UNIMORE
UNIVERSITÀ DEGLI STUDI DI
MODENA E REGGIO EMILIA

Università degli Studi di Modena e Reggio Emilia

Dipartimento di
Scienze Fisiche, Informatiche e Matematiche

Corso di laurea in Physics

Tesi di Laurea Magistrale

Uncertainty Quantification in Symbolic Regression for Modelling Ion Diffusion in Electrolytes

Relatori
prof. Federico Grasselli

Correlatore:
dott.ssa Veronica Guidetti

Laureando
Paolo DUGONI

ANNO ACCADEMICO 2024-2025

Contents

Introduction

For centuries, scientists have sought to identify mathematical laws capable of describing and predicting the behaviour of a physical system starting from empirical data and observations. For example, Johannes Kepler discovered the famous third law of planetary motion, which would later bear his name, by identifying patterns in experimental data collected by astronomers in the preceding decades; similarly, Max Planck initially proposed his law as a mathematical trick to fit experimental data on black-body radiation, without yet understanding its revolutionary conceptual importance[1].

This optimization process, based on experimental data and human intuition, is known as *symbolic regression*[2,3,4,5]. Such an approach aims to generate simple symbolic formulas, in contrast to alternative models, such as deep learning models, which tend to provide more complex results, favouring predictive power at the expense of interpretability. The production of interpretable results could be crucial for the theoretical development of pure sciences such as physics and chemistry.

The symbolic regression process can be very time-consuming even for low-dimensional systems[6]. All the more so, when working with modern high-dimensional datasets, it becomes impossible without resorting to an automated mechanism. Over the years, dedicated symbolic regression applications have been developed, which are capable of producing one or more analytical formulas starting solely from a set of descriptors and an objective function, without any prior knowledge of the system's physics or geometry[7]. Most of these tools, including PySR[8,9] and SISSO[10,11], are based on genetic algorithms and tree-based structures[12], in which the best formula emerges through a process of natural selection among populations of candidate expressions, minimizing a given loss function. These software packages tend to suffer from *formula bloat*, meaning they produce overly complex expressions when optimized solely for accuracy[13], and they lack a mechanism for uncertainty estimation.

The present work aims to develop an error estimation method for formulas generated by symbolic regression software packages such as SISSO. To this end, a Bayesian extension of the symbolic regression process[14,15,16] has been implemented: specifically, an approximation of the posterior probability distribution,

i.e. the probability distribution of the models given an empirical dataset is identified. This approach makes it possible to associate each generated symbolic formula with a probability density describing the likelihood of the model given the empirical estimates. Once the posterior probabilities of each model are known, we can identify the most probable formula through a maximum a posteriori estimate and obtain an error estimate for the predictions, both at the model level and for the associated parameters.

The developed strategy is then applied to a dataset of physical interest containing data on sodium diffusivity in amorphous crystals[17]. This field presents challenges both from a theoretical perspective, since the transport of Na^+ in solid-state electrolytes is not yet fully understood, and from an applied standpoint in optimizing material fabrication and synthesis processes. An approach of this kind, leading to the production of symbolic expressions supported by robust uncertainty estimation, could foster progress in both directions.

Chapter 1

Symbolic Regression

This chapter provides an overview of the fundamental topics addressed in this work. In section 1.1, we introduce and define symbolic regression. In section 1.2, we present the Genetic Programming algorithm, which plays a central role in the symbolic regression methods considered in this study.

1.1 Symbolic Regression

Symbolic Regression[2,3,4,5] (SR) is a class of machine learning techniques aimed at discovering mathematical relationships and patterns within data. Unlike traditional regression methods, SR seeks to identify a compact, human-readable mathematical expression that accurately captures the underlying relationship between input features and the target variable, while maintaining interpretability.

Over the years, various approaches have been proposed to uncover these relationships. A widely used technique for searching symbolic expressions is evolutionary Genetic Programming[12] (GP). GP is an iterative algorithm inspired by biological evolution. In the context of symbolic regression, candidate solutions are typically represented as tree-structured graphs, where nodes correspond to input variables, operators, and basic algebraic functions. During the evolutionary process, individuals (i.e., mathematical expressions) are evaluated according to a fitness function, and the fittest ones are more likely to be selected for reproduction, thus surving into the next generations. New offspring are generated through genetic operations such as recombination (crossover), which combines parts of parent structures, and mutation, which introduces random modifications to nodes. This process gradually improves the population of expressions over successive generations. A more detailed description is provided in section 1.2. In the present work, we will focus on two state-of-the-art codes for practical SR with GP-based search algorithms: PySR[8,9] and SISSO[10,11].

More formally, given some observed dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N_d}$ of paired examples, with features $\mathbf{x} \in \mathbb{R}^{N_x}$ and target $y \in \mathbb{R}$, the goal of SR is to identify a closed form mathematical model space $\mathcal{M} : \mathbb{R}^{N_x} \to \mathbb{R}$, mapping the inputs to the outputs. SR operates under the assumption that the underlying data-generating mechanism can be described by a *sparse* and *algebraic* input-output relationship able to generate the observations in $\mathcal{D}$:

$$y_i = m\left(\mathbf{x}_i, \boldsymbol{\theta}\right) + \epsilon_i, \tag{1.1}$$

where $m\left(\mathbf{x}, \boldsymbol{\theta}\right) \in \mathcal{M}$ is a possible closed-form model within the space of candidates, $\boldsymbol{\theta} \equiv \boldsymbol{\theta}\left(m\right) \in \mathbb{R}^{N_\theta(m)}$ are a set auxiliary model-dependent parameters, and $\epsilon_i \sim \mathcal{N}\left(0, \sigma^2\right)$ is an observational noise, assumed to be normally distributed.

SR seeks the most representative model function (equation), which best maps the inputs to their corresponding output, by searching the space of expressions, $\mathcal{M}$, and parameters, $\boldsymbol{\theta}$. This substantially translates into solving the following optimization problem:

$$\hat{m} = \underset{m \in \mathcal{M}}{\arg\min}\, \ell_{\text{pred}}\left[m\left(\mathbf{x}, \boldsymbol{\theta}\right), \mathbf{y}\right] + \lambda C\left(m\right), \tag{1.2}$$

thus finding the model $\hat{m}$ which minimizes the chosen predictive loss, e.g. mean squared error, plus a regularization penalty which depends on a measure of the complexity of the model, $C\left(m\right)$. The penalty constrains the search toward relatively simpler expressions to favour interpretability.

Complexity

Controlling the complexity level of the produced symbolic expressions is crucial to retain interpretability and avoid formula bloat. Different approaches have been attempted in that sense over the years.

Complexity can be controlled through feature selection, i.e. limiting the number of surviving features after the evolutionary process. This allows for some user control on the complexity, but it produces an element of operator-induced bias in the generated symbolic expressions.

Alternatively, the solution to a SR problem can be considered as a family of Pareto-optimal solutions:

$$\hat{m}^{(c)} = \underset{m \in \mathcal{M}}{\arg\min}\, \ell_{\text{pred}}\left[m\left(\mathbf{x}, \boldsymbol{\theta}\right), \mathbf{y}\right] \;\text{ such that }\; C\left(m\right) = c, \tag{1.3}$$

where $\hat{m}^{(c)}$ refers to the best fitting expression with complexity c. Thus, the optimal expression is found by striking a balance between two objective functions: goodness of fit (error reduction) and function complexity c, see Figure 1.1. This approach removes any explicit user control on the number of selected features or on the complexity level per se, but curbs complexity bias.

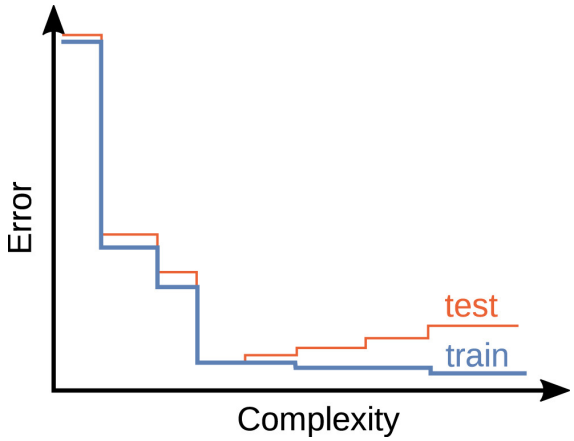


Figure 1.1: Illustration of a Pareto curve for solutions to a SR problem. The Pareto front consists of solutions that achieve an optimal trade-off between the considered objective functions: error and complexity. In addition to the training error, which monotonically decreases with increasing function complexity, the figure also depicts a hypothetical test error which highlights the onset of overfitting at higher levels of complexity. Picture adapted from [2].

1.2 Genetic Programming

In the natural world, complex and intricate structures do not arise via explicit design or from the application of human intelligence, but they evolve over a period of time as the consequence of Darwinian natural selection and sexual recombination, responding to stimuli applied by the surrounding environment.

Since computer programs and algorithms can be as complex and intricate as some natural phenomena, so arose the idea of devising an analogue of natural selection and genetics for the automatic creation of a program that would enable a computer to solve a specific problem. Thus, first *Genetic Algorithms* [18] (GA) and then *Genetic Programming* [12] (GP) were born. The difference between the two lies mainly in the nature of the solutions they evolve. GA represents solutions as fixed size character strings, while GP uses a hierarchical tree-based representation with an indefinite size an shape, which more naturally describes a computer program. Although the distinction is conceptually clear, the two terms are nowadays used interchangeably in the literature.

GP is an evolutionary algorithm based on tournament selection for individual draft [19,20] and genetic operations, like mutations and crossovers, for generating new individuals. It starts with an initial population of randomly generated computer programs, composed by a combination of functions such as arithmetic operations, programming operations, mathematical and logical functions. Each individual member of the population is measured in terms of how it performs

in the particular environment of interest. This measure is called *fitness measure*, and its nature varies with the problem. It can describe the error produced by the computer program or another parameter describing the correctness of the produced result. Thus, the closer this measure is to zero, the better the program. In other cases, it could be a gauge of the parsimony or efficiency of the program, e.g. in a problem of optimal control the fitness may be the amount of time, fuel or money needed to bring the system to a desired target state. The smaller this amount, the better. The GP process can be described as follows.

- **Random Initial Population.** The first generation of programs will have exceedingly poor fitness. Nonetheless, some individuals will perform slightly better, and thus be slightly fitter, than their peers. These minute differences are leveraged to create a new offspring population based on the Darwinian principle of survival of the fittest through reproduction and the genetic operation of sexual recombination (crossover).

- **Reproduction.** The reproduction process involves selecting a computer program from the current population on the basis of fitness (i.e. the higher the fitness measure, the more likely it is to be selected) and allowing it survive into the next generation either as an exact copy or with slight random modifications introduced through mutation.

- **Crossover.** The genetic process of sexual recombination or *crossover* generates offspring programs from two parental programs, selected on the basis of fitness. The generated offsprings are composed of subexpressions: subtrees, subprograms, subroutines and building blocks, acquired from their parents. The idea being that if a program is fitter than its peers, the merit lies in some of its parts. Thus, crossover creates new computer programs using efficient parts of pre-existing parental programs, allowing them to survive into future generations.

- **New Population.** After the operations of reproduction and crossover are completed on the current generation, a new generation of programs is created and the process can be repeated. This algorithm will produce populations of computer programs which, over many generations, tend to exhibit increasing average fitting in dealing with their environment. The single best individual in the given generation is usually designated as the result produced by GP.

Level Tree Representation of a Symbolic Expression

A *Level Tree* is a special data structure used for data storage purposes. It is comprised by a collection of nodes and links. The hierarchical ordering of the nodes, conveys a causative parent-child relationship.

When applied to the context of SR, the nodes indicate the instructions to be executed, while the links indicate the arguments for each instruction, see Figure 1.2. A node can have many children (two at most, if the tree is binary), but at most one parent. The topmost node, without a parent node, is called root, while the downmost nodes are called leaves. The height of a tree is defined as the length of the longest path from its root to one of its leaves, counting the number of values on the path.

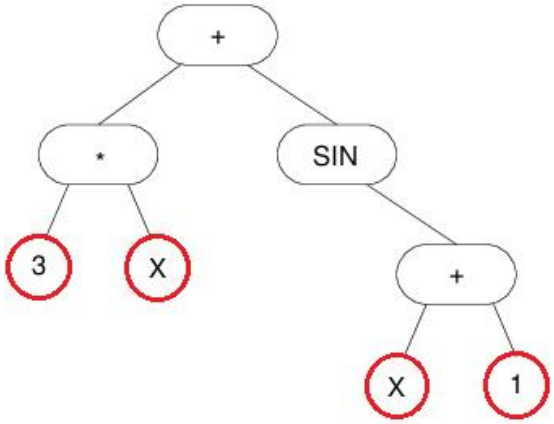


Figure 1.2: Level tree representation of the symbolic expression $3x + \sin(x + 1)$. Here, the leaves are circled in red and the root, the topmost "+", identifies the mathematical operator connecting the two subtrees. The height of the above tree is 4. Picture adapted from [21].

Chapter 2

Search Algorithms

In this chapter, we describe the search algorithms implemented in the software packages SISSO[10,11] (section 2.2) and PySR[8,9] (section 2.3). But first, in section 2.1, we provide an overview of the compressed sensing mechanism that underlies the feature selection process employed by SISSO.

2.1 Compressed Sensing

Data based and Artificial Intelligence (AI) approaches are becoming a vital tool for describing physical and chemical processes. The main advantage of the AI approach is its ability to find correlations between different sets of properties without needing to know which ones are important beforehand. Among the sets of methods for interpretable AI we have Symbolic Regression (SR)[2,3,4,5], in which the underlying algorithm identifies the optimal symbolic expression for a given target property from a set of input features, also called primary features.

The *Sure-Independence Screening and Sparsifying Operator*[10] (SISSO) approach provides an algorithm for practical SR, purportedly specialized in materials development. SISSO employs a compressed-sensing based methodology for feature selection[22,23], specifically for identifying descriptors, i.e. sets of parameters capturing the underlying mechanisms of a material's property, such as atomic radii, ionization energies, valences, bond distances and so on.

Mathematical Description

Let us assume to have a set of properties $P_1, ..., P_N \in \mathbb{R}$, and a set of descriptors $\mathbf{d_1}, ..., \mathbf{d_N} \in \mathbb{R}^M$, that we presume are related. In the spirit of symbolic regression, we want to look for an interpretable equation $P = f(\mathbf{d})$, connecting the given property to a small number of significative descriptors. In order to achieve that goal, some constraints are imposed to find a low-dimensional linear solution

for $P = f(\mathbf{d})$. By introducing non-linear transformations to the vector $\mathbf{d}$, the resulting expression will effectively become a linear superposition of non-linear candidate descriptors, thus accounting for non-linear contributions.

When looking for a linear dependence $P = f(\mathbf{d}) = \mathbf{d} \cdot \mathbf{c}$, the most simple approach is the *Method of Least Squares*:

$$\underset{\mathbf{c}\in\mathbb{R}^M}{\arg\min} \sum_{j=1}^{N} \left(P_j - \sum_{k=1}^{M} d_{j,k}\, c_k \right)^2 = \underset{\mathbf{c}\in\mathbb{R}^M}{\arg\min} \left\| \mathbf{P} - \mathbf{D}\mathbf{c} \right\|_2^2, \tag{2.1}$$

where $\mathbf{P} = (P_1, ..., P_N)^T$ is the column vector of the outputs, $\mathbf{D}$ is the $(N \times M)$-dimensional matrix of the inputs (sensing matrix) and $\mathbf{c}$ is the unknown column vector. Equation 2.1 can be solved analytically by

$$\mathbf{c}^* = \left(\mathbf{D}^T\mathbf{D} \right)^{-1} \mathbf{D}^T\mathbf{P}, \tag{2.2}$$

provided that $\mathbf{D}^T\mathbf{D}$ is non-singular. Since the problem is convex, the solution can be computed efficiently even for large-dimensional sensing matrices. The vector $\mathbf{c}$ is subject only to the requirement of linearizing $\mathbf{P}$ while minimizing the Euclidean norm.

Some previous insight can be added into Equation 2.1 through regularization. For example, if we prefer to have solution vectors $\mathbf{c}$ which are small, i.e. shrunk toward zero magnitude, we can add a ℓ_2 norm, also called Tikhonov norm, and perform a so-called *Ridge Regression*:

$$\underset{\mathbf{c}\in\mathbb{R}^M}{\arg\min} \left(\left\| \mathbf{P} - \mathbf{D}\mathbf{c} \right\|_2^2 + \lambda \left\| \mathbf{c} \right\|_2^2 \right), \tag{2.3}$$

where the penalty parameter $\lambda \in \mathbb{R}^+$, weights the magnitude of the vector $\mathbf{c}$. The larger λ, the smaller the vector $\mathbf{c}$, while for $\lambda \to 0$ we retrieve the solution of Equation 2.1.

In this case, we aim for $f(\mathbf{d})$ to depend only on a small number of significant descriptors $\mathbf{d}$ to promote interpretability. This requires that most components of the solution vector $\mathbf{c}$ be zero. We denote the number of non-zero components of $\mathbf{c}$ by

$$\left\| \mathbf{c} \right\|_0 = \#\{ j : c_j \neq 0 \}, \tag{2.4}$$

where $\{ j : c_j \neq 0 \}$ is the set of indices j corresponding to non-zero components c_j, and # denotes the cardinality of that set. The quantity $\left\| \mathbf{c} \right\|_0$ is commonly referred to as the ℓ_0 norm of $\mathbf{c}$. We say that a vector $\mathbf{c} \in \mathbb{R}^M$ is Ω-*sparse* if $\left\| \mathbf{c} \right\|_0 \leq \Omega$, i.e., if at most Ω of its components are non-zero.

