VC-(x)PWDF Instruction Manual

Version 2. April 12, 2023

Contents

Quick Start 1

Overview 2

Limitations 2

Instructions 2

Installation of critic2 2

VC-PWDF commands in critic2 3

Running a VC-PWDF comparison 4

Printing out the distorted crystal structure file 4

Plotting the simulated powder diffractograms 4

Comparing a list of structures with a target structure 5

Performing a VC-xPWDF comparison 6

Appendix – Detailed installation instructions for critic2 10

Quick Start

If you have already fully installed critic2 and are looking to compare two crystal structures with the VC-PWDF method, the following outlines the quickest way to do this.

1) Create a directory containing your two crystal structures (eg. xtal1.cif and xtal2.cif)

```
amayo@garrison:~/example$ ls
xtal0.cif  xtal1.cif  xtal2.cif
```

2) Start critic2 within this directory

```
amayo@garrison:~/example$ critic2

critic2:1>
```

3) Enter the command `comparevc xtal1.cif xtal2.cif`

```
critic2:1> comparevc xtal1.cif xtal2.cif
```

4) The VC-PWDF score is the `+ FINAL Diff` value give at the end of the critic2 printout

```
+ FINAL DIFF = 0.000536684

critic2:2>
```

5) If you have additional crystals to compare, you can enter them next, eg:

```
critic2:2> comparevc xtal1.cif xtal0.cif
+ FINAL DIFF = 0.445259147

critic2:3>
```

Overview

The VC-PWDF method is a protocol that compares two crystal structures and yields a numerical value that is related to the similarity of the two structures being compared. The protocol uses the simulated powder diffractograms of the two structures in order to yield a dissimilarity value using a cross-correlation function (ie. a measure of peak overlap, [J. Comput. Chem., 2001, 22, 273]). The value yielded by the method is a number between 0 (identical) and 1 (completely dissimilar). We have called this value the "VC-PWDF score", and a score < 0.05 indicates considerable similarity and a possible match. The protocol is specifically designed to be highly effective for the comparison of crystal structures obtained under different conditions; low/high temperatures, high pressure, or *in silico*-generated by force field/MM or electronic structure theory/DFT computational methods.

The VC-xPWDF method is used to compare experimentally collected powder diffractograms to the simulated powder diffractograms from crystal structures in order to identify the matching crystal structure to the experimental powder diffractogram. The VC-xPWDF method requires that experimental powder diffractogram be indexed, our recommendation for accomplishing this is the *Crysfire2020* program. A VC-xPWDF score < 0.1 implies notable similarity but does not guarantee a match. It is recommended to plot an overlay of the simulated powder diffractogram of the best-matching crystal structure and the experimental data in order to confirm a match. The VC-xPWDF method also provides an optimal starting point for the model structure if one has sufficiently high quality PXRD data to perform Rietveld refinement. Minimal processing of the experimental PXRD data is done, so if there is a substantial baseline (eg. transmission mode collection), or the instrument lacks a Cu Kβ filter, pre-processing to baseline correct and strip extraneous peaks from radiation not matching Cu Kα_1 is highly recommended for viable results.

This document is intended to provide the step-by-step instructions for using the VC-PWDF method to compare two crystal structures. Details on the development, abilities, and applications (identification of target crystal structures in CSP landscapes [CrystEngComm, 2021, 23, 7118], distinguishing the same structure from polymorph structures in the CSD [CrystEngComm, 2022, 24, 8326], matching experimental PXRD to crystal structures [Chem. Sci., 2023]) are provided elsewhere.

Limitations

The VC-(x)PWDF methods currently only simulate/compare powder diffractograms from Cu Kα_1 radiation, cannot be used for disordered structures, and will yield low scores for certain polytype and conformational phase structures.

