Models, Parameterization, and Software: Epistemic Opacity in Computational Chemistry

Frédéric Wieber
Université de Lorraine, Archives Henri Poincaré—PReST

Alexandre Hocquet
Université de Lorraine, Archives Henri Poincaré—PReST

Computational chemistry grew in a new era of "desktop modeling," which coincided with a growing demand for modeling software, especially from the pharmaceutical industry. Parameterization of models in computational chemistry is an arduous enterprise, and we argue that this activity leads, in this specific context, to tensions among scientists regarding the epistemic opacity transparency of parameterized methods and the software implementing them. We relate one flame war from the Computational Chemistry mailing List in order to assess in detail the relationships between modeling methods, parameterization, software and the various forms of their enclosure or disclosure. Our claim is that parameterization issues are an important and often neglected source of epistemic opacity and that this opacity is entangled in methods and software alike. Models and software must be addressed together to understand the epistemological tensions at stake.

1. Introduction

"Many who do semiempirical calculations accept they are voodoo quantum mechanics and one has to go to the right witchdoctor" (Evleth 1993a). This extract from a post in a scientific mailing list, the "Computational Chemistry List," illustrates the issues of epistemic opacity in computational modeling methods in chemistry, their relations to computers and the development and use of software in a scientific milieu.

Perspectives on Science 2020, vol. 28, no. 5
 https://doi.org/10.1162/posc_a_00352

Computational chemistry[1] is in some respects the heir of quantum chemistry in the age of growing computing power available to computational science, but its lineage also traces back to other scientific and instrumental fields in chemistry (physical organic chemistry, protein chemistry, spectroscopies). As a scientific field, it has already been discussed by numerous authors, from the perspective of the history and philosophy of quantum chemistry (Gavroglu and Simões 2012; Park 2003, 2009), its more recent transformation into computational quantum chemistry (Lenhard 2014; Fisher 2016a, 2016b), and from the perspective of the development of molecular mechanics force fields in protein chemistry (Wieber 2012).

Computational chemistry emerged in a certain epoch and a certain context. It began as a relative minor client of the supercomputing resources of the 1960s and 1970s but rapidly grew to become a dominant actor in the scientific computing field (Bolcer and Hermann 1994). In the 1980s and 90s, when the Personal Computer and the workstation made computation more widely available in the laboratory, a new era of "desktop modeling" (Johnson and Lenhard 2011) coincided with a growing demand for modeling software, especially from the pharmaceutical industry (Richon 2008). The molecular modeling software became a huge potential market for hardware manufacturers to sell graphics terminals and computing power to Big Pharma, that was eager to dive into the techno-scientific promise of "rational drug design" (Hocquet and Wieber 2017). Moreover, in the 1980s, during the Reagan years, the universities (in the US at least) launched "technology transfer" programs. Spin-off companies were encouraged. Software produced in the university turned out to be viewed as potential revenue for academia (Berman 2012). Molecular modeling software became a central artifact in computational chemistry. The issue of how to transfer it from the developer to the potential users arose and raised tensions in the community. Computational chemistry software was local in the beginning. The only people involved with a scientific program developed for implementing a computational chemistry method were the group of scientists who coded it and the few colleagues who were scientifically collaborating with the developers. It progressively became oriented towards a "market." Software suites were envisioned, conceived, and designed, aggregating various methods of various research groups, and they had to be distributed, supported, and maintained. The relations with the pharmaceutical industry, with its culture of secrecy, exacerbated these tensions. This particular context allows us to emphasize that epistemological

1. Even if it can be argued that other scientific fields like planned synthesis (see for example Hepler-Smith 2018a) or database searching belong to "computational chemistry," we limit ourselves for the purpose of this study to the restrictive meaning of computational modeling of molecules, because it describes a community of practice, one that is sharing an interest on specific scientific methods, but also specific software.

issues associated with scientific modeling in computational chemistry are not only about methods but also about software.

The relations between computing and scientific activity have been the subject of numerous studies. Themes such as the "computerization of science" (Agar 2006), the mutual shaping of computing and biology (Chow-White and García-Sancho 2011), or the emergence of computerized evidence-based medicine (November 2011) explore their interplay. Yet, the history of software within computational scientific communities has attracted less attention. In his study of software used for simulating fluid dynamics, Spencer (2015) has nevertheless shown how analyzing the evolution of a program within a research group can lead to better comprehension of the evolving workability of computational models and the associated transformations of the practices and contexts of actions of computational scientists.

We advocate, along with Spencer, that scientific software deserves more attention in the epistemological discussions of models in computational science, and we argue that computational chemistry and its particular context is a pivotal case study. As Lenhard has pointed out, one of the dimensions of the transformation of *quantum* to *computational quantum* chemistry is that "the field of computational quantum chemistry became organized in a market-like fashion" (Lenhard 2014, p. 90). The adoption of this new type of organization was being driven by the commercialization of software, which has enlarged the community of its users. In his works on the various approximations and idealizations used by computational quantum chemists to study pericyclic reactions, Fischer has shown that computational chemists are in "a position of partial epistemic opacity with respect to the computational processes that produce numerical results" (Fisher 2016a, p. 320), and that what he calls computational diagnostics is a way to unpack computational models "in order to probe the impact of approximations and idealizations on the results" (Fisher 2016b, p. 253). He also adds, in a footnote: "another factor [of epistemic opacity] is ownership and access to proprietary software. [...] Perhaps the algorithms will always be partially epistemic opaque for reasons in addition to the automation of the computational processes" (Fisher 2016b, pp. 253–4, n10).[2] We follow Lenhard and Fischer in pointing out the interest

2. Since its introduction by Humphreys (2004, 2009), the notion of epistemic opacity has been debated in the epistemology of computer simulations (see for example Durán and Formanek 2018, or Jebeile 2018). As Jebeile (2018) sums up this notion: "computer simulations are epistemically opaque (Humphreys 2004); mainly they run too fast for one to follow the computational processes in detail and, even if it was possible to slow down the simulation, the simulation would still be too long to be cognitively grasped by a human mind" (p. 214). This kind of epistemic opacity is what Fisher (2016b) refers to when he considers, in the footnote we quoted above, "the automation of the computational processes" as a factor of epistemic opacity. In this article, we are more interested in the second factor of epistemic opacity he emphasizes on, that is "ownership and access to proprietary software."

of studying software in computational chemistry, and we aim at engaging into discussing the issue of epistemic opacity in the context of software.

