Digichem: Computational Chemistry For Everyone

Oliver S. Lee[a,b], Malte C. Gather[b,c] and Eli Zysman-Colman[a*]*

[a]Organic Semiconductor Centre, EaStCHEM School of Chemistry, University of St Andrews, St Andrews, UK, KY16 9ST

[b]Organic Semiconductor Centre, SUPA School of Physics and Astronomy, University of St Andrews, St Andrews, UK, KY16 9SS.

[c]Humboldt Centre for Nano- and Biophotonics, Department of Chemistry, University of Cologne, Greinstr. 4-6, 50939 Köln, Germany.

Abstract

We describe a new tool for the efficient management of computational chemistry. Digichem is a program that automates and simplifies nearly the entire computational pipeline, including large-scale batch submission of calculations, analysis and results parsing, the generation of 3D density plots and 2D graphs of calculation data, storage and retrieval of calculation results to a database, and automated handling of multi-step jobs. The program is designed to reduce the tedium and likelihood of human error for researchers of all skill-levels but is particularly targeted towards novice users who otherwise may find the barrier to entry to computational chemistry unnecessarily high. To date, this program has been used to successfully run and analyse over 50,000 individual calculations, evidencing its usefulness and utility. The Digichem program is presently released under a free-to-use license, and components of the Digichem system are additionally available under an open-source license.

Introduction

Since its inception in the 1920s, computational chemistry has had a revolutionary impact on the way chemists perform research.[1] Once limited to describing the bonding interactions of the hydrogen molecule,[2] modern computational chemistry can accurately predict a multitude of molecular properties for systems containing hundreds or thousands of atoms,[3–6] including molecular orbitals and their energies, molecular vibrations, electronic excited states, transition states, nuclear magnetic resonance (NMR) shielding and coupling constants, and reaction pathways. Much of this explosion in scale was driven by the seminal works of Hohenberg, Kohn, and Sham,[7,8] building upon the works of many others,[9–12] in developing the fast but accurate methodology that is known as density functional theory (DFT). Today, a plethora of computational models are available to the theoretician. This not only includes various flavours of DFT, including the original local-density approximation functionals, the nowadays ubiquitous hybrid functionals such as B3LYP[13–16] and PBE0,[17–19] and the modern double-hybrid functionals of Truhlar,[20] Grimme[21] and others, but also post Hartree-Fock (HF) methods, such as Møller–Plesset perturbation theory[22,23] and coupled cluster theory.[24–27] In the latter case, the popularity of these post-HF wavefunction based methods has been greatly accelerated by time-saving approximations, such as the resolution of the identity (RI) approximation,[28–34] and the closely related chain of sphere exchange algorithm (COSX),[35,36] permitting their application to larger systems than has been possible previously. These developments, amongst others, have led to computational chemistry being routinely used to understand natural phenomena that are otherwise inscrutable, and to predict the properties of molecules, either as a screening tool to help guide synthetic efforts, or to provide additional evidence for synthetic pathways or molecular properties.

Over its lifetime, the accuracy, speed, and breadth of problems that can be addressed by computational chemistry has been steadily expanded. However, comparatively little has been done to improve the usability, accessibility, or productivity of computational chemistry. Most computational tasks are still too demanding in terms of memory, hard-drive space, and central-processing unit (CPU) cores to be performed on a personal computer and completed in a reasonable amount of time. Instead, most computational chemistry is relegated to distributed, remote super-computing clusters. These clusters invariably operate without a graphical user interface, forcing the user to interact solely *via* the command line. For experienced computing users, this setup enables faster and more efficient

workflows because of its inherent support for scripting, but for many non-expert users the opposite is true, and the lack of a familiar interface can make even basic computing tasks, such as opening a file, arduous. Further, solving a computational chemistry problem rarely involves executing only a single program. Instead, it typically consists of several programs operating in sequence, which together make-up a pipeline, each program taking the output from the last as input to the next. The core of this pipeline is the computational chemistry program, or engine, which solves the computational chemistry problem itself, but other steps are equally important in order to correctly set up the computing environment and analyse the results (Figure **1**). These steps may include, but are not limited to: (1) drawing of the molecules/systems of interest; (2) choice of computational parameters (level of theory, CPU/memory usage, molecular properties of interest, etc.); (3) uploading of the molecule files to the cluster where the calculations are performed; (4) interfacing to the cluster queuing system; (5) running the computational chemistry engine; (6) regular monitoring to determine if the calculation has completed; (7) analysis of output files to determine if the calculation succeeded correctly (for example, checking for convergence of the wavefunction); (8) optional post-processing of results, such as the generation of orbital density plots; (9) downloading of completed calculation results; (10) extraction of results from completed calculations, collation and further processing/analysis.

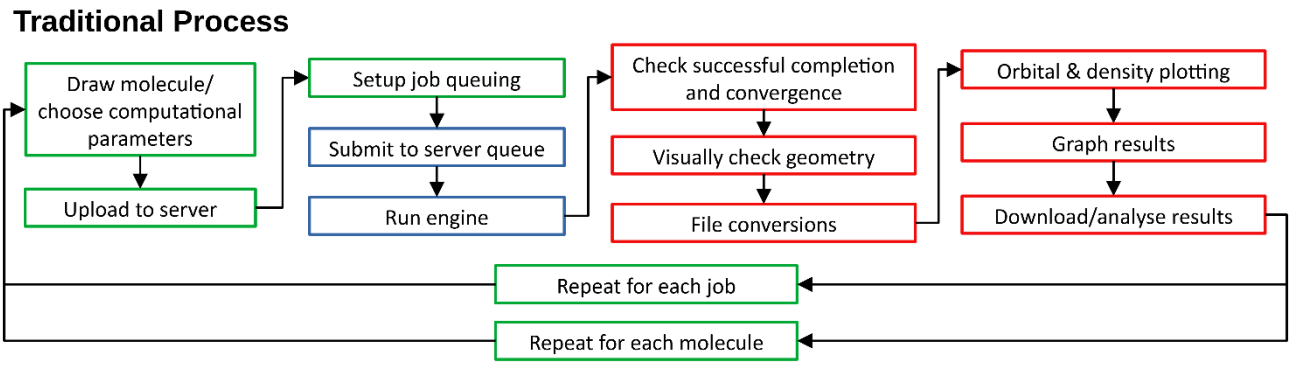


Figure **1**. Flow-diagram of the traditional computational chemistry pipeline. Green tasks relate to setup, blue to executing the computational chemistry engine, and red to analysis.

This pipeline is complex and difficult to learn because it is made up of many different individual programs, and the user must learn how to use all of these programs correctly before any results can be obtained. Complicating matters is that most of the setup is performed using text files, with little-to-no validation, and error reporting varies greatly in quality between the different programs. Further,

it is normally necessary to perform the pipeline multiple times for each molecule because the properties of interest to the chemist (e.g., HOMO-LUMO gap) are dependent on the correct molecular geometry being obtained first, and the geometry optimisation calculation which provides this must normally be run separately. The process must then be repeated again for each molecule in the study, and so large computational studies can quickly swell to thousands of individual calculations.

Together, these problems place an enormous learning burden on the would-be computational chemist and make the field challenging to break into. Even for more experienced users, the daily process of performing computations can be tedious, time consuming and error prone, because so much of the process needs to be curated manually. Many computational chemists will know the anguish of discovering a mistake in an input file only after the calculation has returned with useless data, wasting days of CPU time, and demanding the entire process be started again. Even when performed correctly, this process is inefficient. Many parts of the pipeline, most noticeably between separate calculation steps, require manual intervention by the chemist to set up the next stage. This results in the overall execution time being limited not by the speed of the computer, but by how attentive the user is in checking the calculation progress. For computational screens involving many molecules, the inefficiencies are multiplied because only one molecule can be prepared by the user at a time. For very large computational projects, this essentially mandates that the scientist writes their own code to help automate the process, but this is not a skill that is accessible to all, nor one that is necessarily linked to scientific proficiency.

Existing Solutions

Many researchers use computational chemistry as a tool to support their experimental research, rather than working full time in the field itself. In these studies, the number of computations is typically lower, and so the impact of the inefficiencies of the computational chemistry pipeline is lessened. In these cases, many scientists will choose to accept the status quo. For researchers who are unable to overcome the learning barrier themselves, they may instead be able to outsource computational tasks to collaborators, thus relieving themselves personally of the burden. For scientists looking to conduct more complex studies however, or who do not yet have a well-established network of collaborators to rely upon, this situation is not tenable. In these cases, it is common for the researcher to develop programs or scripts to assist them in managing the pipeline. In the simplest case, a group may share a set of standardised input templates, which each researcher then modifies to suit their needs. In this

way, the need to write the whole input file from scratch is removed. More advanced programs may automatically merge parts of the input file together, for example to combine a standardised section specifying the method of the calculation with a customizable section containing the geometry for each molecule. For large-scale computational screens, this sort of automation is near-essential because of the number of files that need to be prepared. Sometimes, these in-house developed codes will mature to the point where they will be shared with other researchers, but in many cases they will only be available to members of the same research group or close collaborators. This results in a large duplication of effort because multiple researchers are attempting to solve the same problem. The generalizability of the scripts can also be a problem, and they may need to be substantially modified to address a new problem. This demands a significant investment of time at the start of each new project. While human error should be reduced because of the automation they offer, care needs to be taken in modifying these programs, because any errors that are introduced can propagate to many calculations. Finally, the quality of these programs depends on how much time the researcher can dedicate to maintaining them, and this is not always in abundance.

Alternatively, or in addition, researchers can use an existing code or product.. As examples, CalcUS,[37] WebMO,[38] ChemCompute[39] and MolCalc[40] are all web-based, calculation submission platforms. They provide tools to draw and/or upload molecular structures, chose calculation options, and submit a calculation to a number of backend computational chemistry programs. They also offer common analysis options, such as the parsing of molecular orbital energies, the plotting of molecular orbitals and the visualization of vibrational modes, although the exact feature set differs between each program. They all interface to a queuing system, either coming bundled with an internal queuing manager (CalcUS) or interfacing to external programs, such as SLURM[41] (WebMO). CalcUS and WebMO are targeted towards academic research, while ChemCompute and MolCalc are designed for undergraduate teaching. CalcUS and MolCalc are free and open-source software, while ChemCompute is offered as a free-to-use service only, and WebMO is commercial, although it does offer a free basic version with a reduced feature set. Winmostar[42] is a similar calculation submission platform, but is designed to be run on the user's desktop rather than in a web-browser, and is also commercial. Maestro,[43] Spartan,[44] TMoleX[45] and GaussView[46] are graphical user-interfaces (GUIs) to the computational engines Jaguar,[47] Q-Chem,[48] Turbomole,[49] and Gaussian,[50] respectively, and each is specific to its own engine. Maestro is typically distributed as part of the commercial Schrodinger platform, along with Jaguar, which, according to the authors,[43] is targeted primarily towards researchers in drug discovery. Spartan is sold as a standalone product, with Q-Chem bundled

inside of it, while Gaussview is sold as an addon for Gaussian, and TMoleX is included in the purchase of Turbomole. All four offer common analysis features, such as parsing of molecular orbital energies, vibrational frequencies, and excited states, along with more advanced functionality such as the plotting of molecular orbital densities. They differ somewhat in their support for submitting calculations. Both Gaussview and TMoleX can either write input files to the hard drive, or call the underlying engine directly, if installed on the same machine. TMoleX additionally supports calling remote installations of Turbomole. Spartan and Maestro, on the other hand, are designed to perform the calculation internally, without calling an external engine, although both can additionally interface to remote installations. Molden,[51] Gabedit[52] and Avogadro[53] are more general tools and interface to a number of backend programs. They offer the usual analysis functions such as molecular orbital plotting, as well as input file creation tools, but do not manage the running of the calculation itself. There are also a number of libraries and support tools available that the intrepid computational chemist can incorporate into their own scripts and programs. Open Babel[54,55] is a tool for the interconversion of chemistry file formats, while cclib[56,57] provides parsing of calculation output. Both are available as a python application-programming interface (API) and a command-line tool. RDKit[58], CDK[59–62] and ASE[63] are three extensive libraries for performing computational chemistry, with RDKit and CDK focusing on analysis and post-processing, and ASE[63,64] on performing atomistic simulations. They are mainly accessed *via* an API, but RDKit and ASE also offer a number of standalone programs. Finally, AQME[65] is a collection of workflows designed to automate various repetitive computational chemistry tasks.

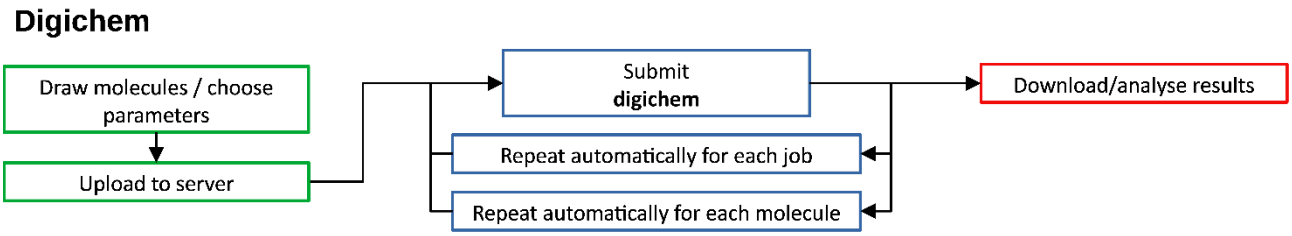


Figure **2**. Flow-diagram of the Digichem pipeline. Green tasks relate to setup, blue to executing the computational engine, and red to analysis.

Despite the plethora of programs and libraries that are available, a complete single solution is not readily available. Many of the codes that are described above offer excellent support for either analysis or submission management, but none offers sufficient support for both for it to be used as a

standalone tool. The user must still navigate multiple different programs in order to manage the complete pipeline. In particular, none of the tools that we have explored offer comprehensive support for large-scale computational screens, where the number of individual computations may reach the thousands, and in these cases the scientist is still expected to develop their own scripts. Many of the programs that offer a more complete set of tools may also be too expensive for a subset of researchers, or exist only as programming libraries, and cannot be used without the scientist having coding experience. To address this problem, we have developed Digichem, a complete computational management tool suitable for computational chemists of all skill-levels, but particularly directed at novice users. Digichem is designed to be the only program the user needs to interact with on the server (Figure **2**). It contains both a command-line and pseudo-graphical interface, the latter powered by the Urwid library,[66] and supports computational screens of unlimited size. Here, we describe this program and its operation in detail.

Program Scope

Computational chemistry is a broad term. Included in this definition is the study of both molecular and periodic (i.e,, crystalline) systems, using techniques such as quantum chemistry (QC), molecular mechanics (MM), molecular dynamics (MD), mixed QC/MM methods,[67] Monte Carlo[68] simulations and, increasingly, machine learning (ML). While some concepts are common to multiple different branches of computational chemistry, the way in which each type of calculation is performed, and the results obtained, can differ substantially. In Digichem, we have chosen to focus on the application of quantum chemistry to molecular systems. This includes common wavefunction methods (HF, MP, CC) as well as DFT, using atomic-orbital type basis sets. We do not explicitly include or exclude certain methods (e.g., different functionals), but rather interface with the functionality that is already presented in each computational engine. Currently, the program supports interfacing with the Gaussian,[50,69] Turbomole,[70] and ORCA[71] programs. All common molecular calculations are supported, including single-point energies, geometry optimisations (including of excited states), vibrational frequencies, excited states (*via* both linear-response type methods[72,73] and ΔSCF[74] for the first triplet state), and nuclear magnetic resonance chemical shifts and coupling constants. The exact feature set supported varies from program to program, and naturally depends on the functionality of the underlying computational engine. The program does not currently support calculations involving multiple structures, most noticeably transition state (saddle point) type calculations, but we look forward to incorporating these in the future.