When trying to regularize Equation 2.1 such that it yields sparse solutions, one has to minimize:

$$\underset{\mathbf{c}\in\mathbb{R}^M}{\arg\min} \left(\left\| \mathbf{P} - \mathbf{D}\mathbf{c} \right\|_2^2 + \lambda \left\| \mathbf{c} \right\|_0 \right). \tag{2.5}$$

Though conceptually straightforward, solving Equation 2.5 quickly becomes computationally unfeasible as M and Ω increase. For example, a brute-force approach would require examining all subsets of indices $T \subseteq \{1, ..., M\}$ of size one and minimizing over vectors supported on that set, then repeating the procedure for subsets of size two, three, and so on up to Ω. This exhaustive search eventually leads to a so-called *NP-hard* problem[24], meaning that while solutions can be verified in polynomial time, finding them generally requires non-polynomial time.

A compromise between convexity and sparsity can be struck by using the *Least Absolute Shrinkage and Selection Operator* (LASSO) approach[25], in which the ℓ_0 norm of Equation 2.4 is replaced by the ℓ_1 norm ($\|\mathbf{c}\|_1 = \sum_k |c_k|$):

$$\underset{\mathbf{c}\in\mathbb{R}^M}{\arg\min} \left(\|\mathbf{P} - \mathbf{D}\mathbf{c}\|_2^2 + \lambda \|\mathbf{c}\|_1 \right). \tag{2.6}$$

The optimization problem is now convex[26] and by looking at the geometry of the ℓ_1-unit surface, one can show that it promotes sparsity[27,28]. As for Equation 2.3, the larger the λ, the smaller $\mathbf{c}$, and vice versa.

The minimization problem of Equation 2.6 is an approximation of the one in Equation 2.5, but there is no guarantee that the two results will necessarily coincide. They can coincide under certain conditions dictated by the notion of *Null Space Property* (NSP) defined within the theory of *Compressed Sensing*[27,28,29] (CS). Let $\mathbf{D} \in \mathbb{R}^{N\times M}$ and let $\Omega \in \{1, ..., M\}$ be an index. Then $\mathbf{D}$ is said to have a NSP of order Ω if:

$$\sum_{j\in T} |v_j| < \sum_{j\notin T} |v_j| \quad \forall\, \mathbf{v} \neq \mathbf{0} \text{ s.t. } \mathbf{D}\mathbf{v} = 0,\ \forall\, T \subseteq \{1, \ldots, M\} \text{ with } \#T \leq \Omega, \tag{2.7}$$

which means that any null-space vector ($\mathbf{v} \neq \mathbf{0}$ s.t. $\mathbf{D}\mathbf{v} = 0$), on any subset of the indices ($\forall\, T \subseteq \{1, \ldots, M\}$ with $\#T \leq \Omega$), never concentrates too much, i.e. the ℓ_1 norm of the vectors within the subset $\left(\sum_{j\in T} |v_j|\right)$ is always smaller than the ℓ_1 norm on the rest of the coordinates $\left(\sum_{j\notin T} |v_j|\right)$, no matter how large the subset. Even if you pick the largest $s \leq \Omega$ entries, the null-space vectors do not concentrate, they spread out. In the limit case of $\Omega = \#T = M$, the strict inequality in Equation 2.7 becomes the $0 = 0$ identity. This condition is never met in CS, where we always have $\Omega \ll M$.

It can be shown[30] that every Ω-sparse vector $\mathbf{x}$ is the unique solution of Equation 2.6 with $\mathbf{P} = \mathbf{D}\mathbf{x}$ if, and only if, $\mathbf{D}$ has the NSP of order Ω. As a consequence, if $\mathbf{D}$ has the NSP of order Ω, the convex ℓ_1 minimization of Equation 2.6 recovers all Ω-sparse vectors, just as the ℓ_0 minimization in Equation 2.5. Unfortunately, given a specific matrix $\mathbf{D}$, it is not trivial to check whether it has the NSP of order Ω, but some guidance regarding the admissible dimension M of $\mathbf{D}$ can be extracted from the CS theory[28,30].

2.2 SISSO

The *Sure-Independence Screening and Sparsifying Operator*[10] (SISSO) approach, combining genetic algorithms-based SR with compressed sensing[27,28,29], provides a deterministic way of performing feature selection[22,23], i.e. determining which among the primary features are the most descriptive, without any pre-existing knowledge about their relative importance, and then finding a set of symbolic expressions relating the target to selected features. The SISSO algorithm provides a tool for practical SR in which the complexity level of the produced expressions is controlled indirectly through the feature selection step. SISSO has been used in numerous materials-science related applications[31,32,33].

The SISSO search algorithm starts from an initial feature space $\mathbf{\Phi_0}$, which is formed by all the descriptors (primary features) that are hypothesized to be relevant to describe the desired target. Then, a set of unary and binary mathematical operators (e.g. addition, multiplication, logarithm, trigonometric functions,...) is exhaustively applied to the initial feature space, extending it. In this so-called *feature creation step*, a pool of generated features is built through the combination of these mathematical operators with the primary features:

$$\hat{\mathbf{H}}^{(m)} \equiv \left\{ I, +, -, \times, /, \exp, \log, |-|, \sqrt{\ }, ^{-1}, ^{2}, ^{3} \right\} [\phi_1, \phi_2] \,, \tag{2.8}$$

where ϕ_1 and ϕ_2 are elements of the feature space $\mathbf{\Phi}$ (for unary operations only ϕ_1 is considered) and the superscript $^{(m)}$ indicates that dimensional analysis is performed at each step to retain only dimensionally meaningful combinations of features. This step is iteratively repeated, applying the mathematical operators $\hat{\mathbf{H}}^{(m)}$ to the previously created generated features, leading to a recursively growing feature space:

$$\mathbf{\Phi}_R \equiv \bigcup_{i=1}^{R} \hat{\mathbf{H}}^{(m)} [\phi_1, \phi_2], \quad \forall \phi_1, \phi_2 \in \mathbf{\Phi}_{i-1}. \tag{2.9}$$

The number of feature creation steps is called rung. If we were to use trees to represent the generated features, the rung would coincide with the height of the resulting binary expression tree. At each step, higher-rung features are constructed by combining lower-rung features with arithmetic operations. As the rung R of each candidate descriptor increases, the number of elements in the feature space increases combinatorially, in a way which depends on the number of primary features and on the cardinality of the set of mathematical operations.

A sparse-solution algorithm is then employed to extract simple and interpretable expressions from the feature space. A Sparsifying Operator[34,35] (SO) combined with a Sure Independent Screening[36] (SIS) procedure were adopted,

since they have been shown to effectively reduce the dimensionality of ultra-high dimensional features spaces[35,37]. SIS evaluates all generated features using the Pearson correlation coefficient and retains only the top-ranking ones[35], which together define a subspace $\mathcal{S}$. After the reduction, compressed sensing (SO) is used to identify the best n-dimensional linear model by performing an ℓ_0-regularized optimization on $\mathcal{S}$, as in Equation 2.5. This process is repeated for multiple feature space dimensions until the leftover residual error falls within a certain threshold.

Out of a huge feature space, the SIS procedure, applied to the target material property $\mathbf{P}$, will produce a subspace $\mathcal{S}_{1D}$ containing the 1D features which are most correlated with such property. From $\mathcal{S}_{1D}$, SO(l_0) finds the best 1D descriptor which, in this case, is trivially the first ranked formula. The residual error for an n-dimensional model is generally defined as

$$\mathbf{\Delta}^0_{nD} = \mathbf{P} - \mathbf{P}^0_{nD} = \mathbf{P} - \mathbf{d}_{nD}\mathbf{c}_{nD}, \tag{2.10}$$

where $\mathbf{P}^0_{nD} = \mathbf{d}_{nD}\mathbf{c}_{nD}$ is the estimate of the best n-dimensional model, $\mathbf{d}_{nD}$ is the column matrix of the selected features and $\mathbf{c}_{nD} = \left(\mathbf{d}^T_{nD}\mathbf{d}_{nD}\right)^{-1}\mathbf{d}^T_{nD}\mathbf{P}$ is the least-square solution of fitting $\mathbf{d}_{nD}$ to $\mathbf{P}$. If the root-mean-square of the residual goes below a certain threshold, then the descriptor is considered fit. Otherwise the model recursively considers an higher dimension, constructing a space $\mathcal{S}_2$ out of the best generated features that can explain the knowledge gap left by the previous step, and so on. In general at dimension n, SIS selects the subspace $\mathcal{S}_{nD}$ with response $\mathbf{\Delta}^0_{(n-1)D}$, ranking the features using a projection score

$$s^0_j = R^2\left(\mathbf{\Delta}^0_{(n-1)D}, \mathbf{d}_j\right), \tag{2.11}$$

where R is the Pearson correlation coefficient, and then SO selects the best nD descriptor, from the union of all previously selected subspaces

$$\mathcal{S}_{1D} \bigcup \mathcal{S}_{2D} \bigcup ... \bigcup \mathcal{S}_{nD}. \tag{2.12}$$

This process is represented in Figure 2.1.

In recent versions of the software[11], the residual definition is extended and uses the best r residuals to calculate the projection score:

$$\max\left(s^0_j, s^1_j, ..., s^{r-1}_j\right), \tag{2.13}$$

where s^1_j accounts for the leftover error of the second best $(n-1)$-dimensional model and so on. The value of r for a calculation is set as a hyperparameter via cross-validation, i.e. the dataset is divided into a training set and a validation set, that are used respectively to train and test the model for a set of values of

r. The hyperparameter with the lowest average validation error is selected and used to retrain the model over the full dataset.

This multiple-residual approach prevents early greedy choices from blocking better higher-dimensional feature combinations by evaluating candidate features against several promising residual directions instead of just the best one.

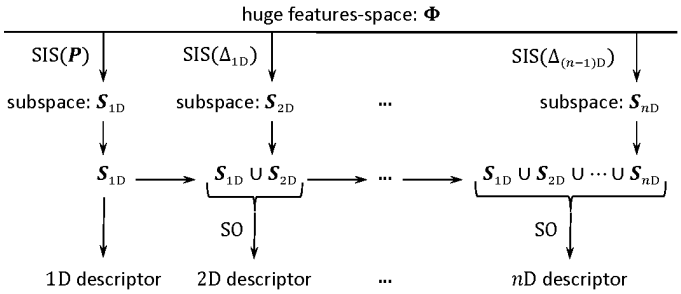


Figure 2.1: The original SISSO method combines unified SIS-generated subspaces having the largest correlation with either **P** or the residual errors **Δ**, with SO (sparsifying operator) to further extract the best descriptor from the union of all previously selected subspaces, Equation 2.12. Picture adapted from [10].

SISSO++

The software SISSO++ [11,38] enhances the original SISSO algorithm and implements it through a C++ code with Python bindings. In the newer implementation, features are represented as binary trees instead of strings, which purportedly yields an higher data storage capability and reduces the overall computational cost of the calculations. Units are treated more accurately and the resulting expressions are explicitly checked for physical consistency, namely addition and subtraction are allowed to act only on features of the same units and all transcendental operations must act only on a unitless quantity. Ranges are also added to prevent an operation from occurring when it would violate its mathematical definition, such as taking the square root of a negative number or dividing by zero.

The search algorithm has now a fully parallelized feature creation step [39], which is embedded with a parametric extension to automatically include scale and bias terms for each of the generated expressions [40]. For a generic generated feature $\hat{h}(\mathbf{x}) \in \hat{\mathbf{H}}^{(m)}$, with a set of scale and bias parameters, $\hat{\mathcal{P}}$, the parametrization scheme updates the operator to be:

$$\hat{h}(\mathbf{x}) \rightarrow \hat{h}^{\hat{\mathcal{P}}}(\alpha_1 \mathbf{x} + \beta_1), \tag{2.14}$$

where α_1 is the scale parameter and β_1 is the bias term. These new operators are then combined with each other and with the predefined set of mathematical operators, to create new parametrized generated feature $\hat{\phi}^{\hat{\mathcal{P}}}(\mathbf{x})$, as is normally done

in SISSO. This process introduces a new hyperparameter called parametrization depth, d_P, which controls the level at which the parametrization procedure is operated. An example of which is provided in Figure 2.2.

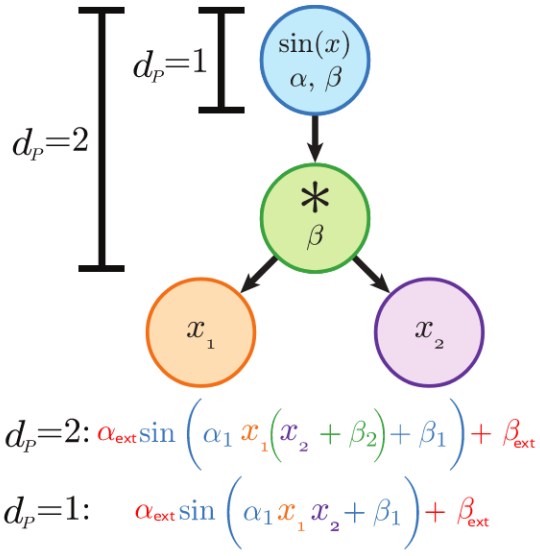


Figure 2.2: A graphical representation of some of the novelties introduced by SISSO++: the binary tree representation of a symbolic expression and the effect of the parametrization depth. If $d_P = 1$ only the last generated feature, $x_1 x_2$, gets parametrized. Whereas, if $d_P = 2$ both operations are separately parametrized: first $x_2 \rightarrow \alpha_2 x_2 + \beta_2$, and then the resulting product $x_1 * \alpha_2 (x_2 + \beta_2) \rightarrow \alpha_1 x_1 (x_2 + \beta_2) + \beta_1$. An external parametrization $(\alpha_{ext}, \beta_{ext})$ is always applied on the final expression. Picture adapted from [11].

Once the parametrized generated feature $\hat{\phi}^{\hat{\mathcal{P}}}(\mathbf{x})$ is defined, all parameters $\hat{p} \in \hat{\mathcal{P}}$ are optimized using the non-linear optimization library NLopt[41]. The Cauchy loss function is used as the objective function for the parameter optimization since it is believed to be more robust against outliers in the dataset, than the mean square error loss function.

$$\begin{aligned} &\min_{\hat{\mathcal{P}}} f\left(\mathbf{P}, \hat{\phi}^{\hat{\mathcal{P}}}\right), \\ &f\left(\mathbf{P}, \hat{\phi}^{\hat{\mathcal{P}}}\right) = \sum_{i}^{n_{\text{samp}}} \frac{c^2}{n_{\text{samp}}} \ln\left(1 + \left(\frac{P_i - \hat{\phi}^{\hat{\mathcal{P}}}(\mathbf{x}_i)}{c}\right)^2\right), \end{aligned} \tag{2.15}$$

where $\mathbf{P}$ is a property vector, c is a scaling factor set to 0.5 for all calculations and n_{samp} is the number of samples.

The parametric extension, though crucial for some applications, will significantly increase the time needed to generate the feature space, and it may constrain the resulting expressions into an artificially linearized form.

2.3 PySR

PySR[8,9] is an open-source library for practical symbolic regression. Its search algorithm is based on a multi-population evolutionary algorithm with multiple evolutions performed asynchronously and independently on each population. The main loop of PySR is based on a modified version of the classic evolutionary genetic programming algorithm, summarized in Figure 2.3.

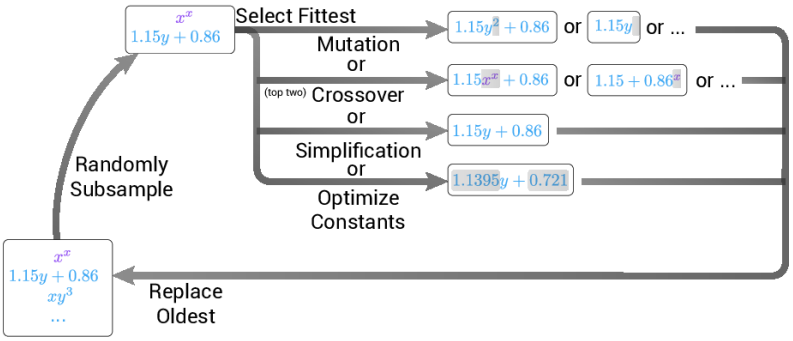


Figure 2.3: Representation of the main loop of PySR, which is applied independently to each population. A subsample of individuals is randomly drawn within each population. Starting from the fittest, each of these individuals have a probability p of undergoing a randomly-selected variation from a set of possible alterations: mutation, crossover, simplification and constants optimization; otherwise the individual is removed from the subsample. Then, the eldest member of the current population is replaced with the mutated individual. Once this process is completed a new generation is formed. Picture adapted from[8].

The PySR algorithm operates several modifications to the original evolutionary algorithm. The algorithm is parallelized, making the process computationally efficient, by evolving each subsample of individuals independently. After a specified number of rounds of evolution, an asynchronous migration of individuals between the groups is performed.

A process of simulated annealing[42] is applied to the mutation probability p. Given a temperature $T \in [0,1]$, a mutation is accepted with some probability

$$p = \exp\left(\frac{L_F - L_E}{\alpha T}\right), \tag{2.16}$$

related to the change in the fitness measure after the mutation, with L_F and L_E

being respectively the fitness of the muted and original individual, and α a hyperparameter. The parameter α controls the temperature scale: as $\alpha T \to \infty$, all mutations are accepted regardless of fitness change, whereas as $\alpha T \to 0$, only mutations that increase fitness (i.e., with $L_F - L_E > 0$) are accepted, and all others are rejected. This allows the evolution to alternate between high temperature phases, increasing individual diversity, and low temperature phases, narrowing in on the fittest individuals, in different moments of the evolutionary process.

Then, instead of replacing the weakest individual, i.e. the least fit, in the subset, as it was originally done, PySR replaces the eldest member. This *age-regularized* evolution[43] prevents the population from specializing too early and getting stuck in a local minimum of the search space.

With the third applied modification, the genetic algorithm is embedded inside an *evolve-simplify-optimize loop*. This serves to further specialize the algorithm to deal with symbolic expressions.