Instructions

Installation of critic2

The VC-PWDF method is implemented within the critic2 program which is free to use under a GNU/GPL v3 license. Currently, the method is only implemented within the development version of the code, which can only be run on a Linux system. Installation on a Windows OS computer can be achieved through the Windows Subsystem for Linux (WSL). Complete installation will allow calling the program with the command `critic2` from the command line. Detailed installation instructions are provided in Appendix – Detailed installation instructions for critic2

```
amayo@garrison:~/example$ critic2
                _  _     _          ___
               (_) | |   (_)        |__ \
    ___   _ __   _  | |_   _    ___    ) |
   / __| | '__| | | | __| | |  / __|  / /
  | (__  | |    | | | |_  | | | (__  / /_
   \___| |_|    |_|  \__| |_|  \___| |____|

* CRITIC2: analysis of real-space scalar fields in solids and molecules.
  (c) 1996-2019 A. Otero-de-la-Roza, A. Martin-Pendas, V. Lua~na
  Distributed under GNU GPL v.3 (see COPYING for details)
  Bugs, requests, and rants: aoterodelaroza@gmail.com
  Website: https://aoterodelaroza.github.io/critic2/
  If you find this software useful, please cite:
  A. Otero-de-la-Roza et al., Comput. Phys. Commun. 185 (2014) 1007-1018.
  A. Otero-de-la-Roza et al., Comput. Phys. Commun. 180 (2009) 157-166.

+ critic2 (development), version
         host: Linux-5.4.0-96-generic
         date: Wed 15 Feb 2023 18:51:17
 compiled dat: /usr/local/share/critic2
      datadir: /opt/critic2/dat
...was found?: T
       spglib: 1.13.0
        libxc: <unavailable>

CRITIC2--2023/4/11, 10:52:10.301

critic2:1>
```

VC-PWDF commands in critic2

critic2 can be run interactively as a command-line interface, or through input files that contain the commands to be run. To start a critic2 session, enter `critic2` into the command line and hit enter. To use critic2 non-interactively with an input and generate an output file, use the following command style:

```
amayo@garrison:~/example$ critic2 input.cri > output.cro
```

A .cri extension is recommended for critic2 input files, and .cro for critic2 output files. The critic2 command for running the VC-PWDF method is `comparevc`, followed by the two crystal structure files that you want to compare. This can be written into your input.cri file and run non-interactively in order to save the output, containing the details from the process and final VC-PWDF score, in the output.cro file. Example input file:

```
amayo@garrison:~/example$ cat input.cri
comparevc NARSOR01.cif NARSOR02.cif
```

The two crystal structure files, NARSOR01.cif and NARSOR02.cif, must be in the same directory where the command is run (ie. the `example` directory, in this case).

```
amayo@garrison:~/example$ ls
input.cri  NARSOR01.cif  NARSOR02.cif  NARSOR.cif
```

Running a VC-PWDF comparison

If your interest is only the VC-PWDF score, piping the output through a grep command can give you this value:

```
amayo@garrison:~/example$ critic2 input.cri | grep FINAL
+ FINAL DIFF = 0.000536684
```

Sending the output to be printed into a file allows you to review the protocol run, and save the VC-PWDF score in a file to review later if necessary.

Printing out the distorted crystal structure file

If you want to view the overlay of the two crystal structures that yield the VC-PWDF score (as a result of the lattice deformation), add `WRITE` to the end of the command:

```
amayo@garrison:~/example$ cat input.cri
comparevc NARSOR01.cif NARSOR02.cif WRITE
```

After running the input file with critic2, two .res format files will be generated of the two crystal structures that yielded the VC-PWDF score obtained (input_structure_1.res and input_structure_2.res).

```
amayo@garrison:~/example$ ls
input.cri  input_structure_1.res  input_structure_2.res  NARSOR01.cif  NARSOR02.cif  NARSOR.cif
```

These structures can be viewed in a GUI of your choice, compared with other methods, etc.. if desired.

Plotting the simulated powder diffractograms

A simulated powder diffractogram can be generated for any crystal structure with critic2 by loading the crystal structure, then entering the POWDER command (combined in the example pxrd.cri critic2 input file below).