Program Design

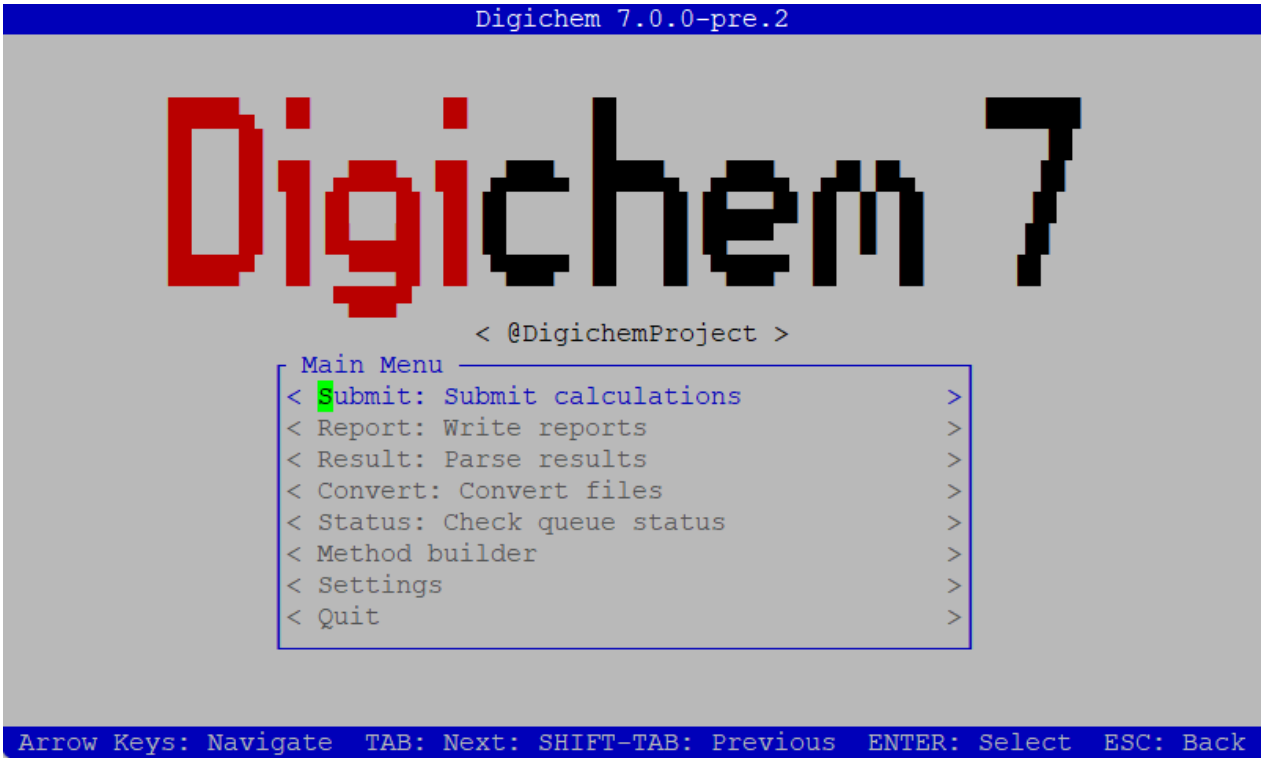


Figure **3**: Digichem main menu.

Digichem is split into several sub-modules, each designed to handle a separate aspect of computational chemistry. All the sub-modules are accessed from a single command on the command-line, which is `digichem`, followed by a keyword to identify the sub-module. Each of the sub-modules can be accessed entirely on the command-line, which is quicker for power-users, while the most important sub-modules also support a pseudo-graphical interactive interface, which can be accessed by specifying the '`-I`' (interactive) option after the sub-module name. The interactive interface can also be launched directly by specifying the '`digichem interactive`' command (Figure **3**), from which all major facets of the program can be accessed. For novice users, this is the only command they are required to learn. The graphical interface formats text and other non-alphanumeric characters, such as the Unicode box drawing characters,[75] to appear like graphical widgets such as buttons, checkboxes, text-entry fields, and scrollable windows. These interactive elements are more familiar to users of traditional GUIs than the opaque command-line. Because the library powering this interface only uses the functionality already available in the text console,[66] this interface does not require a full graphical software stack (e.g., X-server) on the remote cluster. All access to Digichem

is provided *via* the secure-shell (SSH) protocol, which is already the default way users interact with their calculation servers, and it is therefore equally accessible from client machines running Windows, MacOS, or Linux. Digichem is written in Python, is simple to install, contains all its required dependencies internally (even including Python itself), and does not require elevated privileges (neither through `sudo` nor a super-user account) to install or run. It does not require any firewall ports to be opened, except for SSH port 21, which is already required to provide access to the cluster, in contrast to web-server designs.[37,38,40] The currently supported sub-modules are: `digichem submit`, for calculation submission; `digichem convert`, for managing input data types, `digichem result`, for parsing and analysing completed calculation results; `digichem report`, for generating portable document format (PDF) reports of completed results; `digichem image`, for generating graphical data from completed calculations; and `digichem database`, for managing historical calculation data. These are described in more detail below.

Calculation Submission (digichem submit & digichem convert)

a)

```
basis_set:
  internal: 6-31G*
meta:
  class_name: Gaussian
  name: Optimisation
method:
  dft:
    calc: true
    dispersion: GD3BJ
    functional: PBE0
performance:
  memory: 2GB
  num_cpu: 2
properties:
  opt:
    calc: true
solution:
  calc: true
  solvent: Toluene
```

b)

```
basis_set:
  internal: 6-31G*
meta:
  class_name: Turbomole
  name: Optimisation
method:
  dft:
    calc: true
    dispersion: GD3BJ
    functional: PBE0
performance:
  memory: 2GB
  num_cpu: 2
properties:
  opt:
    calc: true
solution:
  calc: true
  solvent: Toluene
```

c)

```
basis_set:
  internal: 6-31G*
meta:
  class_name: Orca
  name: Optimisation
method:
  dft:
    calc: true
    dispersion: GD3BJ
    functional: PBE0
performance:
  memory: 2GB
  num_cpu: 2
properties:
  opt:
    calc: true
solution:
  calc: true
  solvent: Toluene
```

Figure **4**. Comparison of the method file format for three equivalent calculations for the calculation engines a) Gaussian, b) Turbomole, and c) Orca.

Conceptually, each computational chemistry calculation consists of two parts: the method, which describes the level of theory (the DFT functional, the basis set, the solvent model, etc.); and the

coordinates, which describes the geometry of the molecule of interest. In most computational engines these components are combined within a single input file, but in Digichem we made the conscious decision to separate them. This confers a number of advantages, but in particular it makes it easy to reuse a computational methodology across multiple molecules, as is required in a large-scale computational screen. Both the method and coordinate files are written in the non-binary YAML format,[76] which can be edited using any text editor and is more intuitive than many of the calculation engine native formats. In addition to the molecular geometry, the coordinate file also specifies the overall charge and spin multiplicity of the system, something which is missing from many popular cheminformatics formats (XYZ, for example), and this is preserved across Digichem calculations. Both files are calculation engine independent, which means that any geometry or method file is compatible with any of the calculation engines that Digichem supports, so long as the chosen calculation options are supported by the underlying calculating engine (Figure **4**). To improve interoperability, we have also made efforts to standardize calculation options across different engines, for example supporting both PBE0 and the Gaussian specific PBE1PBE as names for this popular hybrid functional.[17,19] Digichem does not currently offer a 3D molecule builder, because this is not possible within the confines of a terminal interface, but instead supports all of the common cheminformatics file formats, including program-independent .xyz and .cml, Gaussian .com/.gjf, ChemDraw .cdx and crystallographic .cif, which are automatically converted to the internal Digichem format. Digichem is able to parse a subset of these formats natively, such as the Gaussian input file and XYZ file, while others are converted by calling the external obabel program,[54] which permits the user to use whichever 3D sketching software they are most accustomed. Files can also be manually interconverted with the `digichem convert` sub-module.

Command:

```
digichem submit Benzene.com Naphthalene.cdx Pyridine.si -m Optimisation.sim Excited_states.sim
```

Result:

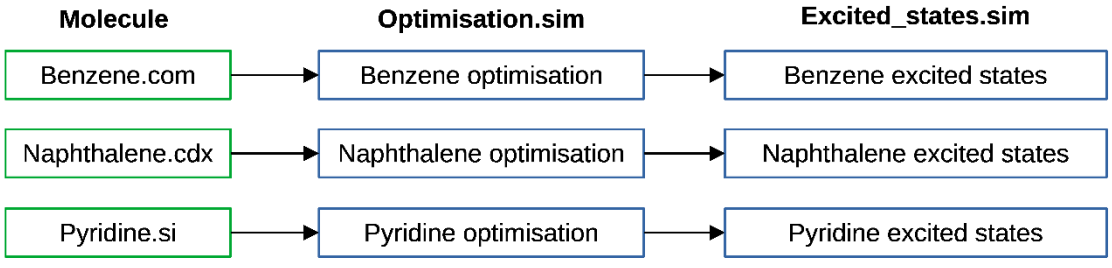


Figure **5**. Example submission process with `digichem submit`. In each case, only the starting geometry is taken from the given input file, while any calculation options (in the case of the Gaussian .com file) are ignored.

Submission in Digichem is achieved using the `digichem submit` sub-module. Any number of coordinate and/or method files can be specified at once. Each coordinate file will spawn a calculation to be run in parallel with the other coordinate files, while each method will be queued up to be performed in series using the output geometry from the previous calculation step (Figure **5**). Once submitted, all the selected calculations will be started automatically, and no further interaction by the user is required. Once each calculation is complete, Digichem will automatically analyse and parse the completed calculation results, generate any requested report and/or image files (see below), and automatically setup and submit the next calculation in the queue. Digichem currently supports the Gaussian,[50,69] Orca[71] and Turbomole[49] programs, and can handle submitting to different programs in series. In this way, a Turbomole calculation can use the output geometry produced from Gaussian, for example, as the coordinates will be formatted automatically. This process can be repeated near-infinitely and is limited only by the resources available on the underlying cluster. In this way, computational screens of any size can be handled in an identical manner. Parallel calculation submission and resource management is handled by external queuing software, and Digichem currently supports both SLURM[41] and PBS.[77] Options to control shared resources, such as the amount of memory, number of parallel CPUs, and maximum job execution time, can all be configured as part of the method file. Digichem has options to ensure that the resources requested by the calculation, such as the amount of memory, do not exceed those requested from the scheduler.

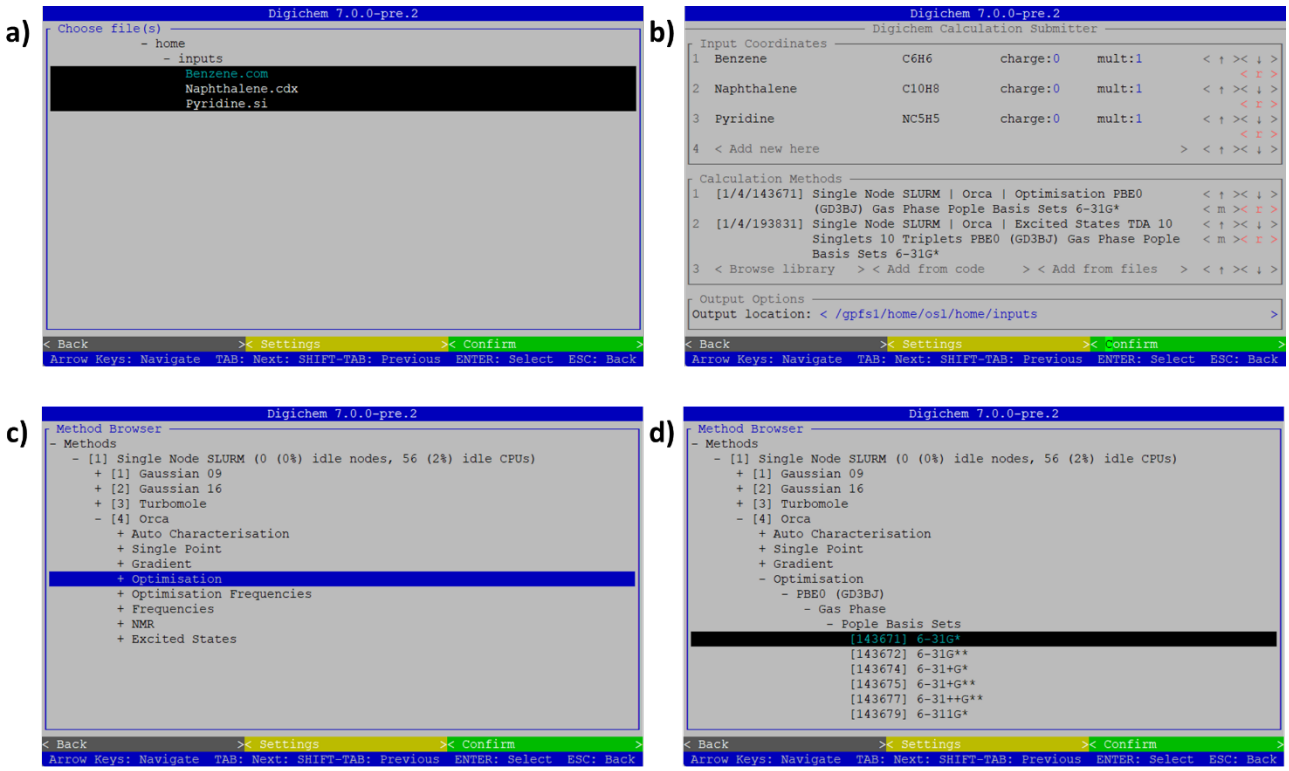


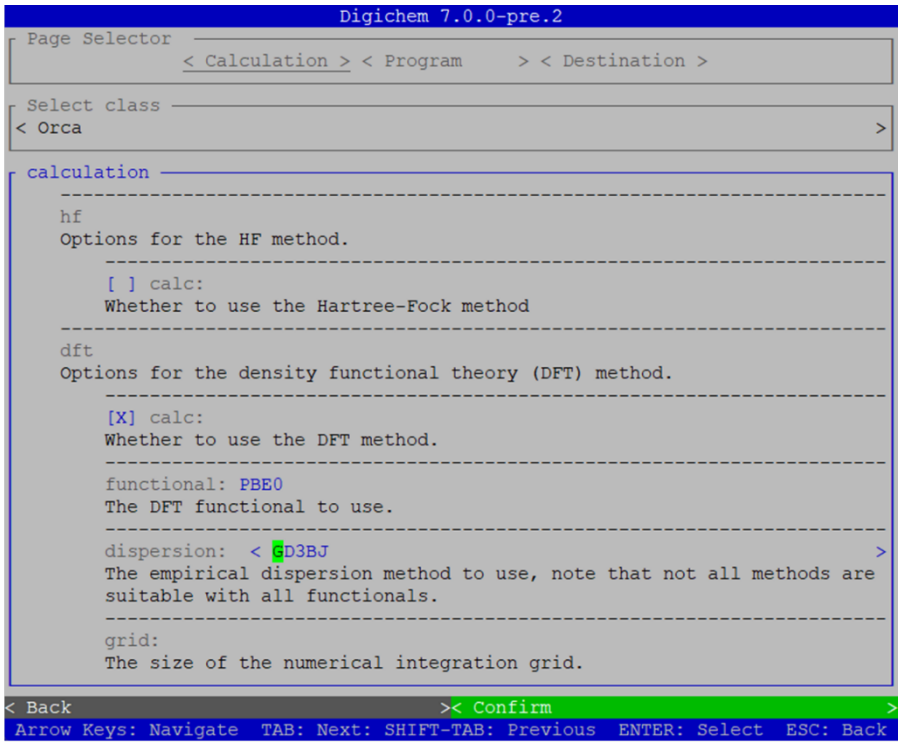
Figure **6**. Screenshots of the submission sub-module showing simultaneous set-up of three molecules (benzene, naphthalene, and pyridine) to be performed in parallel and two calculations (a geometry optimisation followed by TD-DFT excited-states calculation at the PBE0/6-31G** level of theory) to be performed in series. a) input coordinate file-picker. b) main submission interface. c) internal method library, from which the calculation can be chosen. d) the same method library, but further expanded to show more options.

