

Perspective

AI-Assisted Modeling and Coding: The Overlooked Contribution of Generative AI to Chemistry

Didier Mathieu 

CEA, DAM, Le Ripault, F-37260 Monts, France; didier.mathieu@cea.fr

Abstract

In the context of artificial intelligence in chemistry, the first applications that come to mind are widely discussed ones, such as building surrogate models to replace costly numerical simulations or designing new compounds. However, one area where generative AI truly stands out is programming assistance, ranging from code improvement suggestions to vibe coding simple programs from scratch. This article explores how these tools can be used by chemists to test new ideas, make the most of existing open-source software and rapidly develop custom solutions that significantly ease day-to-day work. Using the problem of estimating Hansen solubility parameters as an academic example, their strengths and limitations are illustrated through fully customizable sample codes, including a solubility parameter calculator, a molecular spreadsheet library and a molecular editor. In addition to serving as models, these codes could be of practical interest to some readers due to unique features making them address gaps in the current offerings.

Keywords: Large Language Models; AI-assisted coding; vibe coding; molecular modeling; molecular editor; molecular graphics; chemical informatics

1. Introduction

From the very early days of artificial intelligence (AI), chemistry emerged as a particularly promising field of application. For instance, one of the very first expert systems of practical relevance was DENDRAL, reported in 1965 and designed to help in identifying the structure of unknown organic molecules from mass spectrometry data [1]. Early efforts were put as well in expert systems for retrosynthesis, such as LHASA [2] or PASCOP [3], and in the use of supervised learning methods to uncover quantitative structure-activity relationships (QSARs) for compounds of potential pharmaceutical interest [4]. Despite the second AI winter around 1990, which resulted in a decline in interest for expert systems, QSAR techniques continued to be applied to an increasing number of properties. The resulting quantitative structure-property relationships (QSPRs) became very popular in the early 2000s, due to chemical safety regulations encouraging their use as a low cost alternative to experimentation. Meanwhile, other applications of AI continued to emerge, as reflected in the successive name changes of the journal *QSAR*, launched in 1982, which became *QSAR & Combinatorial Science* in 2003, and finally *Molecular Informatics* in 2010.

While it has become commonplace to point out the impact AI is having on research, it is worth noting that, despite being a privileged application domain for decades, chemistry is no exception. This is evidenced by the recent emergence of new specialized journals such as *Journal of Cheminformatics* (2009), *Digital Discovery* (2022), *Artificial Intelligence Chemistry* (2023), *AI Chemistry* (2026). This acceleration in the adoption of AI in chemistry is driven by



Academic Editor: Giuseppe Trusso Sfrazzetto

Received: 29 June 2026

Revised: 25 July 2026

Accepted: 30 July 2026

Published: 5 August 2026

Copyright: © 2026 by the author.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

the development of generative AI (GenAI) and especially large language models (LLMs), which open up new perspectives. For example, the aims and scope of the journal *Artificial Intelligence Chemistry* list around fifteen applications of AI in chemistry, ranging from well-established ones, like QSAR or AI-assisted materials and drugs design, to more niche areas such as development of AI software and hardware, which concerns few chemists given the wide range of ready-to-use tools provided by the AI research community.

In this context, it may sound surprising that the reliance of chemists on AI for programming does not receive comparable interest, especially since coding is an application of GenAI where particularly dramatic progress has been made in recent years, and which already proves extremely useful in science, whether for checking and improving existing code, writing snippets of code whose implementation is particularly long and tedious, or even designing entire programs from scratch [5,6].

Chemists cannot afford to ignore the widespread adoption of AI-assisted coding in their research field, since its use is not without risks, particularly if it is practiced indiscriminately by researchers lacking programming skills and leads to the proliferation of disposable code. This is particularly clear when it comes to vibe coding, initially defined as a method of developing throwaway projects through specifications provided using natural language to a LLM, and where human code review is considered optional [7,8]. The benefits and limitations of such techniques in science were extensively discussed in recent papers [5–9]. This article extends these discussions in the specific context of chemistry, with a particular focus on molecular modeling, which is the research area of the author.

In this field, AI can be used at every stage of a project, from the design of a model to its implementation, including the installation and updating of the required libraries, as well as the development of ad hoc analysis and visualization tools. In order to adequately reflect this wide range of possible uses, these are illustrated in the sequel through a single specific problem, namely the estimation of Hansen solubility parameters (HSPs). This example is well-suited to this purpose since current HSP models are few in number, simple and likely improvable [10–16]. Moreover, the scarcity of reliable experimental HSP data deters AI tools from routinely steering users toward data-intensive, elaborate machine learning (ML) approaches. The examples mentioned refer to the design, implementation and validation of new models; to code beautification and formatting, to the update of unmaintained software; and to the rapid development of custom visualization and editing tools. While codes similar to those derived in present experiments may be easily obtained from major AI platforms, those of most potential interest to readers are provided as Supplementary Materials (SM) for reproducibility and to avoid redundant requests to AI platforms.

Advanced AI tools hosted locally are increasingly being deployed within large companies and in sensitive sectors, so these techniques can be used effectively while adhering to confidentiality requirements. These include specialized AI-based coding assistance tools like Claude Code, Cursor or Windsurf. In this article, the latter are deliberately ignored as they are not widely adopted by the broader chemistry community, being mainly useful to those few involved in the development of complex software such as quantum chemistry or molecular dynamics codes. Instead, the examples presented in this article rely on freely available online general-purpose platforms, hereafter simply referred to as AI, because they are more widely used in research [6]. More specifically, unless otherwise stated, we use Claude Sonnet 4.5 from Anthropic.

2. Model Design and Implementation

In chemistry, the core expertise of many modelers lies in designing and implementing models to describe and predict the behavior of atoms and molecules, usually within materials, fluids or living organisms. Reading research papers describing such models

immediately sparks numerous ideas for improvements in the minds of specialists. The limiting factor in advancing molecular modeling is not the abstract design of new models but rather the implementation and evaluation of the many possible options.

For example, the introduction of ReaxFF in 2001 was a major breakthrough, enabling molecular simulations of chemical reactions in condensed phases [17]. However, the idea of a reactive force field based on empirical bond orders easily comes to mind, and there is little doubt that it had previously occurred to many chemists frustrated by the inefficiency of *ab initio* calculations. However, developing a reactive potential involves the daunting task of fitting a model with many adjustable parameters against extensive quantum chemical (QC) data. Moreover, in the 1990s, a project to develop a reactive potential was likely to face skepticism from parts of the quantum chemistry community. For sure, it required considerable perseverance and tenacity from the ReaxFF authors to achieve reasonable parameterizations of this model [17,18].