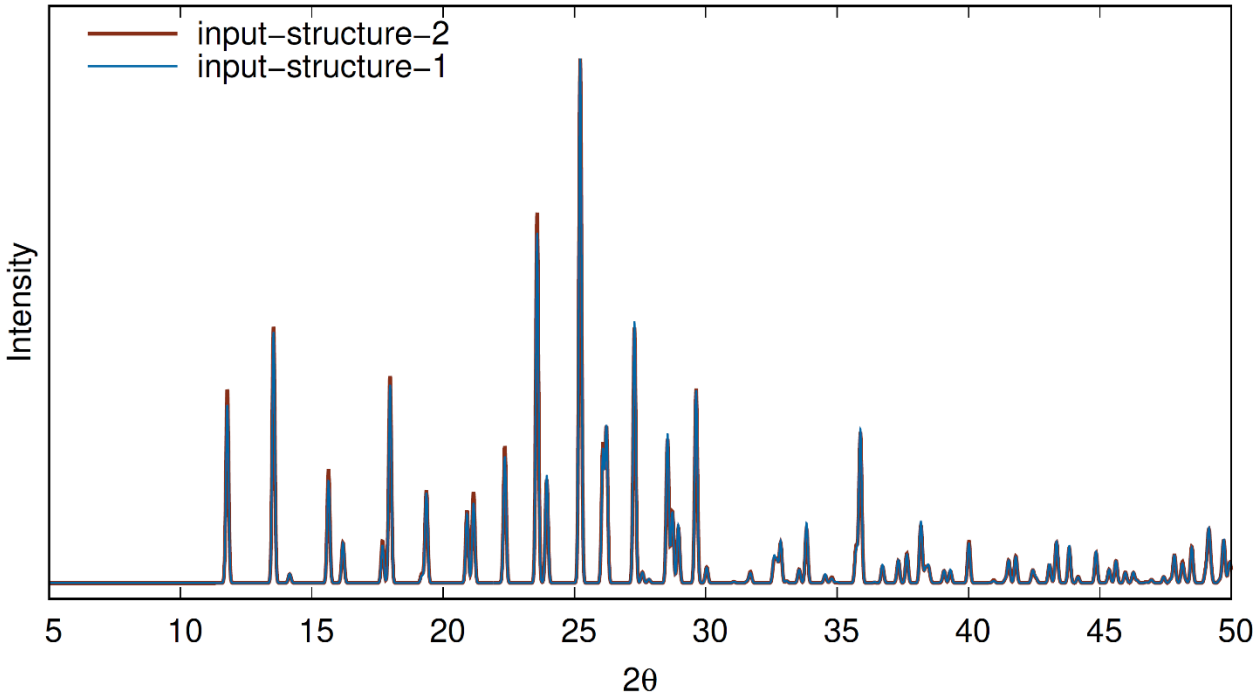
```
amayo@garrison:~/example$ cat pxrd.cri
crystal input_structure_1.res
POWDER
```

Running pxrd.cri with critic2 will print two files, a sample gnuplot command file to plot the data with gnuplot (pxrd_xrd.gnu), and a two-column file of 2θ (°) and intensity (pxrd_xrd.dat).

```
amayo@garrison:~/example$ ls
input.cri             input_structure_2.res  NARSOR02.cif  pxrd.cri      pxrd_xrd.gnu
input_structure_1.res  NARSOR01.cif          NARSOR.cif    pxrd_xrd.dat
```

```
amayo@garrison:~/example$ head pxrd_xrd.dat
#    2*theta     Intensity
   5.0000000       0.0000000
   5.0085000       0.0000000
   5.0170000       0.0000000
   5.0255000       0.0000000
   5.0340000       0.0000000
   5.0425000       0.0000000
   5.0510000       0.0000000
   5.0595000       0.0000000
   5.0680000       0.0000000
```

Use a plotting program of your choice to plot the simulated powder patterns.



More details on the POWDER command in critic2 can be found here. The critic2 output will include a list of Bragg peaks, their positions and relative intensities (max intensity peak = 100).

