Crystal Predictor v2.4.3 Manual

Contents

Introduction

CrystalPredictor is a program designed to predict the crystal energy landscape for a molecule, given its 2D molecular formula. There are many different forms (polymorphs) for any molecule to crystallise into, each with their own set of physical properties; *CrystalPredictor* seeks to rank these possible crystals by their lattice energies, to give a list of sensible crystal structures. *CrystalPredictor* is designed to provide a semi-rigorous model for calculating the lattice energy of crystals such that global optimisation techniques can be applied to locate all relevant minima with an efficient use of computational resources. Structure candidates can be further refined using *CrystalOptimizer* as part of a more general Crystal Structure Prediction workflow. The I/O of *CrystalPredictor* is summarised in Figure 1.

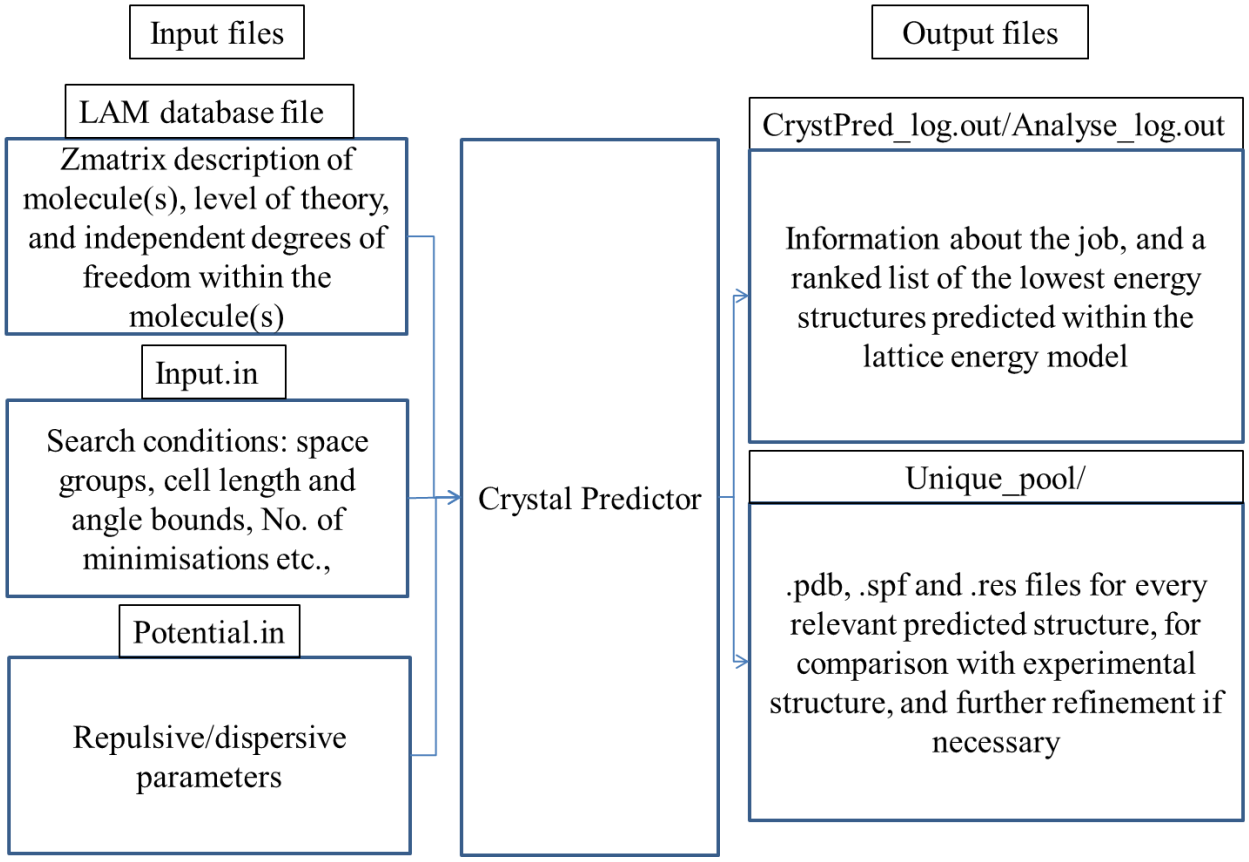


Figure 1, I/O summary

Input files used in *CrystalPredictor*, as shown in Figure 1, are used to specify the system of interest as well as define optimisation specifications. A basic guide to setting up these files is given in the following section.

The overall *CrystalPredictor* algorithm has two major stages in its workflow of first generating LAMS (stage 0) to calculate the intramolecular energy landscape followed by a production run (stage 1) to sample and then minimise points on the lattice energy landscape. The outputs given by *CrystalPredictor* provide information on the results of the optimisation, as well as files describing crystal structures at each minima.

Nomenclature: words in **bold** define files or directories while words in *italic* define executables or utilities.