Computational scientific models usually need the translation of a mathematical formalism into a computationally tractable language but their numerical equations also usually need parameters that are designed and defined to depict the model target and then tested, benchmarked, calibrated, and coded to make the model produce sound results. These tasks, bundled into the wording "parameterization," are, in computational chemistry, an arduous enterprise as we will show in a historical account further into this text, and we argue that this activity leads to tensions among scientists regarding the lack of epistemic transparency of parameterized methods and the software implementing them. We have analyzed previously the multiple tensions among the scientists in this community who develop, license, distribute, and use software in intertwined academic and industrial contexts (Hocquet and Wieber 2017). In the present paper, we emphasize the epistemological issues at stake, specifically regarding epistemic opacity within the pivotal issue of parameterization. We first offer a brief account of the two different epistemic cultures, namely quantum chemistry and molecular mechanics, from which computational chemistry descends. These two cultures share unifying parameterization concerns and the scientific methods associated with them are merged in software, leading to the advent of the common "technical knowledge community" of computational chemistry (Johnson 2009). We then discuss a lively flame war episode of the CCL to emphasize what issues are at stake for a diversity of involved actors, in order to assess, in detail, the relationships between modeling methods, parameterization, software and the various forms of their enclosure or disclosure. Our claim is that parameterization issues are a source of epistemic opacity and that this opacity is entangled in methods and software alike. Models and software must be addressed together to understand the epistemological tensions at stake.

2. Historical Perspective on Methods and Parameterization

Models in computational chemistry have roots in two distinct fields in the history of chemistry. The mathematical modeling of molecules is an idea which came up to chemists long before computers were available. Quantum chemistry is a scientific field that emerged in the late 1920s. Theoretical physicists left quantum chemists with a very practical problem: to imagine theories (and models) to describe the molecules in a way that could be calculable and useful to the chemists (Gavroglu and Simões 2012). They were the heirs of a reductionist world view of micro-physics, and the naming of the most popular quantum chemical approximation ("ab initio") reflects the idea that the scientific soundness of an approximation method is based on the universality of the constructed models which should not have to deal

with the tinkering of parameters (Park 2009). In practice, calculating was done with pencil and paper, desk calculators, and subsequently, during the 1950s, with the use of excess computer time on the first supercomputers (Bolcer and Hermann 1994), with little reward in terms of how big the molecules that could be actually calculated were (Park 2003).

Parallel to this, in the 1950s and 1960s, and because the computing facilities were attracting the interest of many scientific fields, a different way of producing computational models of molecular structure arose, based on far simpler theoretical grounds, on a classical conception of molecules in chemistry, and developed entirely on the pragmatic idea of tackling the modeling of what is actually computable. Organic chemists and biophysicists showed interest in a theory based on a Newtonian classical mechanics view of molecules which appeared in the 1950s for the conformational analysis of strained organic molecules (Westheimer 1956). The very writing of skeletal molecular formulas, and their 3D incarnation, the ball-and-stick model (Francoeur 2001), is the core of a chemist's epistemic culture, one that relies on the chemical concepts of atoms and bonds, in contrast to the microphysical concepts of nuclei and electrons. This simplified Newtonian view was not dissimilar to what chemists knew as a molecule from the nineteenth century molecular models, and to how spectroscopists theorized the molecular vibrations (Wilson et al. 1955).

The benefit from this mathematically simplistic theory was the prospect to compute the structure of molecules ranging from the smallest to the most frequently encountered in organic, biological and pharmaceutical chemistry, by the computational standards of that time (Wieber 2012). It was ad-hoc modeling, based on tinkering parameters to fit experimental results, yet a tractable one, because of its straightforward computability. This tractability was also to the detriment of the universality of the model: this ad-hoc modeling proved successful for a limited (but meaningful) number of molecular families (like cycloalkanes, peptides, sugars, …) and the necessary parameterization to achieve results was limiting, because consistent parameterization was the lengthiest and the hardest task of the modeling activity. It could only be achieved for limited molecular families. Each scientific team developed and parameterized their own method (a so-called "empirical force field"). Each team relied on different (and often competitive) protocols, based on different (and sometimes incompatible) spectroscopic or thermodynamical results, to actually define their parameters, to iteratively refine them, and to apply their parameterization to specific molecular domains, like for example, small organic chemistry, polymer chemistry, or protein chemistry.

Strategies of parameterization were supposed to first define "atom types." Tetrahedral and trigonal carbon, for example, could not use the same linear, angular and torsional parameters. Allegedly, alkene trigonal carbon and ketone trigonal carbon neither. Defining a set of "atom types" was thus a

fragile balance between oversimplification (resulting in poor accuracy of the produced results) and overspecialization (resulting in an exponentially increasing amount of parameters to define, compute, calibrate, fit, benchmark…). Specialization into a molecular family was the only way to mitigate the task. Parameters were also to be validated according to some reference. Results from X-ray diffraction were used to compute crystallographic geometries, which would compare with molecular mechanics geometry optimizations. Another source of geometric parameters could be quantum calculations on benchmark molecules. This lack of consistency regarding parameters validation implies that parameters were not interchangeable from one empirical force field to the other. Yet, one important point is that missing parameters were a common caveat. When parameters were missing, the only workaround to actually make the computation run and not halt, was sometimes the use of ad-hoc "guessed" parameters. Thus, the mere tagging of atom types (ketone trigonal carbon, alkene trigonal carbon…) in a molecule was pivotal to the soundness of the results. It was also the biggest source of an inflation of parameters: the simple addition of one new atom in a molecule could result in an overwhelming avalanche of new parameters to be defined and tested. Finally, this avalanche could be the cause of missing parameters, and thus calculations coming to a grinding halt, which in turn led to the design of coding tricks to avoid them.