Comparing a list of structures with a target structure

If you are searching a list of structures to see if any match a target reference structure that you have (eg. a CSP landscape for an experimental structure), one way to do this is to generate all the input files with a for loop. Given the example of the following directory, where PROGST10 is the reference structure, and the remaining structures were generated computationally:

```
amayo@garrison:~/example$ ls *.cif
PROGST10.cif           PROGST_m65_ak49.cif   PROGST_opt_ab101.cif  PROGST_opt_ak4.cif    PROGST_opt_ca95.cif
PROGST_m110_ak100.cif  PROGST_m65_ak70.cif   PROGST_opt_ab104.cif  PROGST_opt_ak50.cif   PROGST_opt_ca9.cif
PROGST_m110_ak36.cif   PROGST_m65_ak98.cif   PROGST_opt_ab108.cif  PROGST_opt_ak9.cif    PROGST_opt_cb25.cif
PROGST_m110_ca2.cif    PROGST_m65_am26.cif   PROGST_opt_ab111.cif  PROGST_opt_al1.cif    PROGST_opt_cb47.cif
PROGST_m110_cb6.cif    PROGST_m65_am2.cif    PROGST_opt_ab112.cif  PROGST_opt_al80.cif   PROGST_opt_cb48.cif
PROGST_m110_fc11.cif   PROGST_m65_am44.cif   PROGST_opt_ab114.cif  PROGST_opt_am2.cif    PROGST_opt_cb92.cif
PROGST_m110_fc18.cif   PROGST_m65_aq46.cif   PROGST_opt_ab116.cif  PROGST_opt_am4.cif    PROGST_opt_cc11.cif
PROGST_m110_fc29.cif   PROGST_m65_ca10.cif   PROGST_opt_ab123.cif  PROGST_opt_am58.cif   PROGST_opt_cc121.cif
PROGST_m110_fc47.cif   PROGST_m65_ca120.cif  PROGST_opt_ab15.cif   PROGST_opt_am5.cif    PROGST_opt_cc42.cif
PROGST_m190_ab122.cif  PROGST_m65_ca14.cif   PROGST_opt_ab35.cif   PROGST_opt_am65.cif   PROGST_opt_cc95.cif
PROGST_m190_ab89.cif   PROGST_m65_dc11.cif   PROGST_opt_ab36.cif   PROGST_opt_am69.cif   PROGST_opt_dc51.cif
PROGST_m190_ak10.cif   PROGST_m65_fa73.cif   PROGST_opt_ab50.cif   PROGST_opt_am6.cif    PROGST_opt_dd2.cif
PROGST_m190_ak12.cif   PROGST_m65_fc88.cif   PROGST_opt_ab56.cif   PROGST_opt_aq104.cif  PROGST_opt_dd32.cif
PROGST_m190_ak1.cif    PROGST_m65_fc90.cif   PROGST_opt_ab60.cif   PROGST_opt_aq29.cif   PROGST_opt_dd55.cif
PROGST_m190_ak5.cif    PROGST_m80_ab115.cif  PROGST_opt_ab78.cif   PROGST_opt_aq38.cif   PROGST_opt_de16.cif
PROGST_m190_am8.cif    PROGST_m80_ab54.cif   PROGST_opt_ab79.cif   PROGST_opt_aq94.cif   PROGST_opt_de56.cif
PROGST_m190_am96.cif   PROGST_m80_ai109.cif  PROGST_opt_ab94.cif   PROGST_opt_au2.cif    PROGST_opt_de93.cif
PROGST_m190_aq60.cif   PROGST_m80_ak119.cif  PROGST_opt_ab96.cif   PROGST_opt_av1.cif    PROGST_opt_fa1.cif
PROGST_m190_ca55.cif   PROGST_m80_ak121.cif  PROGST_opt_ab97.cif   PROGST_opt_av5.cif    PROGST_opt_fa39.cif
PROGST_m190_ca61.cif   PROGST_m80_ak13.cif   PROGST_opt_ai41.cif   PROGST_opt_bd1.cif    PROGST_opt_fa3.cif
PROGST_m190_ca6.cif    PROGST_m80_ak23.cif   PROGST_opt_ai42.cif   PROGST_opt_ca120.cif  PROGST_opt_fa44.cif
PROGST_m190_ca71.cif   PROGST_m80_ak24.cif   PROGST_opt_ai5.cif    PROGST_opt_ca123.cif  PROGST_opt_fa64.cif
PROGST_m190_ca81.cif   PROGST_m80_ak4.cif    PROGST_opt_ai62.cif   PROGST_opt_ca125.cif  PROGST_opt_fa87.cif
PROGST_m190_dd92.cif   PROGST_m80_aq13.cif   PROGST_opt_ai6.cif    PROGST_opt_ca21.cif   PROGST_opt_fc29.cif
PROGST_m190_de6.cif    PROGST_m80_az3.cif    PROGST_opt_ak101.cif  PROGST_opt_ca25.cif   PROGST_opt_fc31.cif
PROGST_m190_fc67.cif   PROGST_m80_ca35.cif   PROGST_opt_ak113.cif  PROGST_opt_ca33.cif   PROGST_opt_fc3.cif
PROGST_m65_ab49.cif    PROGST_m80_ca6.cif    PROGST_opt_ak14.cif   PROGST_opt_ca42.cif   PROGST_opt_fc41.cif
PROGST_m65_ai3.cif     PROGST_m80_ca97.cif   PROGST_opt_ak16.cif   PROGST_opt_ca50.cif   PROGST_opt_fc52.cif
PROGST_m65_ak3.cif     PROGST_m80_fa44.cif   PROGST_opt_ak1.cif    PROGST_opt_ca76.cif   PROGST_opt_fc8.cif
PROGST_m65_ak42.cif    PROGST_m80_fc44.cif   PROGST_opt_ak42.cif   PROGST_opt_ca84.cif   PROGST_opt_fc9.cif
```

