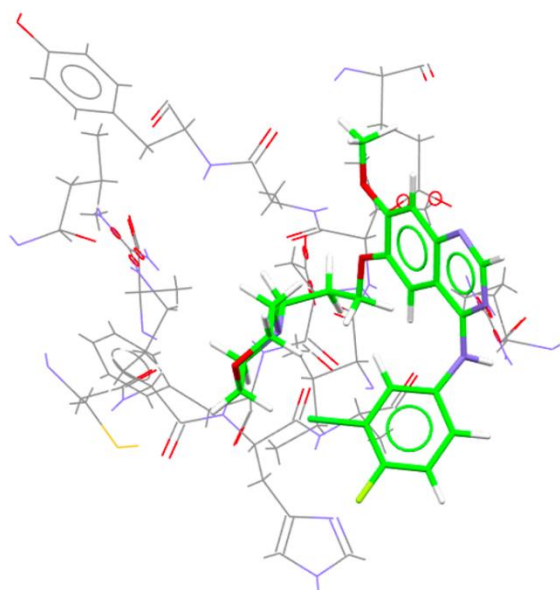


Automating Protein–Ligand Docking with GOLD using the CSD Python API (PYAPI-004)



CCDC
advancing structural science

Contents

Introduction	3
Learning Outcomes.....	3
Pre-required Skills.....	3
Materials.....	3
Docking Tyrosine Kinase Inhibitors	4
Activating the CSD Python API Environment	4
Setting up a Docking Run	5
Generating Conformers	9
Linking Conformer Generation and Docking	11
Conclusions.....	11
Generating and Overlaying Multiple Ligands.....	12
Automatically Docking Multiple Ligands	14
Conclusions.....	16
Extension	16
Summary	17
Next Steps.....	17
How to Cite this Tool.....	17
Feedback.....	17
Glossary.....	18
Basics of Hermes Visualisation.....	20

Introduction

Molecular [*docking*](#) is the most common computational structure-based drug design (SBDD) method and has been widely used ever since the early 1980s. Molecular docking applications in drug discovery are varied, including structure activity studies, lead optimisation and finding potential leads by [*virtual screening*](#). [*GOLD*](#) (Genetic Optimisation for Ligand Docking) is a genetic algorithm for docking flexible ligands into protein [*binding sites*](#). GOLD has been extensively tested and has shown excellent performance for [*pose*](#) prediction and good results for virtual screening. The [*CSD Python API*](#) provides access to the full breadth of functionality that is available within GOLD, and through Python scripting it is possible to build highly tailored custom applications to help you answer research questions, or to automate frequently performed analysis steps. In this workshop you will learn to how to perform a protein-ligand docking using GOLD accessed through the CSD Python API.

Learning Outcomes

After completing this workshop, you will:

- Understand the principles of docking with GOLD
- Know what the CSD Python API is and be familiar with some of its capabilities
- Be able to use the CSD Conformer Generator to generate molecule from SMILES

This workshop will take approximately **30** minutes to be completed. The words in [*Blue Italic*](#) in the text are linked and reported in the [Glossary](#) at the end of this handout.

Pre-required Skills

For this workshop, it is essential that you are familiar with Python. Familiarity with GOLD is desirable but not essential. You may wish to review this free online module [here](https://www.ccdc.cam.ac.uk/community/training-and-learning/csdu-modules/gold-101/) to learn more about GOLD: <https://www.ccdc.cam.ac.uk/community/training-and-learning/csdu-modules/gold-101/>. If you wish to familiarise yourself with the CSD Python API, please try this free online module here: <https://www.ccdc.cam.ac.uk/community/training-and-learning/csdu-modules/python-api-101/>.

Materials

For this tutorial, you will require the following files:

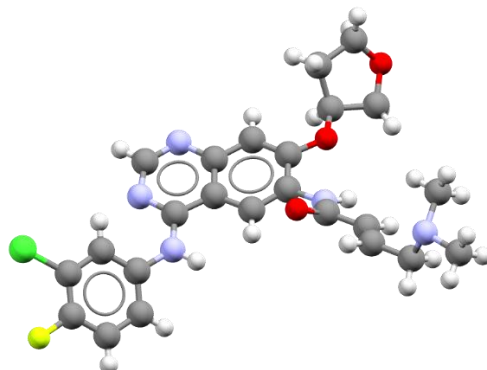
- docking_script_1.py and docking_script_2.py (for reference)
- 7L25.cif and 7L25_prepared.cif – the protein structure files
- gefitinib_example.mol – structure files of the molecules to be docked

All files can be downloaded [here](#).

A text editor is required such as Notepad++, Pycharm or VSCode. The code can also be run through the terminal without editing.

Docking Tyrosine Kinase Inhibitors

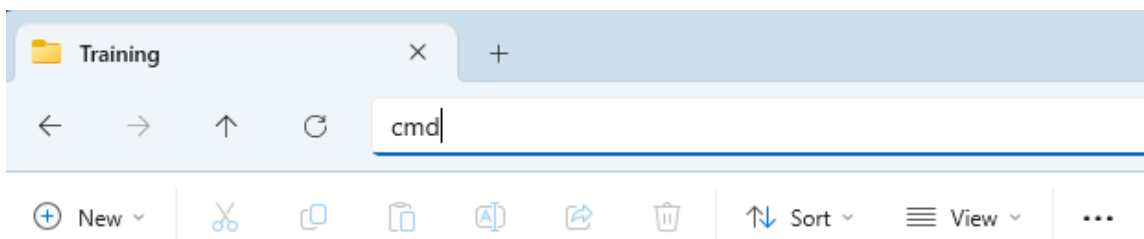
HPK1 (MAP4K1) is a protein kinase which acts as a negative regulator of T-cell receptor (TCR) signaling. It helps dampen immune cell activation, preventing excessive immune responses. Because HPK1 suppresses T-cell activity, it has attracted significant interest in cancer immunotherapy where it is investigated as a way to enhance anti-tumour immune responses by making T cells more active against cancer cells. In this example we will dock three TKIs, Afatinib, Erlotinib and Gefitinib in the crystal structure of Ack1 kinase domain (PDB code 7L25).



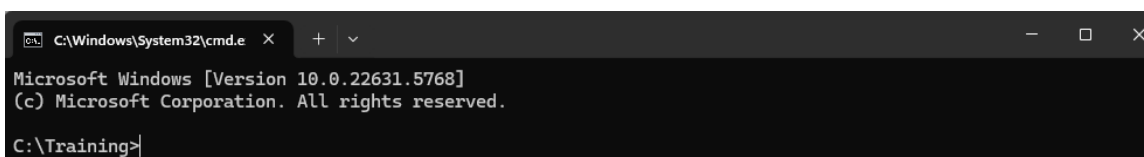
Afatinib, a tyrosine kinase inhibitor (CSD refcode NIKNED).