The demarcation between quantum chemistry and molecular mechanics became blurred throughout the evolution of their respective fields, especially as the promises of the computer arose. The famous Boulder discourse of 1959 by the renowned quantum chemist Charles Coulson at the Conference on Molecular Quantum Mechanics (Park 2003) was a turning point when Coulson acknowledged a schism between two irreconcilable groups within quantum chemistry. In particular, during the 1960s, the so-called "semiempirical" methods emerged. Their name was itself a pun on the compromise they represented. They were based on quantum calculations and thus formed a part of quantum chemistry, but they shared the concern for feasibility with molecular mechanics (aka "empirical" methods, as molecular mechanics were sometimes called). In order to be actually tractable, the quantum methods should be simplified, and above all, parameterized to achieve computability (the most lengthy calculations of the model should be replaced by empirical parameters). Similarly to molecular mechanics, different and sometimes competitive semiempirical methods, based on different parameterizations, appeared in the 1970s.

Semiempirical methods were akin to ab-initio methods because they shared the same theoretical formulations, but parameterization in semiempirical methods shared some concerns with parameterization in molecular mechanics. Different sources of parameters were used, not only regarding geometries, but also thermodynamical or energetic quantities, and the same lack of consistency

regarding parameters definitions and validation protocols led to distinct and competitive methods, just like competitive force fields had arisen in molecular mechanics. Similarly to "atom types" in molecular mechanics, the parameterization in semiempirical methods led to the "missing parameter" caveat, and molecules that included some exotic elements (besides the ever-present carbon, hydrogen, nitrogen and oxygen) could not be calculated, or were calculated with "guessed" parameters.

The plurality of methods (Fisher 2016b) in theoretical and computational chemistry, based on different but communicating epistemic cultures, gradually turned to meet each other with the advent of the computer (and the computer software) as a unifying tool. Quantum chemistry and molecular mechanics were far from ignoring each other, and as a matter of fact, the birth of the computational chemistry discipline, as viewed by its founders, encompassed both fields in a wide spectrum of modeling activities ranging from "pure" ab initio quantum modeling to so-called "empirical" molecular mechanics. Pragmatic quantum chemists, using semiempirical methods, were positioned somewhere in the middle, as Richard Counts (then editorialist of the new "Journal of Computer-Aided Molecular Design") defines it (Counts 1987).

Another quantum-based modeling method arose during the 1990s, the so-called density functional theory (DFT). Like semi-empirical methods, it hybridizes quantum formalism and parametrization. It is based on solving the Schrödinger equation using as a variable the overall electronic density of the chemical object instead of the electronic wavefunctions, as it is the case in strict "ab initio" methods. Electronic density requires the definition of a so-called "exchange and correlation function" whose mathematical expression must be parameterized (again, unlike ab initio methods) to fit to experimental or higher-level theoretical results. The parameters are however independent of the chemical object considered, unlike the parameters of the empirical and semi-empirical methods. DFT has arguably been the most popular quantum-based computational chemistry method in the twenty-first century; it has been used by Lenhard (2014) as the archetypal method of what he calls "computational quantum chemistry."

Using this plurality of modeling strategies is typical in computational chemistry. Moreover, these strategies permeate one another. For example, results from ab initio calculations can be used to define parameters in a molecular mechanics force field. Within this wide spectrum of modeling strategies, parameters tinkering[3] is an important and daily practice. We now focus

3. "Parameters tinkering," a widely used phrasing among computational chemists, consists in the intervention on parameters based on pragmatic ad hoc strategies to improve a model or even merely make the program work. Examples include the modification of a parameter value as a rule of thumb, the replacement of an unknown parameter with a similar one, the programming of a routine to deal with the lack of some parameters, etc…

on semiempirical and molecular mechanics modeling strategies in the next section as parameters tinkering is a core characteristic of the way they work. We discuss how actors debate parameters, modeling methods, and software in a flame war which occurred in 1993 on the scientific mailing list devoted to computational chemistry, the CCL.

3. A CCL Flame War

The CCL was created in 1991 to gather members of the fledgling community. Topics on the CCL include questions and answers about technical details, personal opinions about hardware or software, commercial software announcements, and also scientific matters (Wieber et al. 2018). Because it was public, accessible and open to all, the CCL soon became an informal arena in which all the actors involved with computational chemistry could interact: from graduate students to senior researchers, developers and users from academia or the industry, software vendors, people from supercomputing centers, hardware vendors, etc… The CCL was the arena where all the people linked to molecular modeling software one way or another could debate. From a historical point of view, the lively episodes called "flame wars" are the most interesting in what they reveal of controversies. Controversial debates force the actors to leave their formal and polite stance. The uniqueness of the CCL as a corpus to account for the tensions induced by software in the community has been described in another publication (Wieber et al. 2018).

We focus here on one of these lively episodes, a flame war from 1993, to explore a diversity of opinions regarding models, software and the complex issue of parameterization. The first message is an announcement. Andy Holder, then Assistant Professor of Computational/Organic Chemistry at the University of Missouri-Kansas City and CEO of a scientific software company named Semichem, Inc.[4], announced the publication of a paper providing results for a new quantum chemistry semiempirical method named "SAM1." Holder posts: "This [the paper] is primarily a listing of results for the new method for a vast array of systems. […] A more complete paper describing the model will be forthcoming" (Holder 1993a). This last sentence will launch the debate. Twenty-nine posts from eighteen subscribers will follow for ten days. Graham Hurst (then an employee of the software company Hypercube, Inc.[5]) posts the second message of the thread: "this [Holder's] post disturbs me…" (Hurst 1993a). Hurst considers that "it will be impossible to independently reproduce these results" because the model leading to the results has not already been

4. Semichem, Inc. was a company founded to commercialize the SAM1 method (and other similar semiempirical methods) embedded in the AMPAC software package.