The `digichem submit` sub-module greatly simplifies the submission process for intermediate and advanced users, but for novice users the requirement to use the command-line and to write method files by hand is not ideal. For this scenario, the interactive submission interface is more suitable, which can be accessed using the `digichem submit -I` command (Figure **6**), or from the main menu. From this screen, the user can choose all the parameters of the calculation without using the command-line or generating the files manually. The starting geometries can be selected from a file-picker type interface (Figure **6a**), similar to how files are selected in any commonly used GUI application, and the charge and multiplicity can be adjusted for each using text fields (Figure **6b**). The empirical formula of each molecule is also displayed to permit the user to check that the correct structure has been loaded. Each selected starting geometry will start a separate

calculation that will be run in parallel, the same as if submitting from the command-line. The calculation methods, meanwhile, can be selected from one of the approximately 240,000 pre-built calculations stored in Digichem's internal library, which cover the calculations of optimised geometries, vibrational frequencies, single-point energies and gradients, excited states, and NMR properties. This library is explored in a hierarchy (Figure **6c** and **d**). The first level of the hierarchy corresponds to the chosen SLURM partition or PBS queue, and allows the user to pick, for example, between a high or low priority submission. The second level corresponds to the calculation submission engine. At present, Gaussian, Turbomole and Orca are all supported. The remaining levels select the specifics of the calculation itself, including the properties to be calculated (geometry optimisation, excited states, NMR properties, etc.), the computational method or functional, and the basis set. Multiple calculation methods can be chosen, and each will be performed in series, and the interface provides buttons to re-order the methods to ensure they are performed in the correct sequence. Each calculation method is associated with a unique numerical identifier (ID) consisting of three parts separated by a forward-slash, which in order correspond to: 1) the SLURM/PBS queue, 2) the calculation engine, and 3) the calculation itself. For example, the optimisation calculation shown in Figure **6a** has the unique code 1/4/182628. These calculation codes, if known in advance, can be used to quickly select a calculation from the internal library without navigating the interactive hierarchy, and can be specified on the command line to further speed up the submission process for advanced users. The library provides options for the most popular DFT functionals, including PBE0[17,19] and B3LYP,[13–16], as well as more advanced methods such as Møller–Plesset and coupled-cluster theory, a variety of common solvents and the most common basis sets in the Pople,[78] Karlsruhe,[79] and correlation-consistent families.[80] We expect that, in most cases, the Digichem library will provide all the calculation options that a novice computational chemist would require, but the interactive interface also provides options to load a method file in case the user wishes to write a method from scratch or re-use a method from a previous calculation. Finally, Digichem provides an interface to interactively modify the calculation options from any loaded method (Figure **7**). This interface groups related options together and provides help messages and validation, where appropriate, to help novice users find and understand each calculation option. In this way, a user could select an existing method from the library and modify it to suit their needs; for example, changing the basis set to one that is smaller and increasing the number of requested CPUs so that the calculation will run more quickly. Once the calculation has been set up to the user's specifications, all queued

molecules will be submitted to the queuing manager simultaneously once the green ‘confirm’ button is pressed (Figure **6**).

a)



b)

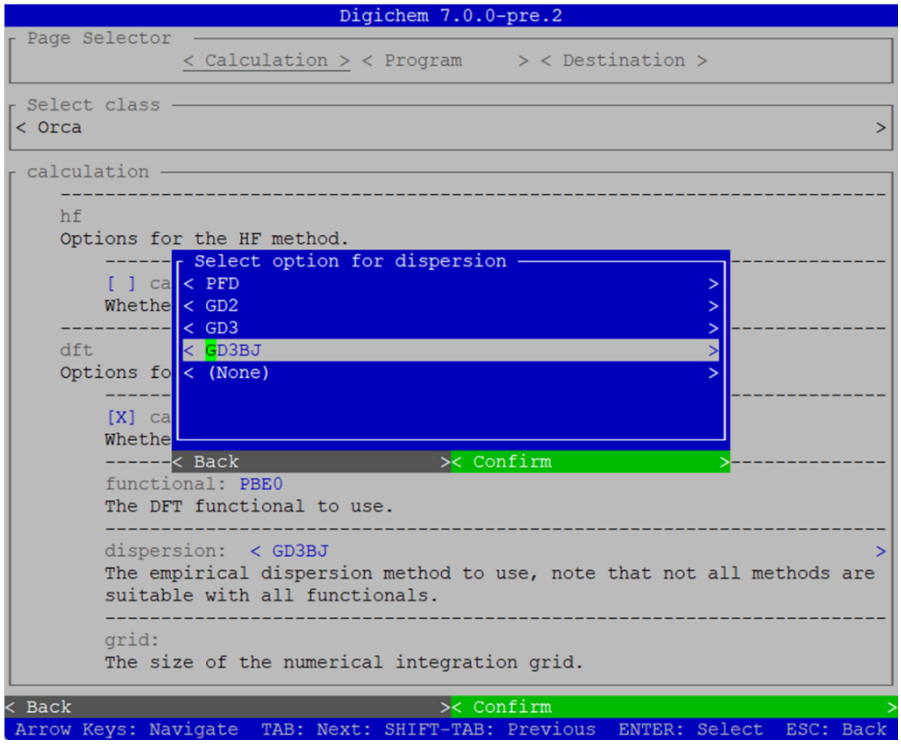


Figure **7**. Screenshots of the calculation method editor interface. a) general overview, showing example options for the calculation level of theory. b) example validation for the DFT dispersion correction option.

Calculation File Storage and Monitoring

The automated handling of submitted calculations is identical regardless of whether the user chooses to use the command-line or interactive interfaces. Digichem automatically partitions calculation data into a hierarchy of files and folders (Figure **8**). At the top of the hierarchy, each molecule is self-contained to its own directory. Within this, each individual calculation is stored in a separate sub-directory, which is named after the calculation in question. Digichem ensures that no two calculations can be performed in the same sub-directory, even if the same calculation is submitted multiple times. In this case, a number is appended to the directory name to distinguish it, and this logic is race-condition free. Within each calculation sub-directory, a third tier of directories store the calculation data. These directories are as follows: 'Flags', which conveys information about the current status of the calculation (see below); 'Input', which stores input files both for the specific calculation engine and Digichem; 'Logs', which stores log messages from Digichem and monitors CPU and memory usage (see below); 'Output', which stores raw output from the calculation engine, including the log file and any checkpoint binary files; 'Results', which stores formatted calculation data in text format, including CSV; and 'Report', which contains the PDF report and any associated image data. The Results and Report folders will naturally only be created after the calculation has completed, as they only contain completed calculation data. A separate 'scratch' directory is also created and managed for each calculation, which is used by many computational engines for input/output (IO) intensive reading and writing. This scratch directory is normally stored outside of the rest of the calculation directory hierarchy to take advantage of faster and/or larger physical file storage media. Each scratch directory is also ensured to be unique for each calculation and is automatically removed following the completion of the corresponding calculation. This hierarchy of folders is managed entirely by the program, and greatly simplifies data management and storage.

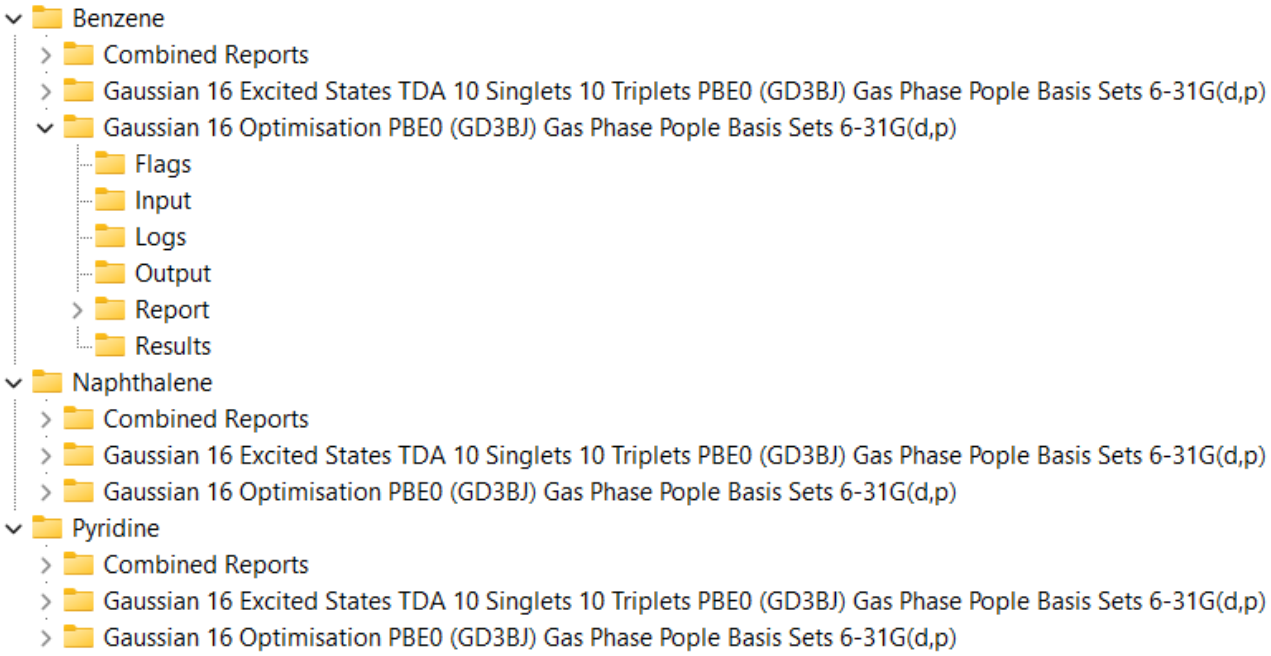


Figure **8** Example of the Digichem directory hierarchy for three molecules (benzene, naphthalene and pyridine) and two calculations (a geometry optimisation and a TDA-DFT excited-states calculation).

Calculation monitoring is traditionally achieved by: 1) checking the status of the job using the relevant queuing manager, for example with the `squeue` command; and 2) checking the final lines of the calculation engine log file, for example using the `less` or `tail` commands. Digichem provides an alternative that does not depend on external tools, and is more convenient for novice users, by means of the Flags sub-directory. Within this folder are a number of text files with distinctive names. These files are all empty, but the name of each file conveys information about the status of the calculation, and new files are created, and old ones are deleted, as certain milestones in the calculation are hit. At the beginning of the calculation, the PENDING flag will be the only file present, which indicates the calculation is currently in the queue, and awaiting resources. Once the calculation reaches the top of the queue and execution begins, the PENDING flag will be deleted and replaced with the STARTED flag. Once the calculation is complete, this in turn will be replaced with the SUCCESS flag, and the POST phase will begin, and so on. In this manner, it is possible to monitor the entire process of the calculation from the file-explorer, without using any external tools. The currently supported file flags, and their meanings, are detailed in Table **1**.

Table **1**. List of currently supported file flags.

Flag Name	Description
PENDING	The calculation has been submitted but has not yet begun. Most commonly, this occurs because the calculation is waiting in the queue for server resources.
STARTED	The calculation has begun. This flag persists even the calculation has completed.
RUNNING	The calculation is currently ongoing. This flag is removed once the calculation has completed.
SUCCESS	The calculation has completed successfully.
CONVERGED	The optimisation converged successfully; only relevant to geometry optimisations.
NOT_CONVERGED	The optimisation did not converge successfully; only relevant to geometry optimisations.
CLEANUP	The main calculation has finished, and Digichem is currently cleaning up.
ERROR	The calculation has stopped because an error occurred.
POST	The main calculation has finished, and Digichem is currently performing post-analysis. This largely involves writing the PDF report and any associated image files.
DONE	All work on the calculation folder is complete, and Digichem will make no further changes. The calculation folder can be safely moved, downloaded or deleted.

