Constructing the CMA Simulation Step by Step

D. J. Goossens*

April 8, 2015

Introduction

This document was begun Wed, Mar 25, 2015 11:35:20 AM to describe the process of setting up a Monte Carlo model of Monoclinic 9-Chloro-10-methylanthracene, and forms part of the deposited material accompanying the publication 'A Process for Modelling Diffuse Scattering from Disordered Molecular Crystals, Illustrated by Application to Monoclinic 9-Chloro-10-methylanthracene' by Darren Goossens. To be used in conjunction with [1] and [2], and with the paper in *Advances in Condensed Matter Physics* to which this deposited material is attached.

This process was undertaken running the `ZMC` suite of code under cygwin (`http://cygwin.com`) on Windows. The suite can also be used on Linux (its native environment) and Mac OS X, and is available as source code, mostly compiling with g95 (`www.g95.org`).

1 CIF files

As a first step, I constructed the .cif files of the +1 and -1 CMA molecules, where ± 1 are the two states the molecules can be in, Cl or CH_3 on the -9- and -10- disordered sites.

The files were derived from versions in the Cambridge Structural Database, and modified to describe the two possible configurations, one with CH_3 on the -9- atomic position in CMA and one with Cl on the -9- atomic position.

These `.cif` files are:
`with_CH3_1.cif`
`with_CH3_2.cif`

The construction was done by hand, editing the downloaded files in a text editor. The main changes to the as-downloaded file(s) were to:

- Put a dummy atom at the centre of the central ring, to act as an origin (not really necessary) and as a single-atom representation of the whole molecule for the calculation of occupancy contact vectors (see below).

*darren.goossens@gmail.com

- Copy the file and in the copy swap the CH_3 and Cl groups on one molecule to make the other configuration.

2 Fill the unit cell

The CIFs were in turn (one by one), imported into Mercury (`http://www.ccdc.cam.ac.uk/Solutions/CSDSystem/Pages/Mercury.aspx`). I selected Calculate and Packing/Slicing and packed out the unit cell. In this case $Z = 4$. I then saved each resulting structure to a separate mol2 file:

`with_CH3_1.mol2`
`with_CH3_2.mol2`

3 Z-matrices

Used zmat_maker to construct z-matrices and (external) z-matrix coordinates for each molecule type [1]. Running zmat_maker outputted a z-matrix (`.zmat` file) and the quaternions and origin atom translations (`.qxyz` file) needed to place the molecule into a unit cell. In this case, these files were:

```
with_CH3_1.qxyz
with_CH3_1.zmat
with_CH3_2.qxyz
with_CH3_2.zmat
```

Note that command line commands were like ('$' is the command prompt):

```
$ zmat_maker.exe with_CH3_1.mol2
```

Note also that

```
$ zmat_maker.exe --help
```

may be useful. Asking for help like this is worth a try with most of the commands used here.

4 Set up initial Short-Range Order (or Not)

I began with a random occupancy structure with the correct global average but no correlation structure. To do this I used make_random_occ to set up an initial random occupancy structure ('occupancy structure' refers to the arrangement of the type 1 and type 2 molecules, which in this case are really just flips of each other). This occupancy structure was taken from [3-6]. Reference [6] was particularly useful (see its figures 4 and 6). Since the orientations and positions generated by zmat_maker were the same for both species (which made sense, given that the two CIF files were made by exchanging the CH_3 and Cl and leaving all other atoms fixed, rather than by flipping the whole molecule over),

it was possible to get the ratios correct by putting 0.686 of type 1 and 0.314 of type 2 on each site. This ensured that the 'Cl rich' sites were always where the 1s are in figure 4 of [6]. This then meant that when the occupancy MC was done the average molecular site occupancy could be maintained by ensuring that occupancies were swapped only *within* a given molecular site (i.e., the the occupancy of a molecule on the first position in a unit cell was only ever swapped with the occupancy of a molecule in the first position of some other cell, for example).