Guide to CrystalPredictor Workflow

Stage 0: Generate LAMS

Three input files (**input.in**, **lam_intra** and **potential.in**) are required to generate Local Approximate Models (LAMS). In **input.in** each molecular type in the asymmetric unit cell is specified in addition to the selection and ranges of internal degrees of freedom. The file name **lam_intra** is also specified here and can be changed (note for co-crystals etc. additional **lam_intra** files will be required to define each structure and must be named separately). The space groups to be searched in, along with other optimisation parameters are also specified in **input.in** as shown in the example file. The optimisation parameters are divided into two categories namely SIMULATION and DETAILS. In the first category the cut-off distances as well as pressure and hydrogen distances are specified. In the second category the minimum and maximum values of unit cell's angles and lengths are defined. Also, the maximum values of density (Kg/m^3), intramolecular, intermolecular energy (kJ/mol) and minimisations steps are determined, respectively. The last line of **input.in (**polym_region) specifies the region in which local minimum are collected. The value of this line refers to the energy difference (in kJ/mole) between the global minimum and the highest energy polymorph under consideration.

lam_intra contains the level of theory and the ranges for degrees of freedom in which LAMS are generated. For ease of use, the range (finish-start) covered by the LAMS should be an integer multiple of the interval. The range and interval specified in **lam_intra** will determine the co-ordinates of each LAM, while the range in **input.in** will determine the regions of these LAMS that will be searched. **lam_intra** also contains information on atom types, atom number, point charges and z-matrix as shown in the example file. Specified internal degrees of freedom should also be moved to the end of the **lam_intra** file in the order that they are defined at the top of the file such that *CrystalPredictor* can correctly identify them. The values of the point charges and crystal structure given in **lam_intra** should describe the structure and point charges of a molecule in the gas phase confirmation as calculated using software such as *GAUSSIAN*.

The **potential.in** file contains information on empirical repulsive/dispersive interactions for each different atomic ~~molecular~~ type. The chosen force field and associated parameters can be input as shown in the example input file.

Once all files are prepared, LAMS can be generated by running the program *LAM_GENERATOR* from the folder containing these input files. *LAM_GENERTOR* will then prompt several questions, if you would like to modify the files generated by *LAM_GENERATOR* before running then input 'y' when asked 'do you want to set up the directories initially?', modify where required and then run *LAM_GENERATOR* again, this time replying 'n' to the same question. Once LAMs are generated, run *LAM_GENERATOR* one more time to analyse the output and create the file **new_lam_intra**.

If zero internal degrees of freedom are specified (i.e., a "rigid" search), LAMS do not need to be generated, and stage 0 should be omitted. If flexible degrees of freedom are being considered the

new_lam_intra file from stage 0 must be concatenated with the original **lam_intra** file. To append the **new_lam_intra** file at the bottom of **lam_intra** execute:

$ cat new_lam_intra >> lam_intra

The new file can be named differently to preserve the original **lam_intra** file, but this name will have to be included in **input.in**. If stage 0 was omitted then **lam_intra** should contain the point charges and gas phase confirmation as shown in the example input file. These can be calculated in standard commercial software such as *GAUSSIAN*.

In addition, several other commands can be included at the end of the **input.in** file, such as DOING_ANALYSE and DOING_CLUSTER X. These commands will be described in the following sections.(explain CASCADE and NON_UNIFORM)

Stage 1: Production Run

CrystalPredictor requires at least 3 processors: 1 bookkeeper, 1 Generator, and the remaining processors being workers. Tests show that the code scales well, up to at least 96 cores; see Figure 2. To begin the production run submit the **runCrystPred.csh** script:

$ qsub runCrystPred.csh

The script can be submitted with the files **input.in** and **potential.in** as described in stage 0 in the same directory, along with **lam_intra.** Being a parallel code, the user can specify the number of processors to be used during the search. Following the completion of a production run further minimisations can be carried out by re-submitting **runCrystPred.csh** with the file **restart.in** (produced from the previous batch of minimisations) in the working directory. This will continue the program from the set of Sobol points that the previous batch finished at. If resubmitting due to failure, **current_restart.in** contains the set of Sobol points reached in each space group after the latest set of 100,000 minimisatons. **current_restart.in** can be copied to **restart.in** to avoid reminimising the same Sobol points.

The term DOING_ANALYSE can be included at the end of the **input.in** to automate stage 1.1 described below. In addition, the term DOING_CLUSTER X (where "X" is the location of the COMPACK file used to cluster structures) can be included along with DOING_ANALYSE to automate stage 1.2. If carrying out batch runs, the DOING_ANALYSE and DOING_CLUSTER keywords should only be included in the final run so to avoid wasting computational resources.

Stage 1.1: Analyse

Once all minimisations are complete, the results can be analysed by submitting **runAnalyse.csh** with the input files **input.in**, **lam_intra** and **potential.in**, as well as **crystals_stable.in** produced by *CrystalPredictor* in the working directory.

$ qsub runAnalyse.csh

This will produce the directory **unique_pool/** containing the structures of each minima ranked in ascending order of energy. Two format types can be found inside unique pool for each minimum, namely the **minimum_X_Y.spf** and **minimum_X_Y.pdb** files, where X is the rank of the structure and Y is the number within the cluster (i.e. if the same structure is found with different unit cells, two files will be produced with different Y).

