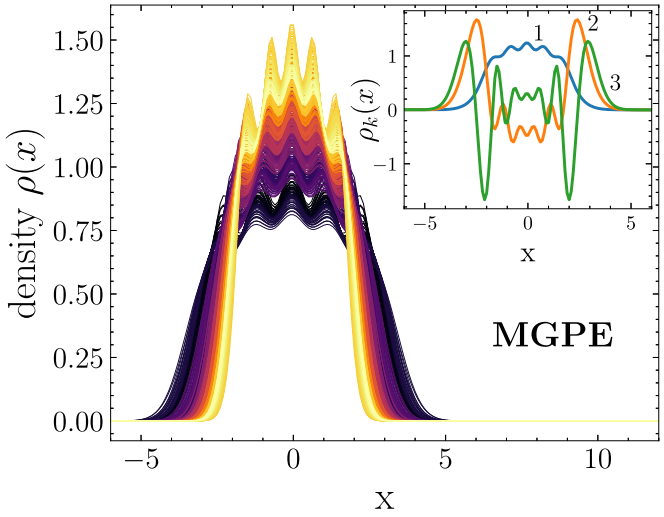

Figure 5. Densities $\rho(x)$ obtained in MGPE for the 492 training evaluations across the parameters listed in table 1. The inset shows the three principal components defined in equation (10) for $k = 1$, 2, 3. All quantities are in arbitrary units.

3.4. Parameter ranges

Table 1 shows the ranges of parameters explored in the three case studies. As an illustrative example, figure 5, pertaining to the MGPE, shows the density $\rho(x)$ for a system with $N = 5$ for various values of the control parameters.

The control parameters in both DFT frameworks, for example C_t^ρ in equation (7) and g_s in equation (9), determine how the solutions, including the nucleonic wave functions ϕ_i, change. For calibration purposes, these parameters are usually expressed in terms of nuclear matter properties (see [78, 87, 88] for definitions and discussion). In the following, we explore two of those parameters that are common to both frameworks: J —the symmetry energy, and L—the symmetry energy slope for symmetric nuclear matter [78]. That is, in this

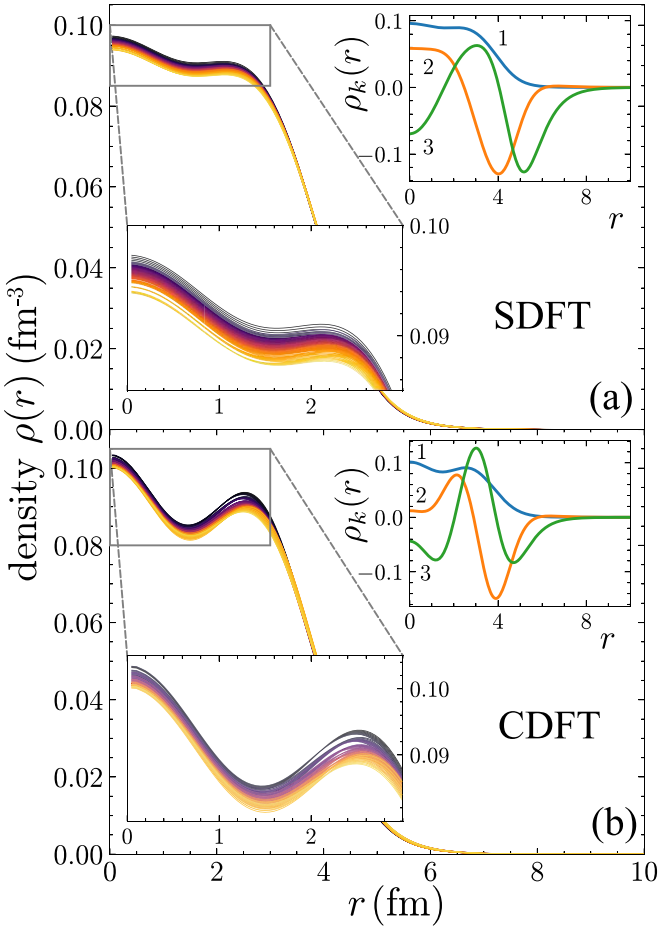

Figure 6. Neutron densities $\rho(r)$ for ^{48}Ca calculated in (a) SDFT and (b) CDFT within the parameter ranges listed in table 1. For both cases, 49 training and 49 testing parameter sets were evaluated, but 9 of the CDFT testing evaluations were excluded due to convergence issues. The inset shows the three principal components (in fm^{-3}) defined in equation (10) for $k = 1, 2, 3$.

case $\alpha = \{J, L\}$, figure 6 shows the neutron density $\rho_n(r)$ of ^{48}Ca for 49 different values of these parameters for both frameworks. The parameters are varied within the ranges specified in table 1, while keeping the other coupling constants fixed at their calibrated values [89, 90].

In all our case studies, the quantum many-body problem is solved using a self-consistent approach, typically involving iterative cycles in which the densities are fixed to compute the wave functions, which are then used to update the densities until convergence is reached. These calculations can become computationally intensive given their intrinsic high-dimensional structure. In the following section, we present an approach based on dimensionality reduction techniques enhanced through GP as an effective way to speed up such calculations.

4. Dimensionality reduction

Dimensionality reduction techniques aim to significantly reduce the active degrees of freedom when solving complex numerical problems while still retaining key features. These

approaches have been broadly applied in the physical sciences and are particularly powerful tools in the context of uncertainty quantification, where one typically needs to evaluate expensive computer models many times. The RBM [4, 5] is such a dimensionality reduction technique that falls under the reduced order modeling umbrella [91, 92].

The RBM identifies a reduced set of coordinates to approximate quantities of interest and then constructs governing equations to describe how those coordinates change with respect to controlling parameters. These reduced coordinates are the amplitudes of a reduced basis of the solution to the parametrized equations one is solving. For the case studies we outlined in the previous section, the RBM approximation could be applied to the densities ρ, taking the form:

$$\rho(x;\,\alpha) \approx \hat{\rho}(x;\,\alpha) = \sum_{k}^{n} a_k(\alpha)\rho_k(x), \tag{10}$$

where x denotes the set of spatial variables, usually represented by a high-dimensional grid of size $\mathcal{N}$, and $n \ll \mathcal{N}$ is a small number of coefficients $a_k(\alpha)$ that are latent reduced coordinates dependent on the control variables α.

The reduced basis, $\rho_k(x)$, is usually informed by previous solutions for different values of α, called snapshots, such as those presented in figures 5 and 6. To construct the basis ρ_k one can apply the proper orthogonal decomposition based on a singular value decomposition (SVD) of a set of snapshots and retaining only the n singular vectors corresponding to the lowest singular values. This has the benefit of better capturing the variability across the training space with a smaller reduced basis, imposing a natural ranking of importance to the basis elements, and introducing additional numerical stability from using an orthogonal basis. A fast (exponential) decay of the singular values will mean that a good approximation can be obtained with a small number n.

The insets of figures 5 and 6 show the reduced basis functions ρ_k for $k = 1, 2, 3$ for our case studies. The corresponding singular values are shown in figure 7. For SDFT and CDFT, the decay of singular values is very fast: It is seen that three reduced basis components with the highest singular values dominate. For MGPE, the decay pattern is significantly slower. This will negatively impact the error budget of the MGPE density emulator.

Once the reduced coordinates have been identified, the next step consists of obtaining equations that describe the response of the latent variables to changes in the control variables $a_k(\alpha)$. In many RBM applications, this is done through the Petrov-Galerkin projection of the original equations into a subspace spanned by test functions. For a system of only one unknown function expanded by n reduced basis, for example, this projection scheme would create n equations to be solved for the n unknown coefficients a_k.

This approach for constructing the reduced equations is well motivated and can provide certain accuracy guarantees, though if one wishes to apply it to the quantum many-body systems described in section 3, it presents three main challenges:

1. Nonlinear and non-affine operators. Here the challenge lies in the inability to precompute the Petrov-Galerkin projections in the offline (training) stage of building the emulator to avoid any scaling of size $\mathcal{N}$. This situation is encountered, e.g. for the MGPE with fractional powers of σ [6]. The RBM literature explores techniques to precompute the required projections when non-affine operators are present, the Empirical Interpolation Method being one of the most widely applied (see [93] for a recent nuclear physics application). Yet when the operators are both nonlinear and non-affine, the applicability of subspace projection-based model reduction schemes become very limited.