- **"Evolve"** applies the tournament selection-based evolution thousands of times for a set of mutations (and crossover), evolving the entire population. The temperature of the system is set to decrease over time, allowing for more drastic variations at the beginning, to widen the search space, and then focusing on fine tuning in the final iterations. Amongst the possible mutations we have: constant mutation, which creates a random multiplier; operator mutation, in which one mathematical operator is switched with another of the same degree, e.g. $+ \to \times$; node addition inside the the expression or to the features, e.g. $x_1 \to \sin(x_1)$; subtree modification or deletion; creation of an entirely new tree.

- **"Simplify"** here refers to equation simplification. Equations are simplified to an equivalent algebraic form using a set of equivalences.

- **"Optimize"** consists of a few iterations of a classical optimization algorithm, e.g. BFGS[44], to explicitly optimize the constants in every equation. This final part of the algorithm significantly improves the discovery of real constants within the symbolic expression, which is one of the strong suits of PySR. The *simplify-optimize* stage is applied after several mutations have been performed, because some equations are accessible only after some redundant, but necessary, intermediate steps take place.

Finally, a novel *adaptive parsimony* metric is adopted. Traditionally, as seen in Equation 1.2, the mechanism to account for complexity would comprise of a constant parsimony, involving a function of the complexity $C(m)$ and a constant factor λ. Instead, PySR adaptively tunes a per-complexity penalty such that the number of expressions at each complexity is approximately the same. This is

expressed as:

$$\hat{\ell}(E) = \ell_{\text{pred}}(E) \cdot \exp\left(\text{frecency}\left[C(E)\right]\right), \tag{2.17}$$

where the "frecency" of $C(E)$ is some combined measure of the frequency and recency of expressions E occurring at complexity $C(E)$ in the population. Complexity in PySR is by default equal to the number of nodes in an expression tree, regardless of the node's content. Frequency is accounted by counting the number of expressions within a given time frame and diving it by a tunable constant. Punishing complexities adaptively by how frequent they are in the population, the evolutionary search is encouraged to explore the problem employing simultaneously simple but less accurate expressions, as well as complex and more accurate expressions, preventing the system from getting stuck in local minima.

Chapter 3

The Bayesian approach

In this chapter we are going to summarize the ideas of Bayesian statistics, in section 3.1, and then describe the state-of-the-art Bayesian approach methods to symbolic regression, section 3.2. In section 3.3, we provide an overview of the statistical methods employed in the post-hoc analysis of the results.

3.1 Bayes' Theorem

Nearly all physical activities, especially in the scientific world, require some ability to reason in presence of uncertainty. Probability theory was developed exactly to deal with uncertainty in different contexts. For events like drawing a certain hand of cards from a deck, the uncertainty estimate is simpler due to their intrinsically repeatable nature. Thus, we can state that an outcome has probability p of occurring if, repeating the experiment infinitely many times, a proportion p of the repetitions would result in that outcome. Mathematically, the probability of an event A is defined as

$$p \equiv P(A) = \lim_{n \to \infty} \frac{N_A(n)}{n}, \tag{3.1}$$

where $N_A(n)$ indicates the number of outcomes favourable to A over the n trials. This approach is called *frequentist probability*.

The frequentist interpretation though becomes ill-defined when applied to a non-repeatable event, like a horse race or the chance of a patient having flu. In these cases, the probability P is interpreted as a *degree of belief*, with 1 indicating absolute certainty that the given event will occur (e.g. the horse will certainly win the race or the patient has definitely the flu) and 0 indicating absolute certainty that the given event will not occur. This alternative interpretation is called *Bayesian probability*[45].

To evaluate the probability of a hypothesis, using Bayesian inference, one has to specify a prior probability. This, in turn, is then updated to a posterior probability in the light of new, relevant data, called evidence and of a likelihood function, derived from a statistical model of the observed data. This updating process is mathematically expressed by *Bayes' Theorem*:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{3.2}$$

where A, the hypothesis, and B, the evidence, are two probabilistic events.

$P(A|B)$ is called *posterior probability* and it represents the probability of the hypothesis given the observed evidence. $P(B|A)$ is called *likelihood* and it represents the probability of the evidence given that the hypothesis is true. $P(A)$ is called *prior probability* and is the estimate of the probability of the hypothesis A before the data B, the current evidence, is observed. $P(B) \neq 0$ is called *evidence probability* or *marginal likelihood*, due to its possible definition as a marginalization of the likelihood $P(B) = \int dA\ P(B|A)P(A)$, it is constant for all hypotheses.

3.2 Bayesian Symbolic Regression

In SR, given some observed dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N_d}$, we aim to identify the closed form mathematical model $m \in \mathcal{M}$ responsible for the generation of some observed variables y_i, through the process

$$y_i = m\left(\mathbf{x}_i, \boldsymbol{\theta}\right) + \epsilon_i. \tag{3.3}$$

Here the model m is a function of the model-dependent parameters $\boldsymbol{\theta} \equiv \boldsymbol{\theta}\left(m\right) \in \mathbb{R}^{N_\theta(m)}$ and of independent variables $\mathbf{x} \in \mathbb{R}^{N_x}$, and $\epsilon_i \sim \mathcal{N}\left(0, \sigma^2\right)$ is an observational noise, assumed to be normally distributed. Given that both the dataset and the model are affected by uncertainty, the most complete description of the SR problem is probabilistic.

The Bayesian approach to SR[14,15,16], referred to as Bayesian SR, involves estimating the joint posterior distribution over models, m, and parameters, $\boldsymbol{\theta}$, given an observed data, $\mathcal{D}$, by application of Bayes' theorem:

$$P\left(\boldsymbol{\theta}, m|\mathcal{D}\right) = \frac{P\left(\mathcal{D}|\boldsymbol{\theta}, m\right) P\left(\boldsymbol{\theta}, m\right)}{P\left(\mathcal{D}\right)} = \frac{P\left(\mathcal{D}|\boldsymbol{\theta}, m\right) P\left(\boldsymbol{\theta}|m\right) P\left(m\right)}{P\left(\mathcal{D}\right)}, \tag{3.4}$$

where $P\left(\boldsymbol{\theta}, m|\mathcal{D}\right)$ is the model-parameter joint posterior, $P\left(\mathcal{D}|\boldsymbol{\theta}, m\right)$ is the likelihood, $P\left(\boldsymbol{\theta}, m\right) = P\left(\boldsymbol{\theta}|m\right) P\left(m\right)$ is the joint prior and $P\left(\mathcal{D}\right)$ is the evidence.

Given that the ultimate goal of SR is to identify models matching the data and that exploration in the trans-dimensional space of models and parameters

is complicated, Equation 3.4 can be marginalized over the parameter dimension, resulting in a posterior computed only over models

$$P(m|\mathcal{D}) = \int_{\mathbb{R}^{N_\theta(m)}} P(\boldsymbol{\theta}, m|\mathcal{D})\, d\boldsymbol{\theta} \propto P(m) \underbrace{\int_{\mathbb{R}^{N_\theta(m)}} P(\mathcal{D}|\boldsymbol{\theta}, m)\, P(\boldsymbol{\theta}|m)\, d\boldsymbol{\theta}}_{\text{Marginal Likelihood}}, \quad (3.5)$$

where $P(m|\mathcal{D})$ is the *model posterior*, $P(m)$ is the model prior and the integral identifies the marginal likelihood. With this marginalization we can explore the model space without requiring trans-dimensional jumps in the parameters $\boldsymbol{\theta}$, leading to a more efficient approximation of the posterior using standard sampling algorithms like Markov Chain Monte-Carlo methods or Sequential Monte-Carlo, discussed in section 3.3. However, this advantage comes at a cost: the marginal likelihood must be evaluated for each candidate model m, leading to a computationally demanding double loop. In other words, we trade improved efficiency in model-space exploration for increased computational cost per model.

Little prior information is usually available on $\boldsymbol{\theta}$, since the model form is unknown a priori. Therefore, the prior is often chosen to be an uninformative, improper uniform distribution, constant over $\mathbb{R}^{N_\theta(m)}$, i.e. $P(\boldsymbol{\theta}|m) \propto 1$, which results in an indeterminate model-dependent constant appearing in the marginal likelihood [46,47,48].

Once the model posterior $P(m|\mathcal{D})$ is obtained, we can determine the most plausible model $\hat{m}$ given the data, which is identified as the *Maximum A Posteriori* (MAP) of the model posterior

$$\hat{m} = \arg\max_{m \in \mathcal{M}} P(m|\mathcal{D}), \quad (3.6)$$

or we can make any prediction about the outcome.

Standard Bayesian inference may fail if the probability model m under consideration is wrong yet useful, i.e. it does not correspond to the true model under which the dataset $\mathcal{D}$ is generated but is still representative of such distribution. This so-called *Misspecification error*, is almost unavoidable in SR. Indeed, due to the intrinsic simplicity of the produced expressions, they will inevitably miss some of the complexity of the true behaviour to retain interpretability, thus leading to a misspecification.

3.3 Post-Hoc Methods

Sequential Monte-Carlo

A direct estimation of the posterior probability distribution, either the joint posterior, shown in Equation 3.4, or the model posterior, shown in Equation 3.5,

becomes computationally intractable for large dimensional models spaces. Indeed, as the number of possible symbolic expressions grows, integrating over all models, which is necessary to compute the evidence, becomes unfeasible. Since full Bayesian inference is impossible, an approximation is introduced: instead of estimating the posterior directly, it is reconstructed through sampling.

Sequential Monte-Carlo[49] (SMC) is a population-based sampling algorithm which combines several statistical ideas, including importance sampling, tempering and Markov Chain Monte-Carlo methods (MCMC). SMC sampling has been preferred to MCMC in the current work and in various state-of-the-art works on Bayesian SR, see[14,15], due to its higher candidate acceptance rate and its intrinsic parallelization, which improves computational efficiency. In the following paragraphs we provide a brief description of the SMC algorithm.

Likelihood-Tempering Let us suppose that we want to sample a target model distribution $P\left(m|\mathcal{D}\right)$, like that in Equation 3.5. Instead of sampling it directly, SMC builds a sequence of easier-to-sample probabilities $P_0\left(m|\mathcal{D}\right), ..., P_T\left(m|\mathcal{D}\right)$, with $P_T\left(m|\mathcal{D}\right) = P\left(m|\mathcal{D}\right)$, by gradually increasing the likelihood weight using an auxiliary (inverse) temperature parameter $\beta_t \in [0,1]$.

In this process, called *likelihood-tempering*, the t^{th} target distribution is defined as:

$$P_t\left(m|\mathcal{D}\right) \propto P\left(m\right)\left[\int_{\mathbb{R}^{N_\theta}} P\left(\mathcal{D}|\boldsymbol{\theta},m\right)P\left(\boldsymbol{\theta}|m\right)d\boldsymbol{\theta}\right]^{\beta_t} = P\left(m\right)P\left(\mathcal{D}|m\right)^{\beta_t}, \tag{3.7}$$

where $P(\mathcal{D}|m)$ denotes the marginal likelihood, and with $0 = \beta_0 < \beta_1 < ... < \beta_T = 1$, such that the target distributions can smoothly transition from the known initial prior $P\left(m\right)$, at $t = 0$, to the true posterior $P\left(m|\mathcal{D}\right)$ of Equation 3.5. The parameter β_t plays the role of an inverse temperature: for $\beta_t = 0$ $(T \to \infty)$ the system is fully disordered and the likelihood is ignored, while for $\beta_t = 1$ the system is ordered and the likelihood is fully incorporated.

Reweighting At each step, SMC starts from N samples, usually denoted as *particles*, $m_{t-1}^{(i)}$, with $i = 1, ..., N$, approximately distributed according to the previous target distribution $P_{t-1}\left(m|\mathcal{D}\right)$. These particles are then reweighted, resampled and propagated (through MCMC), to approximate the subsequent distribution $P_t\left(m|\mathcal{D}\right)$.

For $t = 0$, the initial particles are sampled from the prior and are coupled with normalized weights $W_0^{(i)} = 1/N$, $\forall i$. The particles are reweighted at each step using importance sampling

$$W_t^{(i)} = \frac{W_{t-1}^{(i)} P\left(\mathcal{D}|m^{(i)}\right)^{\beta_t-\beta_{t-1}}}{\sum_{j=1}^{N} W_{t-1}^{(j)} P\left(\mathcal{D}|m^{(j)}\right)^{\beta_t-\beta_{t-1}}}, \tag{3.8}$$

in which the scaling factor is proportional to the ratio between the target distributions $P_t\left(m^{(i)}|\mathcal{D}\right)$ and the previous one $P_{t-1}\left(m^{(i)}|\mathcal{D}\right)$, computed at subsequent tempering steps $t-1$ and t. If a particle is good at explaining the data, the corresponding marginal likelihood will be high and, thus, the corresponding weight will be high as well.

β_t is adaptively updated[50] in order to maintain a user-specified *Effective Sample Size* (ESS), which is defined as:

$$\text{ESS} = \frac{1}{\sum_{j=1}^{N}\left(W^{(j)}\right)^2}, \tag{3.9}$$

for each value of t. The ESS can vary between 1 and N and is an estimate of the number of expressions with non-negligible weights.

Resampling After the reweighting procedure, we will typically have a few large-weight particles and a lot of near-zero-weight particles, that lie far from the region of high posterior density. If the originally-sampled population of expressions is held fixed as $\beta_t \to 1$, it is likely that the ESS will decrease since many expressions will have $W^{(i)} \approx 0$. In the context of SR, this ESS reduction could lead to a drastic depletion in the number of highly-fit equations in the population.

To avoid degeneracy, when the ESS goes below a certain threshold, two strategies are implemented in the SMC algorithm. First, the particle population is resampled with replacement[51] and the weights are renormalized to $W^{(i)} = 1/N$. During replacement, particles with large $W^{(i)}$ tend to be replicated while particles with low $W^{(i)}$ tend to be removed from the population. This, though, leads to a reduction in the number of unique equations in the population. To counteract this, the second strategy involves short runs of a forward MCMC kernel to produce a rejuvenated population approximately distributed as the current target $P_t\left(m|\mathcal{D}\right)$, thus spreading duplicated particles. The combination of resampling and MCMC moves promotes diversity and global exploration.

When $\beta_T = 1$, the final set of particles and weights $\left\{\left(m_T^{(i)}, W_T^{(i)}\right)\right\}_{i=1}^{N}$ is used to approximate the true posterior as a discrete distribution:

$$P\left(m|\mathcal{D}\right) \approx \sum_{i=1}^{N} W_T^{(i)} \delta_{m_T^{(i)}}\left(m\right), \tag{3.10}$$

concentrated at the particles positions, represented by Dirac delta functions δ. The SMC process is summarized in Figure 3.1.

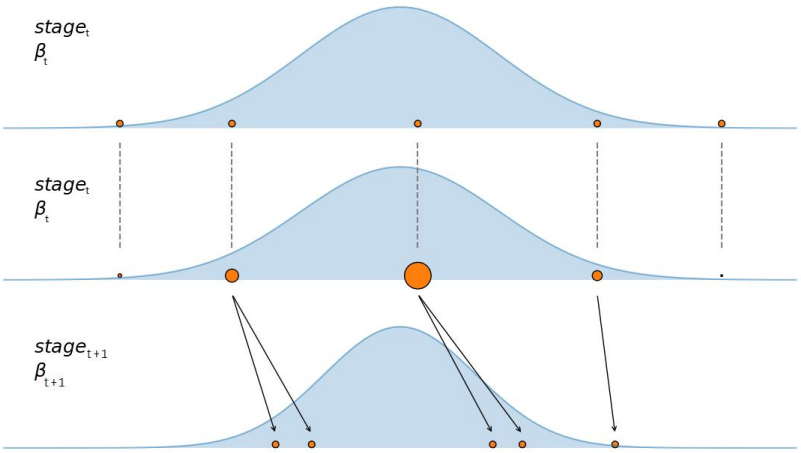


Figure 3.1: Representation of the reweighting and resampling procedure of SMC. In the first subplot, five particles sampled from a target distribution at a given stage β_t are shown, each assigned equal weights. The second subplot shows how these particles are reweighted according to the posterior distribution, using Equation 3.8. The third subplot shows the new initial weights at the $(t+1)$-th step, obtained as a result of resampling and MCMC runs on the previous weights. During resampling, lower-weight particles are discarded while higher-weight ones are duplicated; MCMC runs broaden the pool of different candidate models. Picture adapted from [52].

Comparison with Genetic Algorithms SMC samplers can also be interpreted in the light of genetic algorithms, introduced in section 1.2, by interpreting: *Sampling from Priors at* $\beta_t = 0$ as *Random Initialization*; *MCMC step* as *Mutation*; *Reweighting* as *Fitness-Based Evaluation*; *Resampling* as *Fitness-Based Selection*.

A key aspect of both approaches is maintaining sufficient diversity (mutation) in order to explore the solution space and hence avoid getting trapped in local minima. Then, a selection step is performed to probabilistically keep fitter solutions while also keeping some diversity. Being too greedy and short-sighted could be problematic, bad solutions in a given moment could lead to good solutions in the future. Unlike genetic algorithms, however, SMC provides a principled Bayesian framework in which the population approximates a sequence of well-defined probability distributions.

Marginal Likelihood and Model Selection

The Bayesian approach to model selection involves computing the marginal likelihood $P(\mathcal{D}|m)$ of each model, i.e. the probability of the observed data given the model, and compare them to assess which model is better supported by the data.