Stage 1.2: Clustering

After analysing the results, duplicates can be removed from the list of candidates by submitting:

$ qsub runClustering.csh

It is important that **unique_pool/**, **input** and **Analyse_log.out (**or **CrystPred_log.out** if the keyword DOING_ANALYSE was used in the production run) in the working directory. **input** should contain the location of the *compack* executable used to compare crystal structures. Optionally, the second line of **input** may contain, in the following order, tolerances for the energy and density cut-offs for comparing crystal structures, plus a global energy cut-off. The default tolerances for each of these are set as 3.0, 50.0 and 100.0, respectively. In the input file the location of *runfor* executable is also specified alongside with the argument that determines the time (in seconds) for which the **Clustering.csh** script runs. *runfor* executable is necessary in order to terminate (by force) the **Clustering.csh** script. Successful execution of **Clustering.csh** will produce the final list of candidate structures in **unique_pool/clustered/**.

Note that for the correct execution of **Clustering.csh** directories that contain the **##.res** files should exist. The number of these directories must be equal to the number of the minimised structures. In the current version of *CrystalPredictor* the creation of directories that host the **##.res** files, to be clustered, is done automatically. In older versions the creation of directories was achieved manually by executing the script: *reorder_unique_pool.sh*.

Stage 2: Minimise

It is sometimes necessary to minimise a reference structure using the same computational model as implemented in the search to see whether the reference structure has been found. *Minimise* can be used to analyse the ability of *CrystalPredictor's* energy model at predicting known experimental forms of the compound of interest, and also identify any experimental crystal structures found during the production run. *Minimise* can be executed using the same files as required for stage 1 plus the additional file **expcrys.pdb** which contains the user supplied experimental structure for the compound of interest. **expcrys.pdb** is the reference crystal structure in **.pdb** format. (If you have a **.res** file, you can open it using CCDC Mercury and save it as a **.pdb**. *Minimise* can be executed in the front end by the following command:

$ ./Minimise

Minimise will then complete one optimisation with the experimental structure as an initial point. In the case that a rigid molecule is being studied, the gas phase confirmation from **lam_intra** is "pasted" in the crystal using *OptimalPaste* (visit the **README** file in /OptimalPaste/ if unsure how to do this). The output of *Minimise* (located in **Minimisation_log.out**) can then be compared with generated structures from *CrystalPredictor* to determine in what position the experimental structure was ranked. Comparisons can be made with the rank of all possible structures by extracting Utot from **Minimisation_log.out**

Note that special care must be taken that the atom ordering is the same as the zmatrix in the LAM database. (It is essential that the file **expcrys.pdb** matches the same atom labels and ordering as found in **lam_intra**).

Workflow

Stage	Tasks	Input Files	Output files
0: Generate the LAMs	[$ ./LAM_GENERATOR] When prompted, answer yes to "is this the first uniform run" Concatenate **new_lam_intra** with **lam_intra** [$ cat lam_intra new_lam_intra > lam_intra] CASCADE	**input.in** (optional keyword: DOING_CASCADE) **potential.in** **lam_intra**	**new_lam_intra**
0.1 (optional, to produce non uniform LAMS after initial uniform LAMS are generated)	[$ ./LAM_GENERATOR] When prompted, answer no to this "is this the first uniform run" Concatenate **new_lam_intra** with **lam_intra** Drop "new_" in **new_NON_UNIFORM_LAM_RELEVANCE** before production run [$ mv **new_NON_UNIFORM_LAM_RELEVANCE NON_UNIFORM_LAM_RELEVANCE]**	**input.in** (optional keyword: DOING_CASCADE) (required keyword: NON_UNIFORM) **potential.in** **lam_intra**	**new_lam_intra** **new_NON_UNIFORM_LAM_RELEVANCE**
1: Production run	[$ qsub runCrystPred.csh] Production runs can be performed in batches as long as **restart.in** is specified in the working directory. (**restart.in** if continuing from previous runs)	**input.in** (optional keywords: DOING_ANALYSE, DOING_CLUSTER) **potential.in** **lam_intra** (optional files: **NON_UNIFORM_LAM_RELVANCE**)	**CrystPred_log.out**, **crystals.out**, **crystals_stable.out**, **starting_crystals.out**, **global_statistics.out,** **restart.in**, **current_restart.in**, **restart.message**.
1.1: Generate the polymorphic landscape	[$ qsub runAnalyse.csh] Ranking minima in ascending order of energy	**input.in** **potential.in** **lam_intra** **crystals_stable.out** (optional files: **NON_UNIFORM_LAM_RELVANCE**)	**Analyse_log.out** and **unique_pool/** containing **.pdb**, **.spf** and **.res** files
1.2: Cluster duplicate structures	[$ qsub runCluster.csh] Clusters duplicate structures	**unique_pool/** **Analyse_log.out** (or **CrystPred_log.out** if DOING_ANALYSE keyword was used) **input**	**Clustering_log.out**, **unique_pool/clustered** containing **.res** files and **CrystalPredictor_energy**
2: Identifying experimental form in polymorphic landscape	[$ ./Minimise] Compares experimental structure with the structures generated from CrystalPredictor	**input.in** **potential.in** **lam_intra** **expcrys.pdb** crystals_stable.out	**Minimisation_log.out**, **minimise_initial_strucutre(.pdb/.spf)** and **minimise_final_strucutre (.pdb/.spf)**