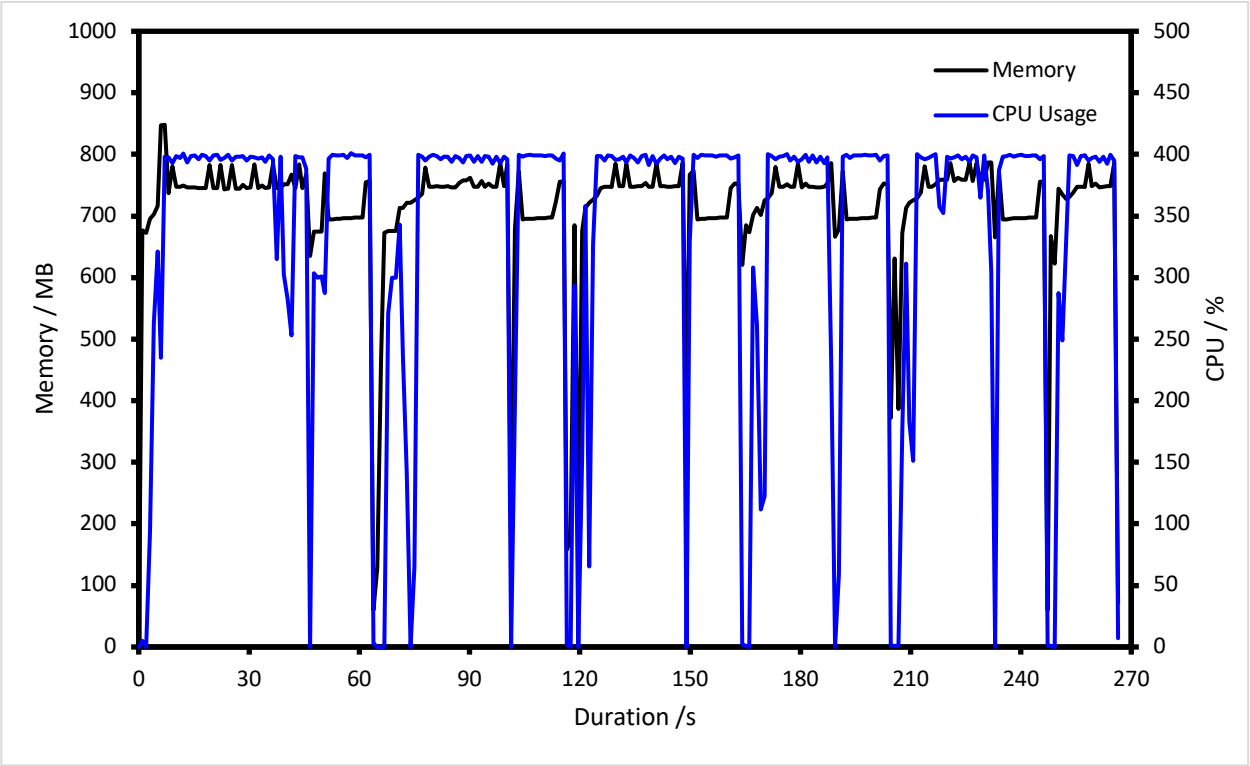


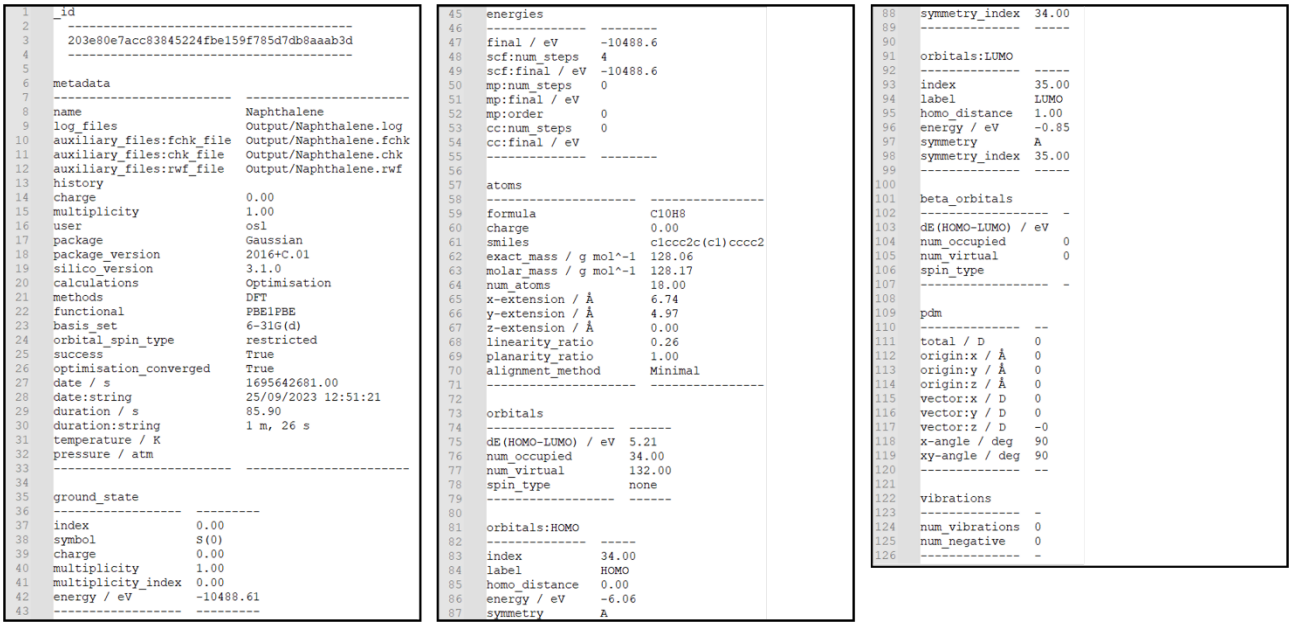
Figure **9**. Graph of CPU and memory usage of an example calculation, plotted from the profiling data provided by Digichem. The calculation was the optimisation of Naphthalene, performed at the PBE0/6-311G**/GD3BJ level of theory, in the gas-phase, using Orca with 4 CPUs and a maximum memory allocation of 10 GB. Profiling was performed every 1 s. The CPU usage is the total for all CPUs, so 400% corresponds to 4 CPUs working at maximum load.

The flags directory conveys information about the status of each calculation and can be used to check for when the calculation has completed, but it conveys no information on the resource usage of the calculation. The CPU usage, memory usage and read/write speed of the calculation over time is of particular importance to both users of computational chemistry and designers of new software, as it helps to identify 'bottlenecks' (i.e., the slow part) of a calculation, and to diagnose out-of-memory failures. Digichem provides calculation profiling for all of its supported calculation engines, and records the following information: 1) CPU usage; 2) real; and 3) virtual memory usage; 4) total number of bytes read from the file system; 5) instantaneous read speed; 6) total number of bytes written to the file system; 7) instantaneous write speed; and 8) the number of new processes spawned since the last profiling step. This information is continuously logged to a CSV file as the calculation

progresses, so it can be inspected even before the calculation has completed, and the profiling frequency can be manually adjusted to favour either higher resolution (more profiling steps) or lower file size (fewer profiling steps). The CPU and memory statistics for an example optimisation calculation are shown graphically in Figure **9**, which clearly details the short, large decreases in CPU usage surrounding a new energy calculation step starting.

Calculation Analysis (`digichem result`)

After each calculation has completed Digichem then automatically performs parsing of the calculation log file. In the first instance, Digichem determines whether the calculation has completed successfully, which can normally be detected by the presence of a trigger line at the end of the calculation (for example, "****ORCA TERMINATED NORMALLY****" for Orca, or "Normal termination of Gaussian" for Gaussian), as well as by checking if the exit code of the calculation engine is 0. For a geometry optimisation, the optimisation must also have converged to a minimum for the calculation to be considered successful. The success/failure status of the program is used to set the SUCCESS or ERROR file flags, as appropriate, and is also used to determine whether to continue to the next calculation in the queue. If the calculation is not successful, no further calculations in the chain will be submitted. This prevents errors being accidentally propagated to subsequent calculations and wasting CPU time.



```
_id
  ----------------------------------------
  203e80e7acc83845224fbe159f785d7db8aaab3d
  ----------------------------------------

metadata
------------------------  ------------------------
name                      Naphthalene
log_files                 Output/Naphthalene.log
auxiliary_files:fchk_file Output/Naphthalene.fchk
auxiliary_files:chk_file  Output/Naphthalene.chk
auxiliary_files:rwf_file  Output/Naphthalene.rwf
history
charge                    0.00
multiplicity              1.00
user                      osl
package                   Gaussian
package_version           2016+C.01
silico_version            3.1.0
calculations              Optimisation
methods                   DFT
functional                PBE1PBE
basis_set                 6-31G(d)
orbital_spin_type         restricted
success                   True
optimisation_converged    True
date / s                  1695642681.00
date:string               25/09/2023 12:51:21
duration / s              85.90
duration:string           1 m, 26 s
temperature / K
pressure / atm
------------------------  ------------------------

ground_state
------------------  ---------
index               0.00
symbol              S(0)
charge              0.00
multiplicity        1.00
multiplicity_index  0.00
energy / eV         -10488.61
------------------  ---------
```

```
energies
--------------  --------
final / eV      -10488.6
scf:num_steps   4
scf:final / eV  -10488.6
mp:num_steps    0
mp:final / eV
mp:order        0
cc:num_steps    0
cc:final / eV
--------------  --------

atoms
--------------------  ----------------
formula               C10H8
charge                0.00
smiles                c1ccc2c(c1)cccc2
exact_mass / g mol^-1  128.06
molar_mass / g mol^-1  128.17
num_atoms             18.00
x-extension / Å       6.74
y-extension / Å       4.97
z-extension / Å       0.00
linearity_ratio       0.26
planarity_ratio       1.00
alignment_method      Minimal
--------------------  ----------------

orbitals
------------------  ------
dE(HOMO-LUMO) / eV  5.21
num_occupied        34.00
num_virtual         132.00
spin_type           none
------------------  ------

orbitals:HOMO
--------------  -----
index           34.00
label           HOMO
homo_distance   0.00
energy / eV     -6.06
symmetry        A
```

```
symmetry_index  34.00
--------------  -----

orbitals:LUMO
--------------  -----
index           35.00
label           LUMO
homo_distance   1.00
energy / eV     -0.85
symmetry        A
symmetry_index  35.00
--------------  -----

beta_orbitals
------------------  -
dE(HOMO-LUMO) / eV
num_occupied        0
num_virtual         0
spin_type
------------------  -

pdm
--------------  --
total / D       0
origin:x / Å    0
origin:y / Å    0
origin:z / Å    0
vector:x / D    0
vector:y / D    0
vector:z / D    -0
x-angle / deg   90
xy-angle / deg  90
--------------  --

vibrations
--------------  -
num_vibrations  0
num_negative    0
--------------  -
```

Figure **10**. Excerpts from an example result summary output file written by Digichem.

If the calculation was successful, Digichem then proceeds to perform a more complete analysis of the output. The calculation log file is parsed using the cclib[56] library, which is able to extract various numerical results, such as orbital energies and vibrational frequencies, as well as vital calculation metadata, including the DFT functional/calculation method, basis set and solvent. Additional data, which is not present explicitly in the calculation output, is also calculated by Digichem. This includes simulated absorption spectra, using Gaussian-broadened electronic excited states or vibrational frequencies, simulated NMR spectra, fluorescence decay rates, the dissymmetry factor of transition dipole moments,[81] and spin-orbit coupling, the latter of which is calculated using a modified version of the PySOC program.[82] The processed data is saved to a number of text files in the Results directory of the calculation. The most important of these results, such as the calculation metadata, HOMO/LUMO energies and geometry information are available in the summary results file (Figure **10**). This file is written in a plain-text format which allows it to be easily read using simple text editors, including those available on the command line (nano, vi, emacs, etc), and thus provides an immediate overview of the completed calculation data. For further processing with other programs or tools, the individual calculation results are saved to a number of tabular CSV files, which can be opened using common spreadsheet management programs or graphing software. This gives the user the opportunity to perform their own analysis and graphing; however, Digichem also generates graphs of the most important results automatically (described below). A CSV version of the summary file is also created. Together, these text-based result files can greatly facilitate an author's ability to conform to FAIR data practices because the files are easily read and are program-independent. Additionally, all the results processed from the completed calculation are stored in a Digichem-specific YAML file. This file can be read by the `result`, `report` and `database` sub-modules to load the full-set of calculation results, without having to re-parse the calculation log file. Finally, the output geometry of the calculation is stored in the Digichem-specific .si format and the program-independent .xyz format, allowing for easy reuse in future calculations or deposition in electronic supplementary information. All of the automatically generated result files, and their contents, are detailed in Table **2**.

Table **2**. List of the currently supported result files, and their contents.

File	Type	Contents
Absorptions	.csv	Simulated UV-Vis absorption spectrum using Gaussian-broadened excited states, plotted on an energy (eV) scale.

Atoms	.csv	Atoms of the studied molecule and their output geometry.
Beta	.csv	Orbital energies and symmetries of any beta orbitals.
CC	.csv	Coupled cluster (CC) energies, including at each optimisation step if relevant.
ES	.csv	Electronic excited state (ES) energies, multiplicities and other data.
IR	.csv	Simulated IR absorption spectrum using Gaussian-broadened vibrational frequencies.
MP	.csv	Møller–Plesset (MP) energies, including at each optimisation step if relevant.
NMR	.csv	Nuclear magnetic resonance (NMR) data. This file contains a calculated NMR shielding, and a matrix of coupling constants between each of the non-magnetically equivalent atoms of the molecule. Additional CSV files are also created of simulated NMR spectra, with and without simulated decoupling, using Gaussian-broadened peaks.
Orbitals	.csv	Orbital energies and symmetries. If the calculation uses unrestricted orbitals (or otherwise contains both alpha and beta orbitals), this file will contain only the alpha orbitals.
SCF	.csv	Self-consistent field (SCF) energies, which normally correspond to calculations at the Hartree-Fock or DFT level, including at each optimisation step if relevant.
SOC	.csv	Spin-orbit coupling (SOC) matrix between each calculated excited singlet and triplet state.
Summary	.csv	Single-row overview of the calculation metadata and principal results.
Summary	.txt	The same information as above, but presented in a human-readable text format.
UV-Vis	.csv	Simulated UV-Vis absorption spectrum using Gaussian-broadened excited states, plotted on a wavelength (nm) scale.

Vibrations	.csv	Calculated vibrational frequencies.
Geometry	.si	Output geometry in a Digichem-native format, including charge and multiplicity.
Geometry	.xyz	Output geometry in a program-independent format, not including charge or multiplicity.
Result	.sir	Complete processed output from the calculation, in a lossless Digichem-native format, can be used for further processing.

In addition to the automated parsing that is performed by Digichem at the end of each calculation, it is also possible to parse any calculation log file on demand. This is achieved with the '`digichem result`' sub-module, which is capable of writing any of the result files that are normally generated automatically. This allows the user, for example, to analyse a calculation result that was not submitted by Digichem, and still obtain the same data. In addition, although Digichem only currently supports submission to the Gaussian, Turbomole, and Orca calculation backends, it can parse the calculation results from a further 13 calculation engines. In the current version, these are: ADF,[83] DALTON,[84] Firefly,[85] GAMESS,[86] GAMESS-UK,[87] Gaussian,[50] Jaguar,[47] Molcas,[88] Molpro,[89] MOPAC,[90] NBO,[91] NWChem,[92] ORCA,[71] Psi4,[93] Q-Chem,[48] and Turbomole.[49] Meanwhile, the desired output format can be selected by specifying the relevant option after the command, for example CSV can be selected with the '-c' option, or the text summary format with '-s'.

Often, the scientist is only interested in a subset of the results from the calculation output. Digichem supports the extraction of targeted data through filters, which can reference any part of the nested hierarchy of data that Digichem stores internally. For example, the filter `` `-f orbitals` `` will return all orbital information associated with the calculation, while `` `-f orbitals:HOMO` `` will only return information related to the HOMO, and `` `-f orbitals:HOMO:energy` `` will only return the energy of the HOMO, and so on. In addition, Digichem supports not only parsing results from a single calculation result file, but also many simultaneously by specifying multiple output files after the `` `digichem result` `` command. In this way, the results from entire studies can be collated into a single spreadsheet file, and specific results can be extracted using the commands described above, giving the user the flexibility to acquire any result they require with ease.

Calculation Result Storage (`digichem database`)

Processing large datasets is one of the more daunting and tedious tasks faced by the computational chemist. The analysis tools described above greatly facilitate the processing of results from each individual calculation, and the ability to parse multiple output files simultaneously allows for easy consolidation of data. However, the `result` sub-module operates on individual log-files (or the Digichem-specific .sir results file), which is only possible while the calculation data remains on the server. Often, each user is granted only a limited file-storage quota, and so it is beneficial to remove completed calculations from the cluster as soon as they are completed. For large computational screens, this means that not all of the log files will be stored on the cluster at the same time, and so multiple rounds of analysis will need to be performed to collect all the data, or alternatively log files will need to be re-uploaded to the cluster for analysis. Completed log files are also occasionally lost, particularly in long-running studies, and in these cases the calculation must be repeated to obtain the desired data. Meanwhile, in studies that involve multiple researchers, the `result` sub-module does not provide a satisfactory solution for merging the data from each researcher without sharing the underlying log files, which may be impractical if the files are large.