2. Scalability. The bulk of the speedup gained when applying the RBM to solve a parametrized differential equation lies in the reduction of the expensive, high-fidelity

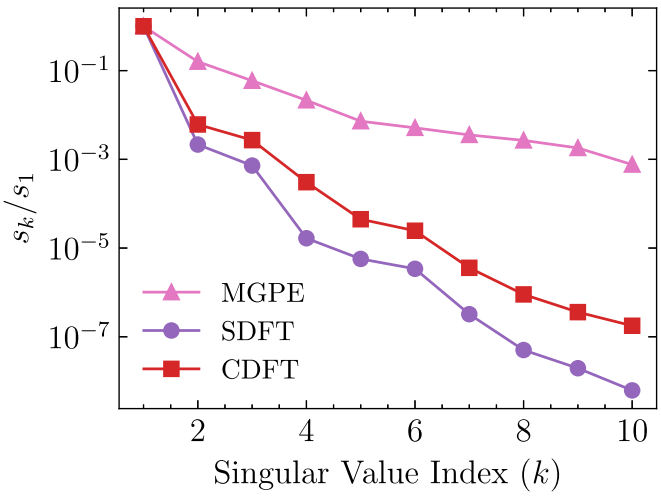

Figure 7. Decay of the singular values of the SVD of the density ρ for MGPE (3), and for the neutron density in ^{48}Ca for SDFT (7) and CDFT (9). The singular values for each density type have been normalized with respect to the first respective value s_1, and their exponential decay for all three cases validates the approximate expansion in equation (10). The decay is much faster in the realistic cases of SDFT and CDFT because the controlling parameters α have been varied over physically calibrated regions [89, 90], while the parameters for the MGPE were varied on a unphysical wide range for the sake of illustration purposes.

calculation to one that scales well with the size n of a reduced basis. If the exponent σ in equation (3) is an integer instead of a fraction, the projection equations can be pre-computed to avoid any dependence on the original spatial domain of size $\mathcal{N}$. However, for a reduced basis expanding the involved wavefunctions $\phi_i(x)$ the number of terms in the equations will scale roughly[6] as n^σ for a fixed σ. This, in turn, can limit the possible speed up that can be obtained by an RBM emulator [90].

3. Reduced model degrees of freedom. Oftentimes, we are interested in understanding how the wave functions that describe the many-body system change with the controlling parameters α. In many cases, however, one aims instead at estimating the global properties of the system, such as the particle density $\rho(x;\alpha)$, and a detailed information on individual states is not needed. In such cases, if the expansion in equation (10) for the system density is accurate enough, we would be interested in focusing on studying only how the density coefficients $a_k(\alpha)$ of $\hat{\rho}(x;\,\alpha)$ change with the controlling parameters α. However, since the density is defined as a function of the solutions to the differential equation (see, for example, equation (4)), it is not possible to use the Petrov-Galerkin projection scheme to separate equations for the reduced coordinates a_k of $\hat{\rho}(x;\,\alpha)$ only, and the full system has to be solved to obtain the density. This, in turn, can limit the speed of an RBM emulator.

The problems that arise when employing the projection framework motivate the development of alternative approaches to obtain reduced equations for a given reduced-order model. The GP framework explored in this Guide offers a very effective avenue. Under this framework, we build expressions that explicitly relate the small set of coefficients $a_k(\alpha)$ of the reduced basis expansion of $\rho(x;\alpha)$ in equation (10) to changes in the controlling parameters α. We achieve this by solving the underlying many-body equations for a training set

[6] For $n = 10$ and $\sigma = 4$ this accounts for exactly 715 terms on each projection equation, for example.

Table 2. Summary of the GP hyper-parameters. $P(C)$ and $P(M)$ indicate the crossover and the mutation probabilities, respectively. CF represents the continued fraction operator $d/(b + c/a)$ and RP represents the rational polynomial function $(b + d + f)/(a + c + e)$.

Parameter	Values
Number of runs	16
Population size	300
Functions (F)	$\{+, -, x, /, -x$, CF, x^2, x^{-1}, RP, $\sqrt{x}$, e^x, $\log(x)$, x^n, $x^{1/3}$, $x^3\}$
Selection	Pareto tournament selection size 30
Genetic operators	{subtree crossover, subtree mutation, depth-preserving subtree mutation}
$P(C)$	0.9
$P(M)$	0.1
Maximum complexity	1000 (Visitation Length [94])
Stopping criteria	5 min

of parameters α and define the set of optimal coefficients $\{\tilde{a_k}(\alpha)\}$ as:

$$\{\tilde{a_k}(\alpha)\} \equiv \underset{\{a_k\}}{\text{argmin}} \,||\, \rho(x;\, \alpha) - \sum_{k}^{n} a_k(\alpha)\rho_k(x)||^2, \tag{11}$$

that is, for a fixed basis size n and each α in the training set, we find the coefficients a_k that best approximates the respective density according to the L2 norm. The L2 norm is defined as $||f(x)||^2 \equiv \int|f(x)|^2 dx$. The GP approach is then used in a traditional ML regression problem: Given the observed relationship between $\tilde{a}_k$ and α in the training set, we build a functional relationship with the objective of generalizing to unobserved parameters α.

Through our developed framework, we are able to create very effective reduced order models by leveraging the best characteristics of both the RBM and GP methods. A good set of reduced coordinates is easily identified by the SVD traditionally performed in RBM applications, effectively compressing high-dimensional densities into a manageable set of coefficients a_k. The GP can then focus on discovering parsimonious expressions to describe the dynamics of this small set of coefficients as a function of α, an easier task than if we attempted to find direct GP expressions for $\rho(x;\alpha)$. In the next section we present our results of applying this framework to the case studies considered.

5. Results

For the computational tests, we selected three basis components $n = 3$ in all case studies. Both the reduced coordinates, i.e. the coefficients of the reduced basis $\rho_k(x)$, and the governing equations are learned from the training data with the parameter ranges listed in table 1. The reduced basis $\rho_k(x)$ is obtained following a SVD on a set of self-consistent solutions, and the optimal coefficients are obtained by means of equation (11). We treat each coefficient as a separate function $a_k(\alpha)$, and as such, we apply the LR and GP frameworks to obtain independent expression for each.

For GP we used the commercially available DataModeler system [67]. The GP parameter set used is shown in table 2, including the hyper-parameters used for GP, along with cross-validation settings. The hyper-parameters reported were from the default set within

Table 3. Summary of the four methods used to calculate the densities of the case studies. The Full Order Model (FOM) represents the high-fidelity MGPE and DFT calculations. Poly-2 is the second-order expansion in model parameters α.

Method	Specification
FOM	Finite element method with an iterative solver with grid size of 300.
LR	Regularized LR with custom feature transformation and selection pipeline, built using scikit-learn API [97, 98].
Poly-2	Second order polynomial fit without regularization built using scikit-learn API [97, 98].
GP	Symbolic expressions generated using the commercially available DataModeler system [67].

DataModeler. Since this default set performed well within this work, it was not necessary to perform a computationally expensive hyper-parameter search. While it was not necessary here, there are common practices within GP for tuning hyper-parameters [95]. Some of these, include strategies that dynamically adapt GP's hyper-parameters during the evolutionary search, as a function of observable evolutionary dynamics [96].

The coefficient of determination R^2 was used as the fitness function [40, 58] as it was found to converge quickly and generalize well, even when only a few data points are available [56].

A common practice in GP is to protect operators from undefined mathematical behavior by defining some ad hoc behavior at those points, such as, for instance, returning the value 1 in the case of a potential division by zero to make it possible for GP to synthesize constants by using x/x [1]. However, these techniques were shown to have several shortcomings in the vicinity of mathematical singularities [53]. In this study, programs that produced invalid values were automatically assigned low fitness values, making them unlikely to be selected. By reducing the selection probability of solutions that produce invalid values, we expect to obtain models whose fitness landscape is less 'sharp' [50].

In addition to the more expressive LR models obtained via the transformation pipeline described in section 2.3, we also include a standard second-order polynomial model composed by squared terms and multiplicative interactions between input features. This model represents the simplest form of nonlinear regression and serves as a baseline for comparison. For example, modeling the coefficients of the neutron density $a_k(\alpha)$ as a function of parameters of J and L, the second order polynomial is:

$$\beta_0 + \beta_1 L + \beta_2 J + \beta_3 L^2 + \beta_4 L J + \beta_5 J^2, \tag{12}$$

where the coefficients β_i are adjustable parameters.

Once the reduced models were trained (see appendix A.4 for the obtained mathematical expressions, and table 3 for the model descriptions), we tested their generalization ability on a test set of previously unexplored combinations of parameters α. The first test consisted of qualitatively comparing the predictions of each method with the optimal coefficients defined in equation (11).