Input Files

input.in

```
********************************************************************************

MOLECULAR TYPES 1       ! add extra "TYPE" sections for each new molecule

-------------------------------------------------------------------------------------------------
TYPE MOL_XXVI
1                       !Number of this type in asymmetric unit cell
62                      !Number of atoms in the molecule
lam_intra               !Name of LAM database file
7                       !N flexible degrees of freedom (<= those in mol file)
dih9   -190     -130    !N lines in the format: degree name - lower bound - upper bound
dih10  -195     -165
dih11   20     80
dih21   -125    -95
dih31   20      80
dih32   -195    -165
dih33    -370     -310
-------------------------------------------------------------------------------------------------
SPACE_GROUPS

P1     P-1     P21     P21/C   P21212  P212121 PNA21   PCA21   PBCA    PBCN
C2/C    CC      C2      PC      CM      P21/M   C2/M    P2/C    C2221  PMN21
CMC21   ABA2    FDD2    IBA2    PNNA    PCCN    PBCM    PNNM    PMMN    PNMA
CMCM    CMCA    FDDD    IBAM    P41     P43     I-4     P4/N    P42/N  I4/M
I41/A   P41212  P43212  P-421C  I-42D   P31     P32     R3      P-3     R-3
P3121   P3221   R3C     R-3C    P61     P63     P63/M   P213    PA-3    P2221 PBA2

END_SPACE_GROUPS
-------------------------------------------------------------------------------------------------
SIMULATION
15.00d0                 !rcut                   in A
12.0d0                  !rcut_quin_spl          in A
12.0d0                  !rcut_elec              in A
1.d-10                  !Ewald_accur_minim      relative accuracy in Ewald summation in minimisation
1.0E-5                  !pressure in Pa
no                      !stand_hyd_dist         =yes standardise hydrogen distances, =no leave unaltered (neutron)
no                      !short_hyd_dis          =yes foreshorten H positions by 0.1A (after standardisation)

SEARCH
50.d0 50.d0 50.d0       !ANG_MIN_SRCH       in ^o, minimum cell angle
130.d0 130.d0 130.d0    !ANG_MAX_SRCH       in ^o, maximum cell angle
3.d0 3.d0 3.d0          !LEN_MIN_SRCH       in A, minimum cell length
40.d0 40.d0 40.d0       !LEN_MAX_SRCH       in A, maximum cell length
300.d0 900.d0           !DENSITY_SRCH       in kg/m3, minimum cell density
50.0d0                  !U_INTER_SRCH       in kJ/mol, maximum intermolecular energy
40.d0                   !U_INTRA_SRCH       in kJ/mol, maximum intramolecular energy
0.4d0                   !W_SRCH             minimum deformation parameter
1000000                 !NO_MINS_MAX        maximum number of minimisations
50.d0                   !polym_region       in kJ/mol region of polymorphism, structures outside rejected

Uintra_cap 2 -2478.04020017
********************************************************************************
```

The values for optimisations in this example make for good defaults. The Space Groups are searched according to their ratios in the 2016 CCDC distribution; for example, P-1 and P21/c are searched more frequently than others. To search the groups uniformly, put “uni” after SPACE_GROUPS, and to define your own ratios put “def” after SPACE_GROUPS, and list the groups and their ratios on separate lines.

Optional Parameters in input.in

Uintra_cap

Including the "Uintra_cap X" or “Uintra_cap X Y” keyword at the end of your **input.in** file allows you to control the levelling out of the intramolecular energy surface.

X=0

Uintra=0 at the lowest energy LAM, and Uintra can be a negative number. There is no control of how low intramolecular energy can go.

X=1

Intramolecular energy can’t be less than 0 (the energy of the lowest energy LAM.). This is the default setting

X=2

By far the most preferable, where Y follows, and Y is the gas phase minimum energy, meaning intramolecular energy can't be lower than the gas phase minimum. This change was necessary as when LAMs are evaluated at highly strained points (for instance, if two benzene rings are pointed at each other), then the gradients for rotating away from the strained conformation are huge, and within the space of one grid increment, can disappear to unrealistically low energy. An alternative way of avoiding this situation would be to not evaluate LAMS at highly strained points; grids could be setup to miss torsions at 180 degrees.

REJECT_LIST

Provide output file for each slave "slave_XXX_rejection_list.out" that outputs data on the reasons for rejected starting points, which might guide the user in making the run more efficient, e.g. if there’s loads of unsound_density, maybe consider using a density range.

TRACK_MIN

.pdb files, and accompanying energy data are output at every stage of minimisation.