To overcome these issues, Digichem provides the `database` submodule. Firstly, the parsed results from each calculation are automatically stored in an internal database, unique to each user. The data stored are identical to that found in the `Result` (.sir) file (Table **2**), and thus can be used to reconstruct any of the result files normally found in the Result folder of each calculation. This parsed data is significantly more compact than the raw output from the calculation, and so the formatted output from many thousands of calculations can be stored in the database without consuming large amounts of file space. The database can then be queried using the `digichem database` command, and Digichem provides tools to insert data, using `digichem database insert`, delete data, using `digichem database delete`, and extract results using `digichem database search`. In addition to the main database, which is activated by default, Digichem also allows for the configuration of any number of additional databases. These can be used, for example, to construct a shared database between a group of users, or to store the results pertaining to one particular study in a separate container. Databases can also be copied, using the `digichem database slice` command, which allows for convenient and efficient sharing of datasets between researchers. Digichem supports both the JSON-based TinyDB backend for human-readable data storage, and the

binary-based Mongita backend for more efficient storage. Both backend programs operate using file-locks to manage concurrent access, and so can be safely used from multiple processes simultaneously.

The true utility of any database, however, is the ability to query it for certain results, and the commands `search`, `delete` and `slice` all support a simple query language to select which calculation results to interrogate. This query language functions similarly to the filter language for the `digichem result` command. Each query starts by identifying an attribute to search against, for example the HOMO energy, followed by a comparison operator, the most common of which are '<' (less than), '<=' (less than or equal to), '=' (equal to), '==' (exactly equal to, which is case sensitive for text), '>' (greater than) or '>=' (greater than or equal to). Digichem also supports more advanced queries to allow comparison of a single item in a list, and for molecular substructure searching. Each query is then completed with a value to search against. As a full example, the command `digichem database main search orbitals:values:any:label==HOMO:energy:value<-0.5` would retrieve all calculation results from the main Digichem database with a HOMO energy of less than -0.5 eV. Multiple queries can be specified simultaneously and combine in a logical AND fashion, which permits querying for results that fall between a given range.

In addition to the typical calculation metadata, Digichem also assigns a unique ID to each calculation result. These IDs are calculated from the checksum of the corresponding log file, meaning they are deterministic, and are used to prevent an identical calculation from being inserted into the same database twice. In addition, each calculation has a `history` attribute, which can optionally contain the ID of the calculation that occurred before it and thus generated the geometry for the following calculation. For example, the `history` attribute of an excited-states calculation might contain the ID of the geometry optimisation that preceded it. In this way, the researcher can easily determine the chain of calculations that lead to each result.

Calculation Reports (`digichem report` & `digichem image`)

For experienced practitioners, the tables of data generated by the `result` and `database` commands are an efficient way to access the results of a calculation or study. However, for novice users, these tables may be daunting and confusing because the data they contain is presented without context. If the user does not understand the headings of the table, then the data within it are useless. Likewise, if the user is unaware of the correct name of the datum they require, they cannot use the filter tools to extract it because they do not know what to query. Even for the experienced users, there

are some data that must be visualized to be understood. A classic example for computational chemistry would be an orbital density plot, which shows the electron density distribution of an orbital throughout the molecule, and these types of results must be shown graphically. Even for data that are purely numerical, the user must typically format or produce graphs of the results before they can be understood or shared with a collaborator. This work is time-consuming, yet can be automated.

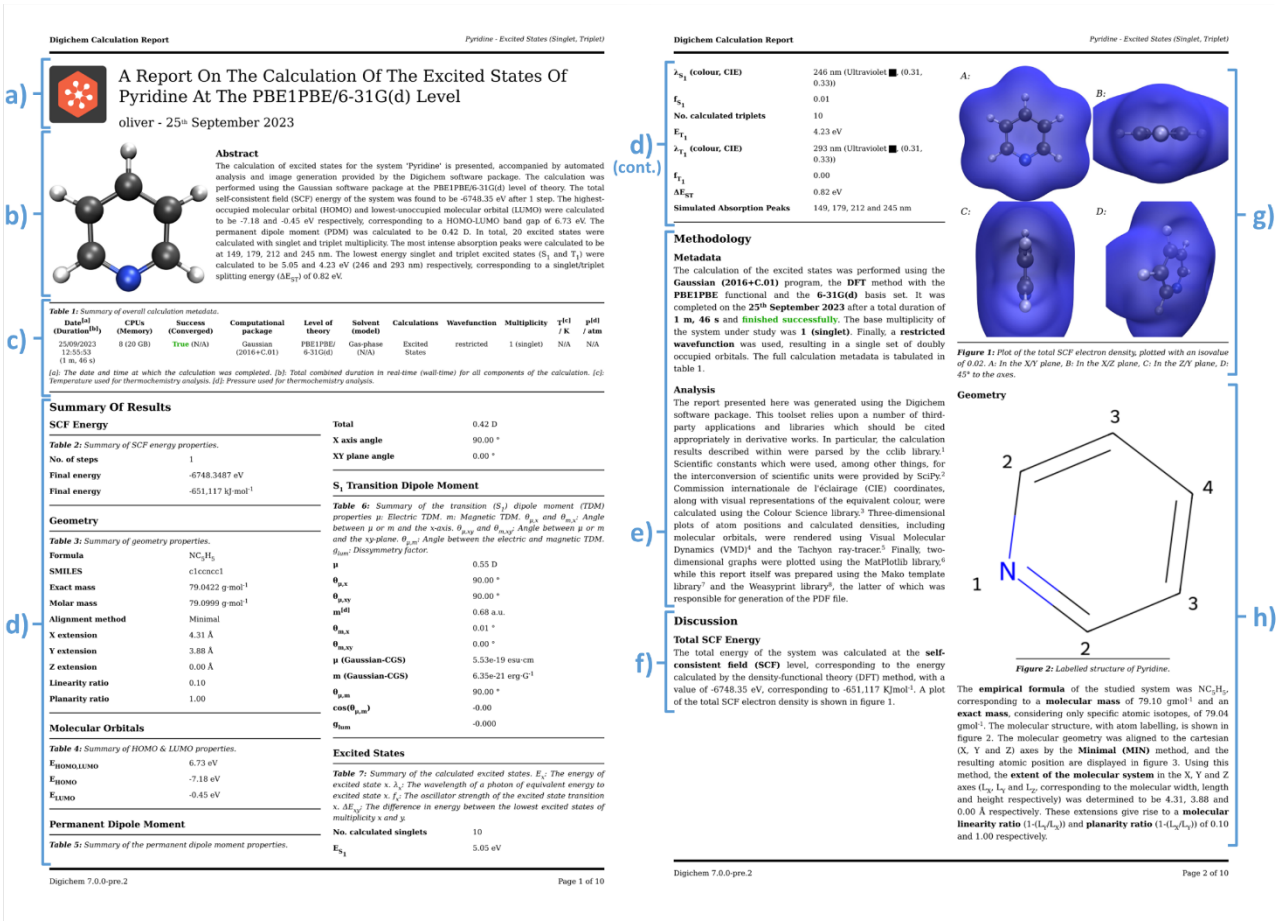


Figure **11**. Excerpts from an example calculation report generated by Digichem. The excited states of pyridine at the PBE0/6-31G* level of theory using the Tamm Dancoff approximation were calculated. a) Header section, b) abstract summary and 3D rendered image of the molecule, c) calculation metadata, d) result summary tables, e) methodology section, f) start of the result discussion section, g) 3D rendered of the total electron density at the DFT level, h) start of the geometry discussion section, including 2D drawing of the molecule.

To achieve this, Digichem provides the `report` sub-module. Following from the successful completion of a calculation, Digichem will automatically generate a single PDF document that

presents all the parsable results of that calculation. This report is presented in a style designed to mimic that of a scientific journal article (Figure **11**), as we envisioned that this format should be familiar to scientists and thus easy to navigate. The report begins with a header section that contains the name of the molecule and the type of calculation that was performed, including which properties were calculated and at what level of theory. This is exemplified in Figure **11a**, which summarizes the calculation of the excited states of pyridine, calculated at the DFT level using the PBE0 functional (named PBE1PBE in Gaussian terminology) and the 6-31G(d) basis set. This section is followed by the 'abstract' (Figure **11b**), which contains a brief textual summary of the most important results of the calculation. This again contains the molecule name and level of theory, but also included are important numerical results, such as the HOMO-LUMO gap, and the energies of the lowest energy singlet and triplet excited states. In place of the traditional graphical abstract, the Digichem report includes a 3D rendered image of the geometry of the molecule. Next is a table of the metadata of the calculation (Figure **11c**), which includes the calculation engine used (in this case, Gaussian 19) and the execution time. Next is the summary section (Figure **11d**), in which headline results from the calculation are displayed in a tabular format. The data presented here are identical to those shown in the summary text result file generated by the `result` sub-module, but here additional captions and more descriptive headings provide context to the tables. A methodology section follows (Figure **11e**), which contains a textual description of the same metadata shown in Figure **11c**, followed by a brief description of the analysis performed by Digichem itself. This latter section also includes references to the various libraries used to help generate the report, such as Weasyprint,[94] Mako,[95] and cclib.[56] The final section shown is the discussion, which occupies the bulk of the report. Visible here is the discussion of the total system energy (Figure **11f**), with an accompanying 3D plot of the total electron density of the system at the PBE0 level (Figure **11g**), and a discussion of the molecular geometry (Figure **11h**) including a 2D drawing of the molecule.

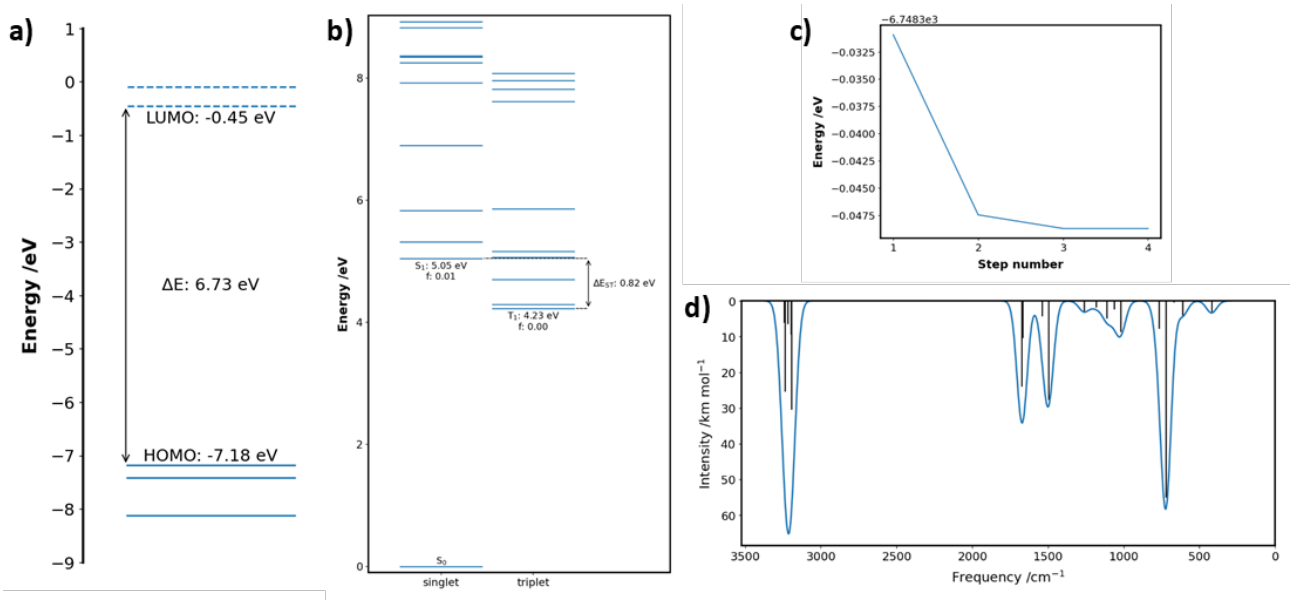


Figure **12**. Example graphs generated by Digichem. a) Graph of HOMO and LUMO energies, and nearby orbitals, b) graph of electronic excited states, c) graph of total system energies with optimisation step number, d) graph of simulated IR absorption graph.

Individual sub-section discussions cover all the results from the calculation that Digichem is able to parse, which includes total system energies (at the SCF/DFT, MP and/or CC levels), molecular geometry, orbitals, permanent and/or transition dipole moments (PDM/TDM), that latter of which includes the calculated dissymmetry factor between the electronic and magnetic TDM,[81] if both are present, electronic excited states, and vibrational frequencies. Each discussion section is populated with important numerical results and provides general context for the results, to help the user to better understand the data that they have computed. We note that we do not use artificial intelligence or machine learning models to write any part of the report or interpret the results, they are compiled entirely from predetermined templates.

Where relevant, these discussion are aided by graphical representations of the data, which are drawn using the Matplotlib library.[96] This includes a graph of the total system energy plotted against each optimisation step gap (Figure **12c**), which is useful for diagnosing convergence problems, a graph of the HOMO-LUMO and close-lying orbitals gap (Figure **12a**), and a graph of the electronic excited-state energies (Figure **12b**). Digichem is also capable of simulating UV-Vis absorption/emission, IR absorption and NMR spectra, by applying a Gaussian line-broadening

function to the calculated electronic excited states, vibrational frequencies (Figure **12c**) and/or NMR shielding parameters and coupling constants.

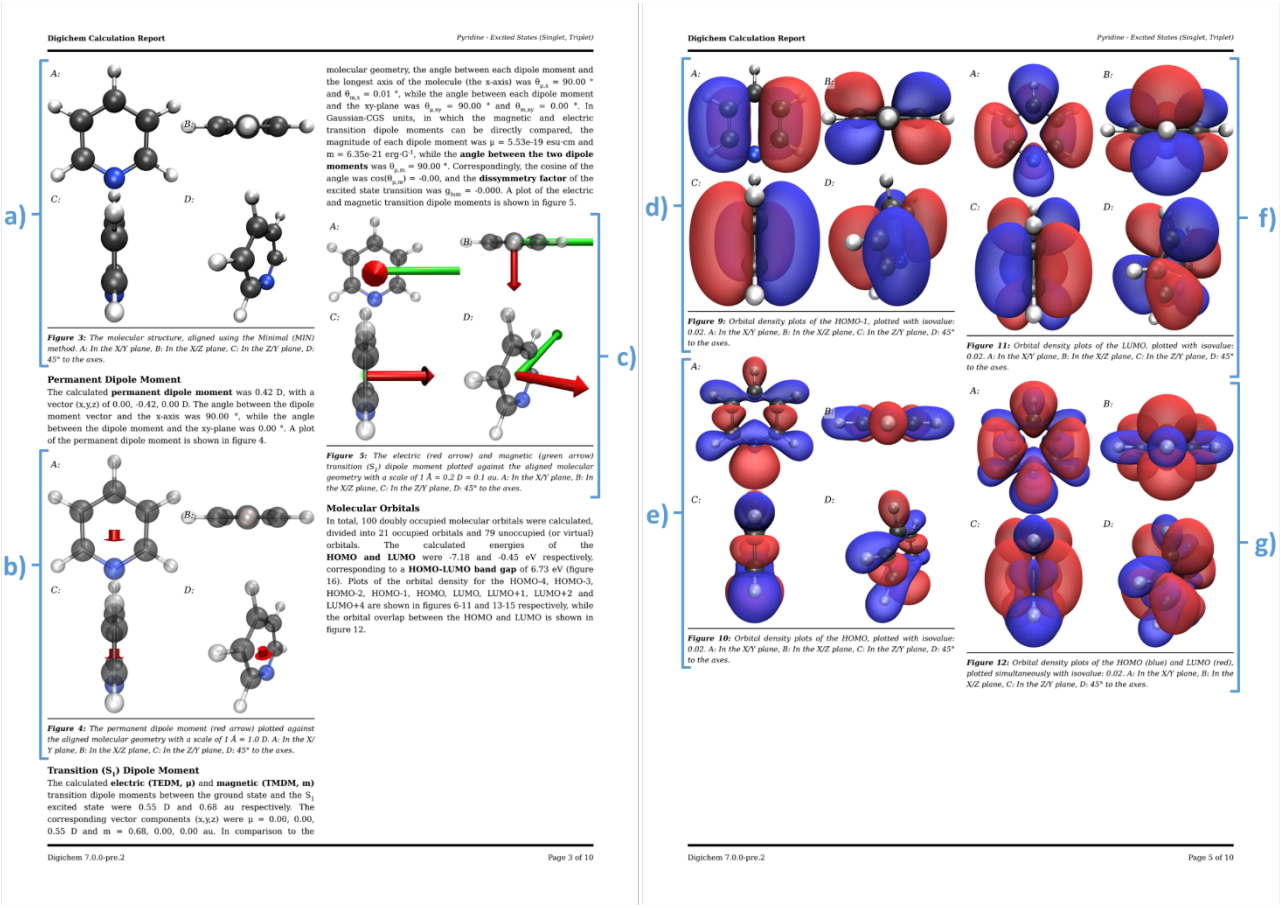


Figure **13**. Excerpts from an example calculation report generated by Digichem, demonstrating the inclusion of 3D renders. The calculation was of the excited states of pyridine at the PBE0/6-31G* level of theory using the Tamm Dancoff approximation. a) render of the molecular geometry, b) the same, showing the calculated permanent dipole moment of the T_1 state, c) the same, showing the calculated electric (red) and magnetic (green) transition dipole moment of the S_1 state, d) render of the HOMO-1 density, e) render of the HOMO density, f) render of the LUMO density, g) render of the HOMO (blue) and LUMO (red) orbital densities, overlayed. The colour of the orbital lobes in d), e) and f) correspond to the different phases of the orbital.