the for loop to make all the critic2 input files could look like this:

```
for i in PROGST_*.cif ; do echo "comparevc PROGST10.cif $i" >
${i%.cif}.cri ; done
```

which can be run from the command line. A similar for loop can be written to run all the new .cri files through critic2:

```
for i in *.cri ; do critic2 $i > ${i%.cri}.cro ; done
```

With about 150 structures to compare, this run takes about 5 minutes. If you have a couple thousand structures to compare, running the comparisons in parallel over N processors will reduce the total time required to 1/N. This can be done with parallel. If your CSP list of structures is all contained within one concatenated file, csplit is ideal for creating a unique file for each structure.

To create a table of results, and sort them by lowest VC-PWDF score the following line can be used:

```
for i in *.cro ; do echo -n "${i%.cro}   " ; grep FINAL $i | awk
'{print$5}' ; done | sort -n -k2 > results.txt
```

```
amayo@garrison:~/example$ head results.txt | column -t
PROGST_m65_aq46  0.006910718
PROGST_m65_ak98  0.097593536
PROGST_opt_ca84  0.098190288
PROGST_opt_am6   0.101631649
PROGST_opt_ab96  0.101989574
PROGST_opt_ab36  0.104097150
PROGST_opt_av5   0.114740892
PROGST_opt_am2   0.117597803
PROGST_m80_aq13  0.119424344
PROGST_opt_ab50  0.119726878
```

The results.txt file contains the structure (column 1) and VC-PWDF score (column 2), and the `head` command prints out the first few lines of the file, `column -t` simply formats the output to align the columns of the printed output. This example clearly shows that the PROGST_m65_aq46 structure matches the PROGST10 target structure with a VC-PWDF score of 0.0069.

Performing a VC-xPWDF comparison

As mentioned in the Overview section, you must have indexed unit cell dimensions from your experimental PXRD data in order to utilized the VC-xPWDF method. The PXRD data are to be provided to the program as a .xy file. The input commands are similar to the VC-PWDF commands:

```
amayo@garrison:~/example$ cat PROGST10-pxrd.cri
trick compare PROGST10.cif PROGST-PXRD.xy 10.3741 12.6059 13.8464 90 90.268 90
```

where `comparevc` is changed to `trick compare`, and the crystal structure file comes first, followed by the filename of the PXRD data and the indexed unit cell dimensions (a, b, c, α, β, γ – in order). The critic2 input file is run the same way. The VC-xPWDF value is obtained in the same way:

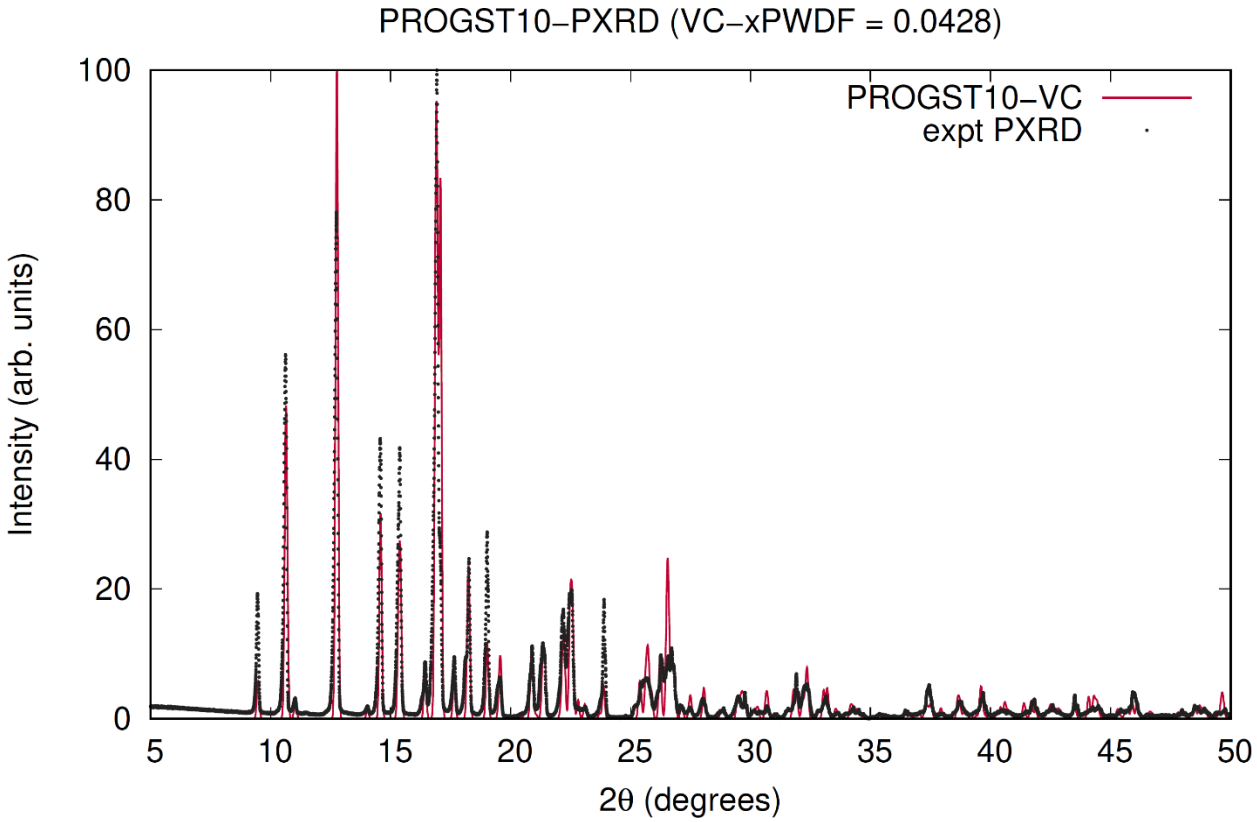
```
amayo@garrison:~/example$ critic2 PROGST10-pxrd.cri | grep FINAL
+ FINAL DIFF = 0.042751228
```

If screening a list of candidate structures, the protocol outlined in the previous section may be useful for rapid and efficient screening.

In order to ensure a matching structure, plotting the overlay of the experimental data and simulated powder diffractogram of the best (lowest VC-xPWDF score) crystal structure is recommended. If you have gnuplot installed, the following file will do this for you. Copy and paste the code (beginning on the following page) into a file named vc-xpwdf-plot.sh and run with the arguments requested, eg:

```
bash vc-xpwdf-plot.sh PROGST10-PXRD PROGST10.cif PROGST-PXRD.xy 10.3741 12.6059
13.8464 90 90.268 90
```