VERIFY_GRAD

Enable "Verify" E04Uff keyword for the experimental minimisation[1].

DOING_ANALYSE

Analyse is performed as part of CrystPred.

DOING_CLUSTER

Clustering is performed as part of CrystalPredictor, clustering is specified as DOING_CLUSTER X where “X” is the location of the compack executable.

NON_UNIFORM

Enables the selection of Non Uniform LAMs this will require the **NON_UNIFORM_LAM_RELEVANCE** file as described below.

NON_UNIFORM_LAM_RELEVANCE

Non Uniform, or adaptive, LAMs can be automatically generated using **LAM_GENERATOR**, once a uniform set of LAMS has already been generated. This is explained in more detail in a recent paper[2].

NON_UNIFORM_LAM_RELEVANCE has the following format:

00001 00649 00650 00651 00652 00653 00688 00xx yyy etc

00002 00649 00650 00651 00653 00654 00655 00656 00657 00658 00698 000xx yyy etc

Where the first number refers to the number of the uniform LAM, and the remaining numbers on the line refer to the number of the non-uniform LAMs that are close enough to that point to be worth considering.

The program "LAM_GENERATOR" automates making this, which responds to user input. For example asking for cutoff, where "cutoff" is the highest acceptable difference in energy predicted by two LAMs to not necessitate a new LAM. NON_UNIFORM. This is currently only available for Z=1 runs.

lam_intra

****

Intramolecular energy/gradients/hessian/charges for:

MOLECULE NAME

Generated at level of theory:

Level_of_theory/Basis_sets

Across No_of_dimensions dimensional grid: No_of_LAMs

start interval finish

tor1 grid_lower_bound grid_inc grid_higher_bound

tor2 …

From starting Z-matrix:

!atom type, atom_num, point charge, the remainder is the zmatrix in traditional form

C1 1 -0.130092

C1 2 -0.155270 1 cc2

C1 3 -0.111365 2 cc3 1 ccc3

C1 4 -0.122888 3 bond4 2 ang4 1 dih4

…

cc2 1.3856

cc3 1.3812

ccc3 119.7184

cc4 1.3903

ccc4 120.9058

dih4 0.7519

…

****

potential.in

****

POTENTIAL MODEL

----------intermolecular soft forces ----------

Name of atom |Name of atom |Soft potential type

A (in kj/mol) |\rho (in A) |C (in kJ*A^6/mol)

\epsilon (in kj/mol)|\sigma (in A) |

----------|----------|----------|

C1 C1 exp-6

369746.1595 0.277778 2439.820883

H1 H1 exp-6

11971.10223 0.267380 136.4011887

H2 H2 exp-6

2263.310395 0.214592 21.49877589

N1 N1 exp-6

254530.1798 0.264550 1378.406323

O1 O1 exp-6

230065.9193 0.252525 1123.59921

Cl1 Cl1 exp-6

924678.8535 0.284900 7740.5164

****

Output Files

crystals.out

This is a file that contains information about all the crystal structures that were generated and minimised (even failed runs, unstable minima, structures that exceed density and cell specifications etc.) on separate lines.

The format is:

Space group Number, info, number of points generated, 1, number of asymmetric molecules(nmolasm), maximum number of minimisations to attempt (hard coded as 6), hist_nattmin(1:6), molecule type, Utot, Uintra, Uvdw, Uelec, Ureal, Urec, Umol_correction, Usurface, Upre, density, volume, time_to_complete, a, b, c, α, β γ, molecule_center_of_mass (3 values for x, y and z), φ, θ, ψ, flexible degrees_of_freedom.

All energy units are in kJ/mol, density in kg/m^3 and angles in radians. *Info* is the value of IFAIL for the final minimisation, given in the E04UFF manual. *Hist_nattmin*(1:6) is the IFAIL results of the attempted minimisations (6 values). *Molecule type* is positive for the identity, and negative for the inverted molecule. *A, b, c, α, β γ,* are the unit cell lengths and angles, and the final values describe the molecule(s); center of mass, orientation (euler angles φ, θ, ψ), and the values for any flexible degrees of freedom, for each molecule in turn.

crystals_stable.out

This is a file that displays all the successsful and stable minimisations. This file will be used for subsequent clustering. Each line is a separate minimisation with the following format:

Minimisation_number, SpaceGroup, nmolasm, moltype, Utot, UnitCellVolume, *a*, *b*, *c*, *α*, *β*, *γ*, nmolasm*molecule_center_of_mass (3 values for x, y and z), φ, θ, ψ, flexible degrees_of_freedom.

Where *a*, *b*, *c*, *α, β, γ,* are the lattice lengths and angles respectively, then follows the fractional coordinates (molecule_center_of_mass(x,y,z)), orientation, and values for the flexible degrees of freedom for each of the molecule's in the asymmetric unit in turn.

In **crystals.out** and **crystals_stable.out** energy is calculated in kJ/mol, density in kg/m^3, volume in Å^3 and angles in radians.