The marginal likelihood corresponds to the normalizing constant of Bayes' theorem, Equation 3.2. We can see this explicitly by combining the model-explicit Bayes' theorem, in Equation 3.4, with the definition of marginalization:

$$P(\boldsymbol{\theta}, m|\mathcal{D}) = \frac{P(\mathcal{D}|\boldsymbol{\theta}, m)\, P(\boldsymbol{\theta}|m)}{\int_{\mathbb{R}^{N_\theta}} P(\mathcal{D}|\boldsymbol{\theta}, m)\, P(\boldsymbol{\theta}|m)\; d\boldsymbol{\theta}} = \frac{P(\mathcal{D}|\boldsymbol{\theta}, m)\, P(\boldsymbol{\theta}|m)}{P(\mathcal{D}|m)}, \tag{3.11}$$

where we have conditioned on a fixed model m, so that $P(m)$ cancels out.

Given two models m_1 and m_2, if $P(\mathcal{D}|m_1) > P(\mathcal{D}|m_2)$, m_1 would better describe the data than m_2, given the priors. Sometimes the main objective is not to just keep a single model but instead to compare different models to determine which ones are more likely and obtain a ranking. The relative hierarchy between to models can be quantified using *Bayes Factors* (BF):

$$\text{BF} = \frac{P(\mathcal{D}|m_1)}{P(\mathcal{D}|m_2)}, \tag{3.12}$$

i.e. the ratio between the marginal likelihoods of the two models. The larger the BF the better the model in the numerator, and vice versa. It is customary to use the best model as a benchmark and compute the BF of the other models w.r.t. it.

It is also interesting to observe the fluctuations in the ranking order as the dimension of the probability distribution changes. This can be achieved through likelihood-tempering, in Equation 3.7, in which the likelihood weight is artificially controlled by a temperature parameter. Reliable models tend to maintain a relatively stable ordering over different temperatures.

Ensemble Calibration

An ensemble-based *calibration*[53] method was employed to assess the prediction accuracy of the models by analysing the relationship between the ensemble variance and the squared error of the ensemble mean.

To do this, we consider M independent evaluation cases (or validation points), indexed by $j = 1, ..., M$. For each point j, an observation $\hat{y}_j$ is available, and the ensemble provides N model predictions $y_j^{(1)}, ..., y_j^{(N)}$. The ensemble members for each case can be considered independent draws from a probability distribution with mean μ_j and variance σ_j^2. The average over ensemble members is represented as $\langle\cdot\rangle_N = (1/N)\sum_{k=1}^{N}$, such that the ensemble mean for case j is denoted $\langle y_j\rangle_N = (1/N)\sum_{k=1}^{N} y_j^{(k)}$.

The ensemble variance is defined as

$$s_j^2 = \frac{1}{N}\sum_{k=1}^{N}\left(y_j^{(k)} - \langle y_j\rangle_N\right)^2, \tag{3.13}$$

and the squared error of the ensemble mean by

$$\epsilon_j^2 = \left(\hat{y}_j - \langle y_j\rangle_N\right)^2. \tag{3.14}$$

In a perfectly reliable ensemble, the exchangeability of observations $\hat{y}_j$, and ensemble members $y_j^{(k)}$, implies the following relationships between σ_j^2, the variance of the true distribution, and the expected variance of members about the ensemble mean, s_j^2:

$$\frac{1}{M}\sum_{j=1}^{M}\left(\frac{N}{N-1}s_j^2 - \sigma_j^2\right) \to 0 \quad \text{as } M \to \infty, \tag{3.15}$$

$$\frac{1}{M}\sum_{j=1}^{M}\left(\frac{N}{N+1}\epsilon_j^2 - \sigma_j^2\right) \to 0 \quad \text{as } M \to \infty, \tag{3.16}$$

such that

$$\frac{1}{M}\sum_{j=1}^{M}\left(\epsilon_j^2 - \frac{N+1}{N-1}s_j^2\right) \to 0 \quad \text{as } M \to \infty. \tag{3.17}$$

The factors $N/(N-1)$, $N/(N+1)$, and $(N+1)/(N-1)$ ensure that these expressions are unbiased with respect to the ensemble size and are required because the spread and error refer to the ensemble mean $\langle y_j\rangle_N$ and not the population mean μ_j. In other words, in a perfectly reliable ensemble, the average ensemble variance will converge towards the average squared error of the ensemble mean, after correction for the finite ensemble size.

One can alternatively define the parameter[54]:

$$\gamma^2 = \frac{1}{M}\sum_{j=1}^{M}\frac{\epsilon_j^2}{s_j^2}, \tag{3.18}$$

which, according to Equation 3.17, is expected to tend to 1, up to a constant dependent on the ensemble size, as $M \to \infty$, for a well calibrated model. Values of $\gamma > 1$ indicate that the ensemble spread underestimates the actual prediction error (under-dispersed ensemble), whereas $\gamma < 1$ indicates over-dispersion.

Within Alpha Methods

Predictions from SR or Neural Network models tend to be affected by two types of uncertainty. First, there is uncertainty associated with the estimated model $\hat{m}$ appearing in Equation 1.2. This type of uncertainty arises from limited data and imperfect model estimation and is commonly referred to as *model* or *epistemic* uncertainty. On the other hand, if we want to predict new y-values using the estimated SR model, we have an additional source of uncertainty due to the inherent randomness of ϵ_i, defined in Equation 3.3. The uncertainty due to ϵ_i is often referred to as *aleatoric* uncertainty. If the distribution of ϵ_i does not depend on the independent variables $\mathbf{x}_i$, the noise is called *homoscedastic*. Otherwise, it is referred to as *heteroscedastic*.

For practical applications of the SR models, it is often necessary to quantify this aleatoric uncertainty. This quantification can be achieved by accompanying a prediction interval to the estimate.

A $(1-\alpha)\cdot 100\%$ *pointwise prediction interval* $\mathrm{PI}^{(\alpha)}$ is a random mapping

$$\begin{aligned} &\mathrm{PI}^{(\alpha)}\left(\mathcal{D}, \mathrm{U}, \cdot\right) : \mathbb{R}^{N_d} \to \mathcal{P}\left(\mathbb{R}\right), \text{ such that} \\ &\mathbb{E}_{\mathcal{D},\mathrm{U}}\left[\mathbb{E}_{y|\mathbf{x}}\left[\mathbb{1}\left\{y \in \mathrm{PI}^{(\alpha)}\left(\mathcal{D}, \mathrm{U}, \mathbf{x}\right)\right\}\right]\right] = 1-\alpha \quad \forall \mathbf{x}. \end{aligned} \tag{3.19}$$

Here $\mathrm{PI}^{(\alpha)}$ depends on the observed dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N_d}$ but also on some random effects of the training process represented by the random variable U. $\mathcal{P}\left(\mathbb{R}\right)$ is the power set of $\mathbb{R}$ and $\mathbb{1}\{\cdot\}$ represents the indicator function, which yields 1 if the condition in parenthesis is true and 0 otherwise. Intuitively, Equation 3.19 says that, by randomly sampling the targets, the probability that an observation y falls inside the created prediction interval is $1-\alpha$, for all values of $\mathbf{x}$.

The calibration of the model can be tested using a PI based metric. The idea is to split the dataset into a training and a test set, create prediction intervals using the training set and calculate the fraction of test points that fall within each interval.

The *Within-α Probability* or *Prediction Interval Coverage Probability*[55] (PICP), is the fraction of observations in a test set that falls inside the corresponding prediction interval:

$$\mathrm{PICP} = \frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} \mathbb{1}\left\{y_i \in \mathrm{PI}^{(\alpha)}\left(\mathbf{x}_i\right)\right\}. \tag{3.20}$$

For a well calibrated model, the fraction of observations whose true values fall within the predicted intervals should be close to the nominal coverage value of $(1-\alpha)\cdot 100\%$, as shown in Figure 3.2.

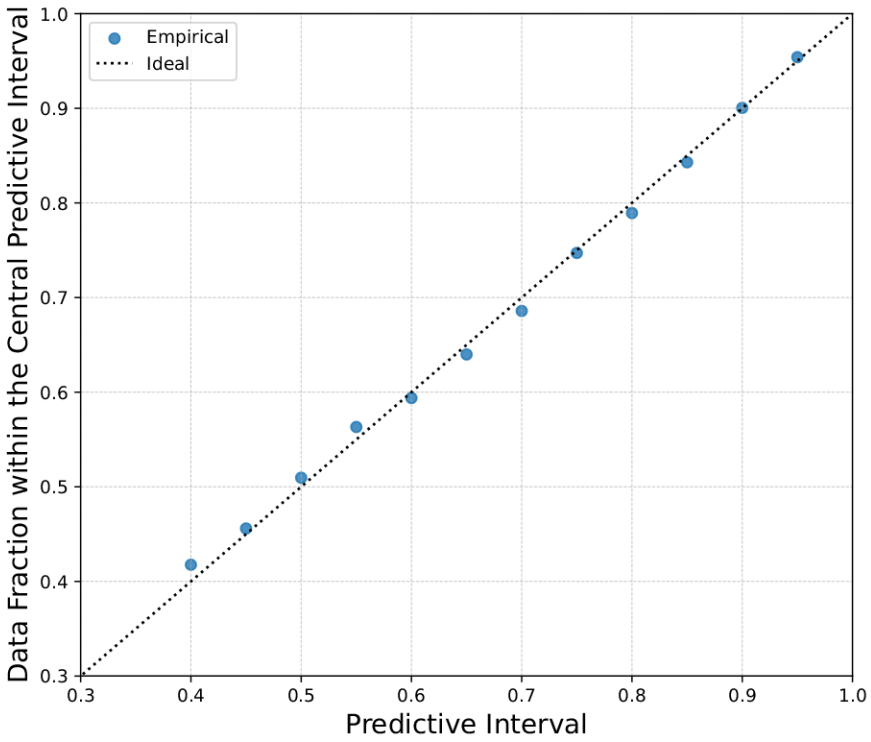


Figure 3.2: Representation of a well-calibrated model according to the PICP metric in Equation 3.20. Ideally, for perfect coverage, all empirical points should lie exactly on the diagonal line, although minor oscillations are expected due to finite sample effects.

Chapter 4

Diffusivity

In this chapter, we introduce the main subject of this work: solid-state electrolytes (see section 4.1). We begin by defining them and discussing their relevance in the context of energy storage devices. We then describe the mechanisms of ionic conductivity and its relationship with diffusivity (section 4.2), followed by a discussion about the general temperature dependence of diffusivity (section 4.3). Finally, in section 4.4, we present the descriptors selected to characterize diffusivity.

4.1 Solid-State Electrolytes

Energy storage has evolved into a priority area within global research, with governments and industry investing heavily to achieve net zero targets. The lithium-ion (Li^+) battery technology has been essential to this challenging task and it has been investigated for many years leading eventually to a major breakthrough: the invention of solid-state batteries

All-Solid-State-Batteries[56,57,58,59] (ASSBs) are promising new technologies that have the potential to revolutionize the way in which we store and use energy. Unlike traditional Li^+-ion batteries, which use a flammable liquid electrolyte to transfer ions between the electrodes, ASSBs use a *Solid State Electrolyte* (SSE), which offers several advantages over their liquid counterparts[60,61], such as an improved safety, a more environmentally friendly synthesis process and potentially higher energy density. Increased safety results from isolating the electrodes with the solid electrolyte, avoiding Li^+ build-up that can trigger short-circuiting. In addition, ASSBs have the potential to be recharged faster than conventional batteries in certain conditions and could be used in a wide range of applications, such as consumer electronics, electric vehicles and various industrial operations[62,63].

Sodium (Na^+) solid-state batteries[64] are also receiving notable attention for

safe and economical energy storage. Since sodium is much more abundant in nature than lithium, it makes for an attracting low-cost alternative, though generally offering lower energy densities than its lithium counterparts.

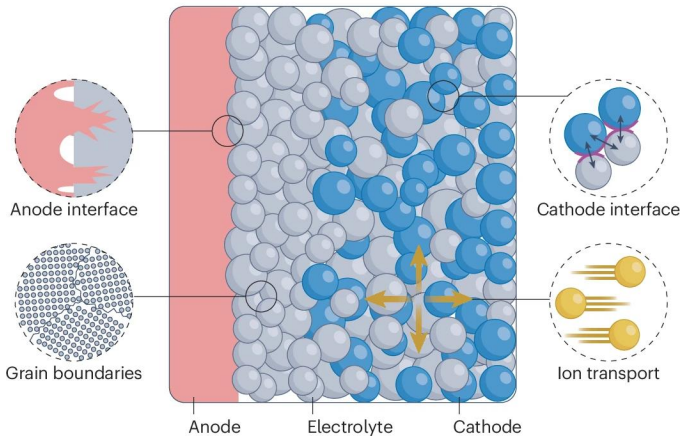


Figure 4.1: Representation of a solid-state battery, highlighting its main components and some of the major challenges affecting this technology. Picture adapted from[56].

Solid-State Batteries: Advantages & Challenges

A solid-state battery, schematically represented in Figure 4.1, has three main components: the metal anode (traditionally made of lithium metal with lithium alloy anodes or sodium metal anodes recently emerging as promising alternative candidates), the solid electrolyte and the cathode, composed by metal oxides, like Lithium Cobalt Oxide (LCO) or Lithium Nickel Cobalt Aluminum Oxide (NCA).

Traditional lithium batteries use liquid electrolytes, which allow the ions to be transported between the electrodes, and a separator layer, which enables ion transport while preventing electrical contact between the two electrodes. Instead, solid-state batteries use a solid-state electrolyte, whose particles are mixed with those of the cathode to form composite cathodes that serve both purposes: they help establishing efficient electron and ion conduction pathways, and they act as a separator layer.

In addition to the safety benefits of using a solid electrolyte in place of a combustible liquid electrolyte, solid-state batteries offer the potential advantage of using Li or Na metal anodes and high-voltage ($> 5V$) cathode materials to design high-energy-density batteries. Inorganic solid electrolytes can also support battery operation at a wide temperature range (for example, -50 °C to 200 °C

or more) in which conventional liquid electrolytes would freeze, boil or decompose.

Despite extensive efforts to realize the advantages of solid-state batteries, they have yet to achieve their full potential. In the following work we will focus exclusively on bulk electrolytes and attempt to derive a symbolic formula for bulk ionic conductivity. But the performance of solid-state batteries is also limited by several other critical factors related to the solid electrolyte and its interfaces[56]. Key issues include:

- **Dendrite Propagation**. Dendrites can grow out of the anode interface and propagate through the electrolyte separator, possibly coming into contact with the cathode and thus causing a short-circuit[65,66].

- **Crack Propagation**. High local current densities and mechanical stresses can force lithium (or sodium) metal into existing microcracks or grain boundaries in the solid electrolyte. This process can widen cracks and lead to the formation of metallic filaments that propagate through the electrolyte, eventually causing internal short circuits.

- **Interface Resistance**. The composition and structure of the electrolyte-electrode interface often deviate substantially from those of the bulk materials, leading to increased interfacial resistance. Chemo-mechanical challenges, such as the maintenance of physical contact between solid particles during electrochemical cycling and the formation of a stable and conductive interphase, are important to enable efficient ion diffusion across the interface.

- **Grain Boundaries**. The internal nanometre-scale to micrometre-scale interfaces that exist within crystalline solid electrolytes can cause issues because they tend to strongly influence the ionic conductivity. In particular, grain boundaries are known to act as defect sinks and can severely inhibit ionic conductivity in many solid electrolytes

Materials Modelling for Solid-State Electrolyte

SSEs are the key components of solid-state batteries. It is important to obtain SSE materials with high ionic conductivity ($\geq 10^{-3}\,\mathrm{S\ cm^{-1}}$)[67] and very low electronic conductivity ($< 10^{-10}\,\mathrm{S\ cm^{-1}}$), a wide electrochemical window (EW), i.e. the voltage range within which an electrolyte remains stable, neither oxidizing at the cathode nor reducing at the anode, good chemical compatibility with the relevant electrodes, excellent thermal stability and mechanical properties[68], and that are easy to manufacture on a large scale and at a low cost.

A broad range of SSE materials have been proposed and investigated for

both lithium and sodium batteries, including polymers[69,70] and inorganic materials like oxides[71], sulphides[72], and halides[73] (i.e. binary chemical compounds, of which one part is a halogen atom, like fluorine, chlorine or bromine, and the other is an element that is less electronegative than the halogen, in this case lithium or sodium). Structurally, SSEs are typically categorized as crystalline, amorphous (glassy), and glass-ceramic (partially crystalline), with many different compositions being recognized as superionic conductors ($\sigma > 10^{-4}$ S cm^{-1})[74].

In general, two main classes of atomistic simulation techniques have been used to study SSE materials: electronic structure methods using density functional theory (DFT) and ab initio molecular dynamics (AIMD); and interatomic-potential-based (or forcefield) methods, including classical Molecular Dynamics (MD). A rapidly growing addition to these conventional materials modelling methods is the use of Machine Learning (ML) techniques[75]. The general approach in the development of accurate ML potentials for materials modelling usually involves using diverse and numerous ab initio datasets (typically generated from high-level DFT calculations or based on available experimental structures) to train the effective ML model. This training procedure then enables MD simulations based on ML potentials with orders of magnitude larger cell sizes and longer timescales than current AIMD, while maintaining near quantum mechanical accuracy at reduced computational cost. These simulations generate valuable outputs on solid electrolyte properties, such as ion conduction mechanisms, stability trends and interfacial effects.

In addition to accelerate the discovery of new solid electrolyte materials, ML-based modelling methods have also been used to improve the fundamental understanding of the mechanisms which govern ionic transport, described in section 4.2.

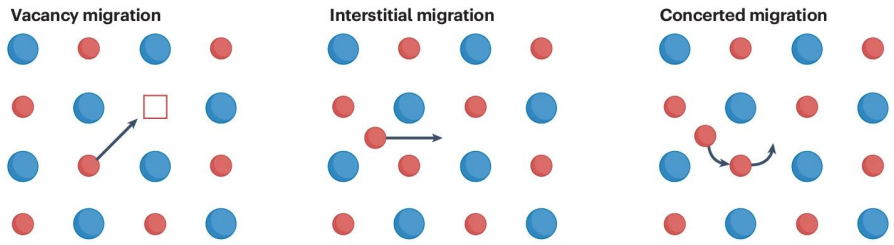


Figure 4.2: Schematic representations of the three principal ion migration mechanisms: vacancy migration, direct interstitial migration, and concerted or interstitialcy migration. Picture adapted from[56].