In the case of the modified MGPE equation (3), figure 8 shows the dependency of the three basis coefficients a_k as a function of κ when the other two parameters $\{q, \sigma\}$ are kept constant. As can be seen, GP has appreciably better performance in reproducing these

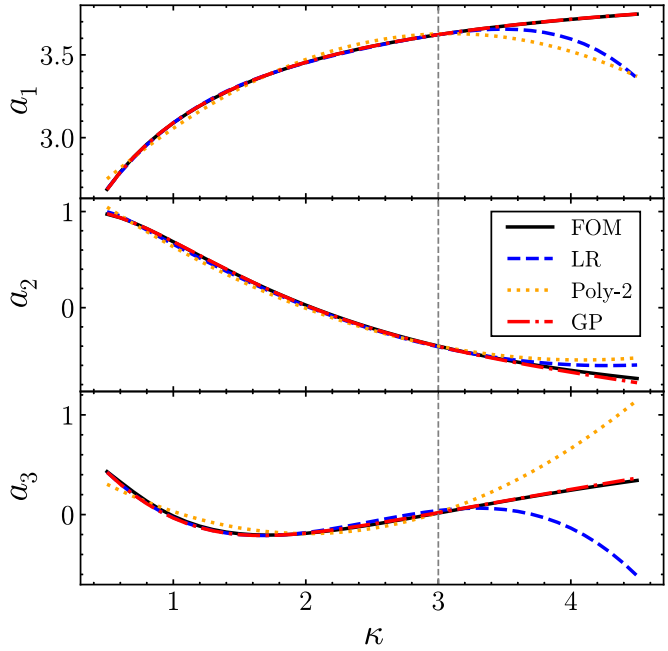

Figure 8. Evolution of the three coefficients a_k in the density reduced order expansion (10) for the MGPE (3) as a function of the dimensionless parameter κ at $q = 1$ and $\sigma = 1.25$. The FOM results (solid line), are compared to the LR (dashed line), Poly-2 (dotted line), and GP (dash-dotted) calculations. The vertical dashed line at $\kappa = 3$ divides the region into interpolation ($\kappa \leqslant 3$) and extrapolation ($\kappa > 3$), with GP being the only method that fully recovers the behavior in the extrapolated region. We obtained similar results when varying q and σ while keeping other parameters constant.

optimal coefficients compared to the other methods. In particular, it excels in extrapolating beyond the dashed vertical line which depicts the maximum parameter value $\kappa = 3$ used for the training of models, see table 1. This suggests that GP has learned expressions that can extrapolate the reduced dynamics with less overfitting.

For the ^{48}Ca neutron density in SDFT and CDFT, figure 9 shows the evolution of the third coefficient $a_3(L, J)$ in the reduced basis expansion (10) as a function of the two controlling parameters L and J. The figure shows the optimal (target) coefficients, obtained from the FOM calculations (11) as gradient-colored solid lines, and compares three of them with the calculations using the different methods listed in table 3, as colored dashed lines. In SDFT, all models demonstrate high generalization performance in predicting the coefficient $a_3(L, J)$. In contrast, for CDFT, the second-degree polynomial model does not extrapolate accurately, particularly at higher values of L ($L > 125$). Although visual comparison may suggest comparable accuracy between custom LR and GP, the symbolic expressions in appendix A.4 reveal that GP-derived models are significantly more compact. This contributes not only to improved inference efficiency but also to greater interpretability, reinforcing GP's advantage in scientific model discovery.

The second test we developed for the methods specified in table 3 focused on a quantitative assessment of their performance. For our purpose, the two main qualities we seek in a reduced-order model are its speed and its accuracy. We define the speed of the emulator—once it has been trained and all relevant quantities have been precomputed—as the time it

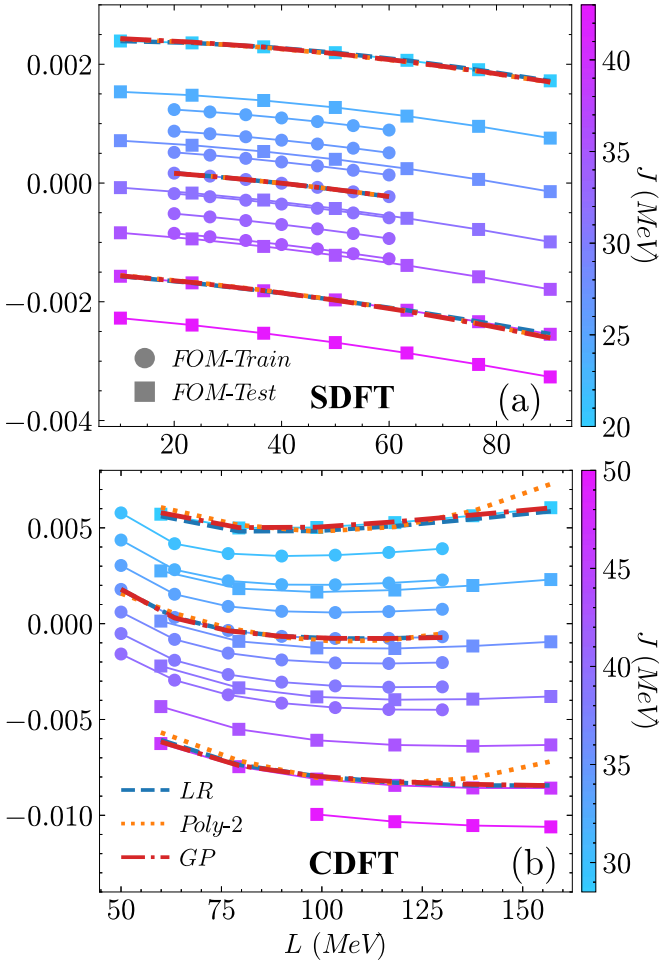

Figure 9. Evolution of the third coefficient $a_3(L, J)$ in the ^{48}Ca neutron density reduced basis expansion (10) as a function of the two controlling parameters L and J for the case of SDFT (a) and CDFT (b). The color grading curves represent the optimal coefficient a_3 (equation (11)) obtained from the FOM calculations, with circles marking training parameters and squares marking the testing parameters. The dashed and dotted lines show the LR, Poly-2, and GP results for three values of J, one in the training region and two in the testing (extrapolation) region.

takes to map controlling parameters α to the quantities of interest, in this case the amplitudes of the reduced basis for the particle densities for the two quantum many-body systems. We quantify the accuracy for a given parameter α_i through the relative root mean squared error (or L^2 norm) between the density obtained from the high-fidelity solver and the one obtained from the reduced order model:

$$\text{Relative Error}(\alpha_i) = \frac{|| \rho(x;\, \alpha_i) - \hat{\rho}(x;\, \alpha_i)||}{|| \rho(x;\, \alpha_i)||}, \tag{13}$$

where $\hat{\rho}$ is the approximation in equation (10) with coefficients $a_k(\alpha_i)$ predicted by the respective method.

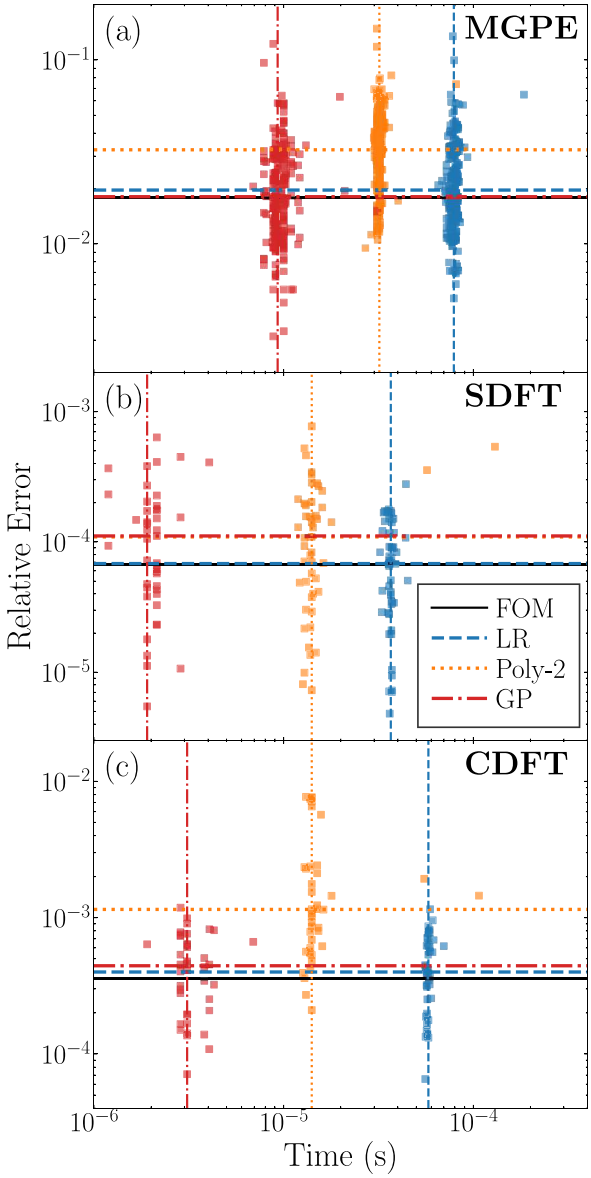

Figure 10. Computational Accuracy vs Time plot for (a) MGPE, (b) SDFT, and (c) CDFT density calculations. The squares show individual calculations for the testing parameters using the reduced order models with LR, Poly-2, and GP. A total of 212 (a), 49 (b), and 40 (c) testing parameters are shown. The horizontal lines represent the median relative error (equation (13)) for each method, with the FOM line representing the 'best case scenario'. The vertical lines represent the average time of the calculations for each of the methods. For FOM, the time is orders of magnitude slower and hence it is not shown.