In previous versions:

NB: before restarting rename the **crystals.out** and **crystals_stable.out** files as they will be overwritten with the new results.

You can then write all the **crystal**(**_stable**)**.out** files into one main file using:

[$ cat crystals.out1 crystals.out2 crystals.out3 > crystals.out]

Version 2.3:

A new feature of V2.3 is that the proceeding **crystals_stable.out** is read in, and outputted at the end of the run, sorted by energy, so renaming it is no longer necessary.

CrystPred_log.out

CrystalPredictor log file used mainly for troubleshooting and timings.
Contains:

- Input data
- Search and Minimisation data
- Cluster data
- Space group propensity

starting_crystals.out

A new feature of CrystalPredictor_v2.3 is that the starting positions of calculations are retained in the file "starting_crystals.out", with a similar format to crystals.out. This can be used to identify the shape of the lattice energy surface, and potentially estimate solid-solid transition reaction coordinates.

global_statistics.out

Optimisation and search statistics as function of the number of the minimisations. It prints a statement whenever the global minimum is updated along with the energy, unit cell volume space group and unit cell dimensions.

restart.in, current_restart.in and **restart.message**

If the search was not complete, it is possible to resubmit CrystalPredictor from the point where it finished using the **restart.in** file. If **restart.in** is in the same directory as CrystalPredictor is submitted from, then the sobol sequence will be continued from the last point recorded in **restart.in**. To begin a new CrystalPredictor run from the start of the sobol sequence remove the file **restart.in**.

current_restart.in records the position in the sobol sequence every 100,000 sobol points. To restart from this point mv **current_restart.in** to **restart.in**.

restart.message tells the user if **restart.in** was used to initialise current CrystalPredictor production run

Analyse_log.out

The Analyse log file is mainly used for troubleshooting.
Contains:

- Input data
- Space group propensities
- Global Optimisation Statistics
- Cluster data

Clustering_log.out

Minimisation_log.out

Log output which has the energy and density of the minimised structure which you can try to match with a structure in the search.

minimise_final_structure.pdb and **minimise_final_structure.spf** - crystal structure files

minimise_initial_structure.pdb and **minimise_initial_structure.spf** – crystal structure of starting point: use to check how well OptimalPaste fits the confirmation in lam_intra to the experimental crystal.

Utility Programs

Scan

executed using:

./scan

or:

./scan tor1 tor2 inc

- Where tor1 and tor2 are integers relating to the torsions you want to scan across, in the order they are given in the input file. if these don't exist, then the default is 1 and 2. "inc" is the increment of the grid, default is 2.0º.
- all other torsions are set as the midpoint between high and low bounds.
- starting values of torsions are printed at the start of the output, for reassurance, then the grid representing all Uintra values for the selected torsions, in 2 degree increments, in the format:

space space tor1_label tor1_val_lbound tor1_val_lbound+inc... tor1_val_hbound

tor2_label tor2_val_lbound Uintra(tor1_val_lbound,tor2_val_lbound) Uintra(tor1_val_lbound,tor2_val_lbound+inc) ... Uintra(tor1_val_lbound,tor2_val_hbound)

tor2_label tor2_val_lbound+inc Uintra(tor1_val_lbound+2,tor2_val_lbound) Uintra(tor1_val_lbound+2,tor2_val_lbound+inc) ... Uintra(tor1_val_lbound+2,tor2_val_hbound)

..

tor2_label tor2_val_hbound

For Example:

space space dih22	-125	-122	-120	-118	-116	-114	-112	-110	-108 ....
dih35	212	13.98293	12.55832	11.29654	10.19761	9.26151	8.48825	7.87783	7.43024 ...
dih35	214	13.97115	12.54235	11.27639	10.17327	9.23299	8.45555	7.84094	7.38918 ...
dih35	216	13.97299	12.54000	11.26986	10.16256	9.21809	8.43646	7.81768	7.36173 ...
dih35	218	13.98844	12.55127	11.27694	10.16546	9.21681	8.43100	7.80803	7.34789 ...

dih35 220 14.01750 12.57615 11.29764 10.18197 9.22914 8.43915 7.81199 7.34767 ...

.....

....