However, although classical software engineering techniques and tools are clearly beneficial for more efficiently implementing models designed by chemists and physicists, resorting to recent generative AI tools for this purpose seems counterproductive at first glance. Indeed, as discussed earlier [19] and illustrated in Figure 1, using natural language introduces a loss of precision that is hard to overcome without resorting to a proper programming language. Since models are naturally expressed through non-ambiguous mathematical equations, using natural language seems undesirable.



Figure 1. While GenAI allows us to converse with a computer in natural language, programming languages are better suited for specifying a model in full details [20]. Reproduced with permissions from <https://www.holybird.fr/productions/commitstrip/> (accessed 1 July 2026).

In fact, this issue disappears if AI takes over both the design of the model and its implementation. While AI-assisted coding tools primarily address the implementation step, which is the most labor-intensive and time-consuming part of the process, the free online AI platforms preferred by casual coders in research are more generic tools trained using a huge body of scientific material that could in principle help the user in designing models. While prompting an AI is obviously not enough to build a full-blown quantum chemistry package, most models reported in chemical literature are quite simple. For instance, models to predict the dispersion (δ_d), polar (δ_p) and hydrogen bond (δ_h) HSP components fall into the following three categories: group contribution (GC) methods [10,11], QSPRs [12–15], and a semi-physical pencil and paper (P&P) model involving simple additivity schemes for the molar volume V_m and for the two energy contributions $V_m\delta_p^2$ and $V_m\delta_h^2$ [16]. QSPR models are the most numerous, which is explained by the fact that they are particularly fast to develop, as they are based on procedures that are largely independent of the property to be modeled. As a result, an AI trained against published QSPRs could, in principle, compete with QSPR developers for new properties.

As a simple test, Claude was first simply prompted: “Please write me a Python code based on RDKit to predict the Hansen Solubility Parameters of any molecule given by a SMILES string provided as input”. In response, it returned an implementation of the Hoy method [10], a well-established but somehow outdated model introduced in 1989. Hence the next prompt: “There is no value in replicating existing methods. I expect you to propose better alternatives.”. Claude then suggested three options: 1: Supervised ML model, preferably XGBoost or Random Forest (RF); 2: Graph neural network (GNN); 3: Simplified COSMO-RS approach based on approximate 3D estimates of the COSMO-RS descriptors. These suggestions are nothing new: XGBoost, RFs and GNNs are currently trending methods for supervised learning, while COSMO-RS descriptors were used in the most recent QSPRs for HSP prediction [13–15]. A more unique model, such as P&P which is based on considerations specific to the modeled property [16], is inherently less accessible to AI since it has not led to many publications, in contrast to mainstream QSPR methodologies.

Finally, after it was informed that the models should be trained on a very limited data set, namely the one previously used to fit the P&P model [16], the AI eventually opted for Gaussian Process Regression (GPR), the Bayesian machine learning method underpinning the gpHSP model based on 2D and 3D quantum chemical COSMO-RS descriptors [14]. More specifically, it proposed GPR models for δ_d , δ_p and δ_h , based on a Matérn-5/2 kernel with automatic relevance determination plus a constant amplitude and a white-noise term, as implemented by *GaussianProcessRegressor* in scikit-learn [21].

However, the AI departed from gpHSP by proactively dismissing 3D descriptors to avoid the variability inherent to 3D conformer selection, and replacing 2D fingerprints with 28 descriptors calculated using RDKit. They were selected independently of the data, as a standard, broad-coverage feature set spanning molecular size, topology, shape, flexibility, lipophilicity, polarizability, polar surface area and hydrogen-bonding capacity.

In fact, using only 2D descriptors may be difficult to justify, especially for δ_p which is determined by the spatial charge distribution. Therefore, for this particular component, we prompted the AI to follow his third suggestion and use 3D descriptors inspired by those commonly obtained from COSMO-RS, which are electrostatic in nature [13–15]. However, to avoid costly quantum calculations, we requested that they be calculated from the MMFF charge distribution. One might object that this is impossible, due to the absence of implicit solvent and a cavity defining the molecular surface. However, in line with the tendency of current chatbots to satisfy user requests, the AI proposed drastic simplifications of the COSMO-RS descriptors, obtained using the MMFF atomic charges in place of a continuous

electron cloud and a solvent-accessible surface (SAS) in place of the COSMO-RS cavity. As a result, these descriptors do not capture any solvation effect, and should rather be viewed as descriptors of the electrostatic field arising from isolated molecules, i.e., a low-cost version of the General Interaction Properties Function (GIPF) descriptors introduced to the predict condensates-phase properties [22].

To evaluate these simplified descriptors, the SAS is divided into regions associated with the different atoms, in proportion to the van der Waals surface areas of the atoms considered in isolation. The surface charge density σ_i is assumed to be constant in the region associated with atom i , equal to q_i/A_i where A_i is the surface area of this region and q_i the partial charge of atom i . On the basis of this simple model, the seven descriptors suggested by the AI are the dipole moment μ arising from the atomic charges, the total SAS area A_{tot} as well as the areas A_{pos} and A_{neg} of the regions of this surface where the charge density is respectively positive and negative, and finally, the first (M_1), second (M_2) and third (M_3) moments of the surface charge density, as shown in Table 1.

Table 1. The 3D descriptors proposed by Claude and calculated from MMFF charge distribution after we suggested drawing inspiration from COSMO-RS descriptors.

Definition	Description
$\mu = \left \sum_i q_i \mathbf{r}_i \right $	MMFF dipole moment derived from partial charge q_i and 3D position $\mathbf{r}_i$ of atom i .
A_{tot}	Total molecular surface area calculated using FreeSASA as provided by RDKit; split into individual atomic contributions A_i proportional to van der Waals atom surface areas.
$A_{\text{neg}} = \sum_{i: \sigma_i < 0} A_i$	H-bond acceptor surface area (electron-rich zones).
$A_{\text{pos}} = \sum_{i: \sigma_i > 0} A_i$	H-bond donor surface area (electron-poor zones).
$M_2 = \frac{\sum_i \sigma_i^2 A_i}{\sum_i A_i}$	Second moment (variance) of the surface charge density; proportional to the COSMO-RS misfit energy; primary predictor of δ_p .
$M_3 = \frac{\sum_i \sigma_i^3 A_i}{\sum_i A_i}$	Third moment (skewness) of the surface charge density; encodes donor/acceptor surface asymmetry.