to yield the following image in .pdf format:



```
#!/bin/bash

### ARGS
# $1 - title
# $2 - a crystal structure file
# $3 - an expt PXRD as .xy
# $4 - a
# $5 - b
# $6 - c
# $7 - alpha
# $8 - beta
# $9 - gamma

title=$1
xtal=$2
pxrd=$3
a=$4
b=$5
c=$6
alpha=$7
beta=$8
gamma=$9

# basic processing of the PXRD pattern
minI=$(sort -n -k2 $pxrd | head -n 1 | awk '{print$2}')
maxI=$(sort -nr -k2 $pxrd | head -n 1 | awk '{print$2}')
pxrd_bc=${pxrd%.xy}-bc.xy
awk -v min="$minI" -v max="$maxI" '{print $1, (($2-min)/((max-min)/100))}' $pxrd > $pxrd_bc

# run VC-xPWDF
cat > ${xtal%.*}_vc.cri << EOF
trick compare $xtal $pxrd_bc $a $b $c $alpha $beta $gamma WRITE
crystal ${xtal%.*}_vc_structure_2.res
powder
symm recalc
write ${xtal%.*}_vc.cif
EOF
critic2 ${xtal%.*}_vc.cri > ${xtal%.*}_vc.cro
vcpwdf=$(grep FINAL ${xtal%.*}_vc.cro | awk '{printf "%.4f", $5}')
xtal_vc=${xtal%.*}_vc_xrd.dat
xtal_label=$(echo ${xtal%.*} | sed 's/_/-/g')
rm ${xtal%.*}_vc_structure_2.res
#mv ${xtal%.*}_vc_structure_2.res ${xtal%.*}_vc.res

### write gnuplot instruction file
cat > overlay.gnu << EOF
set term post enhanced color solid "Helvetica" 18
set encoding iso_8859_1
set output 'overlay-${title}-${2%%.*}VC-${3%%.*}.ps'
set size ratio 0.6

set style line 1  pt 7  lc rgb "#222222" lw 0.3 ps 0.3 lt 0.3 #black
set style line 9  pt 4  lc rgb "#BE0032" lw 2 ps 2 lt 1 #red

set title '$title (VC-xPWDF = $vcpwdf)'
set xlabel "2{/Symbol q} (degrees)"
set ylabel "Intensity (arb. units)"
set xrange [5.0000000:50.0000000]
plot "$xtal_vc" u 1:2 w lines ls 9 t "${xtal_label}-VC",\
```

```
     "$pxrd_bc" u 1:2 w points ls 1 t "expt PXRD",

!ps2pdf overlay-${title}-${2%%.*}VC-${3%%.*}.ps
!pdfcrop overlay-${title}-${2%%.*}VC-${3%%.*}.pdf
!mv overlay-${title}-${2%%.*}VC-${3%%.*}-crop.pdf overlay-${title}-${2%%.*}VC-${3%%.*}.pdf
EOF

gnuplot overlay.gnu
```

Appendix – Detailed installation instructions for critic2

These instructions do not deal with installation of the dependencies for critic2, which will vary by system and may or may not be necessary. If there is a failure during the installation commands outlined below, generally the error message will provide details of the missing dependencies which will need to be installed with the

```
apt install <dependency>
```

command or similar, depending on your Linux distribution. Additional and customizable installation options are provided with the critic2 documentation, here.

1) Clone the critic2 github repository
```
amayo@garrison:~/example$ git clone https://github.com/aoterodelaroza/critic2.git
```
2) Enter the created critic2 directory
```
amayo@garrison:~/example$ ls
critic2
amayo@garrison:~/example$ cd critic2
amayo@garrison:~/example/critic2$
```
3) Make a directory named “build” and change into this directory
```
amayo@garrison:~/example/critic2$ mkdir build
amayo@garrison:~/example/critic2$ cd build
amayo@garrison:~/example/critic2/build$
```
4) Enter the command `cmake ..`
```
amayo@garrison:~/example/critic2/build$ cmake ..
```
5) Enter the command `make`
```
amayo@garrison:~/example/critic2/build$ make
```
6) Return to your home directory and enter your .bashrc file in your home directory with `vi .bashrc` or your preferred text editor (vim, nano, notepad++, etc..).
```
amayo@garrison:~/example/critic2/build$ cd ~
amayo@garrison:~$ vi .bashrc
```
7) (vim) Enter “insert” mode with the ”i” key on your keyboard. Enter the following two export commands to link to the critic2 directory (first export line, adjust according to your directory path) and the source code files (second line, which should be exactly the same).
```
export CRITIC_HOME=/home/amayo/example/critic2
export PATH=${CRITIC_HOME}/build/src:${PATH}
```
8) (vim) Hit the “Esc” key on your keyboard to exit “insert” mode, then save and quit by entering `:wq` (this will send you back to the command line)
9) Refresh your .bashrc file by entering `. .bashrc`
```
amayo@garrison:~$ . .bashrc
```
10) Done! You should now be able to call critic2 from any directory on your computer with the command `critic2`
```
amayo@garrison:~$ critic2
```