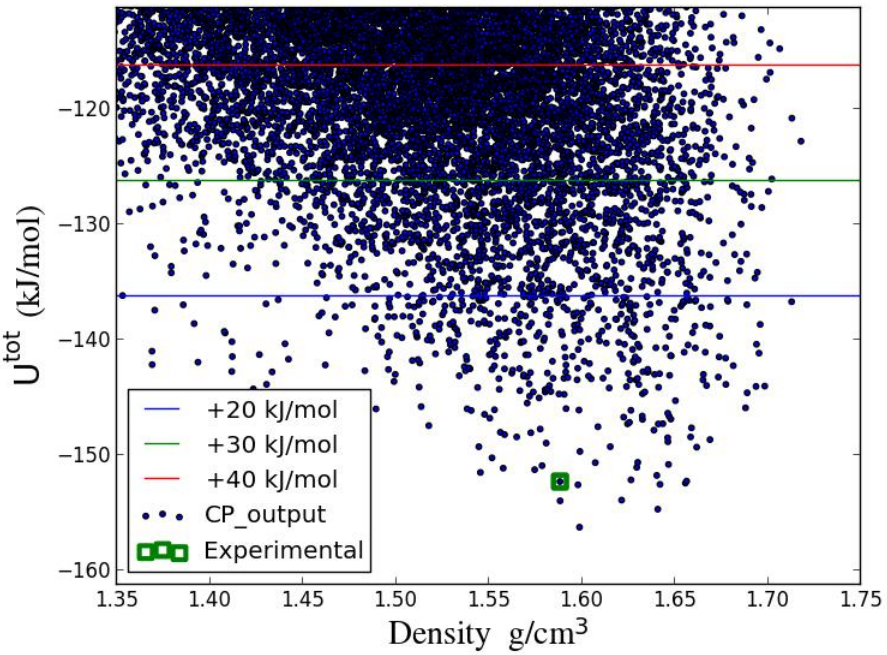
"space space " is included in order to line up the axes when pasted into excel. The label for tor1 (dih22 in this example) must be moved in order to make a surface plot, but is worth retaining in order to make labelling axes easier.

This code is intended to show the user the real intramolecular energy grid a molecule with the associated input files will experience in a CP2 minimisation run. LAMs are intended to only be accurate in the "local" environment, too coarse a grid of LAMs can lead to severe mismatches in energy as the molecule traverses the surface, which can lead to IFAIL=6 errors from the NAG E04UFF minimiser. At highly curved LAM points, highly negative energies can be predicted at medium distances from the LAM; these are currently handled by "if Uintra<0 then Uintra=0", but non-uniform LAM points that accurately describe the surface at these points would be more robust, and wouldn't risk excluding those areas.

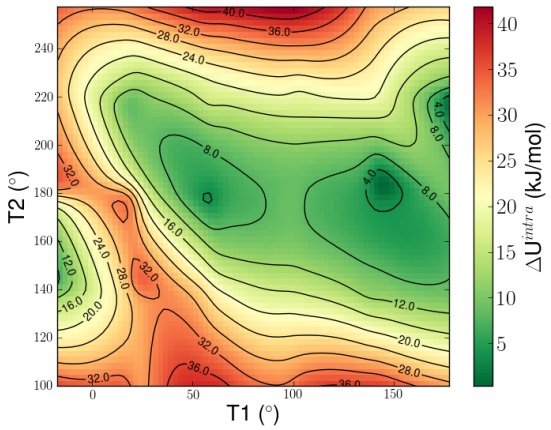
Python utilities

Proportion_of_failure.py: after a production run, analyses crystals.out to see the types of failures exhibited, indicating if search ranges need be broader for instance

Plot.py: plots the CrystalPredictor output from Analyse_log.out (if Analyse_log.out isn't in the directory then CrystPred_log.out is used). Optionally, add a file "Experimental" with the values of density and energy for any experimental forms. Example:



Scan_plotter.py: plots the output of the scan utility. Example:



CP_to_CO.py: reformats LAMs from CrsytalPredictor format to CrystalOptimizer format.