5. Hypercube, Inc. is a software company commercializing the multipurpose Hyperchem package. Hyperchem implements many semiempirical and molecular mechanics methods.

published. "If the method has not yet been published, then the results should not have been accepted for publication since they cannot be verified," he adds.

Three directions of discussion are opened up in response to Hurst's post. First, the issue of how possible it is to verify the validity of the results is discussed as the possibility to reprogram the computational method by oneself. Is the information necessary to reprogram the method available? Mark Thompson, then research scientist at the Pacific Northwest Laboratory and developer of a freely licensed molecular modeling program called Argus, explains the difficulties he came across while trying to reprogram the MNDO method with only the publications at hand:[6] typos, inconsistencies in quantities units (due to the importation of geometrical or energetical parameters from experimental results in units used by experimentalists like angstroms for distances or kcal/mol for energies), and non-explicit importation of parameters from other semiempirical methods. He concludes his post by asking "Would anyone else out there who has implemented the MNDO-family of methods care to comment on their experiences?" (Thompson 1993a). Other posters comment about the long lasting of parameters errors in the published literature (Rzepa 1993) or the anomaly of sulfur containing molecules claimed in a submitted paper to have been calculated with a program version that officially does not include any sulfur parameter for the AM1 method. While checking how this is possible, Evleth, a researcher at the CNRS French institution in Paris, says he learned that the sulfur parameters had been implicitly imported from MNDO, another method, without any testing at all (Evleth 1993b). Authors of said paper had then to acknowledge retrospectively that both methods were using "mixed AM1-MNDO parameters" for some chemical elements. The issue of difficult attempts at reprogramming methods illustrates the messiness of parameterization. Evleth concludes in another message, the same day, that "many who do semiempirical calculations accept they are voodoo quantum mechanics and one has to go to the right witchdoctor" (Evleth 1993a).

Similarly, on molecular mechanics, Hurst (1993b) then adds his experience of coding various force fields in the Hyperchem package, and his having a hard time obtaining official lists of parameters to refer to. Said parameters are sometimes claimed to be listed in a reference publication (or doctoral thesis) that happens to contain only partial and incomplete specifications.

A more general discussion then opens up regarding the issue of the parameters that are used in semiempirical and molecular mechanics methods. These

6. MNDO, AM1, and SAM1 are all semiempirical methods of the same "family," in the sense that they are developed in the same research group. Holder is the only poster that belongs to this research group. Thompson attempts to implement MNDO in his "Argus" free software package.

parameters, central in the different methods used, are not always made publicly available. They are sometimes hidden away in the source code of the program, which is not always made public.[7] Hurst cites (perhaps apocryphally) Allinger, the father of the MM2 molecular mechanics method: "the only complete specification [of the force field] is the program itself" (Hurst 1993b). It exemplifies the difficulty to get a hand on parameters, the fact that parameters are not always available publicly in publications, and moreover that the parameters available in programs are sometimes enclosed in a non-open code. As one of the participants of the thread writes: "[...] we should like to know your opinion on the actual trend in commercializing computational packages without source codes. Does this trend encourage the development of science? And also: up to what limit a computational package can be considered as a product of a single research group?" (Adamo 1993).

In the second direction of the discussion, the tension between the world of academic research and the world of scientific software corporations is underlined. In response to Hurst's first message, Holder concedes that it is not always easy to clearly distinguish scientific from entrepreneurial activities. The scientists' implication in scientific software corporations, along with the costs necessary to develop software, clashes with the values the actors associate with science. As Holder puts it: "So, while Dr. Hurst's point is well-taken and fully subscribed to by me both in my capacity as a university researcher and president of Semichem, there is no intention to "hide" anything. I understand the sensitivity of this issue and I am committed to the pursuit of science in an *open atmosphere*. [...] The development of SAM1 is my primary research activity [...], but Semichem is also spending money to develop this method and will be giving it to the scientific community freely. We withhold only our code. [...] It should be noted, however, that *some interests are not scientific, but competitive*" (emphases added) (Holder 1993b).

In the third direction of the discussion, the problem of publication ethics is discussed. The importance of the peer review process in scientific publishing is underlined and some contributors ask if reviewers do a good job when accepting for publication results which have been obtained by a computational method not fully (and openly) described. The question leads more generally to contrast proprietary methods and open scientific literature. As Mark Thompson writes down: "I feel very strongly that when a new method is developed and implemented that it must pass the peer review process to gain

7. Computational programs may be distributed in the form of ready to use "executables" or "binaries" that lack the possibility to scrutinize what is actually written in the code. The "source code" offers this possibility, but then must be compiled into an executable by the user. An "open" source code can mean that it is readable (but not necessarily intelligible), but a precise definition of what "open" source code means (modifiable? reusable?) is embedded into the software licensing and may vary.

legitimacy in the scientific community, regardless of whether most other scientists care to re-implement that method or not. Proprietary methods are fine, as long as it is openly known that they are proprietary. Results of proprietary methods do not belong in the open scientific literature" (Thompson 1993b). Of course, these three directions of discussion are interrelated.