Although the AI was requested to draw inspiration from COSMO-RS descriptors and to describe the electron distribution simply using atomic charges, it autonomously made many design choices that demonstrated a remarkable domain expertise, in addition to its widely recognized ability to generate code. For example, having chosen the exact form of the simplified descriptors used and proposed the seven descriptors above, it decided to introduce an eighth one obtained from the dipole moment μ and molecular weight M_w of the compound as $\mu/\sqrt{M_w}$. This descriptor is justified by the AI on the basis of the Böttcher formula $\delta_p \propto \mu/V_m$ where μ and V_m are the dipole moment and molar volume of the compound, using (somewhat surprisingly) $\sqrt{M_w}$ as a proxy for V_m , rather than one of the volume descriptors provided by RDKit. Furthermore, the AI noted that these quantities are conformation-dependent and thus decided to calculate them on a random sample of 5 conformers generated using RDKit, using their means and standard deviations as actual descriptors. However, despite noticing the small magnitude of standard deviations, it failed to observe that M_1 is necessarily equal to zero due to the charge neutrality of the systems considered.

In order to demonstrate the value of the AI to develop a QSPR, we instructed it, as the only directives, to ignore M_1 (always equal to zero) as well as all standard deviations

derived from the dispersion of values across the different conformers, except for that of the dipole moment μ , which we denote as $\Delta\mu$. The retained 3D descriptors are finally as follows: μ (along with $\mu/\sqrt{M_w}$ and $\Delta\mu$), A_{tot} , A_{pos} , A_{neg} , M_2 , M_3 . They are precisely defined in Table 1. These 3D descriptors were only used to model the polar HSP. More specifically, three models referred to as GPR/2D, GPR/3D and GPR/2D+3D were developed to estimate δ_p depending on the pool of descriptors considered. In contrast, only 2D descriptors were used to model δ_d and δ_h . The models were fitted and validated using the training and test sets compiled in *HSPVm_ACSO_Database.csv*, using the script *HSP_gpr.py* where they are described in full details. These files are provided as SM.

3. Validation of Implemented Models

Statistical and ML techniques are made easy to use with the help of existing libraries. This is made even easier by using an AI such as Claude, which proves very good at leveraging these tools. For example, given a list of modeling techniques and a data set, it is enough to specify that the status of each reference value is indicated by an integer equal to zero for the training set and one for the test set, for the AI to implement both the model fitting and their validation. Unlike a common practice in the field of GC models, whose performance are usually measured based on fitting errors [10,11], the reference P&P models were assessed using leave-one-out (LOO) cross-validation (CV) and application to an external test set, as summarized in Table 2 [16].

Table 2. Summary of MAE data ($\text{MPa}^{1/2}$) reported previously for P&P models [16].

MAE	LOO-CV (Training Set)			External Validation (Test Set)		
	δ_d	δ_p	δ_h	δ_d	δ_p	δ_h
MAE	0.68	2.00	1.55	0.75	2.08	1.67

For comparison, the present GPR models were assessed using similar indicators derived from LOO-CV and external test set predictions, keeping in mind that data included in the test set are deemed unreliable (due to the lack of experimental confirmation) and must therefore be regarded with caution [23].

Importantly, LOO-CV is performed only for the final models. It is in no way used to optimize hyperparameters or to select a model among several variants, which implies that LOO estimates may be used to derive relevant performance indicators, including mean absolute and root mean square deviations (MAD and RMSD) from reference values and determination coefficient R^2 . The main focus is on MAD as corresponding values were reported in recent reference HSP models [14–16].

Training GPR models is particularly expensive, so their LOO cross-validation is often prohibitive and thus replaced by validation using a smaller number of folds. Interestingly, Claude seems aware of this issue. In response to our request to perform LOO-CV on the models discussed here, it proposes a cross-validation that it describes as “analytical,” which leads in a surprisingly short time to performance metrics that are surprisingly optimistic. In fact, when asked for clarification, it indicated that this was an approximate LOO in which the model hyperparameters obtained from fitting on the training set are not reoptimized. This explains the good metrics obtained in reply to our first request. To avoid considering artificially inflated performance indicators, the LOO-CV data reported in the sequel were obtained from a genuine cross-validation where a new model is derived from scratch for each fold, ignoring the held-out sample.

Regarding present δ_d estimates, the LOO-CV yields a MAD of $0.65 \text{ MPa}^{1/2}$, close to the value of $0.68 \text{ MPa}^{1/2}$ obtained for both the P&P and gpHSP models [14]. P&P performs remarkably well considering that it involves just three fitting parameters and might suffer

from underfitting. In contrast, the use of genuine COSMO-RS descriptors in gpHSP does not allow it to obtain a superior accuracy. Regarding out-of-sample predictions for the test set, P&P yields the best results with a MAD of $0.75 \text{ MPa}^{1/2}$, in line with its lower empiricism, whereas the present model exhibits a MAD of $0.96 \text{ MPa}^{1/2}$, close to that of $0.98 \text{ MPa}^{1/2}$ reported by Wojeicchowski et al. [15].

Regarding the δ_h component, the LOO-CV yields a MAD of $1.55 \text{ MPa}^{1/2}$ for the present model, in line with corresponding values of $1.55 \text{ MPa}^{1/2}$ and $1.57 \text{ MPa}^{1/2}$ reported respectively for P&P and gpHSP. In contrast, application to the test set yields a MAD of $1.95 \text{ MPa}^{1/2}$, larger than the values of $1.67 \text{ MPa}^{1/2}$ and $1.44 \text{ MPa}^{1/2}$ reported respectively for P&P and by Wojeicchowski et al. [15]. This could reflect limitations in the ability of present descriptors to reflect subtle differences in the chemical environment of labile protons and proton acceptor atoms.

Regarding the δ_p component, three different GPR models are considered in this work. However, since 2D descriptors are in principle ill-suited to described this HSP, as confirmed by the results in Table 3, we focus on the model based only on 3D descriptors. The LOO-CV of this model yields a MAD of $1.76 \text{ MPa}^{1/2}$, lower than the values of $2.00 \text{ MPa}^{1/2}$ and $1.93 \text{ MPa}^{1/2}$ obtained for the P&P and gpHSP models. Test set predictions yield a MAD of $2.12 \text{ MPa}^{1/2}$ similar to that obtained using P&P ($2.08 \text{ MPa}^{1/2}$) but higher than the state-of-the-art value of $1.44 \text{ MPa}^{1/2}$ reported in ref. [15].