Here is the make_random_occ session:

```
$ make_random_occ.exe
 How big is the model crystal? (asize bsize csize)
32 32 32
 How many locations in the unit cell?
4
 How many types of zmatrix in the structure?
2
How many instances of zmatrix   1 on location   1
1
How many instances of zmatrix   2 on location   1
1
How many instances of zmatrix   1 on location   2
1
How many instances of zmatrix   2 on location   2
1
How many instances of zmatrix   1 on location   3
1
How many instances of zmatrix   2 on location   3
1
How many instances of zmatrix   1 on location   4
1
How many instances of zmatrix   2 on location   4
1
What is the probability of getting instance   1
of zmatrix   1 on location   1
0.686
What is the probability of getting instance   1
of zmatrix   2 on location   1
0.314
What is the probability of getting instance   1
of zmatrix   1 on location   2
0.686
What is the probability of getting instance   1
of zmatrix   2 on location   2
0.314
What is the probability of getting instance   1
```

```
of zmatrix   1 on location   3
0.686
What is the probability of getting instance   1
of zmatrix   2 on location   3
0.314
What is the probability of getting instance   1
of zmatrix   1 on location   4
0.686
What is the probability of getting instance   1
of zmatrix   2 on location   4
0.314
 What is output filename?
CMA_initial_random_occ.txt
 Done.
```

5 Contact vectors (interatomic interactions) Part 1

The next step was to set up an input file for `ZMC` to generate the contact list. This file is shown in `Gen_contacts.inp`

Here are the last few fields:

```
NEWCONTACTS CMA_set9_contacts
VMIN 1.0
VMAX 7.0
CONTATOMS ZMAT 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28
CONTATOMS ZMAT 2 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28
```

These lines tell `ZMC` to look for contacts involving z-matrices 1 and 2 and atoms 2–28, with contact lengths of between 1.0Å and 7.0Å.

To run it:

```
$ ZMC --getcontacts Gen_contacts.inp > Gen_contacts.screen_output
```

Where `Gen_contacts.screen_output` just captures the screen dump of the program and the results are in `CMA_contacts_initial`, which is referred to in the input file, `Gen_contacts.inp`.

6 Sort the contacts

This can be done any number of ways. In this case, it was a two step process. First, the list (`CMA_contacts_initial`) was imported into Excel, then sorted on (see the header line of the contact file for symbols): (1) Length (2) oz (‘**o**rigin **z**-matrix’, basically the type of z-matrix that the contact is coming ‘from’) (3) dz (‘**d**estination **z**-matrix’, basically the type of z-matrix that the contact is going ‘to’) The result was stored in `CMA_contacts_excel_sort`.

7 Continue sorting the contacts

In this case, I wanted to give contacts which were symmetry-related (ie, the same contact) the same type number. For this, I wrote a bit of code called Classify, with the source code in `classify_v2.f`. Compiling was just:

```
$ gfortran.exe -o Classify_v2 classify_v2.f
```

Then running:

```
$ ./Classify_v2.exe
 Classify
 Assumes contacts are already sorted into
 ascending order by length, and within that
 that they have been sorted by oz and within
 that by dz.  Assumes om=dm=1 always !!!!
 This is for CMA mono only!!!
 Feb 2015
 What is increment? 4 for disorder, 1 for no disorder.
4
 Infile?
CMA_contacts_excel_sort
 Outfile?
CMA_contacts_grouped
 Reading CMA_contacts_excel_sort to CMA_contacts_grouped
 How many rows?
49332
       49332  rows.

 Header is...
 ol oz om oat da db dc dl dz dm dat length type
```

And output was to `CMA_contacts_grouped`.

What the program does is fairly crude. It looks at the length of the contact and at what types of molecules it connects. For a given length, the contact could connect molecules 1 to 1, 1 to 2, 2 to 2 or 2 to 1. Note that 2-2 and 1-1 (and 1-2 and 2-1) are not necessarily equivalent, given that molecules can have low symmetry. Hence all four possibilities are given different types. If size-effect is being implemented, it may make sense for the size effects on these four types to be related, and to consider these four types together as a single type with four variations.

Note also that aspects of the CMA problem are hard-coded into `classify_v2.f`, and that while it should act as a template for further studies, it may not be directly applicable. In particular, the `classify_v2.f` program *assumes* the spreadsheet sort noted in the previous section was performed *exactly* as specified!

8 Calculate the force constants on the contact vectors

This was done following the formula developed in [7], which uses an exponential form and the van der Waals radii of interacting atoms to determine a force constant. This work was undertaken primarily by Eric Chan, and the program to calculate the force constants has a name that reflects this.