Figure 10 shows the trade-off between accuracy and speed, quantified as the relative error (vertical axis) and the inference times (horizontal axis) for the test parameters across the studied systems. Individual predictions for the testing parameters are represented by squares. The plots clearly show that the GP-based models achieve the fastest inference times while

maintaining a low error. In the MGPE case, GP outperforms all other models in both accuracy and speed. Both GP and custom LR achieve relative errors that are at the theoretical limit of $\sim$1.5% (horizontal black line), which means that any further improvement in precision will come from increasing the number of bases beyond $n = 3$, and not from obtaining more accurate coefficients $a_k(\alpha)$.

For the SDFT framework, LR yields a smaller error—on top of the theoretical limit of $\sim$0.007%—but at the cost of significantly longer inference times; GP, however, performs inference in more than one order of magnitude faster while only slightly compromising the prediction quality with an error of $\sim$0.01%. In the CDFT case, GP and custom LR report similarly small errors—also comparable with the theoretical limit of $\sim$0.03%—but GP does so with more than an order of magnitude faster inference than LR. In all cases, all methods provide appreciable speedups between 4 and 6 orders of magnitude when compared with the standard FOM solvers.

It is worth noting that the faster inference times of GP-based models are a direct consequence of their compact functional form (see appendix A.4) and reduced data preprocessing requirements. In particular, both custom LR and Poly-2 models require 0-1 scaling of the input data at each inference event, which contributes to higher run-times. On the other hand, GP models operate directly on raw input values. This justifies the fact that even a relatively simple second-order polynomial lags behind GP in terms of run-times.

6. Conclusions and outlook

In this Guide we developed and presented a general framework based on GP to aid in the construction of reduced order models used in expensive nuclear physics computations. The GP approach employs evolutionary algorithms to propose, test and evolve a set of candidate governing equations that control how response variables change with respect to controlling variables. With the specific application to three many-body case studies, we used the GP approach to create the mathematical expressions that describe the change in the reduced coordinates of high-dimensional solutions. By using the GP approach to build the equations, we have been able to successfully craft effective reduced-order models for these systems, allowing speedup gains of 5–6 orders of magnitude in comparison to their respective full-order solvers at a negligible loss in accuracy.

It should be noted that although surrogate models based on reduced-order modeling, such as the RBM used here, have provided spectacular success in terms of many orders of magnitude computational speedup gains at negligible loss in accuracy, certain classes of problems are not amenable to these approaches. Furthermore, in general, the development of the surrogate model requires an appreciable upfront effort to construct, often necessitating access to and manipulation of the underlying complex equations of the full system.

Future work on using information on the underlying operators to inform the evolution algorithm and the construction of implicit equations [99–101] relating the control variables and the response variables could further improve the performance of the built surrogate models and also help in the interpretability of the obtained equations obtained through GP. The development of a pipeline to straightforwardly build effective and more interpretable reduced order models for nuclear science will enable a broad adoption of uncertainty quantification approaches [102], and allow the theory to provide data across thousands of nuclei necessary to face modern challenges in this new era of discovery [103].

Acknowledgments

We are grateful to Edgard Bonilla for useful discussions. This work was supported by the U.S. Department of Energy, Office of Science and Office of Nuclear Physics under Awards Nos. DOE-DE-SC0013365, SC0024586, and DE-SC0023175 (Office of Advanced Scientific Computing Research and Office of Nuclear Physics, Scientific Discovery through Advanced Computing). W.B. gratefully acknowledges funding provided through the John R. Koza endowment to Michigan State University.

Data availability statement

The data that support the findings of this study are openly available at the following URL/DOI: https://gitlab.com/ibakurov/beacon-frib-gp.

Appendix

In this work, we use GP and compare its performance to construct nonlinear and interpretable emulators against an augmented version of LR. LR is one of the most popular and frequently used SML tools for regression. It is known for its simplicity, computational efficiency and interpretability. It assumes a linear relationship between k input variables/features (x) and the output target (y). Formally: $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n$, where y is the target, β_0 is the intercept, and β_k is a learnable weight associated with input feature x_k. The learnable weights and the intercept are usually estimated from the data using Ordinary Least Squares (OLS) estimation procedure to minimize the sum of squared residuals (i.e. difference between the observed target values in the training dataset, and the values predicted by the linear model). Formally: $\min_{\beta} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$, where $\hat{y}_i$ is the model's prediction for data instance i.

The assumption of linearity reduces the ability of LR to produce models in complex and multi-dimensional feature spaces. This structural rigidity makes it fundamentally different from symbolic regression methods, like GP, which simultaneously search for both the functional form and parameters of a mathematical expression to best fit the data. To overcome this limitation and enhance the expressiveness of LR, we use a custom feature transformation pipeline. This pipeline integrates tailored feature transformations, including several mathematical functions (such as logarithmic and exponential functions, roots, powers, etc.), and feature interactions (products and quotients). These transformed features are concatenated with the original inputs and the bias term. However, when the feature space is inflated and comprises potentially highly nonlinear relationships, the risk of overfitting becomes particularly high. For this reason, we include several levels of feature selection and weight-decay regularization in our pipeline. In this way, our goal is to make LR more competitive (and comparable) to GP in the capture of complex patterns. Additionally, we include a standard second-order polynomial LR model with multiplicative interaction terms as a baseline for comparison.

We use randomized cross-validation to find the most suitable hyper-parameters for our pipeline. This includes the choice of feature transformations, feature interactions, feature selection and regression model-specific parameters such as regularization strength (these aspects are described in detail in the following sections). In total, 250 parameter settings are sampled and the best models are selected after aggregating validation errors from 3 fold

cross validation. To perform the experiments, we use predictive models, cross-validation and pipeline construction tools provided by scikit-learn [97, 98]. Full details about our LR models, pipeline composition and training are provided in our source code available at [104].

A.1. Feature transformation

Different combination of feature transformations for LR are seen as different hyper-parameter values in our pipeline. These transformations range from polynomials, such as $\{x^2, x^3\}$, to more complex terms $\{\sqrt{x}, \sqrt[3]{x}, e^x, \ln(x), x^{-1}\}$. We include two types of feature interactions, one through multiplication (e.g. $x_1 \times x_2$) and another through division (e.g. x_1/x_2). In such a way, LR can be equipped with simpler or more complex feature spaces, balancing model interpretability and complexity. Exploring the space of feature transformations in this manner allows for a fairer comparison with GP. Recall that GP operates on a highly expressive feature space, and can employ arbitrary powers (e.g. square and cube), roots (e.g. square and cube roots), logarithms, and exponentials. In this sense, our custom feature engineering process for LR allows it to approximate the expressiveness of GP, ensuring that the comparison between methods is equitable.

A.2. Feature selection

We are aware that inflating the input feature space by applying a variety of transformations introduces significant challenges in the SML context. Deliberately increasing the proportion of features relative to the number of data instances (which remains fixed) is expected to penalize the generalization of estimated regression coefficients (i.e. leads to overfitting). To mitigate this challenge, we incorporate up to two consecutive steps of feature selection. In the first step, we allow the pipeline to perform feature selection that uses a tree-based ensemble strategy, called Extremely Randomized Trees (ExtraTrees) [75], known for its robustness and computational efficiency. The ExtraTrees method fits multiple randomized decision trees to the training data in parallel, and then aggregates their results. The inference of randomized trees relies on randomly choosing a set of splitting points and then selects the one that minimizes a given loss function, instead of exhaustively enumerating all possible splits. Among them is the feature importance ranking provided by each tree. Note that in decision tree learning, features that are more frequently used at the top of trees (where the first splits occur) are considered more important. The aggregation across multiple trees allows to reduce variance and dependency from a single model's output [105]. The second step of feature selection is optional and implements a refinement of the previous step. It involves a LR model to make the selection based on the predictive power of the remaining features and their linear relationship with the target variable. This results in potentially simpler and possibly more interpretable models.

A.3. Regularization

While feature selection helps to reduce the dimensionality of the input space, it does not fully eliminate the risk of overfitting. To further improve the generalization ability of our models, we also incorporate regularization techniques. Regularization adds an explicit penalty term to the regression loss function, discouraging overly complex models and helping to prevent the model from fitting the noise. Several levels of regularization are included, L1 regularization (Lasso) [72], L2 regularization (Ridge) [106], and a combination of the two (ElasticNet) [74]. L1 regularization promotes model sparsity and is useful in high-dimensional tasks (i.e. when there are many input features), especially if many are expected to be irrelevant. L2

regularization is frequently used when the input features are expected to be informative but their impact has to be reduced to avoid overfitting, and/or if they are highly correlated (multicollinear). Formally, a L1 term is given by $\lambda_1 \sum_{k=0}^{n} |\beta_k|$, while a L2 term is given by $\lambda_2 \sum_{k=0}^{n} \beta_k^2$. ElasticNet uses combines and balances both L1 and L2.