Table 3. Performance of present GPR models: number of entries (N), mean absolute deviation (MAD), root mean square deviation (RMSD) and determination coefficient (R^2) between predicted and reference values for leave-one-out (LOO) and test set predictions ($\text{MPa}^{1/2}$). Standard deviations were obtained from bootstrapping over 1000 samples.

	N	R^2	MAD	RMSD
<i>LOO against training set:</i>				
δ_d	151	0.56 ± 0.06	0.65 ± 0.06	0.96 ± 0.08
δ_h	153	0.81 ± 0.03	1.55 ± 0.15	2.44 ± 0.29
δ_p (2D)	144	0.69 ± 0.04	1.85 ± 0.15	2.61 ± 0.24
δ_p (3D)	144	0.70 ± 0.04	1.76 ± 0.14	2.56 ± 0.20
δ_p (2D+3D)	144	0.74 ± 0.04	1.78 ± 0.13	2.37 ± 0.19
<i>Predictions for test set:</i>				
δ_d	707	0.50 ± 0.04	0.96 ± 0.14	1.30 ± 0.18
δ_h	707	0.67 ± 0.08	1.95 ± 0.34	2.76 ± 0.48
δ_p (2D)	545	0.12 ± 0.13	2.23 ± 0.33	2.98 ± 0.41
δ_p (3D)	545	0.54 ± 0.16	2.12 ± 0.34	2.99 ± 0.52
δ_p (2D+3D)	545	0.50 ± 0.14	2.25 ± 0.30	3.12 ± 0.42

Finally, although they were derived in a couple of hours using AI and crude approximations to the 3D electrostatic descriptors, the present GPR models compare well with published ones. According to out-of-sample predictions, their accuracy is similar to that reported in ref. [15] for δ_d , and similar to that reported using P&P for δ_p . To our knowledge, along with P&P, present AI-derived models yield state-of-the-art accuracy among models avoiding quantum calculations. Accordingly, they constitute a promising starting point for building better models, for example by addressing poorly justified approximations such as replacing molar volume with the square root of molar mass. They are therefore provided as SM to the reader.

4. Code Beautification and Formatting

Although human programmers remain essential for developing complex and large-scale software, such development work requires writing many lines of code and carrying

out long, tedious testing and debugging work, regardless of the size or complexity of the code. Since AI can quickly identify questionable routines, it can prove invaluable in helping researchers who program occasionally to improve their code without needing to master tools designed for experienced programmers. This is all the more true given that in addition to the fact that they may lack familiarity with software engineering principles, researchers are often short on time to properly implement the models they have in mind.

For example, in order to take advantage of a limited-time waiver of publication fees in an open-access journal, the above-mentioned P&P models for HSP components were published in a rush, together with an implementation in the form of a script serving as a first draft, left unrevised and uncorrected due to lack of time [16]. While these models, originally developed to estimate gas sensors responses, are now used in a wide range of applications, from selecting solvents for the extraction of phytochemicals [24] to calculating enthalpies of vaporization [25], the published Python script remained in a draft state, and had to be adapted in order to be included in the HSPmol software, whose first pre-release was recently described in full detail [26].

Therefore, we asked AI to rewrite it more cleanly. While no improvement was made to the structure of the code, the revised version appeared cleaner and was enriched with relevant comments. At first glance, this new code produced exactly the same results as the original, which is the least that could be expected. However, while this observation holds true in the vast majority of cases, a more careful examination revealed a few instances where the new estimates diverged significantly from the original ones. After this difference in results was pointed out, the AI admitted that it had uncommented two lines originally intended to test whether a polar group detected in a molecule contains an atom belonging to another group already accounted for. Indeed, as the model was being developed, it was initially considered to define polar groups such that they could not share any atoms. Ultimately, a more flexible definition of polar groups, allowing the possibility that two groups may share atoms, was adopted. However, the tests designed to exclude this possibility were simply commented out rather than deleted.

Interestingly, the AI took the initiative to reactivate them, as if it could not conceive of the presence of unnecessary commented-out code lines remaining from an earlier version. This experience highlights that free online AIs cannot be used without caution, even for minor code modifications to the simplest scripts. Hopefully, commercial programming assistance tools would not silently take such initiatives without referring to the user.

5. Updating Outdated Software

The popularity of the HSP approach stems from the fact that it exploits a pre-averaged representation of interactions between chemical compounds, based on the δ_d , δ_p and δ_h indices. This approach may be used for instance for binder selection, as it allows to identify those with good affinity for molecular crystals they are intended to coat [27]. However, it cannot predict how the growth medium affects crystal morphology, since it does not make any distinction between the different faces of the crystal. For this purpose, it would be necessary to define solubility parameters specific to each crystal face, for example by simulating the latter in the presence of various solvents whose HSPs are known.

This requires a tool that enables generation of the relevant faces while respecting the integrity of the molecules. For many years, this feature was only widely available through commercial suites like Materials Studio, which do not easily integrate into custom research workflows [28]. In 2020, Yang et al. described an open-source implementation into a code named OGRE [29]. However, we observed that this code does not run on the most recent computing environments. The debugger indicates that certain lines of the code contain calls to unknown functions. Unfortunately, in the lack of familiarity with the

code and its dependencies, determining the library that used to provide these missing functions and finding suitable replacements may prove time-consuming. In this context, it is not rare for a feature to be independently reimplemented by different research teams. A new independent implementation of an algorithm to build molecular crystal faces is now provided by the Los Alamos Crystal Cut package [30].

Another option is to upgrade unmaintained software, which is now much easier than before thanks to AI. Regarding OGRE, the error messages returned by the code when executed within a modern environment were submitted to the default version of the perplexity.ai platform. In reply, the latter suggested several possible causes, two of which seemed highly plausible, namely changes in the interfaces of the NetworkX and Pymatgen libraries. After implementing the suggested edits following these upstream changes, the code once again functioned as expected, using the latest versions of these two libraries. Without AI assistance, identifying the necessary changes would have required a significant amount of time, especially for authors without prior direct experience using Pymatgen and NetworkX.

6. Custom Visualization and Editing Tools

In the above-mentioned examples, AI is used to write or improve simple pieces of software, short enough to be subsequently read, checked and validated. However, as mentioned in the introduction, it is increasingly used by scientists to develop throwaway software through natural language specifications (vibe coding) as a means to address specific, short-term needs [7].