4.2 Ionic Conductivity

At the atomic scale (in the order of Å), the diffusion of mobile Li^+ or Na^+ cations within crystalline solids can be visualized as ion jumps or hops between ground-state sites and/or intermediate sites of the anionic sublattice, typically formed by anions such as O^{2-} or S^{2-}. The sites and their energies are mainly defined by their local ion coordination, that is, bonding environment, which in crystalline compounds tends to be tetrahedral or octahedral. Consequently, the migration pathway of an ion in a material is a function of the availability and interconnectivity of different sites as defined by the arrangement of anions[65].

The three principal migration mechanisms[76], schematically represented in Figure 4.2, are:

- **vacancy diffusion**, in which an ion migrates into a neighbouring vacant lattice site;
- **direct interstitial migration**, in which an ion hops between adjacent vacant interstitial sites;
- **concerted or interstitialcy (knock-on) mechanism**, where the migrating interstitial ion displaces a neighbouring lattice ion into the adjacent site.

Vacancy and direct interstitial migration are seen as conventional hopping processes, whereas concerted migration is a highly correlated process in which several ions move together in a concerted manner. Such concerted dynamics can give rise to high liquid-like ionic conductivity, a key feature for solid electrolytes in ASSBs.

A key descriptor of ionic transport is the *ionic conductivity* σ. Ionic conductivity can be expressed using linear response theory and the Green–Kubo formalism[77,78] as a time integral over a current-current autocorrelation function,

$$\sigma = \frac{1}{3Vk_BT} \int_0^\infty dt \ \langle \mathbf{J}(t) \cdot \mathbf{J}(0) \rangle \,, \tag{4.1}$$

where V is the volume, k_BT is the thermal energy, and $\mathbf{J}(t)$ is the total electric current density of the system at time t, which can be expressed as

$$\mathbf{J}(t) = e \sum_{i=1}^{N} z_i \mathbf{v}_i(t). \tag{4.2}$$

Here $\mathbf{v}_i(t)$ is the velocity of the centre of mass of the i-th ion with charge $e \cdot z_i$, and N is the total number of ions. The charge constants z_i can be treated as integers, even from an ab-initio perspective, and correspond to the atomic oxidation numbers[79].

By making the current dependence on the microscopic ionic motion explicit (Equation 4.2), the Green-Kubo formula, in Equation 4.1, can be decomposed into autocorrelation and cross-correlation terms[80]

$$\sigma = \frac{e^2}{3Vk_BT}\sum_i z_i^2 \underbrace{\int_0^\infty dt\ \langle \mathbf{v}_i(t)\cdot\mathbf{v}_i(0)\rangle}_{3\cdot D_i} + \frac{e^2}{3Vk_BT}\sum_{i\neq j} z_i z_j \int_0^\infty dt\ \langle \mathbf{v}_i(t)\cdot\mathbf{v}_j(0)\rangle\,. \tag{4.3}$$

Under the assumption of non-interacting ions, the cross-correlation terms vanish and thus Equation 4.3 can be approximated by the Nernst-Einstein (NE) equation

$$\sigma = \frac{e^2}{Vk_BT}\sum_k z_k^2 N_k D_k, \tag{4.4}$$

where the sum now runs over the different ionic species k and N_k identifies their number. Thus, within the NE approximation, the issue of computing σ is reduced to the determination of the *Self-Diffusion Coefficient* or *Diffusivity* D_k, obtained by integrating the velocity auto-correlation function of each ionic species.

4.3 The Arrhenius Relation

Unlike transport in liquids, where ions can propagate through a continuous medium, in solids they move via discrete jumps between fixed sites (interstitial positions, vacancies, or saddle points in a crystal lattice), following the mechanisms described in section 4.2. Initially trapped in a local potential energy minimum, in order to move to an adjacent site ions must overcome an energy barrier by acquiring sufficient thermal energy from the environment. The average height of this barrier is quantified by an effective *activation energy* E_a.

The self-diffusion coefficient D of type-A ions in a metal A, first presented in Equation 4.4, often follows the *Arrhenius relation*[81,82]

$$D\,(p,T) = D_0 \exp\left(-\frac{\Delta H_a}{k_BT}\right) = D_0 \exp\left(-\frac{\Delta \mathrm{U}_a + p\Delta V_a}{k_BT}\right), \tag{4.5}$$

where D_0 denotes a pre-exponential factor depending on system-specific descriptors, k_BT is the thermal energy, $\Delta H_a = \Delta \mathrm{U}_a + p\Delta V_a \equiv E_a$ is an activation enthalpy, i.e. the enthalpy difference between the transition state (saddle point) and the initial site. p is the pressure, $\Delta \mathrm{U}_a$ and ΔV_a are respectively the activation internal energy and activation volume.

The diffusivity generally depends on both temperature and pressure, but since the pressure term is much smaller than the thermal energy $p\Delta V_a \ll k_BT$, even at moderately high pressures, the pressure dependence is typically omitted

and Equation 4.5 is approximated as

$$D(T) = D_0 \exp\left(-\frac{E_a}{k_B T}\right), \tag{4.6}$$

where the activation energy E_a is approximated by the activation internal energy ΔU_a. This approximation is good under ambient conditions.

E_a and D_0 are the so-called activation parameters of diffusion, and their physical interpretation depends non-trivially on the diffusion mechanism, the type of diffusion process and the lattice geometry. The aim of the present work is to derive a robust yet interpretable symbolic expression of these two quantities in terms of descriptors, introduced in section 4.4.

4.4 Descriptors

Table 4.1 summarizes the selected descriptors used in the symbolic regression of the diffusivity D and the activation energy E_a, together with a brief description of their physical meaning.

Table 4.1: Descriptors used in the symbolic regression of the diffusivity D and the activation energy E_a. N_i is the number of atoms of species i, m is the mass, V is the volume, $\mathbf{R}_i$ is the atomic radius of atom i from the materials science Python library *pymatgen*[83], and X_i is the electronegativity of atom i. Charge neutrality was enforced by adjusting the number of anions based on the most typical oxidation states of the cations and anions in glass systems: $\{Li^+, B^{3+}, N^{3-}, O^{2-}, F^-, Na^+, Mg^{2+}, Al^{3+}, Si^{4+}, P^{5+}, S^{2-}, Cl^-, K^+, Ca^{2+}, Ge^{4+}, Br^-, I^-\}$. $\mathbf{d}_{\mathrm{Na}}(s)$ identifies the position vector of Na atoms, which depends on the lattice structure s. The polyhedral distortion Δ_{poly} is defined in Equation 4.7 and $\mathcal{V}_i$, defined in Equation 4.8, identifies the Voronoi cell.

Feature	Symbol	Formula
Average Atomic Volume	AAV	N/V
Volume Per Anion	VPA	N_{anion}/V
Na Bond Ionicity	NBI	$\sum_{i\neq \mathrm{Na}} N_i \left(1 - e^{-0.25(X_i - X_{Na})^2}\right)/N$
Set-of-non-Na-atoms Bond Ionicity	SBI	$\frac{\sum_{i<j,\, i,j\neq \mathrm{Na}} N_i N_j \left(1 - e^{-0.25(X_i - X_j)^2}\right)}{\sum_{i<j,\, i,j\neq \mathrm{Na}} N_i N_j}$
Ratio of NBI and SBI	RBI	NBI/SBI
Packing Fraction	PF	$\sum_{i=1} \frac{4}{3}\pi R_i^3 / V$
Density	$\boldsymbol{\rho}$	m/V

Continued on next page

Feature	Symbol	Formula
Weighted Electronegativity	WE	$\sum_i N_i X_i / N$
B atomic fraction	B%	N_{B} / N
N atomic fraction	N%	N_{N} / N
S atomic fraction	S%	N_{S} / N
O atomic fraction	O%	N_{O} / N
F atomic fraction	F%	N_{F} / N
Na atomic fraction	Na%	N_{Na} / N
Al atomic fraction	Al%	N_{Al} / N
Si atomic fraction	Si%	N_{Si} / N
P atomic fraction	P%	N_{P} / N
Cl atomic fraction	Cl%	N_{Cl} / N
Ge atomic fraction	Ge%	N_{Ge} / N
Br atomic fraction	Br%	N_{Br} / N
I atomic fraction	I%	N_{I} / N
Cubic Root of the Volume	cub-root	$V^{1/3}$
Minimum Na-Na interatomic distance	Na-Na	$\min_{i<j} \lvert \mathbf{d}_{\mathrm{Na},i} - \mathbf{d}_{\mathrm{Na},j} \rvert$
Maximum Na-Na interatomic distance	Na-Na-max	$\max_{i<j} \lvert \mathbf{d}_{\mathrm{Na},i} - \mathbf{d}_{\mathrm{Na},j} \rvert$
Average Na-Na interatomic distance	Na-Na-avg	$\frac{2}{N_{\mathrm{Na}}(N_{\mathrm{Na}}-1)} \sum_{i<j}^{N_{\mathrm{Na}}} \lvert \mathbf{d}_{\mathrm{Na},i} - \mathbf{d}_{\mathrm{Na},j} \rvert$
Minimum Bottleneck	$\min(R_{\mathrm{Bottleneck}})$	$\min_{\mathrm{s}} \left[\min_{i=\mathrm{O,S}} \left(\lvert \mathbf{d}_{\mathrm{Na}}(s) - \mathbf{R}_i \rvert - R_{\mathrm{Na}} \right) \right]$
Average Bottleneck	$\langle R_{\mathrm{Bottleneck}} \rangle$	$\langle \min_{i=\mathrm{O,S}} \left(\lvert \mathbf{d}_{\mathrm{Na}}(s) - \mathbf{R}_i \rvert - R_{\mathrm{Na}} \right) \rangle_{\mathrm{s}}$
Maximum Polyhedral Distortion	$\Delta_{\max}$	$\max_s \left(\Delta_{\mathrm{poly}}(s) \right)$
Standard Deviation of the Polyhedral Distortion	Δ_{std}	$\mathrm{std} \left(\Delta_{\mathrm{poly}}(s) \right)$
Average Polyhedral Distortion	Δ_{avg}	$\langle \Delta_{\mathrm{poly}}(s) \rangle_s$
Average Polyhedral Distortion with Fixed Structure except for O & S atoms	$\Delta_{\mathrm{avg,fix}}$	$\langle \Delta_{\mathrm{poly}}(s) \rangle_{\mathrm{O,S}}$
Maximum Face Area between two Voronoi Polyhedra	$fa_{\max}$	$\max_s \left[\mathrm{Area} \left(\mathcal{V}_i \cap \mathcal{V}_j \right) \right]$
Average Face Area between two Voronoi Polyhedra	fa_{avg}	$\langle \mathrm{Area} \left(\mathcal{V}_i \cap \mathcal{V}_j \right) \rangle_s$
Minimum Face Area between two Voronoi Polyhedra	$fa_{\min}$	$\min_s \left[\mathrm{Area} \left(\mathcal{V}_i \cap \mathcal{V}_j \right) \right]$

Continued on next page

Feature	Symbol	Formula
Number of Faces between two Voronoi Polyhedra containing a Na Atom	fa_{count}	$\#\left(\mathcal{V}_i \cap \mathcal{V}_j\right) \geq R_{\text{Na}}$
Pockets per Sodium Ion Pockets per Volume	$\text{pockets}_{\text{Na}}$ $\text{pockets}_{\text{V}}$	$\#\text{Pockets}/N_{\text{Na}}$ $\#\text{Pockets}/V$
Paths per Sodium Ion Paths per Volume	paths_{Na} paths_{V}	$\#\text{Paths}/N_{\text{Na}}$ $\#\text{Paths}/V$

The choice of the descriptors and the corresponding dataset are partially based on the work of R. Christensen & M. M. Smedskjaer[17]. Additional physical descriptors are included, as they play a crucial role in the microscopic mechanisms of ionic transport described in section 4.2. These newly-added descriptors include:

- **Na-Na**, i.e. the relative interatomic distance between Na atoms;
- **bottleneck**, i.e. the free distance available for a Na^+ ion to pass through the narrowest part of a diffusion pathway formed by surrounding framework atoms (typically O^{2-}, S^{2-});
- **pocket**, free-volume region in the crystal structure where a Na^+ ion can reside with relatively low energy, typically corresponding to a stable or metastable Na site;
- **polyhedral distortion**, i.e. the degree to which the bond lengths and bond angles of a coordination polyhedron differ from those of an ideal reference polyhedron, measured as the weighted sum

$$\Delta_{\text{poly}} = \frac{w_d}{N_{\text{bonds}}^{\text{Na}}} \sum_{k=1}^{N_{\text{bonds}}^{\text{Na}}} \frac{|d_k - d_0|}{d_0} + \frac{w_\theta}{M_{\text{Na}}} \sum_{j=1}^{M_{\text{Na}}} |\theta_j - \theta_0|, \tag{4.7}$$

  where d_k and θ_j are respectively the measured bond length and bond angle, d_0 and θ_0 are respectively the ideal bond length and bond angle, $N_{\text{bonds}}^{\text{Na}}$ is the number of Na bonds and M_{Na} the number of Na bond angles, w_d and w_θ are weighting parameters. The ionic conductivity is expected to be sensitive to symmetry breaking in the coordination environment, and thus to the degree of polyhedral distortion quantified by Δ_{poly};
- **area of the shared face** between two neighbouring Voronoi polyhedra corresponding to adjacent Na sites, representing the cross-sectional area available for Na^+ migration between those sites;

- **path**, i.e. connected sequence of pockets linked by Voronoi faces and constrained by bottlenecks, forming a continuous route that allows Na^+ ions to move through the crystal lattice.

Here a *Voronoi polyhedron* or cell is defined as the region of space closer to a given seed point, than to any other, generalizing the Wigner-Seitz construction to disordered structures. It is mathematically defined as:

$$\mathcal{V}_i = \left\{ \mathbf{x} \in \mathbb{R}^3 \mid ||\mathbf{x} - \mathbf{d}_{\mathrm{Na},i}(s)|| \leq ||\mathbf{x} - \mathbf{d}_{\mathrm{Na},j}(s)|| \; \forall j \neq i \right\}, \tag{4.8}$$

where $\mathbf{x} \in \mathbb{R}^3$ is a generic point in 3D space and $\mathbf{d}_{\mathrm{Na},i}$ identifies the position of the i-th seed (Na) site.

The activation energy E_a, introduced in section 4.3, was included among the descriptors for the diffusivity, even though E_a itself depends non-trivially on some of the other descriptors in Table 4.1. This choice is motivated by the desire to explicitly look for an Arrhenius-like structure, as in Equation 4.6, in the learned model.

Chapter 5

Numerical Experiments

In this section, we present the results of the current study. In section 5.1, we report the symbolic expressions obtained using the SISSO code, along with a post-hoc analysis of the results. In section 5.2, we compare these findings with those obtained using PySR. Finally, in section 5.3, we interpret the derived expressions in the context of the existing literature on the subject.

5.1 SISSO-obtained Models

While SR methods like SISSO and PySR can generate complex expressions, these models often lack a mechanism for uncertainty quantification. The aim of the current work is to devise a method for distilling robust yet interpretable symbolic expressions by combining state-of-the-art SR software packages with Bayesian post-hoc methods, described in chapter 3.

The current method was tested on SISSO[10,11] and the resulting symbolic expressions were confronted with those obtained using PySR[8,9]. Both are state-of-the-art SR codes, introduced in chapter 2. The dataset used for training the models includes sodium diffusivities for solid-state electrolyte candidates, which are crucial for understanding ionic conductivity in batteries, as described in chapter 4. This dataset incorporates a variety of physical descriptors, summarized in Table 4.1, and is based on the work of ref[17], which focuses on material properties relevant to energy storage applications.

The fitting procedure was divided in two stages to guide the SR process toward physically meaningful expressions. In the first stage, we fit the logarithmic diffusivity $\ln(D)$ using all descriptors in Table 4.1 and the activation energy E_a, introduced in section 4.3. This stage encourages the model to recover an Arrhenius-like dependence (see Equation 4.6), which is often observed in diffusion processes. The term $-E_a/(k_B T)$ explicitly accounts for temperature dependence, while the remaining symbolic expressions help define the pre-exponential

factor $\ln(D_0)$. In the second stage, we perform symbolic regression for E_a using the same set of descriptors. Combining the results of these two stages allows us to derive a robust symbolic expression for $\ln(D)$ in terms of the descriptors.

A couple of preliminary steps were also taken. First of all, it was decided to perform the fit on $\ln(D)$ instead of D. This allows the expected result to be more adherent to what SISSO naturally tends to produce, i.e. a linear composition of non-linear terms, rather than forcing it to reconstruct a complex exponential dependency. Then, all dimensional quantities in the dataset were standardized using the zeta-score. This transformation rescales each feature so that its mean is zero and its standard deviation is one and it is mathematically expressed as:

$$f \to \frac{f - \langle f \rangle}{\sigma(f)}, \tag{5.1}$$

where f is a given dimensional feature, $\langle f \rangle$ is its average and $\sigma(f)$ its standard deviation. Standardization removes units and scale dependencies, ensuring that all features are treated on equal footing and preventing any individual feature from disproportionately influencing the regression process, while retaining the original distributions. Moreover, when performing Bayesian post-hoc analysis, the Monte-Carlo sampling requires well-behaved input distributions to produce reliable results and standardizing the features ensures that. The main drawback is that standardized features generally lead to symbolic expressions which are different to those obtained with dimensional features, at least in scale.