The sixth message of the thread, written by Douglas Smith (then Assistant Professor of Chemistry at the University of Toledo) is particularly revealing. In this long post, Smith responds point-by-point, using interleaved posting, to Thompson's whole message. The tensions produced by software within the community are interestingly expressed by contrasting how scientists believe they should act with what they actually do. Thompson has written that "good science is that of reproducibility and independent verification" (Thompson 1993b). Smith points out that it is "universally true and accepted" but "rarely followed" (Smith 1993). Smith uses as an example the issue of parameters used in molecular mechanics, which are regularly modified and adjusted for a particular study without being published in the paper relating to that particular study. More generally, the very nature of such a method (and of semiempirical methods) leads to a multiplication of the parameters used without a clear display of which parameters are used when producing such or such results. In actuality, computational chemists act in a way that differs from what they say they should do. Thompson has also written: "If the results of a new method are published without sufficiently describing the method to fulfill the above criteria [reproducibility and independent verification], then I personally could not take the results seriously" (Thompson 1993b). Here again, Smith considers that if this position points to "a real problem," it is "utopian and most likely not practical," because of "the proprietary nature of commercial software" (Smith 1993), and because some people use this type of software as a "black box." He then adds: "Besides, who ever said we had to reveal all our secrets and make them readily available and accessible? When software copyrights and patents really provide adequate protection, maybe I will agree with that attitude" (Smith 1993). Finally, if "results of proprietary methods do not belong in the open scientific literature," as Thompson has written, "where do they belong?" Smith replies. According to him, the situation is complicated: "what about the difference between someone in industry who paid for the source code for MacroModel[8] as compared to the academic, such as myself, who only gets binaries? Are my results to be less acceptable because I don't have the absolute method available? Or are the industrial results less acceptable because they can be the results of tweaking the code?" (Smith 1993).

8. MacroModel is a software package implementing several molecular mechanics methods, including Allinger's MM2 force field, and adding in-house parameters.

It is worth noting that posters switch easily from semiempirical methods to molecular mechanics and back when they talk about their concerns regarding publications, methods, parameterization and software. These two domains share the same concerns and are bound in similar and sometimes even the same software, even if the theoretical formulations they use are very different.

In Smith's post, the discrepancy between the values the actors associate with science and their actual practices associated with computational methods and software is clearly highlighted. The issue of the norms of sound science is, in practice, difficult for them to address.

Moreover, computational chemists ask the question of how the difficult and tedious work of programming can be recognized. Can this recognition be obtained by publishing programs or by adequately protecting them ("When software copyrights and patents really provide adequate protection [...]" Smith writes)? The complexity of the issue of software copyrights and patents is then stressed in many subsequent posts of the thread.

Some posters try to separate the issue of copyright and patenting, as a software issue, from the issue of transparency of methods and publication, as a science issue. Hurst (1993b) makes the strong claim that "it is important to distinguish 'science' from 'code'" (his emphasis). The former should "include everything a researcher needs to know to reproduce numbers" and the latter "need not be fully or publicly disclosed." But this dichotomous view is criticized. Balducci (1993), a research associate and systems manager at the university of Texas in Austin, opposes that a lot of work in the code, such as defining molecular geometry (and especially ring structures), belongs to science: "in several cases it would be impossible to even describe (much less to reproduce) the 'science' of a method without a clear definition of the structure of the 'coded' solution."[9] Mercier, then at Cornell school of medicine, regrets that method coding is often "hardwiring tables of parameters into the code" (Mercier 1993) and makes them difficult to comprehend. He refers to modular programming, such as in Mathematica software, as a mean to separate parameters from an enclosed code in order to give the community the possibility to enhance the parameterization of a hard-coded method.

Finally, Fernandes, then an undergraduate student at the University of Waterloo, insists that even an open (in the sense of readable) source code is not sufficient to understand the method and its parameterization. Lack of code annotation and obscure versioning do not help: "what guarantee do we have that G92 (or any of Biosym's or AutoDesk's products) actually do what they are supposed to? Even having the source code just doesn't help... who wants to

9. Defining computationally a chemical structure with atoms and bonds is actually one of the other scientific domains belonging to computational chemistry in a broader sense we mentioned in the first footnote (Hepler-Smith 2018b).

root through 350,000 lines of someone else's code for any reason?" (Fernandes 1993). The thread finally dies of attrition after a general sense of uncertainty about what the future holds regarding the relationships between intellectual property notions and the tensions they expressed beforehand.

4. Parameters as Opacity in Methods and Software

The initial problem of the flame war is an epistemological problem entangled with a problem of publication ethics: as the details of the model used to produce the published results have not been published, the results cannot be independently reproduced and verified and are then not considered publishable. Beyond this problem, the tensions revealed by the conversation thread are the symptomatic expression of opacity regarding parameters, methods and software.

Andy Holder's first post speaks about a future paper that will "describe the [SAM1] model" and Hurst's infuriated answer states that "the SAM1 *method* still does not have an *official* reference" (his emphasis). Two subsequent posters speak of disclosure of "semiempirical parameterization" (Evleth 1993b) and "disclosure of parameters" (St-Amant 1993). As a matter of fact, if the titles in the subject headers of the first posts are about "SAM1 reference," there is a shift in titles towards "full disclosure of methods" after two days of posting, and actors even insist on the issue of "disclosure of programs" in subsequent messages. The issue is about what exactly has to be disclosed—models, methods, parameters, programs, software—and how to ensure transparency.

The epistemological nature of the models in computational chemistry implies that epistemic transparency is very difficult to reach in practice. The very nature of the models, for example in semiempirical methods like SAM1 or in molecular mechanics approaches like MM2, requires the time-consuming work of parameterization. Parameterization poses a problem of reproducibility and transparency. Tinkered parameters that are designed to make the model actually work are subject to their own epistemic problems.

Numerous research groups build numerous molecular mechanics force fields, and the essential work of parameterization in this construction is sometimes developed in a competitive atmosphere. Force field success is measured in terms of parameters efficiency (to produce sound results out of well-defined parameters for the calculation of properties of a molecule), consistency (to produce reproducible results from consistently defined parameters across a variety of molecules) but also workability (to avoid calculation failures because of the absence of parameters that lead to a program halt). In force field construction, some parameters are missing, some generic parameters are designed to replace missing parameters to avoid program halt, and some parameters are lacking a sufficient description in order to know if a parameter is proper or simply fills a hole. The situation of force field multiplicity is even made more complex by the fact

that competitive software packages may implement a certain force field differently, especially in the treatment of missing parameters. Furthermore, other research groups may adapt a force field to their particular calculation needs by adding layers of "in-house" parameters.