Activating the CSD Python API Environment

1. Start by creating a folder where you will save your Python files in a place where you have read and write access, for example C:\training\ for Windows, or something equivalent on macOS or Linux. Please make sure 7L25.cif, 7L25_prepared.cif and gefitinib_example.mol from the workshop [materials](#) are also saved in this folder.
2. Open the command prompt from this folder. In Windows you can type 'cmd' in the File Explorer tab and press 'Enter'. In Linux you can right click on the folder and select Open in Terminal. In macOS, right click on the folder, select Services then click New Terminal at Folder.



The command prompt should now appear.



- To run your Python scripts from the command prompt, you will first need to activate your environment. The activation method will vary depending on the platform:

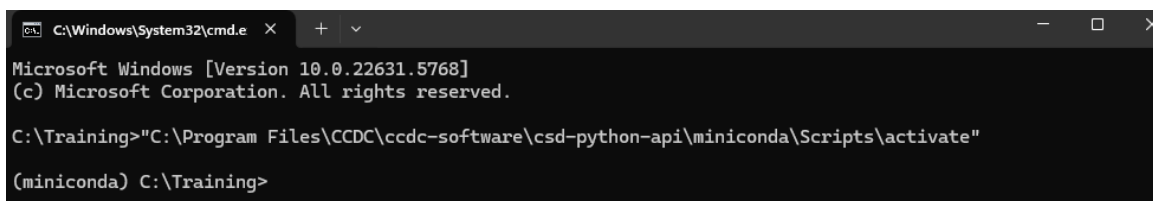
- Windows:** Open a command prompt window and type (including the " marks):
`"C:\path\to\CCDC\ccdc-software\csd-python-api\miniconda\Scripts\activate"`

- MacOS/Linux:** Open a terminal window and change directory to the CSD Python API bin folder:
`cd /Applications/CCDC/ccdc-software/csd-python-api/miniconda/bin`

Then activate the environment with:

```
conda activate
```

Replace "path\to" with the exact path to your ccdc-software install. If the activation is successful, (miniconda) will appear at the beginning of your command prompt:



```
C:\Windows\System32\cmd.e  X  +  v
Microsoft Windows [Version 10.0.22631.5768]
(c) Microsoft Corporation. All rights reserved.

C:\Training>"C:\Program Files\CCDC\ccdc-software\csd-python-api\miniconda\Scripts\activate"
(miniconda) C:\Training>
```

Setting up a Docking Run

- Open a text editor of your choice and begin a new file for your script.
- Let's start by importing the modules we will need. We will be using the `docking` module which contains everything needed for GOLD docking.

```
from ccdc import docking
```

We will now work towards creating a function to perform a single ligand docking. The following section describes the general configuration workflow, **you do not need to add anything to your script until step 3.**

We'll start by creating the docker. This class contains all code for generating instances of GOLD docking.

```
docker = docking.Docker()
```

When running programs through the API, it's common for a Settings object to be used. This contains all the default settings which can be found in a default configuration file. In the case of docking, the `Docker` itself has one of these `Settings` objects.

```
settings = docker.settings
```

Left with no arguments, the docker settings will be left as the defaults.

Tip: If you have a configuration file you would like to use, you can instead load it like this:

```
settings = docker.settings.from_file("gold.conf")
```

GOLD has a number of example configuration files which you can find in your installation folder. E.g. C:\Users\USERNAME\CCDC\ccdc-software\gold\GOLD\templates (replace USERNAME with your username).

Tip: You can also set up a configuration file in Hermes and save it for reuse later. This is a nice way of setting up a consistent configuration for future docking if you tend to use the same settings such as fitness function or constraints. All you need to do is change the files and the rest of the settings remain.

If you are starting from the default configuration and want to edit in through code, rather than editing directly the gold.conf text of the file, you can do as follows.

To select the pre-prepared protein file:

```
settings.add_protein_file("7L25_prepared.cif")
```

To select the ligand file:

```
settings.add_ligand_file("gefitinib.mol")
```

To select the output directory:

```
settings.output_directory = "results"
```

To select the fitness function:

```
settings.fitness_function = "plp"
```

You also need to set a binding site, based on a particular residue. In this case, it's known to be ALA149. We'll loop over all residues, finding the one with the correct identifier. Then, you can save that binding site to the settings, along with a radius of 8.0 Angstrom.

```
ALA149 = [r for r in settings.proteins[0].residues if r.identifier == 'A:ALA149'][0]
```

```
settings.binding_site = settings.BindingSiteFromResidue(settings.proteins[0], ALA149, 8.0)
```

To run the docking simulation we use `docker.dock()`, and we can save the results to an argument `results` to be inspected later:

```
results = docker.dock()
```

- Now that we have covered minimal code for docking, we will create a function for efficiency. We'll make the ligand and protein files arguments so that they're easy to change. We'll also set the binding site on the known residue. Add the following code to your script. If you copy this from this PDF, please check that the indentation is correct and if in doubt, compare with the example scripts in the [workshop materials](#).

```
def dock_single_ligand(protein: str, ligand: str, results_dir: str =
'results', n_poses: int = 10):
    """
    Dock a single ligand to a protein using GOLD.

    The protein file must be prepared (no waters, hydrogens added)

    :param protein: prepared protein file location.
    :param ligand: ligand file location.
    :param results_dir: directory to save results to.
    :param n_poses: number of poses to generate.
    :return: None. Pose scores are printed as soon as the run finishes.
    """
    docker = docking.Docker()
    settings = docker.settings

    settings.add_protein_file(protein)
    ALA149 = [r for r in settings.proteins[0].residues if r.identifier ==
'A:ALA149']
    settings.binding_site =
settings.BindingSiteFromResidue(settings.proteins[0], ALA149, 8.0)

    settings.add_ligand_file(ligand, n_poses)
    settings.output_directory = results_dir

    results = docker.dock()

    return results
```

- To use the function, add the following code to your script underneath the function. In this we have specified the prepared protein and ligand files.

```
dock_single_ligand("7L25_prepared.cif", "gefitinib_example.mol")
```

- Save your script and run it. If you are running the script from the command line, type

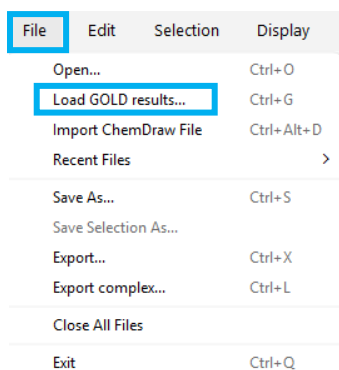
```
python docking_script.py
```

where `docking_script.py` is the name of the Python script you have just saved.

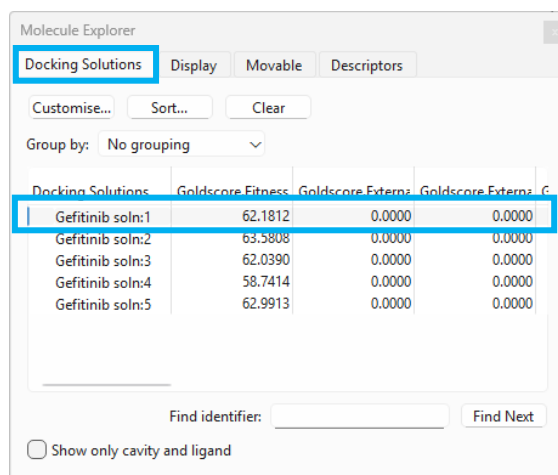
6. Open Hermes from the Start Menu or desktop icon.