After obtaining symbolic expressions from SISSO, we apply Bayesian post-hoc analysis to refine and validate these models. While SISSO provides a ranked list of symbolic expressions based on their performance, it does not quantify the uncertainty or robustness of the results. The Bayesian approach addresses this by assigning a normal distribution to each model, with uninformative priors based on the SISSO-estimated constants. A likelihood function is centered on the SISSO predictions, and we compute the posterior distribution using the Sequential Monte-Carlo method (described in section 3.3). This allows us to estimate not only the constants of the symbolic models but also their uncertainties, providing a more reliable and interpretable final model.

At the end of the SR process, SISSO produces an user-defined number of symbolic expressions, typically in the form of linear combinations of non-linear functions of the descriptors, ranked from most to least descriptive and with an estimate of the additive and multiplicative constants, without quantifying the uncertainty or robustness of the results. The Bayesian approach addresses this by assigning a normal distribution to each model, with uninformative priors based on the SISSO-estimated constants and a likelihood function centered on the SISSO predictions. An unnormalized posterior is thus computed by multiplying priors and likelihood and it is sampled using the PyMC[52] implementation of the Sequential Monte-Carlo method, described in section 3.3. This allows

us to estimate not only the constants of the symbolic models but also their uncertainties, providing a more reliable final model.

In addition to refining the constants, Bayesian model selection allows us to evaluate the robustness of the SISSO ranking. By analyzing the posterior distributions, we can assess which symbolic expressions are most likely to represent the true underlying physical relationships. This provides a more rigorous framework for model comparison.

Model Structure

By construction, SISSO tends to produce symbolic expressions in the form of linear combinations of non-linear terms, as described in section 2.2. SISSO has two user-defined parameters that directly determine the structure of the resulting expressions:

- **rung**, which controls the maximum height of the binary tree representation of the produced symbolic expressions;
- **descriptor dimension**, abbreviated as **desc_dim**, which determines the maximum number of generated features linearly combined in the final expression and consequently the number of coefficients involved.

For example, setting rung $= 2$ restricts all generated expressions to trees of height 2. Expressions with low rung correspond to shallow trees and are therefore generally simpler and more interpretable than those with higher rung. Similarly, setting desc_dim $= 3$ and rung $= n$ yields expressions that are linear combinations of at most three generated features, each with a maximum rung of n.

The best SISSO-derived models for different combinations of these parameters are reported in Table 5.1 and Table 5.2. It can be observed that, when $\ln(D)$ is used as the target, the Root Mean Square Error (RMSE) generally decreases with increasing values of desc_dim. This trend suggests that the logarithmic target may be better approximated by linear combinations of multiple features. In contrast, when E_a is used as the target, lower RMSE values are obtained for higher values of rung. This indicates that more complex, non-linear feature constructions may be required to accurately model this quantity.

Different combinations of rung and desc_dim lead to different symbolic expressions. This introduces a heuristic component in the selection of the these parameters and thus in the final form of the symbolic expressions, as is common in many symbolic regression approaches. To promote interpretability and reduce overfitting, only combinations of low values of rung and desc_dim were considered.

Model Equation	Rung	Desc_dim	RMSE test
$-\lvert c_0\rvert - \lvert a_0\rvert \cdot \left[(\mathrm{O\%} \cdot \mathrm{WE}) - (fa_{\mathrm{avg}} - E_T)\right]$	2	1	0.409
$-\lvert c_0\rvert + \lvert a_0\rvert \cdot \left[(\mathrm{Cl\%} + \mathrm{P\%}) - (\mathrm{Al\%} - \mathrm{B\%})\right] + - \lvert a_1\rvert \cdot \left[(\mathrm{O\%/PF}) - (fa_{\mathrm{avg}} - E_T)\right]$	2	2	0.292
$-\lvert c_0\rvert - \lvert a_0\rvert \cdot (\mathrm{Al\%} - \mathrm{B\%}) - \lvert a_1\rvert \cdot (\mathrm{O\%} - \mathrm{NBI}) + \lvert a_2\rvert \cdot (fa_{\mathrm{avg}} - E_T)$	1	3	0.291
Equation 5.2	1	4	0.273

Table 5.1: Best SISSO-obtained symbolic expressions for different values of rung and desc_dim using $\ln(D)$ as target.

Model Equation	Rung	Desc_dim	RMSE test
$\lvert c_0\rvert - \lvert a_0\rvert \cdot \left[(\mathrm{WE/RBI}) + (fa_{\mathrm{avg}} + \mathrm{B\%})\right]$	2	1	0.701
Equation 5.3	2	2	0.634
$\lvert c_0\rvert + \lvert a_0\rvert \cdot (\mathrm{Ge\%} \cdot \mathrm{S\%}) + + \lvert a_1\rvert \cdot (\Delta_{\mathrm{avg,fix}} - \mathrm{SBI}) - \lvert a_2\rvert \cdot (\mathrm{paths_V} + fa_{\mathrm{count}})$	1	3	0.661
$\lvert c_0\rvert - \lvert a_0\rvert \cdot (fa_{\mathrm{avg}} \cdot \mathrm{AAV}) + \lvert a_1\rvert \cdot (\mathrm{Ge\%} \cdot \mathrm{S\%}) + + \lvert a_2\rvert \cdot (\Delta_{\mathrm{avg,fix}} - \mathrm{SBI}) - \lvert a_3\rvert \cdot (\mathrm{pockets_V} + fa_{\mathrm{count}})$	1	4	0.645

Table 5.2: Best SISSO-obtained symbolic expressions for different values of rung and desc_dim using E_a as target.

Best Model

Among all candidate models obtained from different parameter combinations, the best model was selected as the one yielding the lowest RMSE on the test set.

According to this criterion, the best SISSO-derived symbolic expression for $\ln(D)$ is:

$$c_0 + a_0 \cdot (\mathrm{P\%} \cdot \mathrm{S\%}) + a_1 \cdot (\mathrm{Al\%} - \mathrm{B\%}) + a_2 \cdot (\mathrm{O\%} - \mathrm{NBI}) + a_3 \cdot \left(fa_{\mathrm{avg}} - \frac{E_a}{k_B T}\right), \tag{5.2}$$

where the parameter values and their uncertainties are reported in Table 5.3, and the descriptors are listed in Table 4.1. Notably, this expression is consistent with the Arrhenius law (see Equation 4.6). The presence of the term $E_a/(k_B T)$ indicates that SISSO can recover a functional dependence consistent with Arrhenius-type behaviour, lending physical interpretability to the model but only if $k_B T$ is explicitly included in the features.

Parameter	Value with Uncertainty
c_0	-12.43 ± 0.06
a_0	5.3 ± 0.3
a_1	-1.69 ± 0.07
a_2	-2.43 ± 0.07
a_3	0.668 ± 0.004

Table 5.3: Values and uncertainties of the parameters of Equation 5.2 obtained through a post-hoc analysis of the SISSO results using $\ln(D)$ as target.

Repeating the same procedure using E_a as the target yields:

$$\begin{aligned} c_0 + a_0 \cdot [(\text{Ge\%} \cdot \text{S\%}) + (\text{P\%/WE})] + \\ + \ a_1 \cdot \left[(\text{O\%} - \text{NBI}) \cdot (fa_{\text{avg}} \cdot \text{NBI})\right], \end{aligned} \tag{5.3}$$

whose parameter values are reported in Table 5.4.

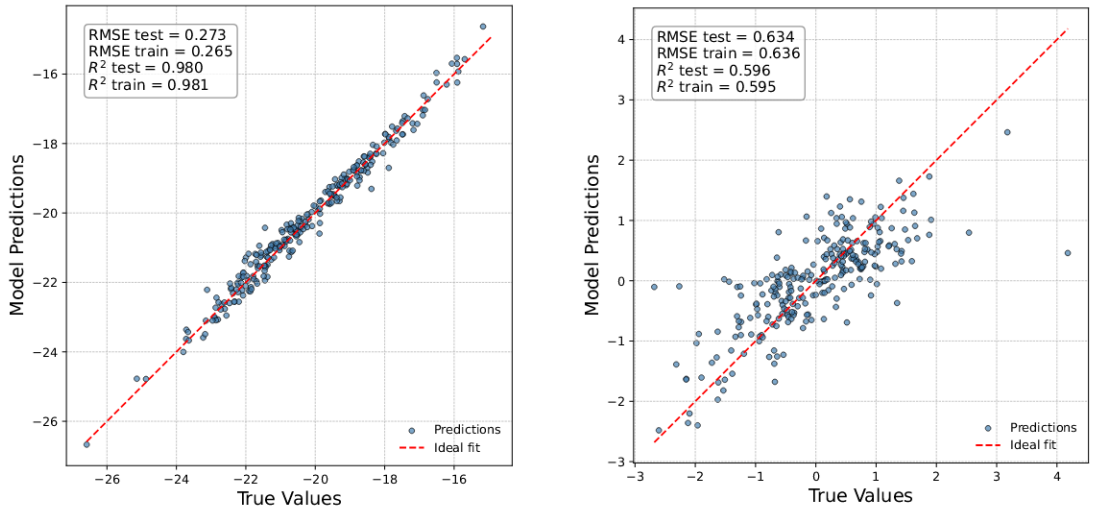
Parameter	Value with Uncertainty
c_0	-0.11 ± 0.02
a_0	9.4 ± 0.6
a_1	4.6 ± 0.1

Table 5.4: Values and uncertainties of the parameters of Equation 5.3 obtained through a post-hoc analysis of the SISSO results using E_a as target.

The parity plots between the two target formulations are shown in Figure 5.1. The model in Equation 5.2 provides a reasonably accurate fit to the sodium diffusivity test data, whereas Equation 5.3 shows significantly poorer agreement with the E_a test values.

In the former case, the residual discrepancies can be attributed to the inherent trade-off in symbolic regression, where interpretability is often favoured over predictive accuracy. Moreover, the underlying functional form is partially known (Arrhenius behaviour shown in Equation 4.6), constraining the search space and facilitating model discovery. The near-identical training and test metrics ($R^2 \approx 0.98$) indicate strong generalization and absence of overfitting.

In contrast, modelling E_a is more challenging, as the functional relationship between target and descriptors is completely unknown. The poor observed performance may reflect missing relevant descriptors or limitations of the search algorithm in capturing complex non-linear dependencies, thus producing models lacking in complexity. Consequently, the derived expression can be interpreted as an approximate, simplified representation of a potentially more complex underlying relationship.



(a) (b)

Figure 5.1: Parity plots of the best SISSO-obtained symbolic expressions having $\ln(D)$ as target (Equation 5.2) in (a) and E_a as target (Equation 5.3) in (b).

Tempering

After the SR procedure, SISSO produces a set of symbolic expressions, ranked according to the Pearson correlation coefficient between predictions and ground truth values. This ranking is subsequently validated by comparing the marginal likelihoods of the derived models via the Bayes factors (defined in Equation 3.12), providing a probabilistic measure of relative model evidence. The model with the highest marginal likelihood is assigned the top rank, with the others ordered accordingly. The marginal likelihood-based ranking broadly mirrors the original SISSO ranking, with near-perfect agreement at the top and bottom positions and some reshuffling in the middle.

A tempering procedure, as described in section 3.3, is then employed to test the robustness of this ranking for different likelihood temperatures. This is achieved by introducing a weighting factor, defined in Equation 3.7, into the likelihood. Lower values of β correspond to a flattened likelihood, whereas higher values emphasize sharper distinctions between model predictions. Overall, the results indicate that the relative ranking of the two target formulations remain stable across the explored range of β. However, a notable difference emerges between the two targets: for models with $\ln(D)$ as the target, the marginal likelihoods spans several orders of magnitude, whereas for models with E_a as the target, the variation is considerably smaller. This suggests that, in the latter case, the descriptive differences between models are limited, while in the former they are more pronounced.

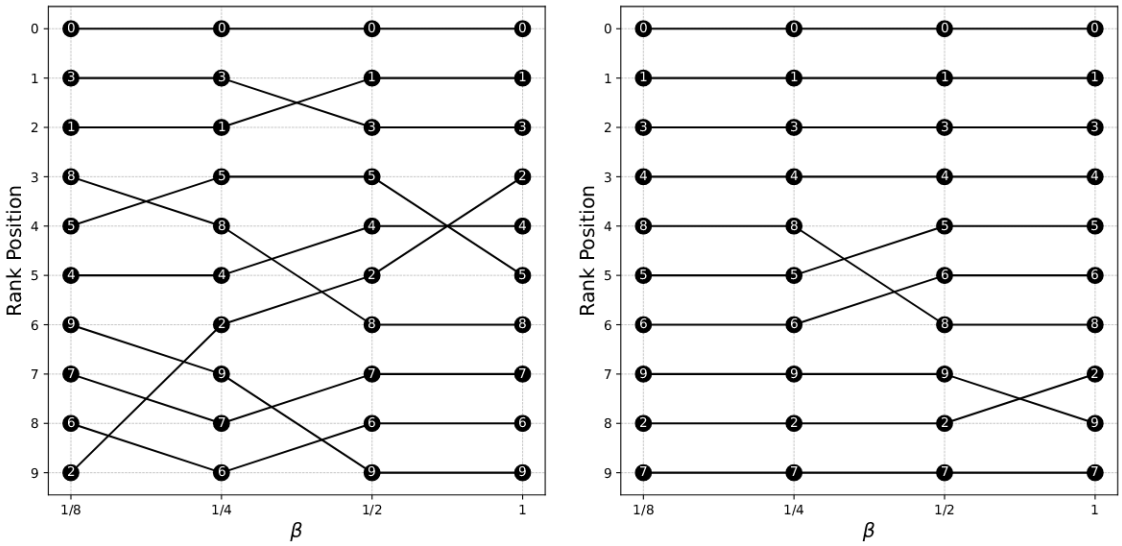


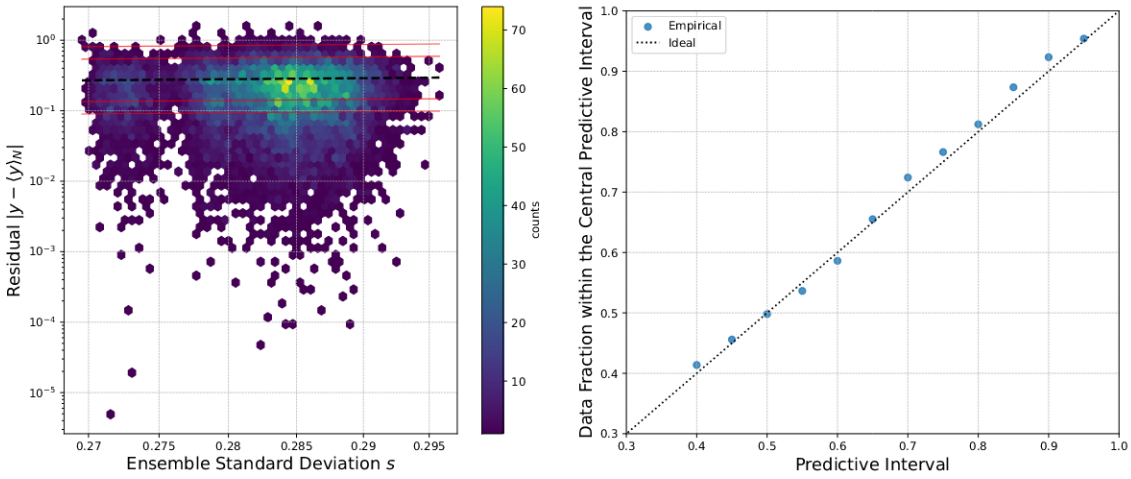
Figure 5.2: Rank plots of the 10 best SISSO-obtained symbolic expressions for different temperature parameter values β, using $\ln(D)$ as target in (a) and E_a as target in in (b). The number in the marker identifies the position of the given model in the original SISSO ranking, with the 0-th being the best model and the 9-th the worst.

Calibration

Two complementary calibration methods are employed to assess whether the uncertainty of the model estimates reflect the true prediction error. The first method is the ensemble calibration, which checks whether the ensemble variance correctly reflects the actual prediction error, i.e. the mean-squared-error between predictions and ground truth values. The second is the Prediction Interval Coverage Probability (PICP) or within-α method, defined in Equation 3.20, which checks whether prediction intervals have the correct coverage, by measuring how often the true values fall within the predicted intervals. Both methods are described in section 3.3.

Calibration results are shown in Figure 5.3 and Figure 5.4 for the models trained on $\ln(D)$ and E_a, respectively. The density plots depict the relationship between residual magnitude and the ensemble standard deviation s, i.e. the square roots of Equation 3.14 and Equation 3.13, respectively).

Both plots exhibit a mild upward trend: as s increases, the residuals tend to increase as well. This indicates that higher predicted uncertainty corresponds, on average, to larger prediction errors. However, the correlation remains weak, and the spread of residuals at fixed s is large (ranging from 10^{-5} to 1) while the



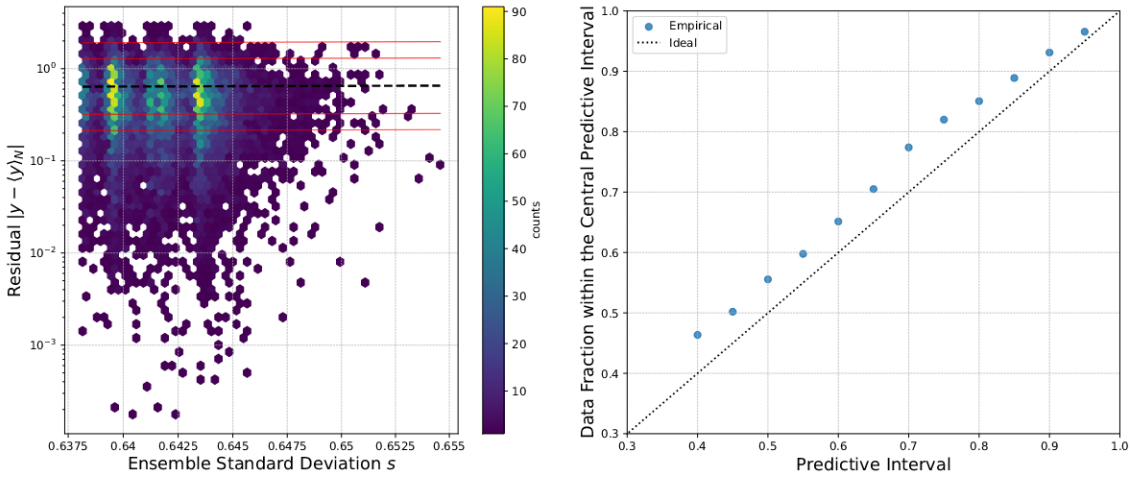
(a) (b)

Figure 5.3: Calibration plots for the best model obtained with $\ln(D)$ as target (Equation 5.2). In (a) we have a density plot illustrating the relationship between the residual magnitude and the ensemble standard deviation s. Hexagonal bins are employed to provide a clear view of data concentration in regions with many overlapping points. The dashed black line identifies the ideal behaviour $|y - \langle y \rangle_N| = s$, while the red lines mark $\pm 2s$ and $\pm 3s$. In (b) the PICP result is depicted, confronting the empirical coverage with the nominal predictive interval.

predicted uncertainty spans a relatively narrow range. This compression limits the discriminative power of the model, i.e. its ability to differentiate between low- and high-error predictions.