Compile:

```
$ gfortran.exe -o Eric_Chan_springs eric_trim_disorder.f
```

Run:

```
$ ./Eric_Chan_springs.exe
 Assumes contacts are already classified as per
 classify_v2.f, i.e.:
 ascending order by length, and within that that
 that they have been sorted by oz and within
 that by dz.  Assumes om=dm=1 always !!!!
 This is for CMA mono only!!!
 Feb 2015, Darren Goossens.
 Infile?
CMA_contacts_grouped
 Outfile? (sprcons written here)
CMA_force_const_Chan
 Outfile? (trimmmed contacts written here)
CMA_contacts_grouped_trimmed
 Reading CMA_contacts_grouped, writing to CMA_force_const_Chan
 How many rows?
49332
       49332  rows.

 Header is...
  ol oz om oat da db dc dl dz dm dat length type
```

This then resulted in a trimmed list of contacts and their force constants. A lot of the generated contacts proved to have zero force constants (i.e., be too long to worry about). Compare the length of the two contact lists:

```
$ wc CMA_contacts_grouped_trimmed
  4769  61997 224197 CMA_contacts_grouped_trimmed

$ wc CMA_contacts_grouped
  49333  641329 2516034 CMA_contacts_grouped
```

So before trimming there were 49333 lines in the file (top line is header, so 49332 contacts). Afterwards, only 4766 contacts. Note that trimming was done by working backwards through the file and eliminating all the contacts *after* the

last one with a non-zero spring constant. That meant that contacts shorter than the last non-zero one but which *were* zero remained in the model. A further improvement would be to remove them and then renumber what is left. This would speed things up a bit but is not needed for demonstration purposes. Also, note that if the parameters that determine the force constants are themselves to be refined (see [7]), then the trimming would also need to be repeated each cycle of refinement.

9 Run a model of CMA

The first model ('Model 1')used random occupancies (no chemical SRO — that is, no correlations amongst the molecule flips apart from the fact that the two species are not randomly 50:50 distributed but more like 70:30). It was run, and then its Fourier transform was calculated to look at the diffuse scattering.

An input file to `ZMC` was set up – `Model_1.inp` – which used random occupancies but many MC cycles to give an indication of the thermal diffuse scattering. The results were fed into DZMC, which also reads in an input file as used by DIFFUSE to define the calculation regions. These files are included in this archive and are called `diffuse.monoXYZ` where `XYZ` = `0kl`, `hk0` or `h0l`. Other cuts can be specified. Note also that the `diffuse.monoXYZ` files can be modified to use different scattering factors (absorption edges, neutrons...), to subtract the single body average (Bragg) scattering in a number of ways ('Y' does a calculation based on average atomic positions and atomic displacement parameters, 'e' does an exact calculation using 5% of the model, 'E' does an exact calculation using the whole model. Similar options are available in DISCUS [8].

Scattering factors were taken from
`http://it.iucr.org/Cb/ch6o1v0001/sec6o1o1/?#sec6o1o1o4`
and the file
`diffuse.mono0kl`
is fairly self-documenting.

The simulation was controlled by two things, the input file `Model_1.inp`, and the `ZMC` command line.

Invoke `ZMC` as

```
$ZMC --help2 --help
```

for some help. It is useful to direct the output to a file for easy reference.

```
$ZMC --help2 --help  > ZMC.help
```

The 'Trimmed' list of contacts written to `CMA_force_const_Chan` was pasted into the input file, `Model_1.inp`. Only the section above the word 'Untrimmed' was pasted in, and not including the word 'Trimmed'....

The small batch job (shell script) `Model_1.sh` is a convenient way to run the calculation. If you have multiple cores, you might want to put ampersands

on the ends of the commands that run the DZMC jobs to let them all run at once, for example.

The command was (might be different in a Windows command prompt):

```
$ nohup bash Model_1.sh > Model_1.screen &
```

And this was followed by much waiting.

10 View the Calculated Diffraction

DZMC uses as its engine DIFFUSE, which outputs binary files. The toolbox program bin2gray converts these to `pgm` files, which can be viewed using any number of programs; a very useful tool is ImageJ. The binary file names were given in the files `Model_1_in???` and were:

```
Model_1_hk0
Model_1_0kl
Model_1_h0l
```

Note that

```
$ bin2gray --help
```

may be useful. Then

```
$ bin2gray --hmirror --vmirror --twofold Model_1_0kl Model_1_hk0
```

and

```
$ bin2gray --twofold  Model_1_h0l
```

(the $h0l$ layer does not have mirror planes due to the monoclinic structure). This resulted in three files:

```
Model_1_hk0.pgm
Model_1_0kl.pgm
Model_1_h0l.pgm
```

Which were viewed and compared to the observations.