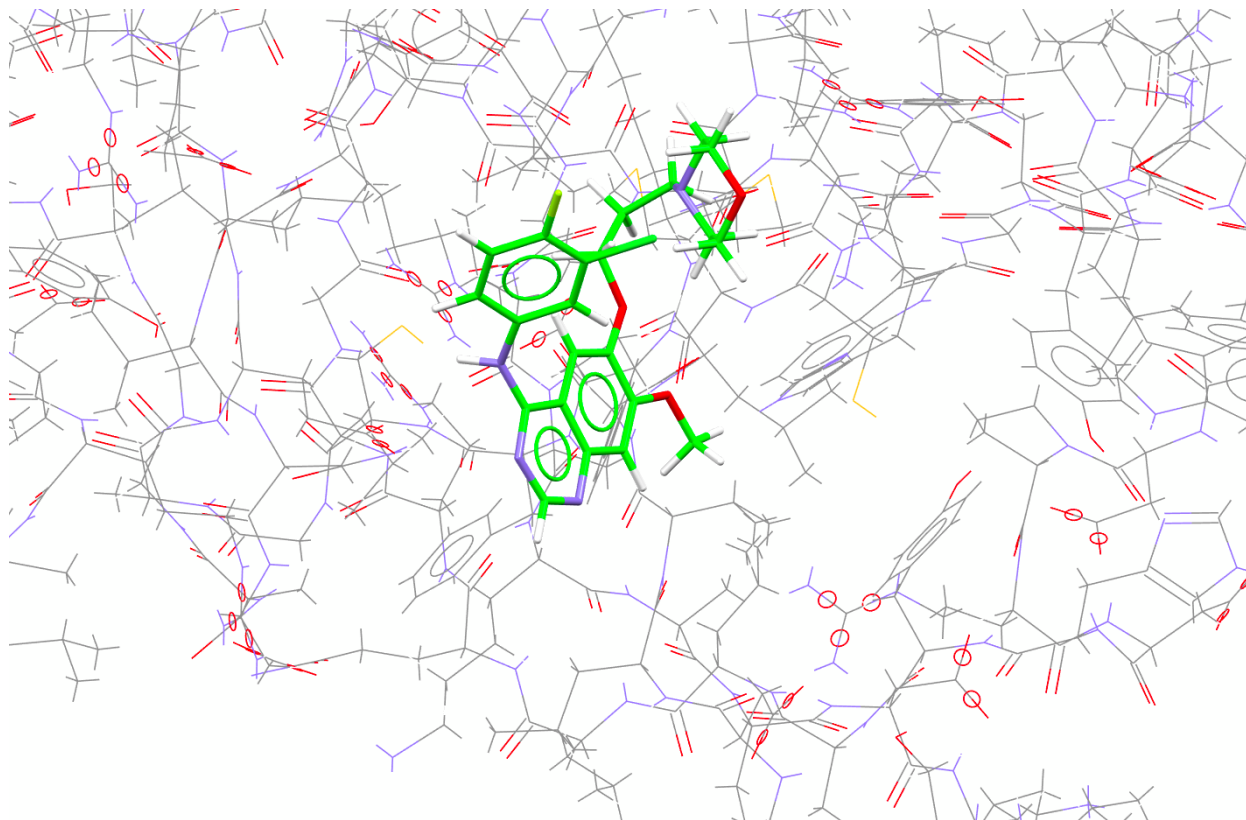
7. From the top menus, go to **File > Load GOLD results**.



8. In the **File Explorer**, navigate to where you have saved your script, select the .conf file (e.g. "api_gold.conf") and click **Open**.
9. In the **Molecule Explorer** window, in the **Docking Solutions** tab, click on a docking solution to load it in the visualiser. You can see an example of a docking solution on the next page.



Please note: GOLD is a stochastic algorithm, and results may differ between runs.



To summarise, we can now supply ligands and proteins to the Python API to perform docking calculations. Settings are editable through a configuration file, or directly in the code. This is all combined neatly into a function to be reused automatically.

Generating Conformers

Now we understand the basics of GOLD docking with the Python API, let's try creating 3D molecules from [SMILES](#) strings to be used in docking. This will use the [conformer generator](#) which builds molecules using the CSD rotamer libraries.

1. First, we need to generate basic molecules from the SMILES strings which uses our io library. It is best practice to add these imports at the start of your script.

```
from ccdc import io
from ccdc import conformer
```

The `io` module is used for loading from the CSD, reading and writing files and molecules. Here we will use it to generate a `Molecule` from a SMILES string. Initially, this will only create the atoms and bonds; no 3D information is yet created. The `conformer` module supplies the conformer generator, which generates conformers of the input molecule.

We will now work towards creating a function to generate conformers of a 3D molecule from a SMILES string. The following section describes the general workflow, **you do not need to add anything to your script until step 2**.

First, we need to generate the molecule from a SMILES string, e.g. for butane it would be:

```
mol = io.Molecule.from_string("CCCC")
```

We can also give the new molecule an identifier with its name:

```
mol.identifier = "butane"
```

SMILES for the ligand molecules will be given later.

To create 3D coordinates, we can set up the conformer generator:

```
conf_gen = conformer.ConformerGenerator()
```

We can change any settings we like, for example, the total number of conformers generated:

```
conf_gen.settings.max_conformers = n_conformers
```

And finally, generate the conformers:

```
conformers = conf_gen.generate(mol)
```

These conformers need to be put into a format that GOLD can use, so any standard molecule file will work. We will make use of the API's molecule readers and writers to read and write files.

```
with io.MoleculeWriter(name) as writer:
    for conf in conformers:
        writer.write(conf.molecule)
```

2. Instead of writing this as a script, also in this case, we can create a function to be reused. We'll supply this with arguments for the SMILES string, the file name output, and how many conformers we want. Add the following to your script.

```
def smiles_to_3d(smiles, name: str, n_conformers):
    mol = io.Molecule.from_string(smiles)
    mol.identifier = name
    conf_gen = conformer.ConformerGenerator()
    conf_gen.settings.max_conformers = n_conformers
    conformers = conf_gen.generate(mol)

    with io.MoleculeWriter(name) as writer:
        for conf in conformers:
            writer.write(conf.molecule)
```

3. We will now use the function to create the gefitinib molecule:

```
smiles_to_3d("C1COCCN1CCC0c2c(OC)cc3ncnc(c3c2)Nc4cc(Cl)c(F)cc4",
"gefitinib_generated.mol", 2)
```