Semiempirical methods show similar issues due to a similar complex situation. Holder (1993b) says that it is impossible to reproduce the MNDO method "from scratch," because publication is incomplete. Yet, Thompson tried to actually reprogram MNDO "from scratch," without the source code, with only the mere paper in hand. Errors, units discrepancies, and mixed parameters (importation of parameters from one method to another to make the program run) were the difficulties encountered by Thompson and others.

Given this diversity of parameterization and parameters descriptions, methods and their parameters are not consistently disclosed, and this is a source of epistemic opacity. Actors ask whether they could be, for example in a publication's supplementary material. In the end, the question actors are asking is whether a fully parameterized method could be properly described and then sufficiently disclosed in a publication.

In practice, this ideal of transparency is hard to reach because parameters are intertwined with the coding of the method. As several actors mention, the parameters are often "hardwired" into the code. In one poster's (quoted) words, "the only full description of the method is the program itself." The entanglement of parameters and code is adding a new layer of opacity in the issue of reproducibility.

A serious issue regarding parameters embedded in code is the fact that the code may not be open, in the sense of not readable. If the code of the program is not open, then executable binaries are what software users get. Parameters, which are in that case hidden in an enclosed program, thus cannot be checked.

So why do developers "withhold the code" in Holder's words (Holder 1993b)? Because software is more than just code, it is also a commodity, and the scientific activity of computational chemistry shares common concerns with the industrial sector of software sales. In a world where software is also a business, issues of intellectual property, or software distribution in general, entwine with the traditional concerns of the scientific world. The difficulty to finance continuous development clashes with scientific ethos concerns. This situation gives rise, for example, to the problem of the lack of scientific recognition for software development. Many actors then judge that it is important to gain recognition for software development, and in this regard, to protect and enclose the code is vital, even if methods should be reproducible. Modular programming is cited as a means to give the community the possibility of disclosing parameters while leaving the code enclosed.

In the same vein, Hurst (1993b) claims that "science and code must be separated," but this attempt to separate methods and software is viewed by

some posters as vain. He is reminded that the entanglement of both is inextricable. For example, Balducci (1993) advocates that beyond mathematical equations formalizing a model, the computational definition of molecular structure itself to effectively compute the calculations (molecular geometry, and especially ring structure) is intermediate between science and code, and belongs to both.

Is code openness, as a proposal to solve this opacity issue, a solution? The idea that any scientific code should be open is not straightforward. Some posters regard the commercialization of computational chemistry packages as having many advantages. First, instead of spending time and energy in building code, the use of already existing code can be viewed as the possibility to dedicate oneself to actual chemistry (publishable) calculations, instead of (unrewarded) programming. Second, financing the computational effort of developing software through software sales is seen in some quarters as the only way to have the possibility that robust, versatile, powerful, and efficient software packages even exist. And third, from an industrial user point of view, buying commercial software grants liability from a corporate software vendor in case of something going wrong. Some posters then view software packages as a metaphoric scientific instrument: commercialization is seen as building trust in the scientific community.

There is however no consensus on this matter. Posters cite many examples of the need of an open source code for sound scientific practices, and one of them is precisely the need for parameters verifiability. Tensions proceed from the confrontation of two viewpoints: one concerned with epistemic transparency as scientific ethos, and the other concerned with stability thanks to software as a commodity. Academic publishing, which constitutes the traditional form of academic reward, is central in the actors' ideal concept of the openness of science. Yet, there is a tension between two stances. One is that modeling software, as a scientific tool, should be considered a public tool, and as such, one that belongs to the scientific community, including its potentiality to be disclosed, enhanced, and maintained. The other is that, software, as a tool developed by a small team, in a commercial context, should be licensed, strict licensing policies helping to keep software stable, which guarantees the production of sound scientific results.

Yet, even if the code is made public, the opacity also lies in the complexity of the programs: checking programs out (beyond merely testing results) is very difficult according to Fernandes (1993). It is not easy to assess whether the program behaves as intended by the code developers: lack of code annotations and versioning issues are again sources of opacity, even in a readable code.

Finally, the opacity also lies in the software package licensing policy that impedes checking of parameters. Smith (1993) evokes the paradoxical issue of industrial or academic users of the same software package. The former get

access to the source code and thus have the possibility to tinker force field parameters "in house" and then publish computational results that can be criticized for unsoundness (are the parameters used thoroughly tested?) and lack of traceability (are the modified parameters published?). The latter only get access to binaries and their calculations may be criticized for using a method whose parameters they don't know exactly. In this regard, packaging and licensing policies add yet another layer of epistemic opacity.

The flame wars as we have depicted one of them are thus the symptoms of tensions about a lack of epistemic transparency regarding methods and software. First, the opacity of methods' parameters lie in their diversity, and this implies issues regarding publications. Methods are opaque if parameters are unavailable. Second, parameters are intertwined with code, which adds a layer of opacity, be it open or not. Third, programs are themselves opaque because of their complexity and lack of traceability. The final layer of opacity lies in the policies of software as packages. We have shown that this epistemic situation and the tensions implied by parameterization at any level, are constitutive of computational chemistry. It also implies that the concepts of verification (does the model perform in a consistent way mathematically and computationally?) and validation (does the model do a good job at depicting its target?) are actually confluent, as Winsberg hints (Winsberg 2018). The entanglement between parameters and software can lead to this confluence.

5. Conclusion

The issue of parameterization as a source of epistemic opacity in computational chemistry is a telling example that models and software must be addressed together in computational science. Interrelations between both imply that transparency and validity of computational methods are complex, and that they are a source of tensions for scientists, in the economic, political and technological context we mentioned.