Note that regularized LR has long been used in scientific applications. Sparse-identification of nonlinear dynamics (SINDy) is a popular method for discovering governing equations in dynamic systems [107]. Essentially, it utilizes LR with the L1 regularizing term to promote sparsity, yielding smaller models, which can offer better interpretability, considered an appealing property. SINDy has been successfully applied to find governing equations in many different applications [108–112]. Nevertheless, SINDy-like approaches present an obvious disadvantage: nonlinear relationships are combinatorial in input variables and must be manually pre-defined, requiring some degree of expertize. This can be particularly limiting when applied to problems where valuable nonlinear relationships are unexpected and is one of the advantages we expect from GP.

A.4. Expressions

In this appendix we present the discovered mathematical expressions for the case studies described in section 3 when using the three surrogate methods (LR, Poly-2, and GP) summarized in table 3. The coefficients a_k are dimensionless in all cases since the respective dimensions of the densities in equation (10) is carried out by the respective principal components ρ_k. The parameters of the MGPE are themselves dimensionless, so we write direct equations between them and the response variables a_k. For the case of SDFT and CDFT we write the equations in terms of the dimensionless parameters $\tilde{J} = J/\mathrm{MeV}$ and $\tilde{L} = L/\mathrm{MeV}$.

A.4.1. MGPE

$$
\begin{aligned}
a_2(\kappa, q, \sigma)_{\mathrm{LR}} = {} & +0.002\sqrt{\kappa}\sqrt{q} + 0.032\sqrt{\kappa}q^2 + 0.469\sqrt{\kappa}q - 0.107\sqrt{\kappa} - 0.014\kappa^2q^2 - 0.083\kappa^2q \\
& + \frac{0.324\kappa^2}{(\sqrt{q}+1)^{0.5}} + \frac{0.526\kappa^2}{(q+1)^{0.5}} + \frac{0.505\kappa^2}{(q^2+1)^{0.5}} - 0.210\kappa\sqrt{q} - 0.051\kappa\sqrt{\sigma} \\
& - 0.003\kappa\sigma - \frac{0.383\kappa}{(\sqrt{q}+1)^{0.5}} - \frac{0.723\kappa}{(q+1)^{0.5}} - \frac{0.356\kappa}{(q^2+1)^{0.5}} - \frac{0.037\kappa}{(\sqrt{\sigma}+1)^{0.5}} \\
& - \frac{0.065\kappa}{(\sigma+1)^{0.5}} - \frac{4.500\cdot 10^{-3}\kappa}{(\sigma^2+1)^{0.5}} - 1.300\kappa - 0.010\sqrt{q}\sigma^2 - 0.011\sqrt{q}\sigma \\
& + \frac{0.040\sqrt{q}}{(\kappa^2+1)^{0.5}} + \frac{0.079\sqrt{q}}{(\sigma^2+1)^{0.5}} - \frac{0.065q^2}{(\sqrt{\kappa}+1)^{0.5}} - 0.176q\sigma^2 - 0.005q\sigma \\
& + \frac{5.800\cdot 10^{-3}q}{(\kappa^2+1)^{0.5}} + \frac{0.160q}{(\sigma^2+1)^{0.5}} + \frac{0.033\sigma^2}{(\kappa+1)^{0.5}} + \frac{0.078\sigma^2}{(q+1)^{0.5}} \\
& + \frac{0.022\sigma^2}{(q^2+1)^{0.5}} + \frac{0.063\sigma}{(\kappa+1)^{0.5}} + 0.802
\end{aligned}
$$

$$
\begin{aligned}
a_3(\kappa, q, \sigma)_{\mathrm{LR}} = {} & -2.819\kappa^3 + 0.358\kappa^2q + 0.196\kappa^2\sigma + 5.669\kappa^2 + 0.269\kappa q^2 + 0.835\kappa q\sigma \\
& - 1.839\kappa q + 1.482\cdot 10^{-3}\kappa\sigma^2 - 0.654\kappa\sigma - 2.296\kappa - 0.027q^3 \\
& - 0.090q^2\sigma + 0.036q^2 + 0.245q\sigma^2 - 0.662q\sigma + 0.677q \\
& + 0.033\sigma^3 - 0.175\sigma^2 + 0.421\sigma - 0.059
\end{aligned}
$$

$$a_1(\kappa, q, \sigma)_{\text{Poly}-2} = -1.023\,\kappa^2 + 0.154\,\kappa q + 0.006\,\kappa\sigma + 1.778\,\kappa + 0.016\,q^2 + 0.210\,q\sigma - 0.390\,q - 3.012\cdot 10^{-3}\,\sigma^2 - 0.104\,\sigma + 3.020$$

$$a_2(\kappa, q, \sigma)_{\text{Poly}-2} = 0.971\,\kappa^2 + 0.204\,\kappa q - 0.033\,\kappa\sigma - 2.629\,\kappa - 0.072\,q^2 - 0.316\,q\sigma + 0.484\,q + 0.015\,\sigma^2 + 0.171\,\sigma + 0.740$$

$$a_3(\kappa, q, \sigma)_{\text{Poly}-2} = 1.695\,\kappa^2 - 0.806\,\kappa q - 0.041\,\kappa\sigma - 1.216\,\kappa + 0.086\,q^2 - 0.101\,q\sigma + 0.316\,q - 1.167\cdot 10^{-2}\,\sigma^2 + 0.070\,\sigma + 0.021$$

$$a_1(\kappa, q, \sigma)_{\text{GP}} = 4.162 - \frac{0.012}{\kappa} - 0.015k - \frac{1.676}{0.738+\kappa} + 0.007q + \frac{0.028q}{0.264+\kappa} - \frac{0.230q}{\kappa + 0.142q + \frac{1}{2\sigma} + \sigma} - \frac{0.372q}{3+\kappa+\sigma^2} + \frac{0.093q}{\sigma + \frac{12.828+q}{10.738+\sigma}}$$

$$a_2(\kappa, q, \sigma)_{\text{GP}} = -1.051 - 1.973\cdot 10^{-4}\,\kappa^4 + \frac{6.549}{3+k^2} - 0.016kq + 0.002kq^2 + 0.031q\sigma^{1/3} - 0.007q\sigma - \frac{0.043q}{\kappa+\kappa^2\sigma} - \frac{0.163k^3q^2\sigma}{(4\kappa^2+\sigma^2)^2} + \frac{0.559kq}{\frac{1}{\kappa}+\kappa^2+\frac{q}{1+\kappa}+\frac{1}{\sigma}+\sigma^2}$$

$$a_3(\kappa, q, \sigma)_{\text{GP}} = 1.276 - 0.079k + 0.020k^2 - \frac{4.077}{\frac{2.036}{\kappa}+\kappa} - 0.199q - 0.019kq + 0.133k^{\frac{q}{\left(\frac{1}{\kappa}+\kappa\right)(\kappa+\sigma)}}q + 1.725\cdot 10^{-4}\kappa^2q\sigma^3 + \frac{0.573q}{\frac{1}{\kappa}+\kappa^3+\sigma}$$

A.4.2. SDFT

$$a_1(\tilde{L}, \tilde{J})_{\text{LR}} = -1.123\cdot 10^{-4}\tilde{L}^3\tilde{J}^3 + 2.206\cdot 10^{-4}\tilde{L}^3\tilde{J}^2 - 8.261\cdot 10^{-5}\tilde{L}^3\tilde{J} - 8.083\cdot 10^{-5}\tilde{L}^3 + 2.071\cdot 10^{-4}\tilde{L}^2\tilde{J}^3 - 3.914\cdot 10^{-4}\tilde{L}^2\tilde{J}^2 + 1.013\cdot 10^{-4}\tilde{L}^2\tilde{J} + 6.635\cdot 10^{-4}\tilde{L}^2 - 4.132\cdot 10^{-5}\tilde{L}\tilde{J}^3 - 1.743\cdot 10^{-4}\tilde{L}\tilde{J}^2 + 0.00199\tilde{L}\tilde{J} - 0.0133\tilde{L} - 6.594\cdot 10^{-5}\tilde{J}^3 + 2.775\cdot 10^{-4}\tilde{J}^2 - 0.005\tilde{J} + 1.007$$

$$a_2(\tilde{L}, \tilde{J})_{\text{LR}} = -5.273\cdot 10^{-5}\tilde{L}^3\tilde{J}^3 + 1.044\cdot 10^{-4}\tilde{L}^3\tilde{J}^2 - 3.772\cdot 10^{-5}\tilde{L}^3\tilde{J} - 4.313\cdot 10^{-5}\tilde{L}^3 + 8.843\cdot 10^{-5}\tilde{L}^2\tilde{J}^3 - 1.693\cdot 10^{-4}\tilde{L}^2\tilde{J}^2 + 2.005\cdot 10^{-5}\tilde{L}^2\tilde{J} + 4.320\cdot 10^{-4}\tilde{L}^2 - 4.879\cdot 10^{-6}\tilde{L}\tilde{J}^3 - 1.558\cdot 10^{-4}\tilde{L}\tilde{J}^2 + 1.127\cdot 10^{-3}\tilde{L}\tilde{J} - 7.211\cdot 10^{-3}\tilde{L} - 2.011\cdot 10^{-5}\tilde{J}^3 + 1.634\cdot 10^{-4}\tilde{J}^2 + 5.522\cdot 10^{-4}\tilde{J} + 2.880\cdot 10^{-3}$$