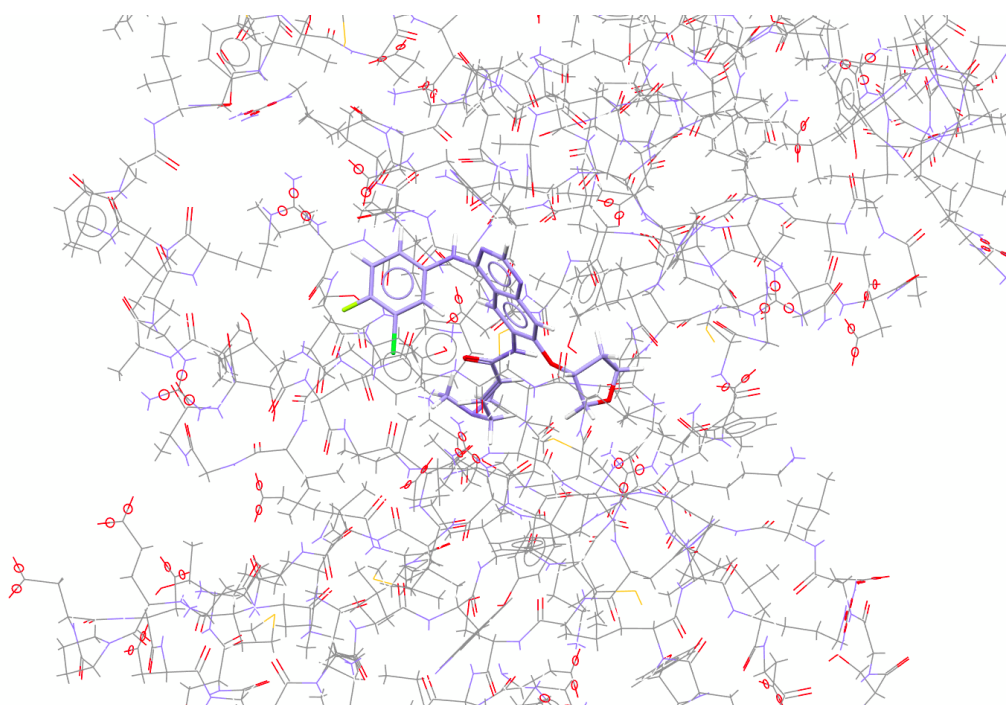
Linking Conformer Generation and Docking

Because we're working with functions, we can easily combine what we've written together to go from 1D SMILES to docked 3D complexes.

1. We can supply the 3D molecule file generated by the conformer generator to the docking code. Before adding the following line, comment out the previous docking code from step 4 of [Setting up a Docking Run](#). Then add at the bottom of your script:

```
dock_single_ligand("7L25_prepared.cif", "gefitinib_generated.mol")
```

2. Save your script and run it, as in step 5 of [Setting up a Docking Run](#). The run might take a few minutes to complete. You should end up with a very similar result to when running straight from a ligand file.
3. In Hermes, from the top menu, go to **File > Close All Files** then open the docking solutions in Hermes as described in steps 7-10 in the [Setting up a Docking Run section](#).



Conclusions

So far, we have built two functions, one to generate conformers of a molecule and the other to individually dock the conformers with the protein. To build on this, we will supply SMILES for multiple molecule variants, then dock them all to the same target.

Generating and Overlaying Multiple Ligands

Before beginning to add more code, it will be best to start with a new file for our script.

1. Open a new text file and copy from your current script copy the following:
 - All `import` instructions
 - The two functions, `dock_single_ligand` and `smiles_to_3d`
2. We will now add the SMILES for the ligands we will use. You can supply the ligands as a dictionary. To your script add the following:

```
example_ligands = {
    "Gefitinib": r"C1COCCN1CCC0c2c(OC)cc3ncnc(c3c2)Nc4cc(Cl)c(F)cc4",
    "Erlotinib": r"COCC0c1cc2c(cc1OCCOC)ncnc2Nc3cccc(c3)C#C",
    "Afatinib": r"CN(C)C\C=C\C(=O)Nc3cc1c(Nc(cc2Cl)ccc2F)ncnc1cc3OC4COCC4",
}
```

One way to visually compare similar ligands is to overlay them using the ligand overlay tool. Before generating the ligands, we will create a function to then overlay them. While not necessary for the docking, this can be useful to understand structural and functional similarities between ligands. The ligand overlay tool identifies functional groups with similar features (hydrogen bond donors and acceptors, ring interactions, etc) and overlays according to their alignments in three dimensions. This automatically invokes conformer generation to find specific, statistically probable conformers which align.

3. Import the molecular descriptors module at the beginning of your script:


```
from ccdc.descriptors import MolecularDescriptors
```
4. Write a small function to overlay molecules using their separate files, as follows:

```
def overlay_separate_files(*mol_files: str):
    overlay = MolecularDescriptors.LigandOverlay('.')

    overlay.add_ligands(
        [io.MoleculeReader(mol_file)[0] for mol_file in mol_files]
    )

    overlay.overlay_ligands()
```

Note that the molecule reader is reading only the first molecule from each ligand's mol file (which may contain multiple conformers).

5. We will now create a `for` loop to generate all three ligand molecules using the `smiles_to_3d` function. To your script add the following code:

```
for name, smiles in example_ligands.items():
    smiles_to_3d(smiles, f'{name}.mol', 1)
```

6. To overlay the ligands, add the following:

```
overlay_separate_files('Gefitinib.mol', 'Erlotinib.mol', 'Afatinib.mol')
```

7. Save your script (with a new name) and run it.

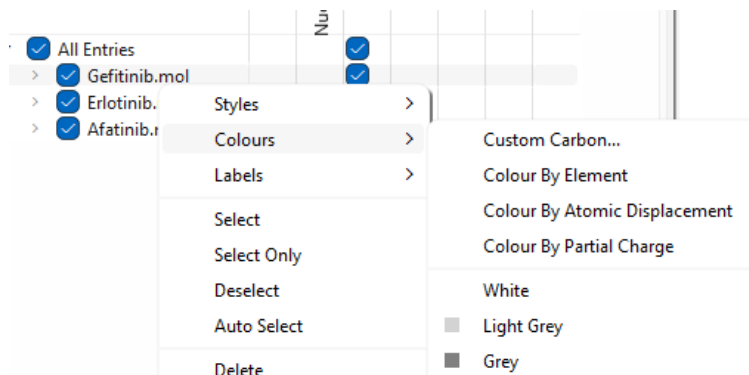
Let's view these overlaid results in Hermes.