This approach is particularly attractive to most chemists for developing graphical user interfaces (GUIs) and visualization tools [5], due to their lack of familiarity with the required libraries and the programming techniques involved. It provides an appealing solution for creating lightweight alternatives (Figure 2) to mainstream software for data or molecular structure visualization, such as DataWarrior [31], PyMOL [32], or Jmol [33]. These tools, while feature-rich, are consequently relatively resource-heavy. Moreover, since DataWarrior and Jmol are written in Java, they do not naturally integrate into the Python-based workflows of many modelers.

Structure	δ _d	δ _p	δ _h	M _w
176/943 95-50-1	19.2	6.3	3.3	111.1 cc/mol
177/943 103-11-7	14.8	4.7	3.4	208.0 cc/mol
178/943 13116-57-9	17.8	15.7	3.4	396.9 cc/mol
179/943 21400-25-9	17.7	15.7	3.4	103.8 cc/mol
180/943 470-82-6	16.7	4.6	3.4	169.6 cc/mol
181/943 95-94-3	21.2	10.7	3.4	134.1 cc/mol
182/943 96-18-4	17.8	12.3	3.4	111.8 cc/mol
183/943 105-58-8	15.1	6.3	3.5	119.7 cc/mol
184/943 108-18-9	14.8	1.7	3.5	135.0 cc/mol
185/943 123-95-5	14.5	3.7	3.5	395.1 cc/mol
186/943 5216-25-1	20.3	5.5	3.5	152.2 cc/mol
187/943 582-83-2	19.6	7.1	3.5	156.9 cc/mol
188/943 709-63-7	18.8	6.1	3.5	150.6 cc/mol
189/943 74-97-5	17.3	5.7	3.5	150.8 cc/mol
190/943 75-79-6	16.5	6.6	3.5	113.1 cc/mol

Figure 2. Exploring HSP data using the molecular spreadsheet *qmogrid.py* provided as SM.

In order to quickly browse large collections of experimental data, one may find it easier to have AI write a simple molecular browser, like the molecular spreadsheet *qmogrid.py*

provided as SM, which was obtained from just a few prompts. Figure 2 demonstrates the use of this library to explore the database of experimentally confirmed HSP values provided as SM and used to fit P&P as well as the present GPR models. Although not optimized for speed and therefore not suitable to browse the huge molecular databases typically used to train machine learning (ML) models, such a tool is much more convenient and versatile than mainstream spreadsheets like Excel, enabling optimal use of the screen space, due to the fact that basic functions like zooming a specific cell or sorting entries by a specific criterion are executed with direct clicks in the grid cells, rather than through menus. On the other hand, given the relative simplicity of this library, it is actually quite easy to convince oneself that it provides the required functionalities, based on a few tests and a quick reading of the code—ignoring the many lines dedicated to managing graphical objects (windows, buttons, input fields, etc.).

This guarantee is more complex to achieve for other kinds of graphical applications, especially editors. With the rise of AI in chemistry, researchers often need to input numerous molecular diagrams from literature or various documents. The available open-source tools for this task (Chemtool, BKChem, jsMolEditor, JChemPaint...) are poorly suited for intensive input, either due to a lack of ergonomics or configurability, or because they are primarily designed to produce high-quality diagrams for publication illustrations. In this context, it is tempting to write code a custom 2D molecule editor, specifically designed to input many structures in minimal time. We thus developed the editor presented in Figure 3, taking precautions to ensure that the molecules input by the user were correctly interpreted. To this end, we provided the AI with the Python data structure to use for storing molecules in memory: essentially a list of nodes (atoms or functional groups) each one possibly featuring a label (a string displayed at the node), as well as a list of bonds represented as pairs of indices pointing to two nodes, each one also featuring a bond order attribute. Additionally, a snapping mechanism prevents two nodes from being so close that they could be visually confused. This ensures that the compounds drawn by the user are correctly interpreted.

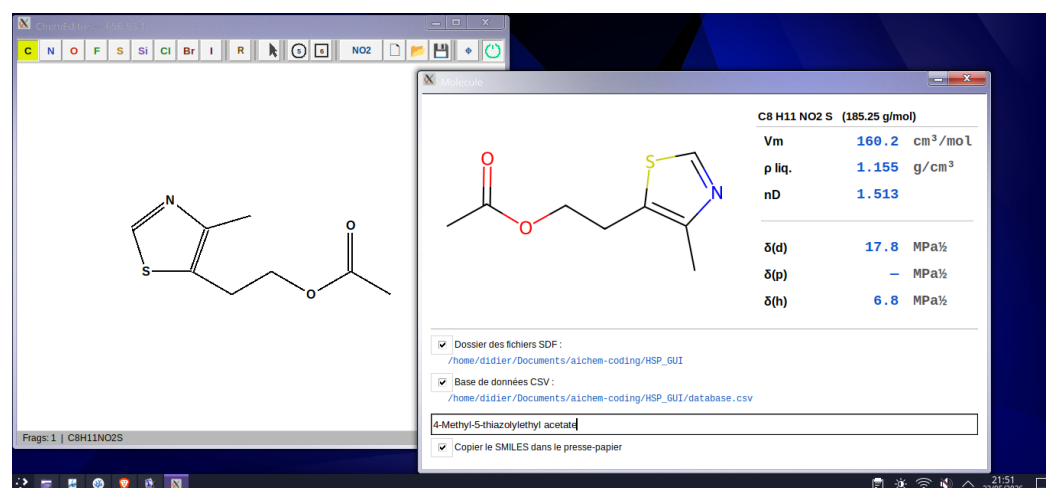


Figure 3. Easily built with vibe coding: a 2D molecular editor connected to Python scripts for estimating HSPs and refractive indices of chemical compounds. See SM for source code and documentation.

As an experiment, we also asked the AI to design this type of editor on its own, still providing technical specifications (Python/Tkinter) but no guidance on data structures or how to manage distances between nodes representing atoms. The resulting tool consists of an editor quite similar to the one shown in Figure 3, but whose internal structure is very different and does not lend itself to easy conversion of the diagrams entered in the canvas into molecular structures. It is primarily suited for drawing molecules, with bonds treated

as simple line segments unrelated to the concept of atoms. This allows segments to overlap on the canvas, potentially producing molecules that differ significantly from what the user intended. Thus, simply telling the AI that the script would be used for molecule input proved insufficient to guide it toward a suitable architecture. The fact that it chooses a data structure suited to creating publication-quality diagrams rather than to entering chemical formulas may stem from the fact that, among open-source tools, drawing applications are far more common than those designed for efficient input rather than aesthetic schematics.