Scaling

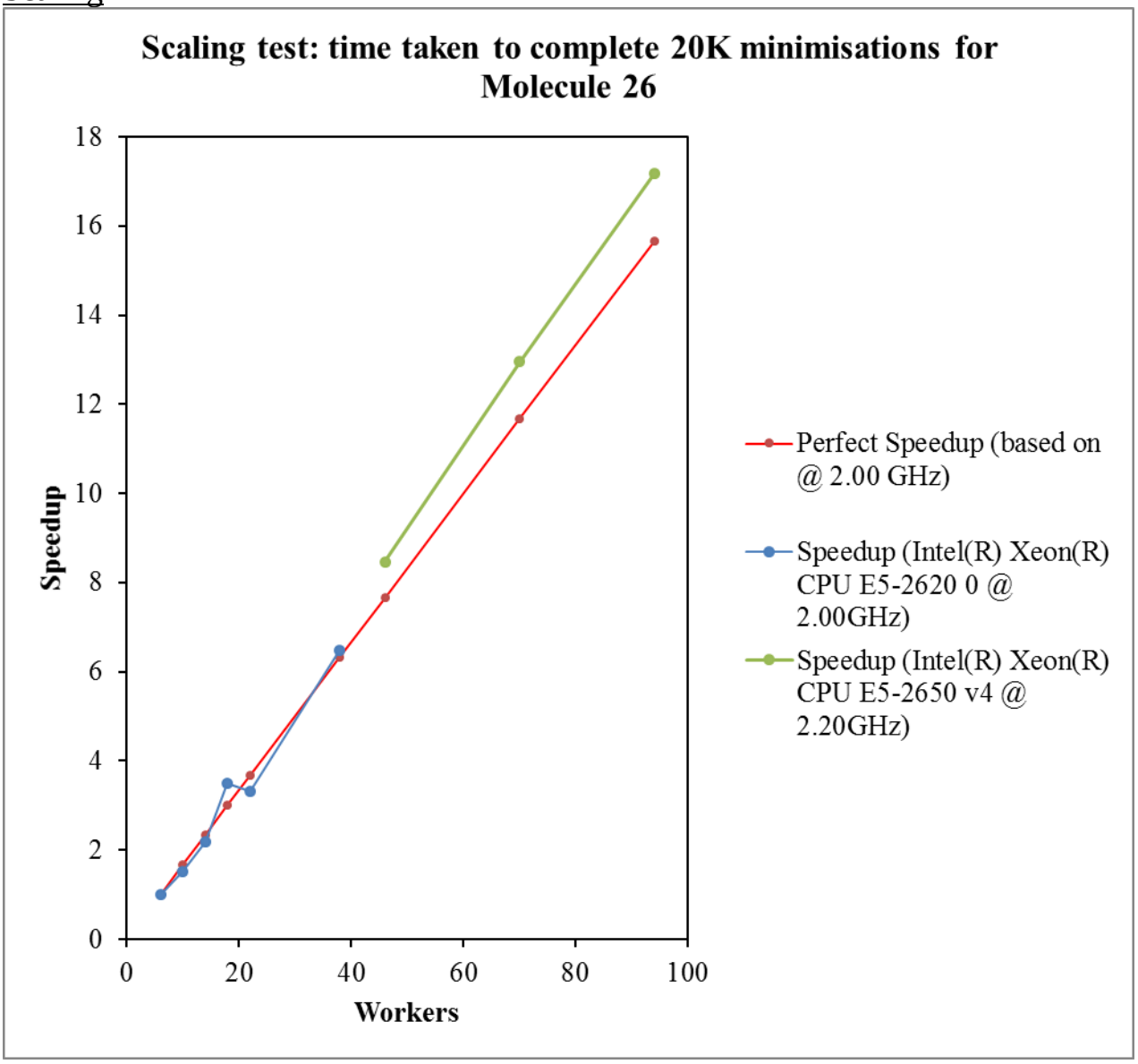


Figure 2, Scaling test.

References

Further reading on theory as follows: CrystalPredictor v1[3, 4], v2[5] and v2.3[2]. Review[6]

[1] https://www.nag.co.uk/numeric/fl/nagdoc_fl25/pdf/e04/e04uff.pdf.

[2] I. Sugden, C. S. Adjiman, and C. C. Pantelides, Acta Crystallogr B **72**, 864 (2016).

[3] P. G. Karamertzanis and C. C. Pantelides, Mol Phys **105**, 273 (2007).

[4] P. G. Karamertzanis and C. C. Pantelides, J Comput Chem **26**, 304 (2005).

[5] M. Habgood, I. J. Sugden, A. V. Kazantsev, C. S. Adjiman, and C. C. Pantelides, J Chem Theory Comput **11**, 1957 (2015).

[6] C. C. Pantelides, C. S. Adjiman, and A. V. Kazantsev, Top Curr Chem **345**, 25 (2014).

Appendix

NonNAG version

Regarding the nonNAG version of the code:

- Routine for constrained local minimizations. I have used Klaus Schittkowski's routine NLPQLP. Professor Schittkowski hence needs to be informed if you are giving out the code to new users, using his routine to make money, etc.
- Use of Numerical Algorithms Group (NAG) libraries. Version 1 made free use of NAG libraries. As these were cost-prohibitive for many potential users, I have created a version which dispenses with them. However, the NAG constrained local minimization routine E04UFF has proven to be faster than NLPQLP. Hence, I have created an alternative version that calls E04UFF instead (but have not replaced the other NAG routines), for users that have access to NAG libraries.

Numerical Recipes routines: This code uses modified versions of the NR routines lubksb, ludcmp, and sort2.

Installation folder

This should contain:

- This Manual
- The CrystalPredictor binaries: CrystPred, Analyse, Minimise, LAM_Generator, Clustering and scan.
- A folder “Utility_Programs/” containing the python scripts: proportion_of_failure.py, plot.py, scan_plotter.py, and CP_to_CO.py, as well as the runCrystPred.csh and runAnalyse.csh runscripts

- A folder “Examples” containing the folders benzoic_acid, BMS and ROY. Each contains rigid_lam_intra for rigid searches with the gas phase conformation and building LAM databases, and flexible_lam_intra for flexible searches after the LAM database has been built. It will also include the excprys.pdb, and a folder called output_files/ containing the output of the CrystPred and Minimise runs and plotty.py.

Software dependencies

The code has been compiled with the intel64 MPI libraries (2018) and the NAG Version 26 libraries, which are required, whilst the version of glibc is 2.17. Please get in contact if there are environment issues, for instance if OpenMPI libraries are required instead. ldd CrystPred on cx1:

linux-vdso.so.1

libstdc++.so.6

libmpifort.so.12

libmpi.so.12

libdl.so.2

librt.so.1

libpthread.so.0

libm.so.6

libc.so.6

libgcc_s.so.1

/lib64/ld-linux-x86-64.so.2