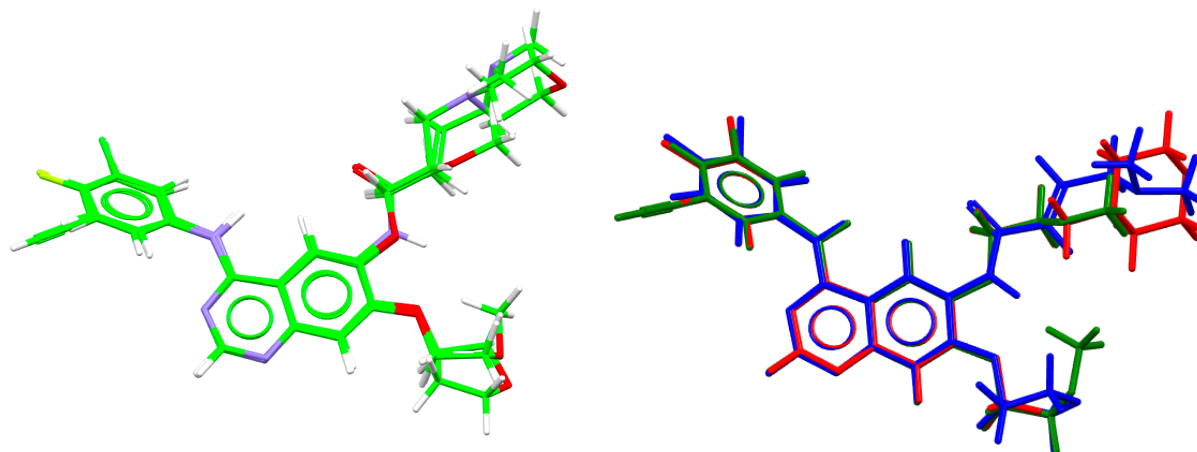
8. If you still have Hermes open from the previous section, go to **File > Close All Files**, otherwise, open Hermes from the Start menu or desktop icon.
9. Click **File > Open** and in the File Explorer, navigate to the folder "job1". This should be in the same directory as your Python script. You should see SDF and MOL2 files like shown below. Choose any of these, e.g. `solution_1.sdf` and click **Open** to display the overlay in the visualiser. An example is shown on the next page.

☐ Name	Date modified	Type	Size
pharmacophores	13/10/2026 14:52	File folder	
solution_1.mol2	13/10/2026 14:52	MOL2 File	11 KB
solution_1.sdf	13/10/2026 14:52	SDF File	6 KB
solution_2.mol2	13/10/2026 14:52	MOL2 File	11 KB
solution_2.sdf	13/10/2026 14:52	SDF File	6 KB

In the image on the next page, Gefitinib, Afatinib and Erlotinib are shown overlaid. We can see how these particular conformers are very similar; features such as the rings overlay almost exactly and key hydrogen bond donors and acceptors are aligned.

Tip: you can change the colour of the molecules by right-clicking on their name in the Molecule Explorer then from the drop-down menu, selecting **Colours** and choosing your preferred colour.





Automatically Docking Multiple Ligands

1. To dock all three ligands, add the following code inside the `for` loop added in [Generating and Overlaying Multiple Ligands](#) step 5 (at the same indentation level as `smiles_to_3d`).

```
results = dock_single_ligand('7L25_prepared.cif', f'{name}.mol',
                             f'{name}_results')
```

2. We can extract information about the ligand poses from the docking results. For example, to see the fitness scores of each pose, add the following inside the `for` loop (at the same indentation level as step 1):

```
for pose in results.ligands:
    print(f' {pose.identifier} {pose.fitness():.2f}')
```

3. The iterations of the `for` loop will overwrite the default `.conf` file. To keep a copy of the `.conf` file specific to each ligand, you can rename it. At the beginning of your script, add:

```
import os
```

4. Within the `for` loop, add the following code (at the same indentation level as in step 2):

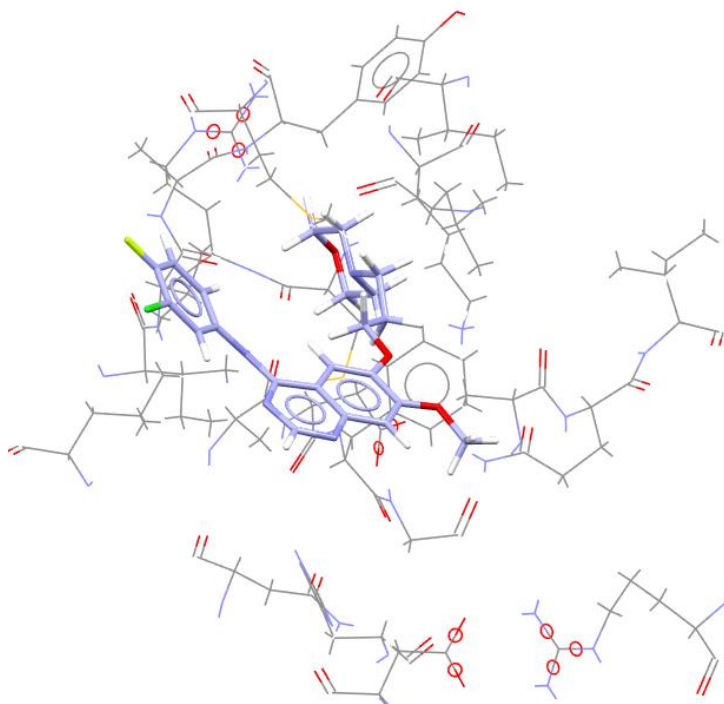
```
os.rename("api_gold.conf", f"api_gold_{name}.conf")
```

Putting this altogether, the final section of your script, which does the conformer generation, docking and ligand overlay should be:

```
for name, smiles in example_ligands.items():
    smiles_to_3d(smiles, f'{name}.mol', 1)
    results = dock_single_ligand('7L25_prepared.cif', f'{name}.mol',
    f'{name}_results')
    for pose in results.ligands:
        print(f' {pose.identifier} {pose.fitness():.2f}')
        os.rename("api_gold.conf", f"api_gold_{name}.conf")
    overlay_separate_files('Gefitinib.mol', 'Erlotinib.mol', 'Afatinib.mol')
```

5. Save your script and run it.
6. Open the docking solutions in Hermes as described in steps 7-10 in the [Setting up a Docking Run section](#).

Shown below is a result for Gefitinib, as part of the entire workflow. This time, we have hidden the broader protein, only showing the ligand and binding site. You can do this by selecting “Show only cavity and ligand” in the Docking Solutions tab of the Molecule Explorer.



Molecule Explorer

Docking Solutions Display Movable Descriptors

Customise... Sort... Clear

Group by: No grouping

Docking Solutions	Chemscore.Fitnes	Chemscore.DECle	Chemscore.DEC
Gefitinib.mol soln:4	32.4327	0.5458	0.54
Gefitinib.mol soln:5	30.4142	1.5330	1.53
Gefitinib.mol soln:2	28.0655	4.1331	4.13
Gefitinib.mol soln:3	27.1287	1.1653	1.16
Gefitinib.mol soln:1	25.5376	2.8469	2.84

Find identifier: Find Next

☒ Show only cavity and ligand

Please note: GOLD is a stochastic algorithm, and results may differ between runs.