The highest density region sits around the bisector line suggesting that, on average, the residuals are consistent with the predicted uncertainty, confirming Equation 3.17. At the same time, the presence of large residuals across all uncertainty levels points to either a heavy-tailed error distribution or model misspecification, indicating that extreme errors are not well captured. Finally, compared to the $\ln(D)$ model, the E_a model exhibits a more diffuse distribution and weaker alignment between uncertainty and error, consistent with its poorer calibration behaviour.

The coverage plots compare nominal and empirical probabilities. For Equation 5.2, the empirical coverage closely matches the nominal predictive interval, with only minor deviations, indicating good calibration. This confirms that, on average, the predictive uncertainty provides an accurate quantification of the expected error. In contrast, Equation 5.3 shows a tendency toward underconfidence, as the empirical coverage systematically exceeds the nominal level for



(a) (b)

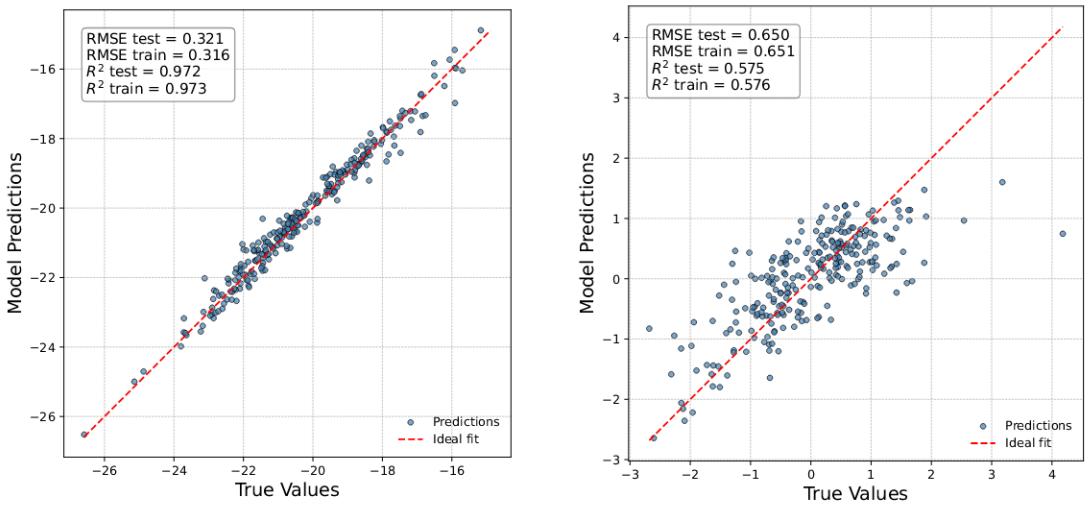
Figure 5.4: Calibration plots for the best model obtained with E_a as target (Equation 5.3). In (a) we have a density plot illustrating the relationship between the residual magnitude and the ensemble standard deviation s. Hexagonal bins are employed to provide a clear view of data concentration in regions with many overlapping points. The dashed black line identifies the ideal behaviour $|y - \langle y \rangle_N| = s$, while the red lines mark $\pm 2s$ and $\pm 3s$. In (b) the PICP result is depicted, confronting the empirical coverage with the nominal predictive interval.

several predictive intervals, consistent with the parity plot in Figure 5.1.

Despite a good global calibration, both models exhibit limited local informativeness. As highlighted by the density plots, the substantial spread of errors at fixed uncertainty levels indicates weak discriminative power. In other words, while the models provide reliable uncertainty estimates on average, they are less effective at identifying which individual predictions are likely to incur in large errors.

5.2 Comparison with PySR

The symbolic expressions obtained using SISSO were compared with those produced by a different state-of-the-art practical SR code: PySR[8,9], described in section 2.3. A direct comparison between the two results is challenging because SISSO and PySR handle model complexity differently. In SISSO, the complexity of the generated expressions can be directly controlled through feature selection, while PySR strikes a balance between complexity and goodness of fit, without any explicit user-control on the complexity.



(a) (b)

Figure 5.5: Parity plots of the best PySR-obtained symbolic expressions having $\ln(D)$ as target (Equation 5.4) in (a) and E_a as target (Equation 5.5) in (b).

To facilitate comparison, we aimed to derive the best possible PySR expressions with a similar number of features as those Equation 5.2 and Equation 5.3. With this ad-hoc definition of complexity, the best PySR-derived model having $\ln(D)$ as target is

$$-\frac{\mathrm{Al}\%}{\mathrm{NBI}} + \mathrm{B}\% + \mathrm{Cl}\% - 24.711 \cdot E_a - 1.789 \cdot \mathrm{O}\% + \\ + 1.203 \cdot fa_{\mathrm{avg}} - 14.992, \tag{5.4}$$

where the non-standardized descriptors are defined in Table 4.1. The derived model respects the Arrhenius law (Equation 4.6), and by explicitly substituting k_B and T, the multiplicative coefficient of the exponent is ~ 0.639, comparable to the corresponding parameter (a_3) obtained with SISSO (Table 5.3).

A key advantage of PySR is its ability to determine multiplicative constants. Although the PySR expression has a higher RMSE than the SISSO model in Equation 5.2, it accurately approximates the thermal energy term $k_B T$ without explicitly including it as a feature. This behaviour suggests that PySR can implicitly recover physically meaningful relationships, even when they are not explicitly encoded in the descriptor set. In applications where interpretability and physical consistency are prioritised, such behaviour may justify a moderate reduction in predictive accuracy.

PySR can also generate more flexible expressions, which is useful for modelling complex or non-linear relationships, such as those involving E_a. The best

PySR-derived model for a standardized E_a is

$$-fa_{\text{avg}} + \frac{\text{VPA}}{\text{RBI}} + \frac{\text{F\%} + \text{Ge\%} - \text{SBI} + \Delta_{\text{avg}}}{\text{PF}}. \tag{5.5}$$

It can also be interesting to look at the result for a non-standardized E_a,

$$-0.204 \cdot \text{SBI} - 0.00530 \cdot \frac{fa_{\text{count}}}{-\text{B\%} + \Delta_{\text{avg}} + 1.14} + 0.508, \tag{5.6}$$

the involved descriptors are defined in Table 4.1. The parity plot of Equation 5.4 and Equation 5.5 are shown in Figure 5.5. Consistent with the SISSO results, the $\ln(D)$ model in Equation 5.4 provides a satisfactory description of the test data ($R^2 \approx 0.972$), reflecting the known functional dependence of $\ln(D)$ on temperature, without overfitting. In contrast, the model in Equation 5.5 underfits the E_a test values, suggesting that the available descriptors or functional forms may be insufficient to capture the underlying behaviour of the activation energy.

However, a limitation of PySR is that it produces only a single model, whereas SISSO generates a ranking of models. Therefore, PySR does not enable a direct Bayesian uncertainty quantification as SISSO does.

5.3 Physical Insights

Both SISSO and PySR are able to derive models that capture physically meaningful relationships in the system. As previously noted, both Equation 5.2 and Equation 5.4 exhibit Arrhenius-like behaviours, as expected for this kind of systems. The multiplicative factor of 1 is systematically underestimated (~ 0.65), which may be attributed to uncertainty in the determination of E_a in the original dataset. The models correctly capture a positive dependence of $\ln(D)$ on the Voronoi face areas and on the S atomic fraction, as well as a negative dependence on the O fraction. This trend can be rationalized by the higher electronegativity and, consequently, denser packing of O relative to S, which hinders ionic transport by creating a tighter bottleneck.

The presence of atomic species other than O and S appears to be less significant in itself and can instead be interpreted as a broader dependence on classes of materials. Exchanging elements within the group of network formers/intermediates $\{\text{B}, \text{Al}, \text{P}, \text{Si}, \text{Ge}\}$, or within the set of anions $\{\text{O}, \text{S}, \text{N}, \text{F}, \text{Cl}, \text{Br}, \text{I}\}$, results only in marginal variations in model performance (with the partial exception of O, which leads to slightly poorer results). The selection of specific elements is therefore likely attributable to local minima in the search space explored by the algorithm rather than to a fundamental physical distinction.

The symbolic expressions obtained for E_a are more complex and not straightforward to compare directly, due to the distinct approaches of SISSO and PySR

and the lack of an established functional form. Nevertheless, some general trends can be observed across both Equation 5.3 and Equation 5.5. In particular, a more connected glass network, with higher O content and stronger formers such as Ge and P, generally increases E_a, as ions encounter higher barriers to hopping. Conversely, features that promote percolation pathways, such as larger Voronoi face areas, can significantly reduce E_a by facilitating ion transport. Thus, E_a reflects a balance between global network rigidity, which hinders ion motion, and local structural features that facilitate it.

Chapter 6

Conclusions

In conclusion, this work demonstrates that combining symbolic regression with Bayesian post-hoc analysis can yield physically interpretable results. SISSO is able to recover expressions that are not only accurate but also consistent with known Arrhenius behaviour and the inclusion of Bayesian inference significantly enhances this approach by quantifying parameter uncertainties, validating model rankings, and offering a probabilistic basis for model comparison, features that are inherently absent in standard symbolic regression outputs.

The results highlight a clear distinction between modelling the logarithmic diffusivity $\ln(D)$ and the activation energy E_a. In the former case, the presence of an underlying physical law for the target guides the regression process and effectively constrains the search space, leading to well-performing models with strong generalization, good calibration, and meaningful uncertainty estimates. In contrast, the lack of an established functional form for E_a makes its prediction substantially more challenging, resulting in simpler models with reduced predictive power and weaker calibration. This suggests that the success of symbolic regression is strongly influenced by the degree of prior physical knowledge embedded in the problem.

At the same time, the inherently heuristic nature of symbolic regression in general must be acknowledged. The construction of the feature space, based on user-defined mathematical operations, and the dependence on user-defined hyperparameters such as rung and descriptor dimension, introduce a non negligible bias in the resulting symbolic expressions, hindering their reliability. In practice, this means that the space of candidate models is strongly shaped by prior design choices, which may favour certain functional forms over others. As a consequence, different but equally plausible configurations can lead to structurally different models with comparable predictive performance, limiting the uniqueness and reproducibility of the discovered expressions. This issue is further amplified by the trade-off between interpretability and predictive power: restricting the search to low-complexity models improves transparency but may

exclude more accurate representations of the underlying physics.

Furthermore, the Bayesian post-hoc analysis itself is not immune to model misspecification, which must be carefully considered when interpreting uncertainty estimates and model evidence. As a result, the inferred uncertainties primarily reflect parameter uncertainty within a fixed model structure, rather than the full epistemic uncertainty associated with model inadequacy and lack of descriptors.

The comparison with PySR further reinforces these observations. While SISSO is limited by its inability to directly determine hidden composite constants (like $k_B T$) unless explicitly encoded in the feature space, and by its restriction to linear combinations of non-linear terms, PySR demonstrates a remarkable ability to implicitly recover physically meaningful relationships and to generate more flexible symbolic expressions. However, this flexibility comes at the cost of reduced complexity control and, crucially, the absence of a model ensemble, which prevents a straightforward application of Bayesian uncertainty quantification.

Overall, the proposed methodology addresses a key limitation of symbolic regression by bridging the gap between symbolic model discovery and uncertainty quantification. Although challenges remain – particularly in capturing complex, poorly understood relationships, improving the local informativeness of uncertainty estimates, and more broadly assessing the reliability of the derived symbolic expressions – the approach represents a significant step toward more reliable interpretable and physically grounded data-driven modelling in materials science.

Data Availability

The data supporting the findings of this study are openly available on GitHub at https://github.com/PDugoni/Symbolic-regression.git.

Bibliography

[1] Max Planck. *Über eine Verbesserung der Wien'schen Spectralgleichung*. Friedr. Vieweg & Sohn, 1900. doi:https://doi.org/10.1007/978-3-663-13885-3_15.

[2] Gustau Camps-Valls, Andreas Gerhardus, Urmi Ninad, Gherardo Varando, Georg Martius, Emili Balaguer-Ballester, Ricardo Vinuesa, Emiliano Diaz, Laure Zanna, and Jakob Runge. Discovering causal relations and equations from data. *Physics Reports*, 1044:1–68, 2023. doi:https://doi.org/10.1016/j.physrep.2023.10.005.

[3] William La Cava, Patryk Orzechowski, Bogdan Burlacu, Fabricio De França, Marco Virgolin, Ying Jin, Michael Kommenda, and Jason Moore. Contemporary symbolic regression methods and their relative performance. *Adv. Neural Inf. Process Syst. 2021*, 07 2021. doi:10.48550/arXiv.2107.14351.

[4] Feiyu Xu, Hans Uszkoreit, Yangzhou Du, Wei Fan, Dongyan Zhao, and Jun Zhu. *Explainable AI: A brief survey on history, research areas, approaches and challenges*. Springer, 2019. doi:10.1007/978-3-030-32236-6_51.

[5] Guilherme Seidyo Imai Aldeia and Fabrício Olivetti de França. Measuring feature importance of symbolic regression models using partial effects. In *Proceedings of the genetic and evolutionary computation conference*, pages 750–758, 2021. doi:10.1145/3449639.3459302.

[6] Marco Virgolin and Solon P. Pissis. Symbolic regression is NP-hard. *Trans. Mach. Learn. Res.*, 2022, 2022. URL https://arxiv.org/abs/2207.01018.

[7] Michael Schmidt and Hod Lipson. Distilling free-form natural laws from experimental data. *Science*, 324(5923):81–85, 2009. URL https://www.science.org/doi/abs/10.1126/science.1165893.

[8] Miles Cranmer. Interpretable machine learning for science with PySR and SymbolicRegression.jl. 2023. URL https://arxiv.org/abs/2305.01582.

[9] Miles Cranmer. PySR: Fast & parallelized symbolic regression in Python/Julia, 2020. URL http://doi.org/10.5281/zenodo.4041459.

[10] Runhai Ouyang, Stefano Curtarolo, Emre Ahmetcik, Matthias Scheffler, and Luca M. Ghiringhelli. SISSO: A compressed-sensing method for identifying the best low-dimensional descriptor in an immensity of offered candidates. *Phys. Rev. Mater.*, 2:083802, 2018. doi:10.1103/PhysRevMaterials.2.083802.

[11] Thomas A. R. Purcell, Matthias Scheffler, and Luca M. Ghiringhelli. Recent advances in the SISSO method and their implementation in the SISSO++ code. *The Journal of Chemical Physics*, 159(11):114110, 2023. doi:https://doi.org/10.1063/5.0156620.

[12] John R. Koza. Genetic programming as a means for programming computers by natural selection. *Stat Comput*, 4:87–112, 1994. doi:https://doi.org/10.1007/BF00175355.

[13] Marco Virgolin, Eric Medvet, Tanja Alderliesten, and Peter A. N. Bosman. Less is more: A call to focus on simpler models in genetic programming for interpretable machine learning. 2022. URL https://arxiv.org/abs/2204.02046.

[14] Geoffrey Bomarito and Patrick Leser. Bayesian symbolic regression via posterior sampling. *Philosophical Transactions Of the Royal Society A*, 2025. URL https://arxiv.org/abs/2512.10849.

[15] Roger Guimera and Marta Sales-Pardo. Bayesian symbolic regression: Automated equation discovery from a physicists' perspective. *arXiv preprint arXiv:2507.19540*, 2025. URL https://arxiv.org/abs/2507.19540.

[16] Roger Guimerà, Ignasi Reichardt, Antoni Aguilar-Mogas, Francesco A. Massucci, Manuel Miranda, Jordi Pallarès, and Marta Sales-Pardo. A bayesian machine scientist to aid in the solution of challenging scientific problems. *Science Advances*, 6(5), 2020. doi:10.1126/sciadv.aav6971.

[17] Rasmus Christensen and Morten M. Smedskjaer. Accelerating the discovery of high-conductivity glass electrolytes via machine learning. *Advanced Energy Materials*, 16(10):e03813, 2026. doi:https://doi.org/10.1002/aenm.202503813.

[18] John H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975. doi:https://doi.org/10.7551/mitpress/1090.001.0001.

[19] Anne F. Brindle. Genetic algorithms for function optimization. 1980. URL https://archive.org/details/Brindle1980/page/n7/mode/2up.

[20] David E. Goldberg and Kalyanmoy Deb. A comparative analysis of selection schemes used in genetic algorithms. In *Gregory J.E. Rawlins*, volume 1 of *Foundations of Genetic Algorithms*, pages 69–93. Elsevier, 1991. doi:https://doi.org/10.1016/B978-0-08-050684-5.50008-2.

[21] Ulrich Bodenhofer. Genetic algorithms: Theory and applications. 2002. URL https://api.semanticscholar.org/CorpusID:265814436.