It is interesting and necessary to discuss the structure, properties and epistemological status of models, as it is common in the philosophy of science. We argue that it is furthermore necessary to understand models in relation with software which embody them, which give them their productivity. In turn, understanding software (in computational sciences) needs to take into account the models they express, which is "the representations of the world" scientists translate in a way the computer can "understand." These representations depend on the communities of scientists involved and the histories of the ways they represent the portion of the world they are interested in (Mahoney 2008). In Mahoney's words, these models, and their translations into software, are "operative representations", which are central to our study, and parameterization is pivotal to this entanglement.

The complexity of the parameterization is central in the modeling activity. This has to be understood in the context of the calculability problems quantum approaches and molecular mechanics approaches have faced. In the case of molecular mechanics methods, the choice of a particular representation of matter, which is consistent with a classical conception of molecules, also leads to a necessary complex work of parameterization. The choices of sets of parameters, made locally by such or such research group for such or such group of molecules, lead to models whose epistemic transparency is questioned by the actors themselves. What is interesting for our argument is that this lack of transparency of models has consequences on the status of software: the issue of the openness of the source code is for example made more salient knowing the importance of parameterization in modeling. What is also interesting is that there are similar concerns with semiempirical methods, even though the latter are quantum methods. In this regard, molecular mechanics and semiempirical quantum methods share a fundamental epistemic issue, especially since they are bound in similar ways to software, and sometimes even within the same package, even though they are attached to different (and even incompatible) theories. The phrasing "Voodoo quantum mechanics," as used by one of the actors of the flame war we have narrated, ironically highlights the issues of opacity in methods, parameterization, and software that literally possess computational chemists, who have then to rely on "the right [but often evasive] witchdoctor."

References

Adamo, C. 1993. Message: 1993.06.28-005, http://www.ccl.net/cgi-bin/ccl/message-new?1993+06+28+005.

Agar, Jon. 2006. "What Difference Did Computers Make?" *Social Studies of Science* 36(6): 869–907. **DOI:** https://doi.org/10.1177/0306312706073450

Balducci, R. 1993. Message: 1993.06.30-013, http://www.ccl.net/cgi-bin/ccl/message-new?1993+06+30+013.

Berman, Elizabeth Popp. 2012. *Creating the Market University: How Academic Science Became an Economic Engine*. Princeton: Princeton University Press. **DOI:** https://doi.org/10.1515/9781400840472

Bolcer, John D., and Robert B. Hermann. 1994. The Development of Computational Chemistry in the United States. Pp. 1–63 in *Reviews in Computational Chemistry*, vol. 5. Edited by Kenny B. Lipkowitz and Donald B. Boyd. New York: Wiley. **DOI:** https://doi.org/10.1002/9780470125823.ch1

Chow-White, Peter A., and Miguel García-Sancho. 2011. "Bidirectional Shaping and Spaces of Convergence: Interactions Between Biology and Computing from the First DNA Sequencers to Global Genome Databases." *Science, Technology & Human Values* 37 (1): 124–164. **DOI:** https://doi.org/10.1177/0162243910397969

Counts, Richard W. 1987. "Strategies I." *Journal of Computer-Aided Molecular Design* 1(2): 177–178. **DOI:** https://doi.org/10.1007/BF01676961

Durán, Juan M., and Nico Formanek. 2018. "Grounds for Trust: Essential Epistemic Opacity and Computational Reliabilism." *Minds and Machines* 28 (4): 645–66. **DOI:** https://doi.org/10.1007/s11023-018-9481-6

Evleth, E. 1993a. Message: 1993.06.28-002, http://www.ccl.net/cgi-bin/ccl/message-new?1993+06+28+002.

Evleth, E.. 1993b. Message: 1993.06.28-001, http://www.ccl.net/cgi-bin/ccl/message-new?1993+06+28+001.

Fernandes, A. 1993. Message: 1993.07.01-006, http://www.ccl.net/cgi-bin/ccl/message-new?1993+07+01+006.

Fisher, Grant. 2016a. Divergence, Diagnostics, and a Dichotomy of Methods. Pp. 306–331 in *Essays in the Philosophy of Chemistry*. Edited by Eric Scerri and Grant Fisher. Oxford, New York: Oxford University Press.

Fisher, Grant. 2016b. "Diagnostics in Computational Organic Chemistry." *Foundations of Chemistry* 18(3): 241–262. **DOI:** https://doi.org/10.1007/s10698-016-9253-4

Francoeur, Eric. 2001. Molecular Models and the Articulation of Structural Constraints in Chemistry. Pp. 95–115 in *Tools and Modes of Representation in the Laboratory Sciences*. Edited by Ursula Klein. Dordrecht: Kluwer Academic Publishers. **DOI:** https://doi.org/10.1007/978-94-015-9737-1_7

Gavroglu, Kostas, and Ana Simões. 2012. *Neither Physics nor Chemistry: A History of Quantum Chemistry*. Cambridge: MIT Press. **DOI:** https://doi.org/10.7551/mitpress/9780262016186.001.0001

Hepler-Smith, Evan. 2018a. "'A Way of Thinking Backwards' Computing and Method in Synthetic Organic Chemistry." *Historical Studies in the Natural Sciences* 48(3): 300–337. **DOI:** https://doi.org/10.1525/hsns.2018.48.3.300

Hepler-Smith, Evan. 2018b. "Paper Chemistry: François Dagognet and the Chemical Graph." *Ambix* 65(1): 76–98. **DOI:** https://doi.org/10.1080/00026980.2017.1418232, **PMID:** 29318947

Hocquet, Alexandre, and Frederic Wieber. 2017. "'Only the Initiates Will Have the Secrets Revealed': Computational Chemists and the Openness of Scientific Software." *IEEE Annals of the History of Computing* 39(4): 40–58. **DOI:** https://doi.org/10.1109/MAHC.2018.1221048, https://doi.org/10.1353/ahc.2017.0030

Hocquet, Alexandre, and Frederic Wieber. 2018. "Mailing List Archives as Useful Primary Sources for Historians: Looking for Flame Wars." *Internet Histories* 2(1–2): 38–54. **DOI:** https://doi.org/10.1080/24701475.2018.1456741

Holder, A. 1993a. Message: 1993.06.23-013, http://www.ccl.net/cgi-bin/ccl/message-new?1993+06+23+013.