Conclusions

By starting from a simple single ligand docking function, we have gradually built up to a SMILES to 3D docking pipeline. The features of the CSD Python API have handled the molecule input, conformer generation, overlaying and docking. Hermes has been used to display the results throughout, and the docked ligands can now be used in further analysis. Even though these drugs were developed for a different target, the cavities are similar. By docking these compounds, we can see they interact with the target's binding site and could be viable drugs for future development.

Extension

Try docking some different molecules and compare the results with those in this exercise. The SMILES strings of some molecules of interest are given below:

Brigatinib: COc1cc(ccc1Nc1ncc(Cl)c(Nc2cccc2P(C)(C)=O)n1)N1CCC(CC1)N1CCN(C)CC1

Osimertinib: C=CC(=O)Nc1cc(Nc2nccc(-c3cn(C)c4cccc34)n2)c(OC)cc1N(C)CCN(C)C

It may also be useful to adjust the settings of the docking, perhaps even adding new constraints or changing the binding site. Specific alterations are defined in the documentation here:

https://downloads.ccdc.cam.ac.uk/documentation/API/descriptive_docs/docking.html

If you would like to a script with a complete workflow, you can download it from GitHub [here](#).

Summary

In this workshop we have built up a Python script that generates 3D molecules from SMILES strings and automatically docks them with a protein, allowing us to efficiently evaluate the different ligand poses. You should now be able to use the CSD Python API to:

- Create molecules from SMILES strings
- Generate conformers of the molecules
- Dock ligands using GOLD.

Additionally, you should be able to overlay multiple ligands to compare their conformations and features.

For further information, you can find the CSD Python API documentation [here](#) and the GOLD user manual [here](#).

Next Steps

If you have found this workshop helpful, you might like to our self-guided workshop on [ligand-based virtual screening](#), which also makes use of the CSD Python API.

Alternatively, take a look at CCDC's open source GitHub [repository](#) to explore other CCDC tools available.

How to Cite this Tool

CSD Python API: R. A. Sykes, N. T. Johnson, C. J. Kingsbury, J. Harter, A. G. P. Maloney, I. J. Sugden, S. C. Ward, S. A. Adcock, P. A. Wood, P. McCabe, A. A. Moldovan, F. Atkinson, I. Giangreco and J. C. Cole, *J. Appl. Cryst.*, **57**, 1235-1250, 2024. [DOI: 10.1107/S1600576724005934](https://doi.org/10.1107/S1600576724005934)

GOLD: G. Jones, P. Willet, R. C. Glen, A. R. Leach and R. Taylor, *J. Mol. Biol.*, **267**, 727-748, 1997. [DOI: 10.1006/jmbi.1996.0897](https://doi.org/10.1006/jmbi.1996.0897)

Feedback

We hope this workshop improved your understanding of automating protein–ligand docking with GOLD using the CSD Python API and you found it useful for your work. As we aim to continuously improve our training materials, we would love to hear your feedback. Follow the [link](#) on the workshop homepage and insert the workshop code, which for this self-guided workshop is PYAPI-004. It will only take 5 minutes and your feedback is anonymous. Thank you!

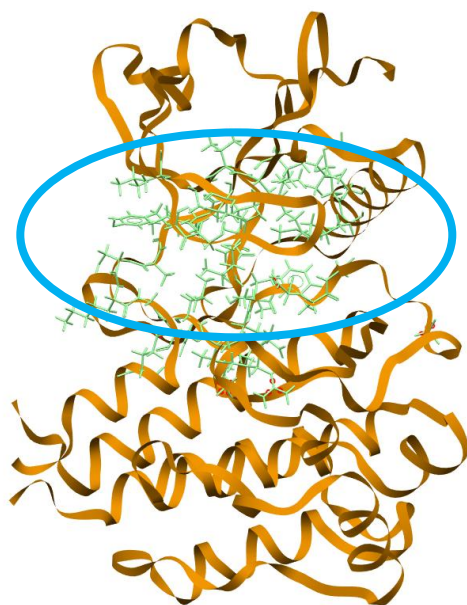
Glossary

API

Application Programming Interface. This is a software intermediary that allows two applications to communicate.

Binding site

A specific region in a molecular entity that is capable of entering into a stabilizing interaction with another molecular entity. An example of such an interaction is that of an active site in an enzyme with its substrate. Typical forms of interaction are by hydrogen bonding, coordination and ion pair formation. Two binding sites in different molecular entities are said to be complementary if their interaction is stabilizing. *Source:* PAC, 1994, 66, 1077. (Glossary of terms used in physical organic chemistry (IUPAC Recommendations 1994)) on page 1089.



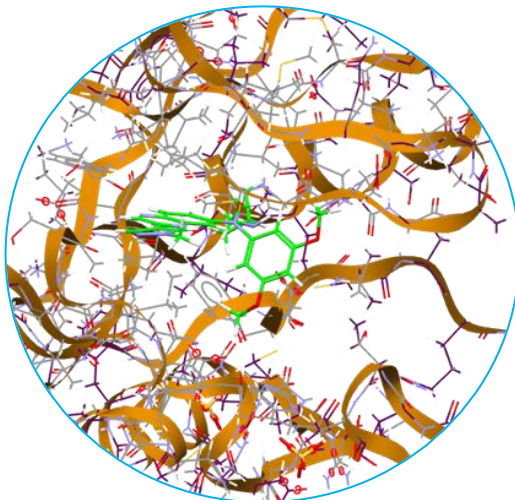
A protein displayed as a ribbon with its binding site represented in capped sticks style in green colour.

Conformer Generator

The CSD Conformer Generator provides the ability to both minimise molecular conformations and generate diverse conformer subsets based on CSD data. The methodology starts from an input 3D molecular structure with all hydrogen atoms present, which is optionally minimised in the first step. Subsequently, conformations are sampled based on CSD-derived rotamer distributions and ring templates. A final diverse set of conformers, clustered according to conformer similarity, is returned. Each conformer is locally optimised in torsion space.

Docking

Docking studies are computational techniques for the exploration of the possible binding modes of a substrate to a given receptor, enzyme or other binding site. *Source:* PAC, 1997, 69, 1137. (Glossary of terms used in computational drug design (IUPAC Recommendations 1997)) on page 1142.



Docking calculates a pose for the ligand and a score for the pose.

GOLD

GOLD (Genetic Optimisation for Ligand Docking) is a genetic algorithm for docking flexible ligands into protein binding sites. GOLD has been extensively tested and has shown excellent performance for pose prediction and good results for virtual screening.

Pose

In molecular docking, a ligand pose is a specific position, orientation, and conformation of a ligand within the binding site of a target molecule (usually a protein).

SMILES

Simplified Molecular Input Line Entry System; a chemical notation for describing the structure of chemical species using short strings.

Virtual screening

Virtual screening Virtual screening is a computational technique used in drug discovery which automatically evaluates a library of small-molecule compounds (which may be very large in number) for their ability to bind to a drug target such as a protein.