$$a_3(\tilde{L}, \tilde{J})_{\text{LR}} = -1.171\cdot 10^{-6}\tilde{L}^3\tilde{J}^3 + 1.864\cdot 10^{-6}\tilde{L}^3\tilde{J}^2 - 2.853\cdot 10^{-6}\tilde{L}^3\tilde{J} + 7.135\cdot 10^{-6}\tilde{L}^3 + 1.081\cdot 10^{-5}\tilde{L}^2\tilde{J}^3 - 1.948\cdot 10^{-5}\tilde{L}^2\tilde{J}^2 + 3.628\cdot 10^{-5}\tilde{L}^2\tilde{J} - 1.362\cdot 10^{-4}\tilde{L}^2 - 1.723\cdot 10^{-5}\tilde{L}\tilde{J}^3 + 6.430\cdot 10^{-5}\tilde{L}\tilde{J}^2 - 1.558\cdot 10^{-4}\tilde{L}\tilde{J} - 2.160\cdot 10^{-4}\tilde{L} - 7.182\cdot 10^{-6}\tilde{J}^3 + 1.261\cdot 10^{-4}\tilde{J}^2 - 2.203\cdot 10^{-3}\tilde{J} + 1.237\cdot 10^{-3}$$

$$a_1(\tilde{L}, \tilde{J})_{\text{Poly-2}} = 5.200\cdot 10^{-4}\tilde{L}^2 + 1.710\cdot 10^{-3}\tilde{L}\tilde{J} - 0.013\tilde{L} + 4.656\cdot 10^{-5}\tilde{J}^2 - 0.005\tilde{J} + 1.007$$

$$a_1(\tilde{L}, \tilde{J})_{\mathrm{GP}} = 0.467 + \frac{697.688}{1243.500 + \tilde{L}} - 5.997 \cdot 10^{-4}\tilde{J} + 4.271 \cdot 10^{-6}\tilde{L}\tilde{J}$$

$$a_2(\tilde{L}, \tilde{J})_{\mathrm{GP}} = -0.261 + \frac{291.699}{1092.852 + \tilde{L}} + 2.420 \cdot 10^{-5}\tilde{J} + 2.298 \cdot 10^{-6}\tilde{L}\tilde{J}$$

$$a_3(\tilde{L}, \tilde{J})_{\mathrm{GP}} = -0.032 + 2.043 \cdot 10^{-6}\tilde{L} - 7.054 \cdot 10^{-8}\tilde{L}^2 - 2.066 \cdot 10^{-7}\tilde{L}\tilde{J} + \frac{4.975}{126 + \tilde{J}}$$

A.4.3. CDFT

$$\begin{aligned} a_1(\tilde{L}, \tilde{J})_{\mathrm{LR}} = & -0.004\sqrt{\tilde{L}}\sqrt{\tilde{J}} - 0.004\sqrt{\tilde{L}}\tilde{J}^2 + 0.035\sqrt{\tilde{L}}\tilde{J} - 0.079\sqrt{\tilde{L}} + 1.939 \cdot 10^{-4}\tilde{L}^2\sqrt{\tilde{J}} \\ & - 0.003\tilde{L}^2\tilde{J}^2 + 0.009\tilde{L}^2\tilde{J} - 0.005\tilde{L}^2 + 0.002\tilde{L}\sqrt{\tilde{J}} + 0.008\tilde{L}\tilde{J}^2 - 0.037\tilde{L}\tilde{J} + 0.036\tilde{L} \\ & + 3.272 \cdot 10^{-4}\sqrt{\tilde{J}}\ln(\tilde{L}) + 0.001\sqrt{\tilde{J}} - 7.290 \cdot 10^{-5}\tilde{J}^2\ln(\tilde{L}) - 0.001\tilde{J}^2 \\ & - 0.001\tilde{J}\ln(\tilde{L}) - 0.008\tilde{J} + 0.003\ln(\tilde{L}) + 3.497 \cdot 10^{-5}\ln(\tilde{J}) + 1.042 \end{aligned}$$

$$\begin{aligned} a_2(\tilde{L}, \tilde{J})_{\mathrm{LR}} = & -7.494 \cdot 10^{-4}\sqrt{\tilde{L}}\sqrt{\tilde{J}} - 5.630 \cdot 10^{-3}\sqrt{\tilde{L}}\tilde{J}^2 + 0.025\sqrt{\tilde{L}}\tilde{J} - 0.042\sqrt{\tilde{L}} \\ & + 3.732 \cdot 10^{-4}\tilde{L}^2\sqrt{\tilde{J}} - 1.653 \cdot 10^{-3}\tilde{L}^2\tilde{J}^2 + 4.115 \cdot 10^{-3}\tilde{L}^2\tilde{J} - 7.376 \cdot 10^{-4}\tilde{L}^2 \\ & - 4.778 \cdot 10^{-4}\tilde{L}\sqrt{\tilde{J}} + 6.697 \cdot 10^{-3}\tilde{L}\tilde{J}^2 - 0.022\tilde{L}\tilde{J} + 0.012\tilde{L} \\ & + 9.966 \cdot 10^{-5}\sqrt{\tilde{J}}\ln\tilde{L} + 3.555 \cdot 10^{-4}\sqrt{\tilde{J}} + 1.520 \cdot 10^{-4}\tilde{J}^2\ln\tilde{L} \\ & + 5.196 \cdot 10^{-4}\tilde{J}^2 - 9.378 \cdot 10^{-4}\tilde{J}\ln\tilde{L} - 1.547 \cdot 10^{-3}\tilde{J} \\ & + 0.002\ln\tilde{L} + 1.643 \cdot 10^{-5}\ln\tilde{J} + 0.020 \end{aligned}$$

$$\begin{aligned} a_3(\tilde{L}, \tilde{J})_{\mathrm{LR}} = & +3.214 \cdot 10^{-4}\sqrt{\tilde{L}}\sqrt{\tilde{J}} - 8.363 \cdot 10^{-4}\sqrt{\tilde{L}}\tilde{J}^2 + 2.905 \cdot 10^{-3}\sqrt{\tilde{L}}\tilde{J} - 7.033 \cdot 10^{-3}\sqrt{\tilde{L}} \\ & + 2.348 \cdot 10^{-4}\tilde{L}^2\sqrt{\tilde{J}} - 7.590 \cdot 10^{-4}\tilde{L}^2\tilde{J}^2 + 1.648 \cdot 10^{-3}\tilde{L}^2\tilde{J} - 1.763 \cdot 10^{-4}\tilde{L}^2 \\ & - 5.384 \cdot 10^{-4}\tilde{L}\sqrt{\tilde{J}} + 1.919 \cdot 10^{-3}\tilde{L}\tilde{J}^2 - 5.936 \cdot 10^{-3}\tilde{L}\tilde{J} + 5.277 \cdot 10^{-3}\tilde{L} \\ & - 3.766 \cdot 10^{-4}\sqrt{\tilde{J}} + 9.651 \cdot 10^{-4}\tilde{J}^2 - 7.959 \cdot 10^{-3}\tilde{J} + \frac{6.054 \cdot 10^{-6}}{\tilde{J}} \\ & + \frac{6.498 \cdot 10^{-5}}{\tilde{L}} + 5.780 \cdot 10^{-3} \end{aligned}$$

$$a_1(\tilde{L}, \tilde{J})_{\mathrm{Poly-2}} = 0.023\tilde{L}^2 - 6.087 \cdot 10^{-4}\tilde{L}\tilde{J} - 0.048\tilde{L} - 3.062 \cdot 10^{-4}\tilde{J}^2 - 1.191 \cdot 10^{-4}\tilde{J} + 1.018$$

$$a_2(\tilde{L}, \tilde{J})_{\mathrm{Poly-2}} = 0.012\tilde{L}^2 + 3.060 \cdot 10^{-4}\tilde{L}\tilde{J} - 0.030\tilde{L} - 3.907 \cdot 10^{-4}\tilde{J}^2 + 0.006\tilde{J} + 0.008$$

$$a_3(\tilde{L}, \tilde{J})_{\mathrm{Poly-2}} = 4.955 \cdot 10^{-3}\tilde{L}^2 - 1.255 \cdot 10^{-3}\tilde{L}\tilde{J} - 0.006\tilde{L} + 1.388 \cdot 10^{-3}\tilde{J}^2 - 0.008\tilde{J} + 0.005$$

$$a_1(\tilde{L}, \tilde{J})_{\mathrm{GP}} = 0.948 + \frac{4373.484}{\tilde{L}} - \frac{10666.073}{\tilde{L} + \frac{2}{\tilde{J}}} + \frac{0.693}{\tilde{J}} + \frac{78.987}{\tilde{L} + \frac{\tilde{J}}{\tilde{L}}} + \frac{6218.963}{\tilde{L} + \frac{\ln\tilde{L}}{\tilde{J}}}$$