Finally, the functionality designed to optimize the layout of the 2D structure by avoiding excessive proximity between non-bonded atoms or non-adjacent bonds proved particularly tedious to implement, despite the possibility for the AI to draw inspiration from analogous algorithms already available in existing open-source editors. Successive AI-generated proposals were consistently dysfunctional, until it was explicitly instructed to limit itself to a simple algorithm based primarily on a short-range repulsive potential between atoms, complemented by a harmonic potential aimed at homogenizing bond lengths. This rudimentary algorithm is the one implemented in the script provided as SM. This illustrates the fact that some experience is likely required to distinguish between requests that will save the user time and those that will waste it.

7. Discussion

The advantages, limitations, dangers and application areas of traditional coding, coding with the help of specialized AI-assisted coding tools and vibe coding using generic AI platforms are well-documented in AI literature [7]. Based on the specific example of HSP modeling, the experiments reported in this article highlight several additional key considerations of general significance for the design of predictive models based on molecular structure. Chief among them is the fact that the value of advanced AI platforms is not limited to coding predefined models, as they prove capable of contributing to how those models are designed and formulated. More precisely, the examples described in this paper show how AI may be useful at every stage of molecular modeling project, as illustrated in Figure 4.

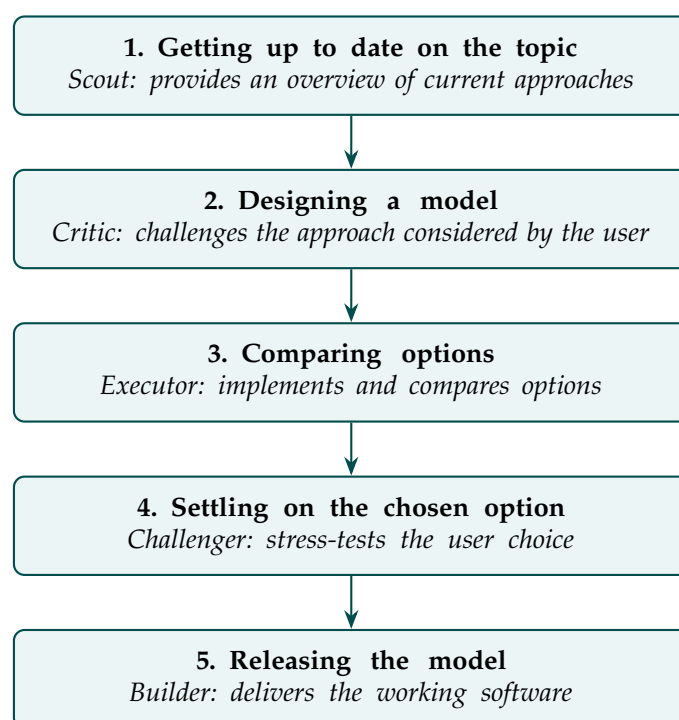


Figure 4. Roles fulfilled by AI throughout the successive stages of a molecular modeling project.

In fact, for developing typical QSPR models—which do not require sophisticated physical reasoning—a generic AI is able, with very little guidance from the user, to propose compelling alternatives to current models published by researchers. This should allow model developers to focus on the most innovative progress in their field, and enable a laboratory with little molecular modeling expertise to set up reasonable models given a training dataset. While this was already possible using specialized QSAR/QSPR software like AutoQSAR [34] or user-friendly statistical modeling tools like KNIME [35], LLMs make this possible without any restrictions on the type of approach used, as they can leverage virtually any tool available online, including not-so-popular ones like genetic programming libraries [36,37] or the deterministic fast function extraction (FFX) method [38].

On the other hand, AI missed the semi-physical P&P approach which currently yield the best test set predictions for δ_d and δ_h , hence possibly suggesting that obtaining state-of-the-art models may still require the help of human researchers. Moreover, when specifying a model to a LLM in natural language, the present examples illustrate the need to verify that the AI has correctly interpreted the provided specifications—despite the inherent ambiguity of natural language—and that it has not taken any misguided initiative. They also reveal its inability to provide satisfactory answers in some cases, such as when trying to optimize the 2D representation of a compound, where it proved unable to converge toward acceptable solutions until it was instructed to stick to a very simple model.

In spite of these limitations, it appears that AI can significantly boost productivity, even for chemists who only occasionally engage in programming but still need to generate and analyze large datasets. As a result, its use is likely to become widespread. Some even envision a future where authors share their prompts instead of their scripts [5]. In the current state of affairs, such a scenario is not desirable. For one, reusing prompts rather than scripts implies redundant requests to AI, and consequently, significant energy waste. Additionally, given the non-deterministic nature of LLMs, this practice undermines the reproducibility of scientific research: there is no guarantee that the same shared prompts will yield the same results. In contrast, and despite the initial perception that vibe coding is specially suited for throwaway projects [7], it is essential for chemists to share these AI-generated applications, if only for the sake of reproducibility. As such codes proliferate, tools will likely need to be developed so that potential users quickly find those best suited for their projects, rather than asking the AI to reinvent the wheel every time a new need arises. Even though vibe coding may produce unreadable or hard-to-understand code, sharing it remains necessary to spot potential errors and allow their authors to have their AI-produced codes reviewed by their peers.

Once authors of AI-generated software publish the code itself, rather than just the prompts used to produce it, the non-deterministic behavior of generative AI platforms ceases to be a problem. Users can rerun the same request to obtain alternative answers. This variability reflects the fact that, for most problems chemists face, there is no single a priori optimal solution. Multiple reasonable solutions are conceivable, and it is generally impossible to determine which is best until all have been implemented and tested. It could be worthwhile to launch several independent conversations with the same AI, starting from an identical base prompt, in order to gauge how many distinct solutions emerge. Such experiments, however, lie beyond the scope of this article, which aims primarily to encourage chemists to adopt LLMs while making them aware of certain limitations currently observed. Moreover, in the highly specific case of HSP estimation, it is evident that both the nature and the number of solutions proposed by the AI directly mirror the existing literature on the topic.

Historically, early molecular modeling software were developed within research laboratories and shared through reciprocal agreements between them or via organizations

like the Quantum Chemistry Program Exchange (QCPE) [39]. Then, the growing interest from industry in modeling spurred the development of commercial software providing easy access to quantum chemistry and molecular simulation methods. Eventually, as in other fields, the advent of the internet enabled the growth of the open-source software (OSS) ecosystem, where most advances now take place. However, with the exception of large collaborative projects such as Psi4 for quantum chemistry [40] or LAMMPS for molecular dynamics [41], open-source chemistry software are often developed by small groups of researchers to meet their own specific, short-term needs. Once they are fulfilled, the developers tend to abandon the code, as it no longer serves their ongoing research. As long as such codes are not widely adopted, their maintenance may be lacking. In this context, major selling points of commercial codes are their ease of use—often resulting from a graphical user interface—as well as that they are typically maintained over long periods, ensuring users a degree of longevity.