In addition to these graphs, the Digichem calculation report sub-module can create three-dimensional renders of the molecular geometry (Figure **13a**), which can be optionally augmented with a plot of the permanent dipole moment (Figure **13b**) and/or a selected transition dipole moment (Figure **13c**). In the case of the TDM, both the electric and magnetic components are plotted, if available, as the

relative orientation of both is an important parameter for the design of circularly-polarized light (CPL) emitters,[81] amongst other applications. Digichem can also generate renders of electron densities, and currently supports orbital densities, total SCF density, spin-density (for open-shell systems), natural-transition orbital densities, and excited-state difference densities. In the case of orbitals, Digichem only plots the densities of the HOMO (Figure **13e**), the LUMO (Figure **13f**), the HOMO-LUMO pair together (Figure **13g**), and any orbitals that are calculated to have a significant contribution to an electronic excited-state transition, by default. The 'significance' of each orbital is based on the probability of an electronic transition involving the orbital; by default, orbitals involved in transitions with > 20% probability are deemed to be significant and are included, although this value can be adjusted by the user. Any additional orbital can also be manually requested for each calculation. As an example, Figure **13d** shows a render of the HOMO-1 orbital, because the T_2 excited state of pyridine has a 67% contribution from a HOMO-1 → LUMO+1 transition. The LUMO+1 orbital was also therefore rendered in this report but is not shown in Figure **13**. Each 3D image is rendered from four different perspectives to allow the researcher to view the molecule from multiple angles, these are: 1) along the z-axis; 2) along the y-axis; 3) along the x-axis; and 4) at 45° to the three axes. Digichem currently supports two rendering engines for generating these images: VMD,[97,98] which is well-established software in the field of computational chemistry, but is only free for academic use; and Blender,[99] using the Beautiful Atoms plugin,[100] which by contrast is less well established but is open-source. All the examples showcased here are rendered using VMD, but the content is the same regardless of the engine used; only the visual style of the render is different.

In the same manner as the `result` sub-module, the report program has both an automated and manual implementation, and reports can be generated at any time using the `digichem report` command. As for the `result` program, this sub-module can also be used to parse calculations that were not submitted with Digichem, and so it is not necessary to re-submit old, completed calculations only to generate a new report. However, unlike for numerical results, image generation requires the binary checkpoint file from the calculation, in addition to the text-based log file, and if this is absent then the calculation will need to be repeated. Regardless of whether it is generated automatically, or through the `digichem report` command, each report contains not only the final PDF file, but also individual image files for every graph and 3D render used within it. These images are available in both a portable, low-quality JPEG format, and a publication-ready PNG format, and can be readily included in the user's own works. Digichem also offers the `digichem image` command to generate

only single images, for situations in which the entire report is not needed or the user requires an image that is not included by default. For example, the command `digichem image output.log --image HOMO-10` can be used to render the HOMO-10, which would typically be absent from the automatically generated report, while the command `digichem image output.log --list` can be used to list all the available images that Digichem can render from a given calculation.

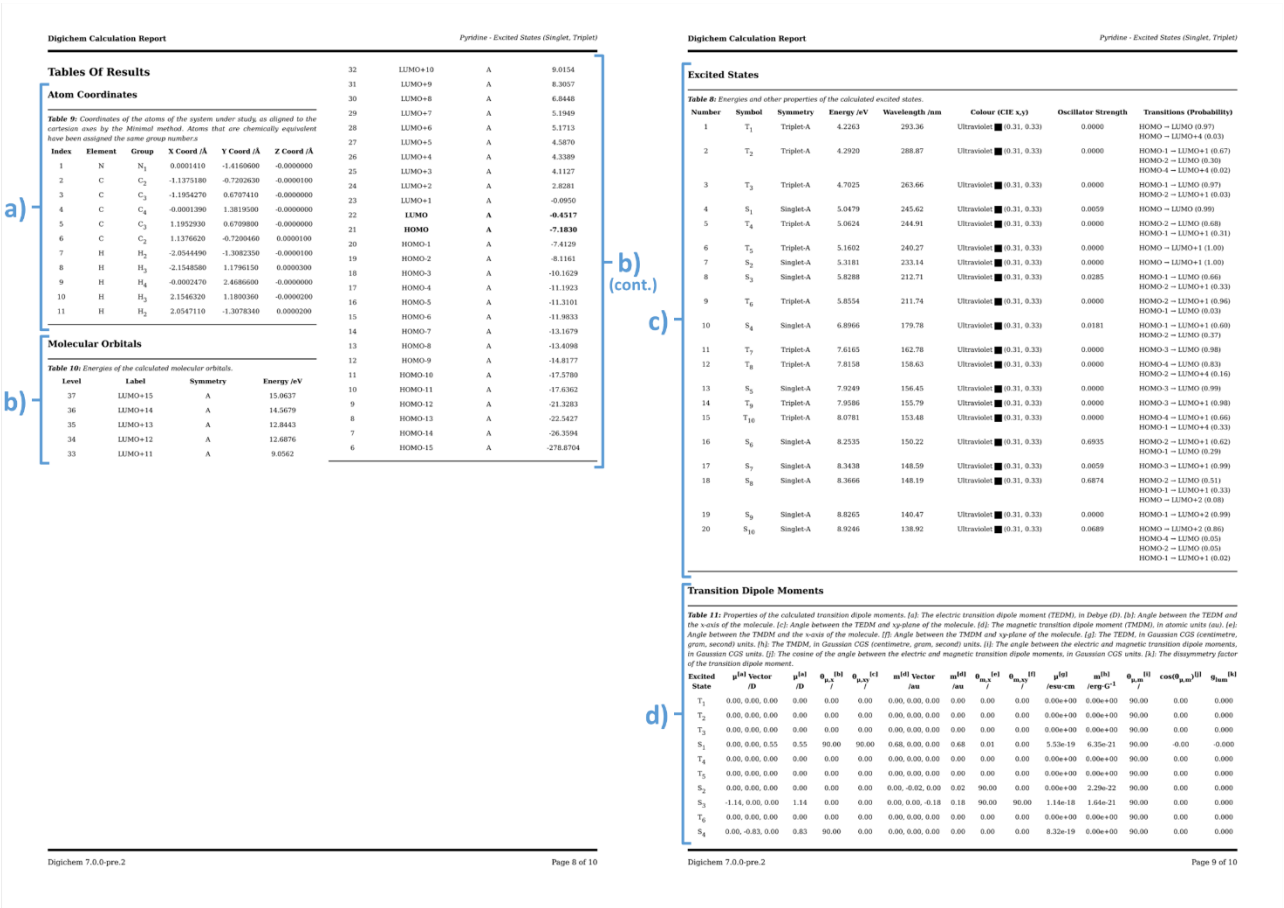


Figure **14**. Excerpts from an example calculation report generated by Digichem, demonstrating tabulated data. The excited states of pyridine at the PBE0/6-31G* level of theory using the Tamm Dancoff approximation were calculated. a) table of molecular geometry, b) table of selected molecular orbitals, c) table of electronic excited states, d) table of transition dipole moments.

Following from the discussion section, the penultimate part of each report contains the tables of results. This section contains tables of all the data that were parsed by Digichem from the calculation, and their structure broadly matches those of the CSV files written to the Result directory at the end of the calculation. The results tabulated here include the molecular geometry (Figure **14a**),

selected molecular orbitals energies and symmetry (Figure **14b**, which orbitals appear here is configurable), electronic excited states (Figure **14c**), including energies, symmetries, multiplicities, equivalent photon wavelengths, oscillator strengths and contributing orbitals, transition dipole moments (Figure **14d**), including parameters important for CPL such as the calculated dissymmetry factor, spin-orbit couplings, vibration frequencies and symmetries, and NMR shielding parameters. These tables serve as an in-depth depository of the calculated results, in contrast to the higher-level analysis provided by the discussion section. Finally, the last part of the report (not shown here), is the bibliography, containing references to the external libraries and programs used by Digichem to generate the report. Together, the data contained in each calculation report should be sufficient for the needs of computational chemistry users of all experience levels, and in many cases, they remove the requirement for the scientist to use external tools to process or format the calculation output. The automatic generation of publication-ready 3D renders saves considerable time for the chemist, both because this task is tedious, and because the renders themselves take time to compute. Lastly, the portable nature of the PDF means it is ideally suited for the sharing of results with collaborators and colleagues, and because all the results of the calculation are included within it (at least to the extent parsable by Digichem), it is rare that the raw calculation output needs to be distributed.

Program Status and Future Work

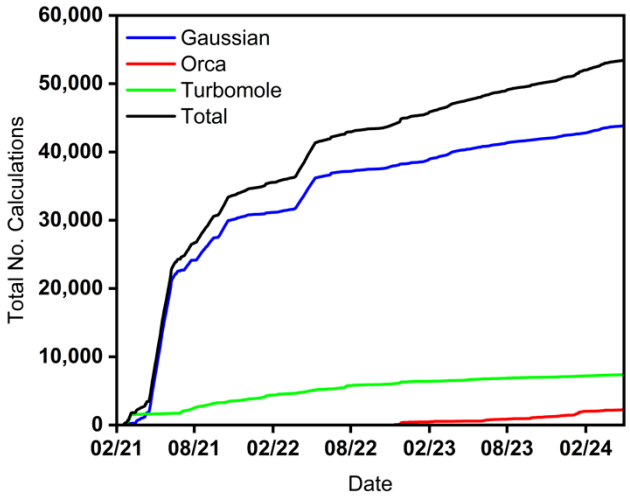


Figure **15**. Graph of the running total number of calculations submitted with Digichem in the Zysman-Colman research group over time.

Digichem has been in development for over three years, during which time it has been extensively tested by the Zysman-Colman research group. Since mid-February 2021 (when we first began recording usage information) to the end of April 2024, Digichem has been successfully used to submit 53,406 calculations (Figure **15a**), an average of 46 calculations per day. By the time of writing of this article, Digichem has entirely replaced the manually-operated calculation pipeline for nearly all of the group's research. The program currently supports three computational engines, which are Gaussian[50], Turbomole,[49] and Orca[71], with the latter only being introduced in November of 2022 (Figure **15**, red line). Due to its more recent inclusion, some of the more advanced analysis features are not yet supported for Orca, including natural transition orbitals and difference density plots, and these are features we intend to incorporate soon. We are actively considering the support of new computational engines, particularly those that offer complementary types of calculations that are not supported by the current roster of backend programs. In particular, the Python native computational package PySCF[101] appears to be an ideal candidate, considering it shares the same programming language as Digichem, it has an open-source license, and it supports a wide-range of double-hybrid DFT functionals, which have recently shown great promise for the prediction of challenging molecular properties in the field of thermally activated delayed fluorescence.[102] Meanwhile, we are continuing to expand the range of metadata that Digichem can parse, and in a forthcoming version we will add support for recognising the different excited states methodologies (e.g., TDA-DFT vs TD-DFT), as well as pertinent performance data, such as the number of CPUs and the amount of memory that was allocated to each calculation. We are also looking to expand upon the program's support for in-series calculation queuing, as Digichem is not currently able to automatically submit multiple calculations from one completed calculation in a branching fashion. Finally, we intend to develop an additional web-based interface to the program to further increase the approachability of computational chemistry, which would be particularly appreciated by the novice user.

Availability and licensing

We have demonstrated through our continuing usage that Digichem is ready for use in active research environments. To facilitate this, the Digichem project has been split into two main components. Core components of the program (Digichem-core) have been released as a python library under the

permissive, open-source BSD-3-clause license. This library is freely available for any purpose and can be incorporated into computational workflows by the user. Currently, the library contains functionality pertaining to results parsing, image generation (both 2D graphs and 3D density plots), file interconversion, simulated spectroscopy (vibrational frequencies, nuclear magnetic resonances etc.), and other miscellaneous functions. Meanwhile, the full program is available in a closed-source format (i.e., compiled binary only) that contains bundled dependencies (produced using PyInstaller[103]). This program is released under a timed license that is free to use for any purpose, but automatically expires after a set duration (currently set to 3 months from release). New releases (compiled automatically every night) are automatically upgraded with a new license with a new expiry. We have chosen this licensing model for two reasons:

- To ensure users remain up-to-date with recent releases. As the software license expires every 3 months, this ensures the user updates the program at least this frequently. This is important, particularly during this rapid development phase, to ensure that crucial bug fixes (as well as new features) are distributed to end-users. We have chosen 3 months as a compromise between convenience (to not force constant updates) and recentness.
- All software requires updates, maintenance, and development to remain useful and relevant. In a fully open-source project, this requires a critical mass of committed developers to sustain, many (if not all) of whom are not directly paid for their time. This can be difficult to achieve, especially in academia, as evidenced by the examples of software that do not see updates past their initial publication, and/or are no longer available.[104–106] We are committed to the continued development of Digichem, and acknowledge that this cannot be done for free, forever. By adopting this licensing scheme, we are able to explore funding strategies for future development, either through commercialisation, sponsorship, or other means.

Digichem-core is available from ref.,[107] while the full Digichem program is available from ref.[108]

Conclusions

We have developed a program designed to automate and simplify the computational chemistry pipeline. We have included tools that reduce the tedium, duration, and likelihood of errors in performing calculation submission, management, and analysis for studies of all sizes, but we have particularly focused on performing large-scale computational screens where these issues are normally

exasperated. The program is designed to be used by computational chemists of all skill levels and experience, but we expect it to be of particular value to novice users, who would normally find the process of learning the intricacies of the pipeline the most daunting. We have extensively tested this program over a period of more than three years, and the future direction of the project has been outlined. Through the continued development of Digichem, we continue to strive towards making computational chemistry accessible for all.