$$\begin{aligned} a_2(\tilde{L}, \tilde{J})_{\mathrm{GP}} = & 126757.788 - \frac{159623.632}{\tilde{L}} + \frac{21868.106}{\tilde{L} - \frac{6.086}{\tilde{J}}} + \frac{137755.287}{\tilde{L} + \frac{1}{\tilde{J}}} + \frac{797441.187}{-6.086 + \frac{1}{\tilde{L}\tilde{J}}} \\ & - \frac{840.950}{-0.197 + \frac{1}{\tilde{L}\tilde{J}}} + \frac{816.101}{\tilde{J}} - 7.738 \cdot 10^{-7}\tilde{L}\tilde{J} - \frac{26.366}{2.836 + \tilde{J}} - \frac{793.962}{-\frac{6.086}{\tilde{L}} + \tilde{J}} \end{aligned}$$

$$a_3(\tilde{L}, \tilde{J})_{\mathrm{GP}} = -0.023 + \frac{49837.925}{\tilde{L}} + 1.312 \cdot 10^{-5}\tilde{L} + \frac{427.181}{\frac{3}{\tilde{L}} + \tilde{L}} - \frac{50263.596}{\tilde{L} + \frac{1}{\tilde{L}\tilde{J}}} - \frac{43.807}{\tilde{J}} - 3.182 \cdot 10^{-4}\tilde{J} + 3.248 \cdot 10^{-7}\tilde{L}\tilde{J} + \frac{44.804}{\tilde{J} + \frac{\tilde{J}}{\tilde{L}}}$$

References

[1] Koza J 1992 *Genetic programming: on the programming of computers by means of natural selection* (The MIT Press)
[2] Banzhaf W, Francone F D, Keller R E and Nordin P 1998 *Genetic Programming: An Introduction: On the Automatic Evolution of Computer Programs and Its Applications* (San Francisco, California: Morgan Kaufmann Publishers, Inc.)
[3] Langdon W B and Poli R 2002 *Foundations of Genetic Programming* (Springer)
[4] Quarteroni A, Manzoni A and Negri F 2015 *Reduced Basis Methods for Partial Differential Equations: An Introduction* vol 92 (Springer)
[5] Hesthaven J S *et al* 2016 *Certified Reduced Basis Methods for Parametrized Partial Differential Equations* (Springer)
[6] Bonilla E, Giuliani P, Godbey K and Lee D 2022 *Phys. Rev. C* **106** 054322
[7] Vanneschi L and Silva S 2023 *Lectures on Intelligent Systems* (Springer)
[8] Bakurov I, Castelli M, Fontanella F, Scotto di Freca A and Vanneschi L 2022 *Swarm Evol. Comput.* **69** 101028
[9] La Cava W, Silva S, Danai K, Spector L, Vanneschi L and Moore J H 2019 *Swarm Evol. Comput.* **44** 260
[10] Keren L S, Liberzon A and Lazebnik T 2023 *Sci. Rep.* **13** 1249
[11] Angelis D, Sofos F and Karakasidis T E 2023 *Arch. Comput. Methods Eng.* **30** 3845
[12] La Cava W, Orzechowski P, Burlacu B, de Franca F, Virgolin M, Jin Y, Kommenda M and Moore J 2021 *Proc. NeurIPS Datasets Benchmarks Track* vol 1 ed J Vanschoren and S Yeung
[13] Zhang H, Chen Q, Xue B, Banzhaf W and Zhang M 2024 *Proc. Genetic and Evolutionary Computation Conf., GECCO'24 (New York, NY, USA, 2024)* (Association for Computing Machinery) pp 998–1006
[14] Neshatian K, Zhang M and Andreae P 2012 *IEEE Trans. Evol. Comput.* **16** 645
[15] Tran B, Xue B and Zhang M 2019 *Pattern Recognit.* **93** 404
[16] Lensen A, Xue B and Zhang M 2022 *IEEE Trans. Evol. Comput.* **26** 661
[17] Bakurov I, Castelli M, Gau O, Fontanella F and Vanneschi L 2021 *Swarm Evol. Comput.* **65** 100913
[18] Sipper M and Moore J H 2021 *Genet. Program. Evolvable Mach.* **22** 357
[19] Haut N, Punch B and Banzhaf W 2023 *Proc. Companion Conf. Genetic and Evolutionary Computation, GECCO 2023 Companion (New York, NY, USA, 2023)* pp 587–590
[20] Correia J, Lopes D, Vieira L, Rodriguez-Fernandez N, Carballal A, Romero J and Machado P 2022 *Genet. Program. Evolvable Mach.* **23** 557579
[21] Correia J, Vieira L, Rodriguez-Fernandez N, Romero J and Machado P 2021 *in Proc. Int. Conf. Artificial Intelligence Music, Sound, Art and Design (Cham, 2021)* ed J Romero, T Martins and N Rodríguez-Fernández pp 82–97
[22] Ul Ain Q, Xue B, Al-Sahaf H and Zhang M 2017 *IEEE Congress on Evolutionary Computation (CEC)* (IEEE Press) pp 2420–2427
[23] De La Torre C, Nadizar G, Lavinas Y, Schwob R, Franchet C, Luga H, Wilson D and Cussat-Blanc S 2025 *Genetic Programming* ed B Xue, L Manzoni and I Bakurov (Springer Nature Switzerland) pp 173–189
[24] Cortacero K *et al* 2023 *Nat. Commun.* **14** 7112
[25] Haut N, Banzhaf W, Punch B and Colbry D 2024 Accelerating image analysis research with active learning techniques in genetic programming *Genetic Programming Theory and Practice XX* ed S Winkler *et al* (Springer) pp 45–64

[26] Olson R S and Moore J H 2019 Tpot: A tree-based pipeline optimization tool for automating machine learning *Automated Machine Learning: Methods, Systems, Challenges* ed F Hutter, L Kotthoff and J Vanschoren (Springer) pp 151–160
[27] Real E, Liang C, So D and Le Q 2020 *Proc. 37th Int. Conf. Machine Learning, PMLR* vol 119 ed III and A Singh pp 8007–19
[28] Pinna G, Ravalico D, Rovito L, Manzoni L and Lorenzo A D 2024 *Genetic Programming* ed M Giacobini, B Xue and L Manzoni (Springer) pp 108–24
[29] Winkler C and Hofmann H M 1997 *Phys. Rev. C* **55** 684
[30] Wilson J and Radford D 1997 *Nucl. Instrum. Methods Phys. Res. A* **385** 108
[31] Agapie A, Fagarasan F and Stanciulescu B 1997 *Nucl. Instrum. Methods Phys. Res. A* **389** 288
[32] Ireland D G 2000 *J. Phys. G* **26** 157
[33] Ireland D, Janssen S and Ryckebusch J 2004 *Nucl. Phys. A* **740** 147
[34] Fernández-Ramírez C, de Guerra E M, Udías A and Udías J M 2008 *Phys. Rev. C* **77** 065212
[35] Acampora G, Chiatto A, Coraggio L, De Gregorio G, Schiattarella R and Vitiello A 2023 *2023 IEEE Congress on Evolutionary Computation (CEC)* pp 1–8
[36] Munoz J M, Udrescu S M and Garcia Ruiz R F 2025 *Commun. Phys.* **8** 101
[37] Cheng J, Wang B, Zhang W, Duan X and Yu T 2024 *Comput. Phys. Commun.* **304** 109317
[38] Brameier M and Banzhaf W 2007 *Linear Genetic Programming* (Springer)
[39] Miller J 2011 *Cartesian Genetic Programming* (Springer)
[40] Kotanchek M, Haut N and Kotanchek K 2025 Representation and reachability: Assumption impact in data modeling *Genetic Programming Theory and Practice XXI* ed S M Winkler *et al* (Springer)
[41] Muni D, Pal N and Das J 2006 *IEEE Trans. Syst., Man, Cybern. B, Cybern.* **36** 106
[42] Gerules G and Janikow C 2016 *2017 IEEE Congress on Evolutionary Computation (CEC)* pp 5034–43
[43] Kotanchek M, Smits G and Vladislavleva E 2007 *Genetic Programming Theory and Practice V* ed R L Riolo, T Soule and B Worzel (Springer) pp 201–220
[44] Markus A F, Kors J A and Rijnbeek P R 2021 *J. Biomed. Inform.* **113** 103655
[45] Archetti F, Lanzeni S, Messina E and Vanneschi L 2007 *Genet. Program. Evolvable Mach.* **8** 413 special issue on medical applications of Genetic and Evol. Comput
[46] Ain Q U, Al-Sahaf H, Xue B and Zhang M 2022 *Expert Syst. Appl.* **197** 116680
[47] Bakurov I, Buzzelli M, Schettini R, Castelli M and Vanneschi L 2023 *IEEE Trans. Image Process.* **32** 1458
[48] Evans B P, Xue B and Zhang M 2019 *GECCO'19: Proc. Genetic and Evolutionary Computation Conf. (New York, NY, USA, 2019)* pp 1012–1020
[49] Smits G F and Kotanchek M 2005 Pareto-front exploitation in symbolic regression *Genetic Programming Theory and Practice II* ed U-M O'Reilly *et al* (Springer) pp 283–99
[50] Bakurov I, Haut N and Banzhaf W 2025 Sharpness-aware minimization in genetic programming *Genetic Programming Theory and Practice XXI* ed S M Winkler *et al* (Springer) pp 151–75
[51] Goodfellow I, Bengio Y and Courville A 2016 *Deep Learning* (MIT Press)
[52] Cristianini N and Ricci E 2008 Support vector machines *Encyclopedia of Algorithms* ed M-Y Kao (Springer) pp 928–32
[53] Keijzer M 2003 *Proc. 6th European Conf. Genetic programming, EuroGP'03* pp 70–82
[54] Vanneschi L, Rochat D and Tomassini M 2007 *Proc. 9th Annual conf. on Genetic and Evolutionary Computation (New York, NY, USA)* pp 1759
[55] Kotanchek M, Smits G and Vladislavleva E 2009 Exploiting trustable models via pareto gp for targeted data collection *Genetic Programming Theory and Practice VI* (Springer) pp 18
[56] Haut N, Banzhaf W and Punch B 2023 Correlation versus RMSE loss functions in symbolic regression tasks *Genetic Programming Theory and Practice XIX* ed L Trujillo *et al* (Springer Nature) pp 31–55
[57] Chen Q, Xue B, Banzhaf W and Zhang M 2023 *Proc. Genetic and Evolutionary Computation Conf.* pp 420–428
[58] Bakurov I, Banzhaf W, Ofria C and Murphy A 2025 *Proc. Genetic and Evolutionary Computation Conf. (New York, NY, USA, 2025)*
[59] Stephens T (2016) *gplearn: Genetic Programming in Python* 2025-06-022025 https://github.com/trevorstephens/gplearn
[60] Fortin F-A, De Rainville F-M, Gardner M-A, Parizeau M and Gagné C 2012 *J. Mach. Learn. Res.* **13** 2171