Regarding the first point, AI now enables many users to setup and share their own bespoke GUI for specific tasks, that are likely to be even easier to use compared with proprietary interfaces that have to accommodate many use cases, as shown in this article with *qmolgrid.py* or *HSPgui.py*. Regarding the second point, GenAI makes it possible to get most from OSS, because it makes it much easier than before to maintain and upgrade even highly specialized and seldom used software, as presently demonstrated in the case of OGRE. Therefore, we can expect that proprietary software vendors will leverage new AI-based assisted programming and vibe coding tools in order to add innovative features setting their products apart from open-source tools for reasons other than ease of use or greater long-term sustainability assurance.

Supplementary Materials: The following supporting information can be downloaded at: <https://www.mdpi.com/article/10.3390/aichem1030012/s1>, Archive S1: ZIP file ArchiveS1.zip containing the Python scripts coded with AI-assistance in the present work.

Funding: This research received no external funding.

Data Availability Statement: The original contributions presented in this study are included in the Supplementary Materials. Readers will find there: an implementation of the HSP models detailed in the article (*HSPgpr.py*); the Python library implementing a molecular spreadsheet (*qmolgrid.py*); and a molecular editor (*HSPgui.py*) calling third-party scripts (*ri_calculator.py* and *HSPs.py*) as the user requests estimates of the refractive index and HSPs. Further inquiries can be directed to the author.

Acknowledgments: In this work, AI tools—namely Claude Sonnet 4.5 and a random version of Perplexity.ai—were used to illustrate some of their applications to chemistry. AI tools were not used to prepare or revise the manuscript, except for the graphical abstract and a few stylistic improvements.

Conflicts of Interest: The author declares no conflicts of interest.

References

1. Lindsay, R.K.; Buchanan, B.G.; Feigenbaum, E.A.; Lederberg, J. DENDRAL: A case study of the first expert system for scientific hypothesis formation. *Artif. Intell.* **1993**, *61*, 209–261. [\[CrossRef\]](#)
2. Corey, E.J.; Wipke, W.T. Computer-Assisted Design of Complex Organic Syntheses: Pathways for molecular synthesis can be devised with a computer and equipment for graphical communication. *Science* **1969**, *166*, 178–192. [\[CrossRef\]](#) [\[PubMed\]](#)
3. Laurencio, C.; Villien, L.; Kaufmann, G. Synthèse assistée par ordinateur de la phosphacarnéine-I Etablissement du plan de synthèse avec l'aide de pascop. *Tetrahedron* **1984**, *40*, 2721–2729. [\[CrossRef\]](#)
4. Roy, K.; Kar, S.; Das, R.N. *A Primer on QSAR/QSPR Modeling: Fundamental Concepts*; Springer International Publishing: Berlin/Heidelberg, Germany, 2015. [\[CrossRef\]](#)
5. Jones, N. Vibe coding for science. *Nature* **2026**, *653*, 348–350. [\[CrossRef\]](#) [\[PubMed\]](#)
6. O'Brien, G.; Eisty, N.U. Scientific Software in the Age of Vibe Coding. *Comput. Sci. Eng.* **2026**, *28*, 112–116. [\[CrossRef\]](#)
7. Horvat, M. What is Vibe coding and when should you use it (or not)? *TechRxiv* **2025**. [\[CrossRef\]](#) [\[PubMed\]](#)