Acknowledgements

The authors would like to acknowledge Mr. Vlad Tataranu for his advice, and Dr. Ettore Crovini, Dr. Tomas Matulaitis, and Dr. Campbell Mackenzie for their assistance in testing early beta versions of the program. This research was financially supported by the European Research Council under the European Union's Horizon 2020 Framework Programme (FP/2014-2020)/ERC grant agreement no. 640012 (ABLASE). We thank the Quantum-Materials Centre for Doctoral Training at the University of St Andrews for support. E.Z.-C. acknowledges the Engineering and Physical Sciences Research Council for support through grants EP/P010482/1 and EP/W007517/1.

Conflicts of Interest

The authors O. S. L. and E. Z.-C. are currently exploring avenues to commercialize parts of the software described in this article.

References

1 Y. Cao, J. Romero, J. P. Olson, M. Degroote, P. D. Johnson, M. Kieferová, I. D. Kivlichan, T. Menke, B. Peropadre, N. P. D. Sawaya, S. Sim, L. Veis and A. Aspuru-Guzik, *Chem. Rev.*, 2019, **119**, 10856–10915.
2 W. Heitler and F. London, *Z. Phys.*, 1927, **44**, 455–472.
3 J. Liu and X. He, *WIREs Comput. Mol. Sci.*, 2023, **13**, e1650.
4 J. M. Soler, E. Artacho, J. D. Gale, A. García, J. Junquera, P. Ordejón and D. Sánchez-Portal, *J. Phys. Condens. Matter*, 2002, **14**, 2745–2779.
5 S. Grimme and P. R. Schreiner, *Angew. Chem. Int. Ed.*, 2018, **57**, 4170–4176.
6 J. Wilhelm, D. Golze, L. Talirz, J. Hutter and C. A. Pignedoli, *J. Phys. Chem. Lett.*, 2018, **9**, 306–312.
7 W. Kohn and L. J. Sham, *Phys. Rev.*, 1965, **140**, A1133–A1138.
8 P. Hohenberg and W. Kohn, *Phys. Rev.*, 1964, **136**, B864–B871.
9 L. H. Thomas, *Math. Proc. Camb. Philos. Soc.*, 1927, **23**, 542–548.
10 E. Fermi, *Rend Accad Naz Lincei*, 1927, **6**, 602–607.
11 P. A. M. Dirac, *Math. Proc. Camb. Philos. Soc.*, 1930, **26**, 376–385.
12 J. C. Slater, *Phys. Rev.*, 1951, **81**, 385–390.
13 S. H. Vosko, L. Wilk and M. Nusair, *Can. J. Phys.*, 1980, **58**, 1200–1211.
14 P. J. Stephens, F. J. Devlin, C. F. Chabalowski and M. J. Frisch, *J. Phys. Chem.*, 1994, **98**, 11623–11627.
15 A. D. Becke, *J. Chem. Phys.*, 1993, **98**, 1372–1377.
16 C. Lee, W. Yang and R. G. Parr, *Phys. Rev. B*, 1988, **37**, 785–789.
17 J. P. Perdew, K. Burke and M. Ernzerhof, *Phys. Rev. Lett.*, 1996, **77**, 3865–3868.
18 J. P. Perdew, K. Burke and M. Ernzerhof, *Phys. Rev. Lett.*, 1997, **78**, 1396–1396.
19 C. Adamo and V. Barone, *J. Chem. Phys.*, 1999, **110**, 6158–6170.
20 Y. Zhao, B. J. Lynch and D. G. Truhlar, *J. Phys. Chem. A*, 2004, **108**, 4786–4791.
21 S. Grimme, *J. Chem. Phys.*, 2006, **124**, 034108.
22 D. Cremer, *WIREs Comput. Mol. Sci.*, 2011, **1**, 509–530.
23 Chr. Møller and M. S. Plesset, *Phys. Rev.*, 1934, **46**, 618–622.
24 F. Coester, *Nucl. Phys.*, 1958, **7**, 421–424.
25 F. Coester and H. Kümmel, *Nucl. Phys.*, 1960, **17**, 477–485.
26 J. ?I?Ek and J. Paldus, *Int. J. Quantum Chem.*, 1971, **5**, 359–379.
27 J. Čížek, *J. Chem. Phys.*, 1966, **45**, 4256–4266.
28 A. M. Burow, M. Sierka and F. Mohamed, *J. Chem. Phys.*, 2009, **131**, 214101.
29 C. Hättig, *J. Chem. Phys.*, 2003, **118**, 7751–7761.
30 C. Hättig and A. Köhn, *J. Chem. Phys.*, 2002, **117**, 6939–6951.
31 C. Hättig and F. Weigend, *J. Chem. Phys.*, 2000, **113**, 5154–5161.
32 C. Hättig, A. Hellweg and A. Köhn, *Phys. Chem. Chem. Phys.*, 2006, **8**, 1159–1169.
33 M. Sierka, A. Hogekamp and R. Ahlrichs, *J. Chem. Phys.*, 2003, **118**, 9136–9148.
34 F. Weigend, A. Köhn and C. Hättig, *J. Chem. Phys.*, 2002, **116**, 3175–3183.
35 R. Izsák and F. Neese, *J. Chem. Phys.*, 2011, **135**, 144105.
36 R. Izsák, A. Hansen and F. Neese, *Mol. Phys.*, 2012, **110**, 2413–2417.
37 R. Robidas and C. Y. Legault, *J. Chem. Inf. Model.*, 2022, **62**, 1147–1153.
38 W. F. Polik and J. R. Schmidt, *WIREs Comput. Mol. Sci.*, 2022, **12**, e1554.

39 M. J. Perri and S. H. Weber, *J. Chem. Educ.*, 2014, **91**, 2206–2208.
40 J. H. Jensen and J. C. Kromann, *J. Chem. Educ.*, 2013, **90**, 1093–1095.
41 A. B. Yoo, M. A. Jette and M. Grondona, .
42 Winmostar V11 X-Ability Co. Ltd., Tokyo, Japan, 2023.
43 Schrödinger Release 2023-3: Maestro Schrödinger, LLC, New York, NY, 2023.
44 Spartan 20 Wavefunction, Inc., Irvine 2023.
45 C. Steffen, K. Thomas, U. Huniar, A. Hellweg, O. Rubner and A. Schroer, *J. Comput. Chem.*, 2010, **31**, 2967–2970.
46 R. Dennington, T. A. Keith and J. M. Millam, GaussView, Version 6 Semichem Inc., Shawnee Mission, KS, 2016.
47 A. D. Bochevarov, E. Harder, T. F. Hughes, J. R. Greenwood, D. A. Braden, D. M. Philipp, D. Rinaldo, M. D. Halls, J. Zhang and R. A. Friesner, *Int. J. Quantum Chem.*, 2013, 2110–2142.
48 Y. Shao, Z. Gan, E. Epifanovsky, A. T. B. Gilbert, M. Wormit, J. Kussmann, A. W. Lange, A. Behn, J. Deng, X. Feng, D. Ghosh, M. Goldey, P. R. Horn, L. D. Jacobson, I. Kaliman, R. Z. Khaliullin, T. Kuś, A. Landau, J. Liu, E. I. Proynov, Y. M. Rhee, R. M. Richard, M. A. Rohrdanz, R. P. Steele, E. J. Sundstrom, H. L. Woodcock, P. M. Zimmerman, D. Zuev, B. Albrecht, E. Alguire, B. Austin, G. J. O. Beran, Y. A. Bernard, E. Berquist, K. Brandhorst, K. B. Bravaya, S. T. Brown, D. Casanova, C.-M. Chang, Y. Chen, S. H. Chien, K. D. Closser, D. L. Crittenden, M. Diedenhofen, R. A. DiStasio, H. Do, A. D. Dutoi, R. G. Edgar, S. Fatehi, L. Fusti-Molnar, A. Ghysels, A. Golubeva-Zadorozhnaya, J. Gomes, M. W. D. Hanson-Heine, P. H. P. Harbach, A. W. Hauser, E. G. Hohenstein, Z. C. Holden, T.-C. Jagau, H. Ji, B. Kaduk, K. Khistyaev, J. Kim, J. Kim, R. A. King, P. Klunzinger, D. Kosenkov, T. Kowalczyk, C. M. Krauter, K. U. Lao, A. D. Laurent, K. V. Lawler, S. V. Levchenko, C. Y. Lin, F. Liu, E. Livshits, R. C. Lochan, A. Luenser, P. Manohar, S. F. Manzer, S.-P. Mao, N. Mardirossian, A. V. Marenich, S. A. Maurer, N. J. Mayhall, E. Neuscamman, C. M. Oana, R. Olivares-Amaya, D. P. O'Neill, J. A. Parkhill, T. M. Perrine, R. Peverati, A. Prociuk, D. R. Rehn, E. Rosta, N. J. Russ, S. M. Sharada, S. Sharma, D. W. Small, A. Sodt, T. Stein, D. Stück, Y.-C. Su, A. J. W. Thom, T. Tsuchimochi, V. Vanovschi, L. Vogt, O. Vydrov, T. Wang, M. A. Watson, J. Wenzel, A. White, C. F. Williams, J. Yang, S. Yeganeh, S. R. Yost, Z.-Q. You, I. Y. Zhang, X. Zhang, Y. Zhao, B. R. Brooks, G. K. L. Chan, D. M. Chipman, C. J. Cramer, W. A. Goddard, M. S. Gordon, W. J. Hehre, A. Klamt, H. F. Schaefer, M. W. Schmidt, C. D. Sherrill, D. G. Truhlar, A. Warshel, X. Xu, A. Aspuru-Guzik, R. Baer, A. T. Bell, N. A. Besley, J.-D. Chai, A. Dreuw, B. D. Dunietz, T. R. Furlani, S. R. Gwaltney, C.-P. Hsu, Y. Jung, J. Kong, D. S. Lambrecht, W. Liang, C. Ochsenfeld, V. A. Rassolov, L. V. Slipchenko, J. E. Subotnik, T. Van Voorhis, J. M. Herbert, A. I. Krylov, P. M. W. Gill and M. Head-Gordon, *Mol. Phys.*, 2015, **113**, 184–215.
49 D. Oelkrug, H. J. Egelhaaf and J. Haiber, *Thin Solid Films*, 1996, **284–285**, 267–270.
50 M. J. Frisch, G. W. Trucks, H. B. Schlegel, G. E. Scuseria, M. A. Robb, J. R. Cheeseman, G. Scalmani, V. Barone, G. A. Petersson, H. Nakatsuji, X. Li, M. Caricato, A. V. Marenich, J. Bloino, B. G. Janesko, R. Gomperts, B. Mennucci, H. P. Hratchian, J. V. Ortiz, A. F. Izmaylov, J. L. Sonnenberg, D. Williams-Young, F. Ding, F. Lipparini, F. Egidi, J. Goings, B. Peng, A. Petrone, T. Henderson, D. Ranasinghe, V. G. Zakrzewski, J. Gao, N. Rega, G. Zheng, W. Liang, M. Hada, M. Ehara, K. Toyota, R. Fukuda, J. Hasegawa, M. Ishida, T. Nakajima, Y. Honda, O. Kitao, H. Nakai, T. Vreven, K. Throssell, J. A. Montgomery Jr., J. E. Peralta, F. Ogliaro, M. J. Bearpark, J. J. Heyd, E. N. Brothers, K. N. Kudin, V. N. Staroverov, T. A. Keith, R. Kobayashi, J. Normand, K. Raghavachari, A. P. Rendell, J. C. Burant, S. S. Iyengar, J. Tomasi, M. Cossi, J. M. Millam, M. Klene, C. Adamo, R. Cammi, J. W. Ochterski, R. L. Martin, K. Morokuma, O. Farkas, J. B. Foresman and D. J. Fox, Gaussian 16, Revision C.01 2016.
51 G. Schaftenaar and J. H. Noordik, .

52 A.-R. Allouche, *J. Comput. Chem.*, 2011, **32**, 174–182.
53 M. D. Hanwell, D. E. Curtis, D. C. Lonie, T. Vandermeersch, E. Zurek and G. R. Hutchison, *J. Cheminformatics*, 2012, **4**, 17.
54 N. M. O'Boyle, M. Banck, C. A. James, C. Morley, T. Vandermeersch and G. R. Hutchison, *J. Cheminformatics*, 2011, **3**, 33.
55 N. M. O'Boyle, C. Morley and G. R. Hutchison, *Chem. Cent. J.*, 2008, **2**, 5.
56 N. M. O'Boyle, A. L. Tenderholt and K. M. Langner, *J. Comput. Chem.*, 2008, **29**, 839–845.
57 The author O.S.L. has contributed code to the open-source cclib project.
58 RDKit: Open-source cheminformatics, https://www.rdkit.org.
59 C. Steinbeck, C. Hoppe, S. Kuhn, M. Floris, R. Guha and E. Willighagen, *Curr. Pharm. Des.*, 2006, **12**, 2111–2120.
60 C. Steinbeck, Y. Han, S. Kuhn, O. Horlacher, E. Luttmann and E. Willighagen, *J. Chem. Inf. Comput. Sci.*, 2003, **43**, 493–500.
61 E. L. Willighagen, J. W. Mayfield, J. Alvarsson, A. Berg, L. Carlsson, N. Jeliazkova, S. Kuhn, T. Pluskal, M. Rojas-Chertó, O. Spjuth, G. Torrance, C. T. Evelo, R. Guha and C. Steinbeck, *J. Cheminformatics*, 2017, **9**, 33.
62 J. W. May and C. Steinbeck, *J. Cheminformatics*, 2014, **6**, 3.
63 A. Hjorth Larsen, J. Jørgen Mortensen, J. Blomqvist, I. E. Castelli, R. Christensen, M. Dułak, J. Friis, M. N. Groves, B. Hammer, C. Hargus, E. D. Hermes, P. C. Jennings, P. Bjerre Jensen, J. Kermode, J. R. Kitchin, E. Leonhard Kolsbjerg, J. Kubal, K. Kaasbjerg, S. Lysgaard, J. Bergmann Maronsson, T. Maxson, T. Olsen, L. Pastewka, A. Peterson, C. Rostgaard, J. Schiøtz, O. Schütt, M. Strange, K. S. Thygesen, T. Vegge, L. Vilhelmsen, M. Walter, Z. Zeng and K. W. Jacobsen, *J. Phys. Condens. Matter*, 2017, **29**, 273002.
64 The author O.S.L. has contributed code to the open-source ASE project.
65 J. V. Alegre-Requena, S. Sowndarya S. V., R. Pérez-Soto, T. M. Alturaifi and R. S. Paton, *WIREs Comput. Mol. Sci.*, 2023, e1663.
66 I. Ward and Contributors, Urwid http://urwid.org/index.html 2020.
67 L. W. Chung, W. M. C. Sameera, R. Ramozzi, A. J. Page, M. Hatanaka, G. P. Petrova, T. V. Harris, X. Li, Z. Ke, F. Liu, H.-B. Li, L. Ding and K. Morokuma, *Chem. Rev.*, 2015, **115**, 5678–5796.
68 E. Paquet and H. L. Viktor, *BioMed Res. Int.*, 2015, **2015**, 1–18.
69 M. J. Frisch, G. W. Trucks, H. B. Schlegel, G. E. Scuseria, M. A. Robb, J. R. Cheeseman, G. Scalmani, V. Barone, B. Mennucci, G. A. Petersson, H. Nakatsuji, M. Caricato, X. Li, H. P. Hratchian, A. F. Izmaylov, J. Bloino, G. Zheng, J. L. Sonnenberg, M. Hada, M. Ehara, K. Toyota, R. Fukuda, J. Hasegawa, M. Ishida, T. Nakajima, Y. Honda, O. Kitao, H. Nakai, T. Vreven, J. A. Montgomery, J. E. Peralta, F. Ogliaro, M. Bearpark, J. J. Heyd, E. Brothers, K. N. Kudin, V. N. Staroverov, T. Keith, R. Kobayashi, J. Normand, K. Raghavachari, A. Rendell, J. C. Burant, S. S. Iyengar, J. Tomasi, M. Cossi, N. Rega, J. M. Millam, M. Klene, J. E. Knox, J. B. Cross, V. Bakken, C. Adamo, J. Jaramillo, R. Gomperts, R. E. Stratmann, O. Yazyev, A. J. Austin, R. Cammi, C. Pomelli, J. W. Ochterski, R. L. Martin, K. Morokuma, V. G. Zakrzewski, G. A. Voth, P. Salvador, J. J. Dannenberg, S. Dapprich, A. D. Daniels, Farkas, J. B. Foresman, J. V. Ortiz, J. Cioslowski and D. J. Fox, Gaussian 09, Revision D.01 citeulike-article-id:9096580 2013.
70 R. Ahlrichs, M. Bär, M. Häser, H. Horn and C. Kölmel, *Chem. Phys. Lett.*, 1989, **162**, 165–169.
71 F. Neese, F. Wennmohs, U. Becker and C. Riplinger, *J. Chem. Phys.*, 2020, **152**, 224108.
72 E. K. U. Gross and W. Kohn, in *Advances in Quantum Chemistry*, Elsevier, 1990, vol. 21, pp. 255–291.
73 S. Hirata and M. Head-Gordon, *Chem. Phys. Lett.*, 1999, **314**, 291–299.
74 A. Hellman, B. Razaznejad and B. I. Lundqvist, *J. Chem. Phys.*, 2004, **120**, 4593–4602.
75 Unicode Consortium, *Unicode Stand. Version 15-1*, 2023, 2500–257F.