[61] Spector L and Robinson A 2002 *Genet. Program. Evolvable Mach.* **3** 7
[62] Bakurov I, Buzzelli M, Castelli M, Vanneschi L and Schettini R 2021 *Appl. Sci.* **11** 4774
[63] Staats K, Pantridge E, Cavaglia M, Milovanov I and Aniyan A 2017 *Proc. Genetic and Evolutionary Computation Conf. Companion, GECCO'17* (ACM) pp 1872–1879
[64] Baeta F, Correia J, Martins T and Machado P 2021 Applications of evolutionary computation *24th Int. Conf., EvoApplications 2021* vol 2021 (Springer)
[65] Hall M, Frank E, Holmes G, Pfahringer B, Reutemann P and Witten I H 2009 *SIGKDD Explor. Newsl.* **11** 1018
[66] Wagner S *et al* 2014 Architecture and design of the heuristiclab optimization environment *Advanced Methods and Applications in Computational Intelligence* ed R Klempous *et al* (Springer International Publishing) pp 197–261
[67] (https://evolved-analytics.com/datamodeler-app/) (2025) *DataModeler* Evolved Analytics LLC
[68] Smits G and Vladislavleva E 2006 *2006 IEEE Int. Conf. Evolutionary Computation* pp 3114–20
[69] Haut N and Kotanchek M 2025 *Proc. Genetic and Evolutionary Computation Conf. Companion, GECCO'25 Companion* (Association for Computing Machinery) pp 2521–2528
[70] Udrescu S-M and Tegmark M 2020 AI Feynman: a physics-inspired method for symbolic regression *Science Advances* **6** eaay2631 arXiv:1905.11481
[71] Kotanchek M, Smits G and Vladislavleva E 2007 Pursuing the pareto paradigm: tournaments, algorithm variations and ordinal optimization *Genetic Programming Theory and Practice IV* ed R Riolo, T Soule and B Worzel (Springer) 167185
[72] Tibshirani R 1996 *J. R. Stat. Soc. Ser. B* **58** 267
[73] Fridmann J *et al* 2005 *Nature* **435** 922
[74] Zou H and Hastie T 2005 *J. R. Stat. Soc. Ser. B: Stat. Method.* **67** 301
[75] Geurts P, Ernst D and Wehenkel L 2006 *Mach. Learn.* **63** 342
[76] Breiman L 2001 *Mach. Learn.* **45** 5
[77] Dalfovo F, Giorgini S, Pitaevskii L P and Stringari S 1999 *Rev. Mod. Phys.* **71** 463
[78] Bender M, Heenen P-H and Reinhard P-G 2003 *Rev. Mod. Phys.* **75** 121
[79] Schunck N 2019 *Energy Density Functional Methods for Atomic Nuclei* (IOP Publishing) pp 2053–563
[80] Vautherin D and Brink D M 1972 *Phys. Rev. C* **5** 626
[81] Engel Y, Brink D, Goeke K, Krieger S and Vautherin D 1975 *Nucl. Phys. A* **249** 215
[82] Serot B D and Walecka J D 1992 Relativistic nuclear many-body theory *Recent Progress in Many-Body Theories* vol 3 ed T L Ainsworth *et al* (Springer) pp 49–92
[83] Ring P 1996 *Prog. Part. Nucl. Phys.* **37** 193
[84] Schunck N 2019 *Energy Density Functional Methods for Atomic Nuclei* (IOP Publishing) pp 2053–563
[85] Boguta J and Bodmer A R 1977 *Nucl. Phys. A* **292** 413
[86] Mueller H and Serot B D 1996 *Nucl. Phys. A* **606** 508
[87] Reinhard P-G, Bender M, Nazarewicz W and Vertse T 2006 *Phys. Rev. C* **73** 014309
[88] Kortelainen M, Lesinski T, Moré J, Nazarewicz W, Sarich J, Schunck N, Stoitsov M V and Wild S 2010 *Phys. Rev. C* **82** 024313
[89] McDonnell J, Schunck N, Higdon D, Sarich J, Wild S and Nazarewicz W 2015 *Phys. Rev. Lett.* **114** 122501
[90] Giuliani P, Godbey K, Bonilla E, Viens F and Piekarewicz J 2023 *Front. Phys.* **10** 1054524
[91] Quarteroni A *et al* 2014 *Reduced Order Methods for Modeling and Computational Reduction* vol 9 (Springer)
[92] Brunton S L and Kutz J N 2022 *Data-driven Science and Engineering: Machine Learning, Dynamical Systems, and Control* (Cambridge University Press)
[93] Odell D, Giuliani P, Beyer K, Catacora-Rios M, Chan M Y-H, Bonilla E, Furnstahl R J, Godbey K and Nunes F M 2024 *Phys. Rev. C* **109** 044612
[94] Keijzer M and Foster J 2007 *Genetic Programming* ed M Ebner *et al* (Springer) pp 33–44
[95] Winkler S M, Banzhaf W, Hu T and Lalejini A 225 *Genetic Programming Theory and Practice XXI* 1st edn (Springer) pp. XIV + 417
[96] Russo M 2020 *Soft Comput.* **24** 16885
[97] Pedregosa F *et al* 2011 *J. Mach. Learn. Res.* **12** 2825
[98] Buitinck L *et al* 2013 *ECML PKDD Workshop: Languages for Data Mining and Machine Learning* pp 108–22

[99] Wang B, Singh H K and Ray T 2015 *2015 IEEE Congress on Evolutionary Computation (CEC)* (IEEE) pp 1129–36
[100] Izzo D, Biscani F and Mereta A 2017 *Genetic Programming: 20th European Conf., EuroGP, Proc. 20 (Amsterdam, The Netherlands, 2017)* (Springer) pp 35–51 April 19-21, 2017
[101] Schmidt M and Lipson H 2009 *Genet. Program. Theory Pract. VII* (Springer) 73–85
[102] Phillips D R *et al* 2021 *J. Phys. G Nucl. Part. Phys.* **48** 072001
[103] 2023 *A New Era of Discovery: The 2023 Long Range Plan for Nuclear Science*
[104] Bakurov I 2025 *https://gitlab.com/ibakurov/beacon-frib-gp*
[105] Domingos P 2000 *Proc. 17th Int. Conf. Mach. Learn., ICML '00* (Morgan Kaufmann Publishers Inc.) pp 231–238
[106] Hoerl A E and Kennard R W 1970 *Technometrics* **12** 55
[107] Brunton S L, Proctor J L and Kutz J N 2016 *Proc. Natl. Acad. Sci. U.S.A.* **113** 3932
[108] Hoffmann M, Fröhner C and Noé F 2019 *J.Chem. Phys.* **150** 025101
[109] Tran G and Ward R 2017 *Multiscale Model. Simul.* **15** 1108
[110] Fasel U, Kutz J N, Brunton B W and Brunton S L 2022 *Proc. R. Soc. A* **478** 20210904
[111] Jiang Y-X, Xiong X, Zhang S, Wang J-X, Li J-C and Du L 2021 *Nonlinear Dyn.* **105** 2775
[112] Shea D E, Brunton S L and Kutz J N 2021 *Phys. Rev. Res.* **3** 023255