[22] Luca M. Ghiringhelli, Jan Vybiral, Emre Ahmetcik, Runhai Ouyang, Sergey V Levchenko, Claudia Draxl, and Matthias Scheffler. Learning physical descriptors for materials science by compressed sensing. *New Journal of Physics*, 19(2):023017, 2017. doi:10.1088/1367-2630/aa57bf.

[23] Luca M. Ghiringhelli, Jan Vybiral, Sergey V. Levchenko, Claudia Draxl, and Matthias Scheffler. Big data of materials science: Critical role of the descriptor. *Phys. Rev. Lett.*, 114:105503, 2015. doi:10.1103/PhysRevLett.114.105503.

[24] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach.* Cambridge University Press., 2009. doi:https://doi.org/10.1017/CBO9780511804090.

[25] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the royal statistical society series b-methodological*, 58:267–288, 1996. doi:https://doi.org/10.1111/j.2517-6161.1996.tb02080.x.

[26] Jerome Friedman Trevor Hastie, Robert Tibshirani. *The Elements of Statistical Learning*. Springer New York, NY, 2009. doi:https://doi.org/10.1007/978-0-387-84858-7.

[27] Mark A. Davenport, Marco F. Duarte, Yonina C. Eldar, and Gitta Kutyniok. *Compressed Sensing: Theory and Applications*. Cambridge University Press, 2012. doi:10.1017/CBO9780511794308.

[28] Gitta Kutyniok, Holger Boche, Robert Calderbank, and Jan Vybiral. *Compressed Sensing and its Applications: MATHEON Workshop 2013*. Birkhäuser Cham, 07 2015. doi:10.1007/978-3-319-16042-9.

[29] David L. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4):1289–1306, 2006. doi:10.1109/TIT.2006.871582.

[30] Albert Cohen, Wolfgang Dahmen, and Ronald DeVore. Compressed sensing and best k-term approximation. *American Mathematical Society*, 22:211–231, 01 2009. doi:10.1090/S0894-0347-08-00610-3.

[31] Akhil S. Nair, Lucas Foppa, and Matthias Scheffler. Materials-discovery workflows guided by symbolic regression: Identifying acid-stable oxides for electrocatalysis. *npj Comput Mater*, 2024. doi:https://doi.org/10.1038/s41524-025-01596-4.

[32] Christopher J. Bartel, Christopher Sutton, Bryan R. Goldsmith, Runhai Ouyang, Charles B. Musgrave, Luca M. Ghiringhelli, and Matthias Scheffler. New tolerance factor to predict the stability of perovskite oxides and halides. *Science advances*, 5(2), 2019. doi:10.1126/sciadv.aav0693.

[33] Han Zhong-Kang, Debalaya Sarker, Runhai Ouyang, Aliaksei Mazheika, Yi Gao, and Sergey Levchenko. Single-atom alloy catalysts designed by first-principles calculations and artificial intelligence. *Nature Communications*, 12, 03 2021. doi:10.1038/s41467-021-22048-9.

[34] Philip Breen. Algorithms for sparse approximation. *School of Mathematics, University of Edinburgh*, 4, 2009.

[35] Jianqing Fan, Richard Samworth, and Yichao Wu. Ultrahigh dimensional feature selection: Beyond the linear model. *Journal of machine learning research : JMLR*, 10:2013–2038, 09 2009. doi:10.1145/1577069.1755853.

[36] Jianqing Fan and Jinchi Lv. Sure independence screening for ultrahigh dimensional feature space. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 70(5):849–911, 10 2008. doi:https://doi.org/10.1111/j.1467-9868.2008.00674.x.

[37] Emmanuel Candès and Justin Romberg. Sparsity and incoherence in compressive sampling. *Inverse Problems*, 23(3):969, 2007. doi:10.1088/0266-5611/23/3/008. URL https://doi.org/10.1088/0266-5611/23/3/008.

[38] Thomas A. R. Purcell, Matthias Scheffler, Christian Carbogno, and Luca M. Ghiringhelli. SISSO++: A C++ implementation of the sure-independence screening and sparsifying operator approach. *Journal of Open Source Software*, 7(71):3960, 2022. doi:10.21105/joss.03960.

[39] Sebastian Eibl, Yi Yao, Matthias Scheffler, Markus Rampp, Luca Ghiringhelli, and Thomas A. R. Purcell. A high-performance and portable implementation of the sisso method for cpus and gpus. *Machine Learning: Science and Technology*, 2025. doi:10.1088/2632-2153/ae0ab3.

[40] Thomas AR Purcell, Matthias Scheffler, Luca M Ghiringhelli, and Christian Carbogno. Accelerating materials-space exploration for thermal insulators by mapping materials properties via artificial intelligence. *npj computational materials*, 9(1):112, 2023. doi:https://doi.org/10.1038/s41524-023-01063-y.

[41] Steven G Johnson et al. The NLopt nonlinear-optimization package, 2014. URL http://github.com/stevengj/nlopt.

[42] Scott Kirkpatrick, C. Daniel Gelatt, and Mario P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 06 1983. doi:10.1126/science.220.4598.671.

[43] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc Le. Regularized evolution for image classifier architecture search. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33:4780–4789, 2019. doi:10.1609/aaai.v33i01.33014780.

[44] C. G. Broyden. The convergence of a class of double-rank minimization algorithms 1. General Considerations. *IMA Journal of Applied Mathematics*, 6:76–90, 1970. URL https://doi.org/10.1093/imamat/6.1.76.

[45] Joseph K. Blitzstein and Jessica Hwang. *Introduction to probability*. Chapman and Hall/CRC, 2014. doi:https://doi.org/10.1201/9780429428357.

[46] Geoffrey F. Bomarito, Patrick E. Leser, N. C. M. Strauss, K. M. Garbrecht, and Jacob D. Hochhalter. Bayesian model selection for reducing bloat and overfitting in genetic programming for symbolic regression. *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2022. doi:https://doi.org/10.1145/3520304.3528899.

[47] Geoffrey F. Bomarito, Patrick E. Leser, N. C. M. Strauss, K. M. Garbrecht, and Jacob D. Hochhalter. Automated learning of interpretable models with quantified uncertainty. *Computer methods in applied mechanics and engineering*, 403:115732, 2023. doi:https://doi.org/10.1016/j.cma.2022.115732.

[48] Deaglan Bartlett, Harry Desmond, and Pedro Ferreira. Priors for symbolic regression. *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*, pages 2402–2411, 2023. doi:10.48550/arXiv.2304.06333.

[49] Pierre Del Moral, Arnaud Doucet, and Ajay Jasra. Sequential Monte Carlo samplers. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 68(3):411–436, 2006. doi:https://doi.org/10.1111/j.1467-9868.2006.00553.x.

[50] Alexander Buchholz, Nicolas Chopin, and Pierre Jacob. Adaptive tuning of Hamiltonian Monte Carlo within sequential Monte Carlo. *Bayesian Analysis*, 16(3):745 – 771, 2021. doi:https://doi.org/10.1214/20-BA1222.

[51] Jeroen Hol, Thomas Schön, and Fredrik Gustafsson. On Resampling Algorithms for Particle Filters. *IEEE Nonlinear Statistical Signal Processing Workshop*, pages 79–82, 2006. doi:10.1109/NSSPW.2006.4378824.

[52] Raul ing Average, Abhipsha Das, Marco Edward Gorelli, Thomas Wiecki, Oriol Abril-Pla, Juan Orduz, Benjamin T. Vincent, Chris Fonnesbeck, Meenal Jhajharia, Christopher Krapu, Michael Osthege, John Goertz, larryshamalama, Farhan Reynaldo, Kavya Jaiswal, Olga Kahn, Ravin Kumar,

ltoniazzi, Byron J. Smith, Christine P. Chai, Cuong Duong, Emilio Jorge, Josh Cook, Kenneth Enevoldsen, Osvaldo Martin, Tyler Burch, Zeel B Patel, ally lee, and rpgoldman. pymc-devs/pymc-examples: January 2022 snapshot, 2022. URL https://doi.org/10.5281/zenodo.5832070.

[53] Christopher D. Roberts and Martin Leutbecher. Unbiased calculation, evaluation, and calibration of ensemble forecast anomalies. *Quarterly Journal of the Royal Meteorological Society*, 151(771):e4993, 2025. doi:https://doi.org/10.1002/qj.4993.

[54] Félix Musil, Michael J. Willatt, Mikhail A. Langovoy, and Michele Ceriotti. Fast and accurate uncertainty estimation in chemical machine learning. *Journal of Chemical Theory and Computation*, 15(2):906–915, 2019. doi:10.1021/acs.jctc.8b00959.

[55] Laurens Sluijterman, Eric Cator, and Tom Heskes. How to evaluate uncertainty estimates in machine learning for regression? *Neural Networks*, 173: 106203, 02 2024. doi:10.1016/j.neunet.2024.106203.

[56] Ana Dutra, Benedek Goldmann, M. Saiful Islam, and James Dawson. Understanding solid-state battery electrolytes using atomistic modelling and machine learning. *Nature Reviews Materials*, 10, 06 2025. doi:10.1038/s41578-025-00817-y.

[57] Aniruddha Joshi, Dillip Kumar Mishra, Rajendra Singh, Jiangfeng Zhang, and Yi Ding. A comprehensive review of solid-state batteries. *Applied Energy*, 386:125546, 2025. ISSN 0306-2619. doi:https://doi.org/10.1016/j.apenergy.2025.125546.

[58] Shun Ai, Xianli Wu, Jintao Wang, Xu Li, Xiaofeng Hao, and Yuezhong Meng. Research progress on solid-state electrolytes in solid-state lithium batteries: Classification, ionic conductive mechanism, interfacial challenges. *Nanomaterials*, 14(22), 2024. doi:10.3390/nano14221773.

[59] Mingming Ma, Menghui Zhang, Bitao Jiang, Yang Du, Bingcheng Hu, and Chengguo Sun. A review of all-solid-state electrolytes for lithium batteries: high-voltage cathode materials, solid-state electrolytes and electrode–electrolyte interfaces. *Materials Chemistry Frontiers*, 7, 01 2023. doi:10.1039/D2QM01071B.

[60] Cong Li, Zhen-yu Wang, Zhen-jiang He, Yun-jiao Li, Jing Mao, Ke-hua Dai, Cheng Yan, and Jun-chao Zheng. An advance review of solid-state battery: Challenges, progress and prospects. *Sustainable Materials and Technologies*, 29:e00297, 2021. doi:https://doi.org/10.1016/j.susmat.2021.e00297.

[61] Shuixin Xia, Xinsheng Wu, Zhichu Zhang, Yi Cui, and Wei Liu. Practical challenges and future perspectives of all-solid-state lithium-metal batteries. *Chem*, 5(4):753–785, 2019. doi:https://doi.org/10.1016/j.chempr.2018.11.013.

[62] Teddy Mageto, Sanket D Bhoyate, Felipe M de Souza, Kwadwo Mensah-Darkwa, Anuj Kumar, and Ram K Gupta. Developing practical solid-state rechargeable li-ion batteries: Concepts, challenges, and improvement strategies. *Journal of Energy Storage*, 55:105688, 2022. doi:https://doi.org/10.1016/j.est.2022.105688.

[63] Joo Gon Kim, Byungrak Son, Santanu Mukherjee, Nicholas Schuppert, Alex Bates, Osung Kwon, Moon Jong Choi, Hyun Yeol Chung, and Sam Park. A review of lithium and non-lithium based solid state batteries. *Journal of Power Sources*, 282:299–322, 2015. doi:https://doi.org/10.1016/j.jpowsour.2015.02.054.

[64] Robert Usiskin, Yaxiang Lu, Jelena Popovic, Markas Law, Palani Balaya, Yong-Sheng Hu, and Joachim Maier. Fundamentals, status and promise of sodium-based batteries. *Nature Reviews Materials*, 6(11):1020–1035, 2021. doi:10.1038/s41578-021-00324-w.

[65] Theodosios Famprikis, Pieremanuele Canepa, James Dawson, M. Saiful Islam, and Christian Masquelier. Fundamentals of inorganic solid-state electrolytes for batteries. *Nature Materials*, 18:1–14, 08 2019. doi:10.1038/s41563-019-0431-3.

[66] Lukas Porz, Tushar Swamy, Brian W Sheldon, Daniel Rettenwander, Till Frömling, Henry L Thaman, Stefan Berendts, Reinhard Uecker, W Craig Carter, and Yet-Ming Chiang. Mechanism of lithium metal penetration through inorganic solid electrolytes. *Advanced Energy Materials*, 7(20): 1701003, 2017. doi:https://doi.org/10.1002/aenm.201701003.

[67] Fudong Han, Andrew S Westover, Jie Yue, Xiulin Fan, Fei Wang, Miaofang Chi, Donovan N Leonard, Nancy J Dudney, Howard Wang, and Chunsheng Wang. High electronic conductivity as the origin of lithium dendrite formation within solid electrolytes. *Nature Energy*, 4(3):187–196, 2019. doi:10.1038/s41560-018-0312-z.

[68] Jitong Yan, Dingding Zhu, Hongjun Ye, Haiming Sun, Xuedong Zhang, Jingming Yao, Jingzhao Chen, Lin Geng, Yong Su, Pan Zhang, et al. Atomic-scale cryo-TEM studies of the thermal runaway mechanism of $Li_{1.3}Al_{0.3}Ti_{1.7}P_3O_{12}$ solid electrolyte. *ACS Energy Letters*, 7(11):3855–3863, 2022. doi:10.1021/acsenergylett.2c01981.

[69] Michel Armand. Polymer solid electrolytes-an overview. *Solid State Ionics*, 9:745–754, 1983. doi:https://doi.org/10.1016/0167-2738(83)90083-8.

[70] Siyang Liu, Hongtai Cheng, Runyue Mao, Wanyuan Jiang, Lin Wang, Zihui Song, Mengfan Pei, Tianpeng Zhang, and Fangyuan Hu. Designing Zwitterionic gel polymer electrolytes with dual-ion solvation regulation enabling stable sodium ion capacitor. *Advanced Energy Materials*, 13(18): 2300068, 2023. doi:https://doi.org/10.1002/aenm.202300068.

[71] Venkataraman Thangadurai, Sumaletha Narayanan, and Dana Pinzaru. Garnet-type solid-state fast Li ion conductors for Li batteries: critical review. *Chemical Society Reviews*, 43(13):4714–4727, 2014. doi:https://doi.org/10.1039/C4CS00020J.

[72] Xiangtao Bai, Tianwei Yu, Zhimin Ren, Shengmin Gong, Rong Yang, and Chunrong Zhao. Key issues and emerging trends in sulfide all solid state lithium battery. *Energy Storage Materials*, 51:527–549, 2022. doi:https://doi.org/10.1016/j.ensm.2022.07.006.

[73] Tetsuya Asano, Akihiro Sakai, Satoru Ouchi, Masashi Sakaida, Akinobu Miyazaki, and Shinya Hasegawa. Solid halide electrolytes with high lithium-ion conductivity for application in 4 V class bulk-type all-solid-state batteries. *Advanced materials*, 30(44):1803075, 2018. doi:https://doi.org/10.1002/adma.201803075.

[74] Austin D. Sendek, Qian Yang, Ekin Dogus Cubuk, Karel-Alexander N. Duerloo, Yi Cui, and Evan J. Reed. Holistic computational structure screening of more than 12 000 candidates for solid lithium-ion conductor materials. *Energy and Environmental Science*, 10:306–320, 2017. doi:https://doi.org/10.1039/C6EE02697D.

[75] Keith T Butler, Daniel W Davies, Hugh Cartwright, Olexandr Isayev, and Aron Walsh. Machine learning for molecular and materials science. *Nature*, 559(7715):547–555, 2018. doi:https://doi.org/10.1038/s41586-018-0337-2.

[76] C.R.A. Catlow. Computer simulation studies of transport in solids. *Annual Review of Materials Science*, 16(1):517–548, 1986. doi:https://doi.org/10.1146/annurev.ms.16.080186.002505.

[77] Ryogo Kubo. Statistical-mechanical theory of irreversible processes. i. general theory and simple applications to magnetic and conduction problems. *Journal of the physical society of Japan*, 12(6):570–586, 1957. doi:https://doi.org/10.1143/JPSJ.12.570.

[78] Martin French, Sebastien Hamel, and Ronald Redmer. Dynamical screening and ionic conductivity in water from ab initio simulations. *Physical review letters*, 107(18):185901, 2011. doi:https://doi.org/10.1103/PhysRevLett.107.185901.

[79] Federico Grasselli and Stefano Baroni. Topological quantization and gauge invariance of charge transport in liquid insulators. *Nature Physics*, 15:1, 09 2019. doi:10.1038/s41567-019-0562-0.

[80] Ashutosh Kumar Verma, Amey S. Thorat, and Jindal K. Shah. Estimating ionic conductivity of ionic liquids: Nernst–Einstein and Einstein formalisms. *Journal of Ionic Liquids*, 4(1):100089, 2024. ISSN 2772-4220. doi:https://doi.org/10.1016/j.jil.2024.100089.

[81] Svante Arrhenius. Über die dissociationswärme und den einfluss der temperatur auf den dissociationsgrad der elektrolyte. *Zeitschrift für physikalische Chemie*, 4(1):96–116, 1889. doi:https://doi.org/10.1515/ZPCH-1889-0408.

[82] Helmut Mehrer. *Diffusion in solids : fundamentals, methods, materials, diffusion-controlled processes*. Springer Berlin, 2007. doi:https://doi.org/10.1007/978-3-540-71488-0.

[83] Anubhav Jain, Shyue Ong, Geoffroy Hautier, Wei Chen, William Richards, Stephen Dacek, Shreyas Cholia, Dan Gunter, David Skinner, Gerbrand Ceder, and Kristin Persson. Commentary: The materials project: A materials genome approach to accelerating materials innovation. *APL Materials*, 1: 011002, 07 2013. doi:10.1063/1.4812323.