Files with 'Bragg' in the title do not subtract out the average structure from the diffraction calculation. This leaves the Bragg peaks in, and leaves in any associated high-frequency ripples that result from having these bright features in the pattern. If these calculations are then thresholded and/or scaled to remove the ripples, it can be useful for comparison with observed data to combine the Bragg spots with the calculated diffuse to get an image that looks more like the observations.

11 Occupancy Correlations

To do occupancy correlations, the first thing was to develop a list of interactions for use with the Ising-like potentials. `ZMC` could be used for this, with only the origin (dummy) atom being used in the calculation. The input file was

`Gen_occ_contacts.inp`

so the command was

```
$ZMC --getcontacts Gen_occ_contacts.inp
```

and the results were written to `CMA_occ_contacts_initial`.

This file 'only' has 96 lines in it, and since it was being used to establish the occupancy structure (the array of 1 and 2 giving molecule orientation) the oz, om, dz, dm fields were all irrelevant, and duplicates were culled. It was then sorted and classified quickly using a spreadsheet to give

`CMA_occ_contacts_sorted`

This file described the same interactions as noted in figure 1 of [3], where there the interactions are denoted *a*,*b*,*c*,*d* and are equivalent to 1,2,4,3 (respectively) in the file `CMA_occ_contacts_sorted` Hence, based on the correlations used in [3], these four interactions need to induce occupancy correlations of 0.4, 0.2, 0.05 and 0.0.

The program to do this was called `occupancy_mc_v2.f90` and was compiled thus:

```
gfortran.exe -o Occ_MC_v2 occupancy_mc_v2.f90 rannum.f
```

where `rannum.f` is a random number generation routine borrowed from somewhere or other. The Occupancy simulation can be run interactively, but it was simpler and better for record keeping to put the inputs into a file and redirect the input/output, giving this command:

```
$ ./Occ_MC_v2.exe < MC_occ.inp > MC_occ_Epstein.screen
```

which took the input from `MC_occ.inp`, generated correlations as suggested by [3], then dumped the screen output to `MC_occ_Epstein.screen` and the actual new occupancy structure to `CMA_occ_correlations_Epstein`.

12 The 'final' model (which is not final)

So the next step was to put the new correlation structure into the input file for `ZMC` and run again.

```
$ nohup bash Model_2.sh > Model_2.screen &
```

which produced images in the same way as before. Clearly at this stage a model existed. It could now be modified, extended, tested more thoroughly, and so forth, to gain the actual structural insight that was desired. Once the diffraction pattern of the model crystal agrees with that of the real crystal, the pattern and the model can be explored to find out about the structures present.

References

[1] D. J. Goossens, A. P. Heerdegen, E. J. Chan, and T. R. Welberry. "Monte Carlo Modelling of Diffuse Scattering from Single Crystals: The Program `ZMC`." Metallurgical and Materials Transactions A, **42A**:23-31, 2010. DOI:10.1007/s11661-010-0199-1. [2] `http://rsc.anu.edu.au/~goossens`

[3] J. Epstein, T. R. Welberry, and R. D. G. Jones. "Analysis of the diffuse x-ray scattering from substitutionally disordered molecular crystals: Monoclinic 9-bromo-10-methylanthracene." Acta Crystallographica, **38**:611-618, 1982.

[4] R. D. G. Jones and T. R. Welberry. "Crystals exhibiting disorder - the monoclinic polymorph of 9-chloro-10-methylanthracene." Acta Crystallographica Section B, **37**(5):1125-1126, May 1981, doi:10.1107/S0567740881005232.

[5] R. D. G. Jones and T. R. Welberry. "Crystals exhibiting disorder - the monoclinic polymorph of 9-bromo-10-methylanthracene." Acta Crystallographica Section B, **36**(4):852-857, Apr 1980, doi:10.1107/S0567740880004682.

[6] T. R. Welberry and R. D. G. Jones. Journal of Applied Crystallography, **13**:244-251, 1980.

[7] E. J. Chan, T. R. Welberry, D. J. Goossens, and A. P. Heerdegen. J. Appl. Cryst., **43**:913-915, 2010, doi:10.1107/S0021889810022260

[8] R. B. Neder and Th. Proffen. Diffuse Scattering and Defect Structure Simulations: A cook book using the program DISCUS. OUP, 2008.