8. Chou, Y.H.; Jiang, B.; Chen, Y.W.; Weng, M.; Jackson, V.; Zimmermann, T.; Jones, J.A. Building Software by Rolling the Dice: A Qualitative Study of Vibe Coding. *arXiv* **2025**, arXiv:2512.22418. [\[CrossRef\]](#)
9. Bianchini, S.; Müller, M.; Pelletier, P. Artificial intelligence in science: An emerging general method of invention. *Res. Policy* **2022**, *51*, 104604. [\[CrossRef\]](#)
10. Hoy, K.L. Solubility Parameter as a Design Parameter for Water Borne Polymers and Coatings. *J. Coat. Fabr.* **1989**, *19*, 53. [\[CrossRef\]](#)
11. Stefanis, E.; Panayiotou, C. Prediction of Hansen Solubility Parameters with a New Group-Contribution Method. *Int. J. Thermophys.* **2008**, *29*, 568–585. [\[CrossRef\]](#)
12. HSPiP: Hansen Solubility Parameters in Practice. Available online: <https://www.hansen-solubility.com/HSPiP/> (accessed on 29 July 2026).
13. Járvas, G.; Quellet, C.; Dallos, A. Estimation of Hansen solubility parameters using multivariate nonlinear QSPR modeling with COSMO screening charge density moments. *Fluid Phase Equilibria* **2011**, *309*, 8–14. [\[CrossRef\]](#)
14. Sanchez-Lengeling, B.; Roch, L.M.; Perea, J.D.; Langner, S.; Brabec, C.J.; Aspuru-Guzik, A. A Bayesian Approach to Predict Solubility Parameters. *Adv. Theory Simul.* **2018**, *2*, 1800069. [\[CrossRef\]](#)
15. Wojeicchowski, J.P.; Ferreira, A.M.; Okura, T.; Pinheiro Rolemberg, M.; Mafra, M.R.; Coutinho, J.A.P. Using COSMO-RS to Predict Hansen Solubility Parameters. *Ind. Eng. Chem. Res.* **2022**, *61*, 15631–15638. [\[CrossRef\]](#)
16. Mathieu, D. Pencil and Paper Estimation of Hansen Solubility Parameters. *ACS Omega* **2018**, *3*, 17049–17056. [\[CrossRef\]](#) [\[PubMed\]](#)
17. van Duin, A.C.T.; Dasgupta, S.; Lorant, F.; Goddard, W.A. ReaxFF: A Reactive Force Field for Hydrocarbons. *J. Phys. Chem. A* **2001**, *105*, 9396–9409. [\[CrossRef\]](#)
18. Senftle, T.P.; Hong, S.; Islam, M.M.; Kylasa, S.B.; Zheng, Y.; Shin, Y.K.; Junkermeier, C.; Engel-Herbert, R.; Janik, M.J.; Aktulga, H.M.; et al. The ReaxFF reactive force-field: Development, applications and future directions. *npj Comput. Mater.* **2016**, *2*, 15011. [\[CrossRef\]](#)
19. Mathieu, D. Modeling Sensitivities of Energetic Materials using the Python Language and Libraries. *Propellants Explos. Pyrotech.* **2020**, *45*, 966–973. [\[CrossRef\]](#)
20. Issartial, E.; Guenoux, T.; Nightingale, M. 2016. Available online: <https://www.holybird.fr/productions/commitstrip/> (accessed on 29 July 2026).
21. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
22. Murray, J.S.; Brinck, T.; Lane, P.; Paulsen, K.; Politzer, P. Statistically-based interaction indices derived from molecular surface electrostatic potentials: A general interaction properties function (GIPF). *J. Mol. Struct. THEOCHEM* **1994**, *307*, 55–64. [\[CrossRef\]](#)
23. Hansen, C.M. *Hansen Solubility Parameters: A User's Handbook*, 2nd ed.; CRC Press: Boca Raton, FL, USA, 2007. [\[CrossRef\]](#)
24. Novaes, F.J.M.; de Faria, D.C.; Ferraz, F.Z.; de Aquino Neto, F.R. Hansen Solubility Parameters Applied to the Extraction of Phytochemicals. *Plants* **2023**, *12*, 3008. [\[CrossRef\]](#) [\[PubMed\]](#)
25. Jeong, S.W.; Kim, Y.; Lim, H.K.; Chan Kim, M.; Song, C.H.; Lee, S. Prediction of enthalpy of vaporization for particulate matter through molecular dynamics using OPLS force field. *J. Aerosol Sci.* **2025**, *188*, 106595. [\[CrossRef\]](#)
26. Ticona-Chambi, J.; Carvalho, E.N.; Almeida, E.A.; Santo, A.M.; Infantes, L.; Cuffini, S.L. Optimized Hansen solubility parameters method for pharmaceutical solids: O-HSP-pharm. *J. Mol. Liq.* **2026**, *446*, 129267. [\[CrossRef\]](#)
27. Mathieu, D.; Ott, E.; Glorian, J. Misconceptions and Open Issues in Energetic Materials. *Propellants Explos. Pyrotech.* **2025**, *51*, 674–685. [\[CrossRef\]](#)
28. Mathieu, D. Fast Conversion of Molecular Diagrams into Plausible Crystal Structures Using Graph-Based Force Fields. *AI Chem.* **2025**, *1*, 2. [\[CrossRef\]](#)
29. Yang, S.; Bier, I.; Wen, W.; Zhan, J.; Moayedpour, S.; Marom, N. Ogre: A Python package for molecular crystal surface generation with applications to surface energy and crystal habit prediction. *J. Chem. Phys.* **2020**, *152*, 244122. [\[CrossRef\]](#) [\[PubMed\]](#)
30. Negre, C.F.A.; Alvarado, A.; Singh, H.; Finkelstein, J.; Martinez, E.; Perriot, R. A methodology to generate crystal-based molecular structures for atomistic simulations. *J. Phys. Condens. Matter* **2023**, *35*, 225001. [\[CrossRef\]](#) [\[PubMed\]](#)
31. Sander, T.; Freyss, J.; von Korff, M.; Rufener, C. DataWarrior: An Open-Source Program For Chemistry Aware Data Visualization And Analysis. *J. Chem. Inf. Model.* **2015**, *55*, 460–473. [\[CrossRef\]](#) [\[PubMed\]](#)
32. Yuan, S.; Chan, H.S.; Hu, Z. Using PyMOL as a platform for computational drug design. *WIREs Comput. Mol. Sci.* **2017**, *7*, e1298. [\[CrossRef\]](#)
33. Herráez, A. Biomolecules in the computer: Jmol to the rescue. *Biochem. Mol. Biol. Educ.* **2006**, *34*, 255–261. [\[CrossRef\]](#) [\[PubMed\]](#)
34. Dixon, S.L.; Duan, J.; Smith, E.; Von Bargen, C.D.; Sherman, W.; Repasky, M.P. Autoqsar: An Automated Machine Learning Tool for Best-Practice Quantitative Structure–Activity Relationship Modeling. *Future Med. Chem.* **2016**, *8*, 1825–1839. [\[CrossRef\]](#) [\[PubMed\]](#)
35. Berthold, M.R.; Cebon, N.; Dill, F.; Gabriel, T.R.; Kötter, T.; Meinl, T.; Ohl, P.; Thiel, K.; Wiswedel, B. KNIME—The Konstanz information miner: Version 2.0 and beyond. *ACM SIGKDD Explor. Newsl.* **2009**, *11*, 26–31. [\[CrossRef\]](#)

36. Turner, A.J.; Miller, J.F. Introducing A Cross Platform Open Source Cartesian Genetic Programming Library. *Genet. Program. Evolvable Mach.* **2015**, *16*, 83–91. [CrossRef]
37. Stephens, T. gplearn: Genetic Programming in Python, with a Scikit-Learn Inspired API. 2026. Available online: <https://gplearn.readthedocs.io/en/latest/> (accessed on 9 June 2026).
38. Kammerer, L.; Kronberger, G.; Kommenda, M. Symbolic Regression with Fast Function Extraction and Nonlinear Least Squares Optimization. In *Proceedings of the Computer Aided Systems Theory—EUROCAST 2022, Las Palmas de Gran Canaria, Spain, 20–25 February 2022*; Moreno-Díaz, R., Pichler, F., Quesada-Arencibia, A., Eds.; Springer: Cham, Switzerland, 2022; pp. 139–146.
39. Indiana University. *Quantum Chemistry Program Exchange*; Indiana University: Bloomington, Indiana, 1962. Available online: https://web.archive.org/web/20070711161206/http://www.qcpe.indiana.edu/cgi-bin/view_catalog.pl (accessed on 31 July 2024).
40. Smith, D.G.A.; Burns, L.A.; Simmonett, A.C.; Parrish, R.M.; Schieber, M.C.; Galvelis, R.; Kraus, P.; Kruse, H.; Di Remigio, R.; Alenaizan, A.; et al. PSI4 1.4: Open-source software for high-throughput quantum chemistry. *J. Chem. Phys.* **2020**, *152*, 184108. [CrossRef] [PubMed]
41. Thompson, A.P.; Aktulga, H.M.; Berger, R.; Bolintineanu, D.S.; Brown, W.M.; Crozier, P.S.; in 't Veld, P.J.; Kohlmeyer, A.; Moore, S.G.; Nguyen, T.D.; et al. LAMMPS—A flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales. *Comp. Phys. Comm.* **2022**, *271*, 108171. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.