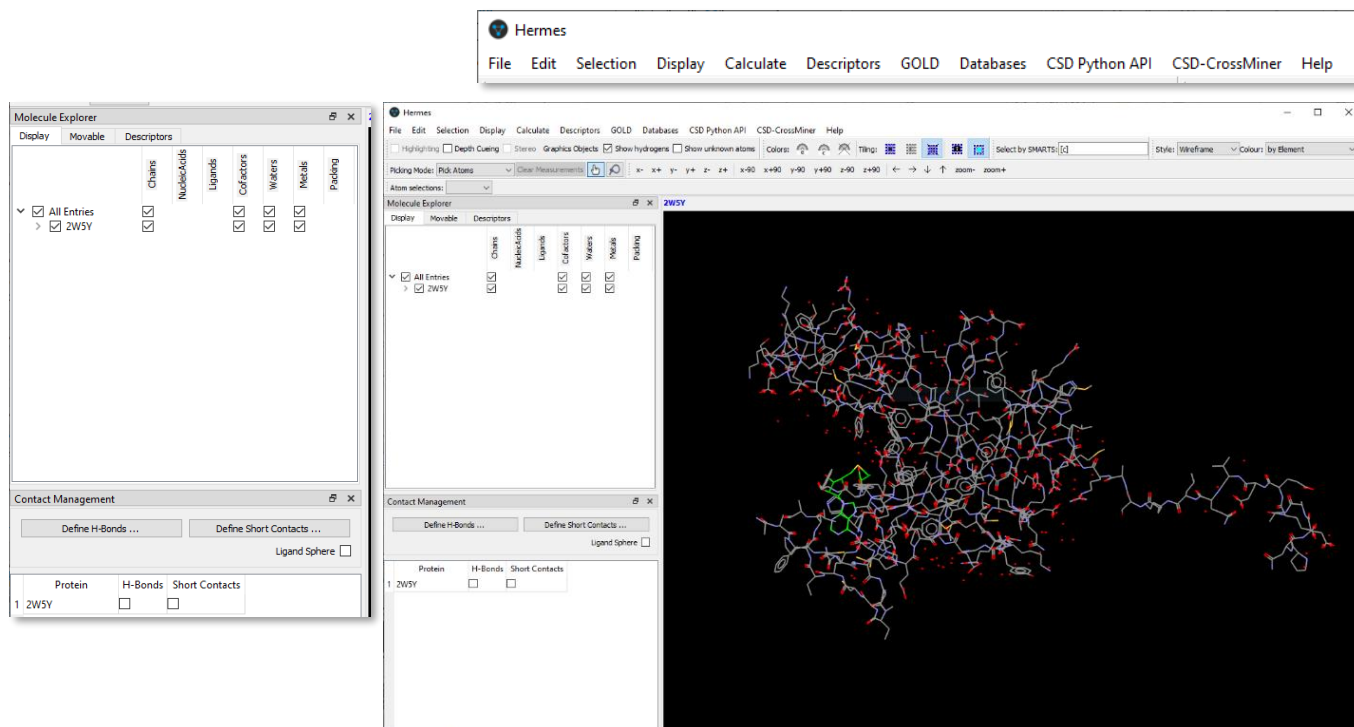
Basics of Hermes Visualisation

Hermes is the CCDC's visualization software to study protein-ligands interaction, and it is the interface for protein docking with GOLD.

In the following we will see some of the basics of navigation and visualization in Hermes that you will find helpful to support your analysis.

In the **Hermes interface** we find:

- **At the top:** list of menus from which we can access visualization and analysis options, and other CSD-Discovery components.
- **On the left-hand side:** the **Molecule Explorer**, where the proteins, ligands, water molecules, and other molecules present in the structures are listed in the *Display* tab. You can expand each entry to see the components, and tick or untick the corresponding boxes to display / not display them. The solutions from a GOLD run will also be displayed in the Molecule Explorer.
- **On the left-hand side,** below the Molecule Explorer: the **Contact Management** box, which allows to display H-bonds and short contacts for selected molecules by ticking the corresponding box.



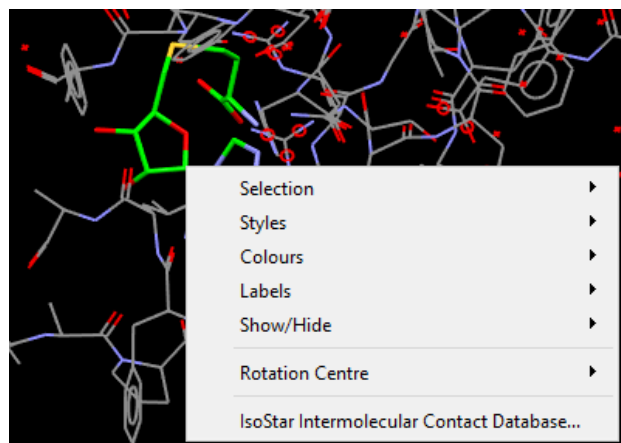
Using the **mouse to enhance visualization**:

- Left mouse button and move – rotate molecules
- Middle Mouse wheel – move molecules up and down
- Right mouse button and move up and down – zoom in and out of molecules
- Shift + Left mouse button and move - rotate in the plane molecules
- Ctrl + Left mouse button and move - translate molecules

Right click:

- On a feature
- Away from a feature

a

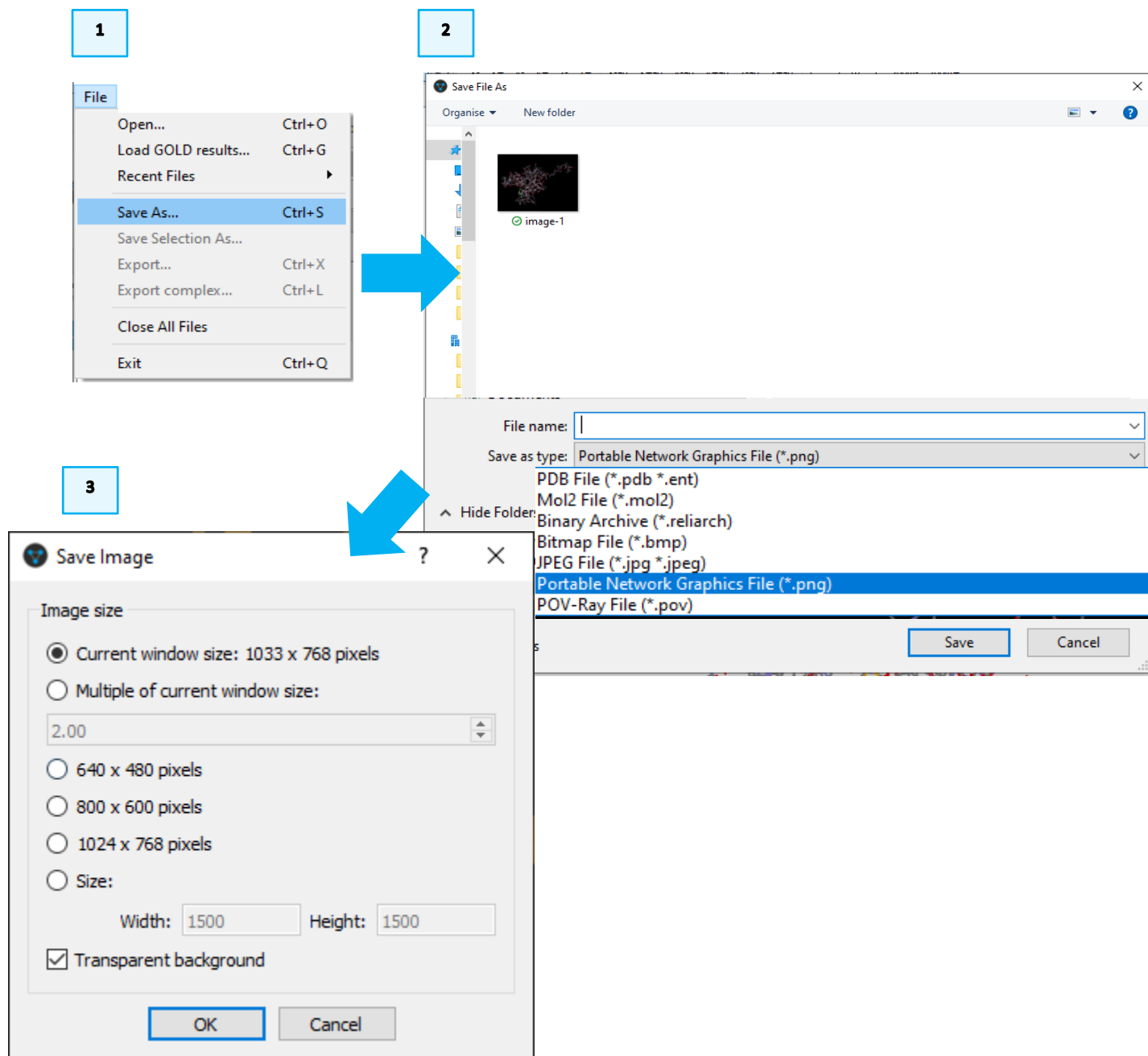


b



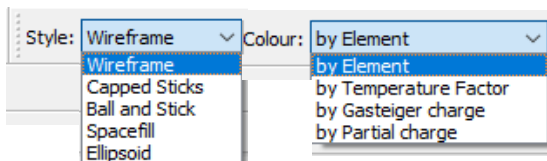
Creating high-resolution images

1. Go to the top menus and click on **File**, then **Save As**. Alternatively you can use the keyboard and do **Ctrl + S**.
2. In the *Save File As* window, select destination folder, file type and choose a name. Then click **Save**.
3. This will bring up the *Save Image* menu. Here you can select the resolution and tick whether you want a transparent background. Click **OK**.

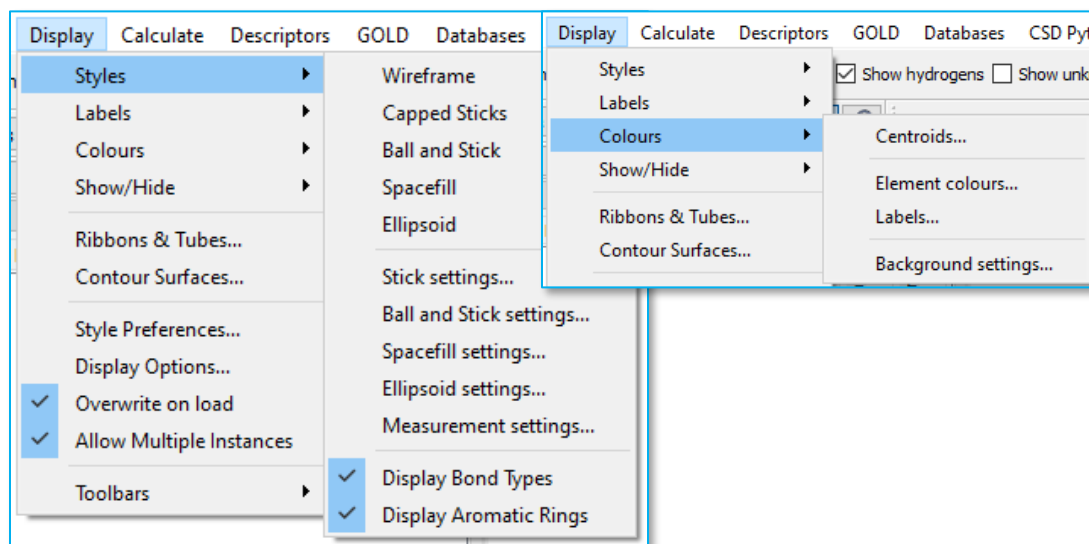


You can access visualization options from the **Display** menu.

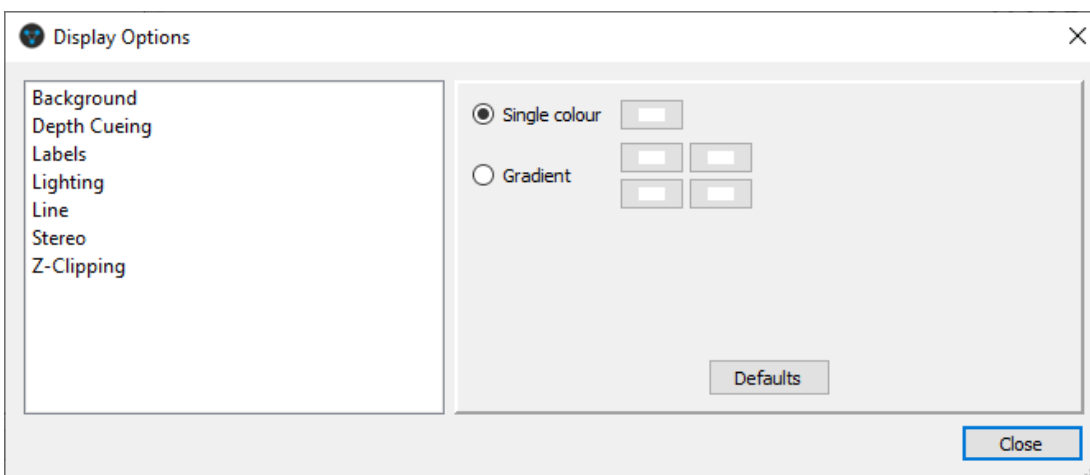
- From **Display > Styles**, you can pick the representation style of atoms and bonds and change settings.
Note: you can also access this feature from the dropdown menu at the top right of the 3D visualizer window.



- From **Display > Colours**, you can edit the colour for elements and object. **Note:** you can change the colouring style for the structure from the dropdown menu at the top right of the 3D visualizer window.



- Clicking on **Display > Display Options** brings up the Display Options menu, where you can edit settings for the 3D visualizer.



- If any of the toolbars at the top of the Hermes interface is not displayed and you wish to have it shown, or vice versa, you can do so from **Display > Toolbars**, where you can ensure to tick the ones you wish to have on.