76 K. Simonov and Contributors, PyYAML (version 6.0) https://pyyaml.org/ 2021.
77 OpenPBS Altair Engineering Inc., 2023.
78 R. Ditchfield, W. J. Hehre and J. A. Pople, *J. Chem. Phys.*, 1971, **54**, 724–728.
79 F. Weigend and R. Ahlrichs, *Phys. Chem. Chem. Phys.*, 2005, **7**, 3297.
80 T. H. Dunning, *J. Chem. Phys.*, 1989, **90**, 1007–1023.
81 H. Kubo, T. Hirose, T. Nakashima, T. Kawai, J. Hasegawa and K. Matsuda, *J. Phys. Chem. Lett.*, 2021, **12**, 686–695.
82 X. Gao, S. Bai, D. Fazzi, T. Niehaus, M. Barbatti and W. Thiel, *J. Chem. Theory Comput.*, 2017, **13**, 515–524.
83 G. Te Velde, F. M. Bickelhaupt, E. J. Baerends, C. Fonseca Guerra, S. J. A. Van Gisbergen, J. G. Snijders and T. Ziegler, *J. Comput. Chem.*, 2001, **22**, 931–967.
84 K. Aidas, C. Angeli, K. L. Bak, V. Bakken, R. Bast, L. Boman, O. Christiansen, R. Cimiraglia, S. Coriani, P. Dahle, E. K. Dalskov, U. Ekström, T. Enevoldsen, J. J. Eriksen, P. Ettenhuber, B. Fernández, L. Ferrighi, H. Fliegl, L. Frediani, K. Hald, A. Halkier, C. Hättig, H. Heiberg, T. Helgaker, A. C. Hennum, H. Hettema, E. Hjertenæs, S. Høst, I. Høyvik, M. F. Iozzi, B. Jansík, H. J. Aa. Jensen, D. Jonsson, P. Jørgensen, J. Kauczor, S. Kirpekar, T. Kjærgaard, W. Klopper, S. Knecht, R. Kobayashi, H. Koch, J. Kongsted, A. Krapp, K. Kristensen, A. Ligabue, O. B. Lutnæs, J. I. Melo, K. V. Mikkelsen, R. H. Myhre, C. Neiss, C. B. Nielsen, P. Norman, J. Olsen, J. M. H. Olsen, A. Osted, M. J. Packer, F. Pawlowski, T. B. Pedersen, P. F. Provasi, S. Reine, Z. Rinkevicius, T. A. Ruden, K. Ruud, V. V. Rybkin, P. Sałek, C. C. M. Samson, A. S. De Merás, T. Saue, S. P. A. Sauer, B. Schimmelpfennig, K. Sneskov, A. H. Steindal, K. O. Sylvester-Hvid, P. R. Taylor, A. M. Teale, E. I. Tellgren, D. P. Tew, A. J. Thorvaldsen, L. Thøgersen, O. Vahtras, M. A. Watson, D. J. D. Wilson, M. Ziolkowski and H. Ågren, *WIREs Comput. Mol. Sci.*, 2014, **4**, 269–284.
85 A. A. Granovsky, Firefly (version 8) http://classic.chem.msu.su/gran/firefly/index.html.
86 G. M. J. Barca, C. Bertoni, L. Carrington, D. Datta, N. De Silva, J. E. Deustua, D. G. Fedorov, J. R. Gour, A. O. Gunina, E. Guidez, T. Harville, S. Irle, J. Ivanic, K. Kowalski, S. S. Leang, H. Li, W. Li, J. J. Lutz, I. Magoulas, J. Mato, V. Mironov, H. Nakata, B. Q. Pham, P. Piecuch, D. Poole, S. R. Pruitt, A. P. Rendell, L. B. Roskop, K. Ruedenberg, T. Sattasathuchana, M. W. Schmidt, J. Shen, L. Slipchenko, M. Sosonkina, V. Sundriyal, A. Tiwari, J. L. Galvez Vallejo, B. Westheimer, M. Włoch, P. Xu, F. Zahariev and M. S. Gordon, *J. Chem. Phys.*, 2020, **152**, 154102.
87 M. F. Guest, I. J. Bush, H. J. J. Van Dam, P. Sherwood, J. M. H. Thomas, J. H. Van Lenthe, R. W. A. Havenith and J. Kendrick, *Mol. Phys.*, 2005, **103**, 719–747.
88 G. Li Manni, I. Fdez. Galván, A. Alavi, F. Aleotti, F. Aquilante, J. Autschbach, D. Avagliano, A. Baiardi, J. J. Bao, S. Battaglia, L. Birnoschi, A. Blanco-González, S. I. Bokarev, R. Broer, R. Cacciari, P. B. Calio, R. K. Carlson, R. Carvalho Couto, L. Cerdán, L. F. Chibotaru, N. F. Chilton, J. R. Church, I. Conti, S. Coriani, J. Cuéllar-Zuquin, R. E. Daoud, N. Dattani, P. Decleva, C. De Graaf, M. G. Delcey, L. De Vico, W. Dobrautz, S. S. Dong, R. Feng, N. Ferré, M. Filatov(Gulak), L. Gagliardi, M. Garavelli, L. González, Y. Guan, M. Guo, M. R. Hennefarth, M. R. Hermes, C. E. Hoyer, M. Huix-Rotllant, V. K. Jaiswal, A. Kaiser, D. S. Kaliakin, M. Khamesian, D. S. King, V. Kochetov, M. Krośnicki, A. A. Kumaar, E. D. Larsson, S. Lehtola, M.-B. Lepetit, H. Lischka, P. López Ríos, M. Lundberg, D. Ma, S. Mai, P. Marquetand, I. C. D. Merritt, F. Montorsi, M. Mörchen, A. Nenov, V. H. A. Nguyen, Y. Nishimoto, M. S. Oakley, M. Olivucci, M. Oppel, D. Padula, R. Pandharkar, Q. M. Phung, F. Plasser, G. Raggi, E. Rebolini, M. Reiher, I. Rivalta, D. Roca-Sanjuán, T. Romig, A. A. Safari, A. Sánchez-Mansilla, A. M. Sand, I. Schapiro, T. R. Scott, J. Segarra-Martí, F. Segatta, D.-C. Sergentu, P. Sharma, R. Shepard, Y. Shu, J. K. Staab, T. P. Straatsma, L. K. Sørensen, B. N. C. Tenorio, D. G. Truhlar, L. Ungur, M. Vacher, V. Veryazov, T.

A. Voß, O. Weser, D. Wu, X. Yang, D. Yarkony, C. Zhou, J. P. Zobel and R. Lindh, *J. Chem. Theory Comput.*, 2023, acs.jctc.3c00182.
89 H.-J. Werner, P. J. Knowles, F. R. Manby, J. A. Black, K. Doll, A. Heßelmann, D. Kats, A. Köhn, T. Korona, D. A. Kreplin, Q. Ma, T. F. Miller, A. Mitrushchenkov, K. A. Peterson, I. Polyak, G. Rauhut and M. Sibaev, *J. Chem. Phys.*, 2020, **152**, 144107.
90 J. J. P. Stewart, *J. Mol. Model.*, 2013, **19**, 1–32.
91 E. D. Glendening, C. R. Landis and F. Weinhold, *J. Comput. Chem.*, 2019, **40**, 2234–2241.
92 E. Aprà, E. J. Bylaska, W. A. De Jong, N. Govind, K. Kowalski, T. P. Straatsma, M. Valiev, H. J. J. Van Dam, Y. Alexeev, J. Anchell, V. Anisimov, F. W. Aquino, R. Atta-Fynn, J. Autschbach, N. P. Bauman, J. C. Becca, D. E. Bernholdt, K. Bhaskaran-Nair, S. Bogatko, P. Borowski, J. Boschen, J. Brabec, A. Bruner, E. Cauët, Y. Chen, G. N. Chuev, C. J. Cramer, J. Daily, M. J. O. Deegan, T. H. Dunning, M. Dupuis, K. G. Dyall, G. I. Fann, S. A. Fischer, A. Fonari, H. Früchtl, L. Gagliardi, J. Garza, N. Gawande, S. Ghosh, K. Glaesemann, A. W. Götz, J. Hammond, V. Helms, E. D. Hermes, K. Hirao, S. Hirata, M. Jacquelin, L. Jensen, B. G. Johnson, H. Jónsson, R. A. Kendall, M. Klemm, R. Kobayashi, V. Konkov, S. Krishnamoorthy, M. Krishnan, Z. Lin, R. D. Lins, R. J. Littlefield, A. J. Logsdail, K. Lopata, W. Ma, A. V. Marenich, J. Martin Del Campo, D. Mejia-Rodriguez, J. E. Moore, J. M. Mullin, T. Nakajima, D. R. Nascimento, J. A. Nichols, P. J. Nichols, J. Nieplocha, A. Otero-de-la-Roza, B. Palmer, A. Panyala, T. Pirojsirikul, B. Peng, R. Peverati, J. Pittner, L. Pollack, R. M. Richard, P. Sadayappan, G. C. Schatz, W. A. Shelton, D. W. Silverstein, D. M. A. Smith, T. A. Soares, D. Song, M. Swart, H. L. Taylor, G. S. Thomas, V. Tipparaju, D. G. Truhlar, K. Tsemekhman, T. Van Voorhis, Á. Vázquez-Mayagoitia, P. Verma, O. Villa, A. Vishnu, K. D. Vogiatzis, D. Wang, J. H. Weare, M. J. Williamson, T. L. Windus, K. Woliński, A. T. Wong, Q. Wu, C. Yang, Q. Yu, M. Zacharias, Z. Zhang, Y. Zhao and R. J. Harrison, *J. Chem. Phys.*, 2020, **152**, 184102.
93 D. G. A. Smith, L. A. Burns, A. C. Simmonett, R. M. Parrish, M. C. Schieber, R. Galvelis, P. Kraus, H. Kruse, R. Di Remigio, A. Alenaizan, A. M. James, S. Lehtola, J. P. Misiewicz, M. Scheurer, R. A. Shaw, J. B. Schriber, Y. Xie, Z. L. Glick, D. A. Sirianni, J. S. O'Brien, J. M. Waldrop, A. Kumar, E. G. Hohenstein, B. P. Pritchard, B. R. Brooks, H. F. Schaefer, A. Yu. Sokolov, K. Patkowski, A. E. DePrince, U. Bozkaya, R. A. King, F. A. Evangelista, J. M. Turney, T. D. Crawford and C. D. Sherrill, *J. Chem. Phys.*, 2020, **152**, 184108.
94 K. Community, Weasyprint: The Awesome Document Factory https://weasyprint.org.
95 M. Bayer, Mako Templates for Python https://www.makotemplates.org.
96 J. D. Hunter, *Comput. Sci. Eng.*, 2007, **9**, 90–95.
97 W. Humphrey, A. Dalke and K. Schulten, *J. Mol. Graph.*, 1996, **14**, 33–38.
98 J. Stone, Computer Science Department, University of Missouri-Rolla, 1998.
99 Blender Online Community, Blender - a 3D modelling and rendering package Blender Foundation, Blender Institute, Amsterdam 2023.
100 X. Wang, T. Tian and U. Aschauer, Beautiful Atoms: A python library for the manipulation and visualization of atomistic structure using blender (Version 2.1.0) https://github.com/beautiful-atoms/beautiful-atoms 2022.
101 Q. Sun, X. Zhang, S. Banerjee, P. Bao, M. Barbry, N. S. Blunt, N. A. Bogdanov, G. H. Booth, J. Chen, Z.-H. Cui, J. J. Eriksen, Y. Gao, S. Guo, J. Hermann, M. R. Hermes, K. Koh, P. Koval, S. Lehtola, Z. Li, J. Liu, N. Mardirossian, J. D. McClain, M. Motta, B. Mussard, H. Q. Pham, A. Pulkin, W. Purwanto, P. J. Robinson, E. Ronca, E. R. Sayfutyarova, M. Scheurer, H. F. Schurkus, J. E. T. Smith, C. Sun, S.-N. Sun, S. Upadhyay, L. K. Wagner, X. Wang, A. White, J. D. Whitfield, M. J. Williamson, S. Wouters, J. Yang, J. M. Yu, T. Zhu, T. C. Berkelbach, S. Sharma, A. Yu. Sokolov and G. K.-L. Chan, *J. Chem. Phys.*, 2020, **153**, 024109.
102 M. Kondo, *Chem. Phys. Lett.*, 2022, **804**, 139895.

103 Collaborators, PyInstaller https://github.com/pyinstaller/pyinstaller 2024.
104 S. G. Chiodo and M. Leopoldini, *Comput. Phys. Commun.*, 2014, **185**, 676–683.
105 V. Sojo, A. Peraza, F. Ruette, M. Sánchez and A. E. Acosta, *J. Comput. Methods Sci. Eng.*, 2012, **12**, 397–406.
106 H. Borkent, J. Van Rooij, O. Stueker, I. Brunberg and G. Fels, *J. Chem. Educ.*, 2003, **80**, 582.
107 O. S. Lee and E. Zysman-Colman, Digichem-core https://github.com/Digichem-Project/Digichem-core 2024.
108 O. S. Lee and E. Zysman-Colman, Build-boy https://github.com/Digichem-Project/build-boy 2024.

TOC Graphic