Holder, A. 1993b. Message: 1993.06.25-004, http://www.ccl.net/cgi-bin/ccl/message-new?1993+06+25+004.

Humphreys, Paul. 2004. *Extending Ourselves: Computational Science, Empiricism, and Scientific Method*. Oxford: Oxford University Press. **DOI:** https://doi.org/10.1093/0195158709.003.0002

Humphreys, Paul. 2009. "The Philosophical Novelty of Computer Simulation Methods." *Synthese* 169(3): 615–626. **DOI:** https://doi.org/10.1007/s11229-008-9435-2

Hurst, G. 1993a. Message: 1993.06.24-000, http://www.ccl.net/cca/archived-messages/1993/06/24.

Hurst, G.. 1993b. Message: 1993.06.29-000, http://www.ccl.net/cca/archived-messages/1993/06/30.

Jebeile, Julie. 2018. "Explaining with Simulations: Why Visual Representations Matter." *Perspectives on Science* 26(2): 213–38. **DOI:** https://doi.org/10.1162/POSC_a_00273

Johnson, Ann. 2009. "Modeling Molecules: Computational Nanotechnology as a Knowledge Community." *Perspectives on Science* 17(2): 144–173. **DOI:** https://doi.org/10.1162/posc.2009.17.2.144

Johnson, Ann, and Johannes Lenhard. 2011. Toward a New Culture of Prediction: Computational Modeling in the Era of Desktop Computing. Pp. 189–200 in *Science Transformed?: Debating Claims of an Epochal Break*. Edited by Alfred Nordmann, Hans Radder, and Gregor Schiemann. Pittsburgh: University of Pittsburgh Press. **DOI:** https://doi.org/10.2307/j.ctt5hjssc.18

Lenhard, Johannes. 2014. "Disciplines, Models, and Computers: The Path to Computational Quantum Chemistry." *Studies in History and Philosophy of Science Part A* 48(Supplement C): 89–96. **DOI:** https://doi.org/10.1016/j.shpsa.2014.05.003, **PMID:** 25571750

Mahoney, Michael S. 2008. "What Makes the History of Software Hard." *IEEE Annals of the History of Computing* 30(3): 8–18. **DOI:** https://doi.org/10.1109/MAHC.2008.55

Mercier, G. 1993. Message: 1993.06.30-011, http://www.ccl.net/cgi-bin/ccl/message-new?1993+06+30+011.

November, Joseph A. 2011. "Early Biomedical Computing and the Roots of Evidence-Based Medicine." *IEEE Annals of the History of Computing* 33(2): 9–23. **DOI:** https://doi.org/10.1109/MAHC.2011.35

Park, Buhm Soon. 2003. "The Hyperbola of Quantum Chemistry: the Changing Practice and Identity of a Scientific Discipline in the Early Years of Electronic Digital Computers, 1945–65." *Annals of Science* 60(3): 219–247. **DOI:** https://doi.org/10.1080/00033790.2018.12016717

Park, Buhm Soon. 2009. "Between Accuracy and Manageability: Computational Imperatives in Quantum Chemistry." *Historical Studies in the Natural Sciences* 39(1): 32–62. **DOI:** https://doi.org/10.1525/hsns.2009.39.1.32

Richon, Allen B. 2008. "Current Status and Future Direction of the Molecular Modeling Industry." *Drug Discovery Today* 13(15–16): 665–669. **DOI:** https://doi.org/10.1016/j.drudis.2008.04.008, **PMID:** 18675761

Rzepa, H. 1993. Message: 1993.06.27-001, http://www.ccl.net/cgi-bin/ccl/message-new?1993+06+27+001.

Smith, D. 1993. Message: 1993.06.26-003, http://www.ccl.net/cgi-bin/ccl/message-new?1993+06+26+003.

Spencer, Matt. 2015. "Brittleness and Bureaucracy: Software as a Material for Science." *Perspectives on Science* 23(4): 466–484. **DOI:** https://doi.org/10.1162/POSC_a_00184

St-Amant, A. 1993. Message: 1993.06.26-005, http://www.ccl.net/cgi-bin/ccl/message-new?1993+06+26+005.

Thompson, M. 1993a. Message: 1993.06.26-004, http://www.ccl.net/cgi-bin/ccl/message-new?1993+06+26+004.

Thompson, M.. 1993b. Message: 1993.06.25-005, http://www.ccl.net/cgi-bin/ccl/message-new?1993+06+25+005.

Westheimer, Frank H. 1956. Calculation of the Magnitude of Steric Effects. Pp. 523–555 in *Steric Effects in Organic Chemistry*. Edited by Melvin S. Newman. New York: Wiley.

Wieber, Frédéric. 2012. Multiple Means of Determination and Multiple Constraints of Construction: Robustness and Strategies for Modeling Macromolecular Objects. Pp. 267–288 in *Characterizing the Robustness of Science*. Edited by Léna Soler, Emiliano Trizio, Thomas Nickles, and William Wimsatt. Dordrecht: Springer. **DOI:** https://doi.org/10.1007/978-94-007-2759-5_11

Weiber, Frédéric, Pisanty, Alejandro, & Hocquet, Alexandre. 2018. 'We Were Here before the Web and Hype...': A Brief History of and Tribute to the Computational Chemistry List. *Journal of Cheminformatics*, 10(1), 67. **DOI:** https://doi.org/10.1186/s13321-018-0322-7

Wilson, Edgar Bright, John Courtney Decius, and Paul Clifford Cross. 1955. *Molecular Vibrations: The Theory of Infrared and Raman Vibrational Spectra*. New York: McGraw-Hill. **DOI:** https://doi.org/10.1149/1.2430134

Winsberg, Eric. 2018. *Philosophy and Climate Science*. Cambridge: Cambridge University Press. **DOI:** https://doi.org/10.1017/9781108164290