

tCindex

SBC01-????-00

May 30, 1995

Prepared by: J.W. Pflugrath
Molecular Structure Corporation
3200 Research Forest Drive
The Woodlands, Texas 77381
(713) 363-1033
(713) 364-3628 FAX
jwp@msc.com

Prepared for: Contract No. 943072401
Mary Westbrook
Argonne National Laboratory
9700 South Cass Avenue
Argonne, Illinois 60439
(708) 252-8914
(708) 252-4021 FAX
westbrook@anl.el.anl.gov

Reviewed by:

Approved by:

Copyright © 1995, 1994 Molecular Structure Corporation
RESTRICTED RIGHT NOTICE SHORT FORM (JUNE 1987)

Use, reproduction, or disclosure is subject to restrictions set forth in Contract No. W-31
109-ENG-38 and Contract No. 943072401 with the University of Chicago, Operator of
Argonne National Laboratory

TABLE OF CONTENTS

1 SCOPE AND PURPOSE	3
1.1. Definitions and Abbreviations	3
2 BACKGROUND	3
3 INPUT IMAGE	6
4 RUNNING TCINDEX	7
4.1. Example.....	8
4.2. Results	10
4.3. Overcoming difficulties.....	11
5 FILE FORMATS	12

1 Scope and Purpose

This document describes how to use the tCindex module of d*TREK. tCindex tests the autoindexing modules of d*TREK. It produces unit cell parameters and crystal rotation values that can be refined with the tCrefine program. Input is a reflection list with observed reflection centroids and an image with a header that describes the experiment that created the observed reflection list. The reflection list output by the dtfind program is suitable for input to tCindex. The output of dtpredict may also be used if the Calc_Pixel1, Calc_Pixel2 and Calc_rot_mid fields are changed to Obs_pixel1, Obs_Pixel2 and Obs_rot_mid, respectively.

In order to index the reflection list and output crystal properties, tCindex also requires initial knowledge of the source, the crystal goniometer, the crystal rotation axis, the crystal rotation range, the detector and the detector goniometer. tCindex gleans all this information from the header of an image file, so that additional input is not required for autoindexing.

1.1 Definitions and Abbreviations

The definitions described in the dtpredict document are appropriate for tCindex.

2 Background

In order to integrate the intensities of Bragg reflections in a single crystal diffraction experiment, their positions in images must be predicted accurately. This can be done provided information about the source, the crystal, the crystal goniometer, the detector, the detector goniometer, and the rotation is known. For the predictions to be accurate, the crystal, source and detector properties must be refined by minimizing the differences between observed and predicted reflection positions. The radius of convergence in the refinement is limited, so approximate values of the crystal and detectors properties must be elucidated beforehand. The detector properties can be physically measured. The crystal properties (the unit cell parameters and crystal rotation) can be determined by autoindexing.

tCindex uses a vectorial description of the crystal, source, detector and their goniostats. This vectorial description has been described in the dtpredict document, in the *Proceedings of the EEC Cooperative Workshop on Position-Sensitive Detector Software (Phases I & II)*, *ibid. (Phase III)*, and in *Computational Aspects of Protein Crystal Data*

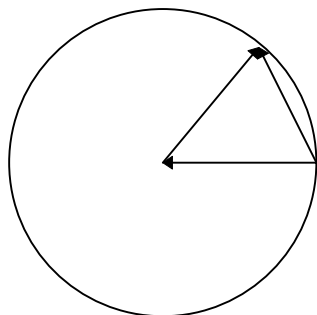
Analysis DL/SCI/R25. The reader is referred to these documents for a complete description.

If a reflection has observed coordinates Px1, Px2, and Rot, they can be transformed to the reciprocal lattice coordinates by first applying a pixel to millimeter function:

$$(Px1, Px2) \rightarrow (Xmm, Ymm)$$

Next this can be mapped onto the surface of the Ewald by multiplying the virtual detector coordinates $\mathbf{v} = (Xmm, Ymm, 1.)$ by the detector projector matrix \mathbf{D} (defined in the `dtpredict` documentation) to yield the scattered beam wavevectors \mathbf{s} scaled by a factor t . Normalizing $t\mathbf{s}$ to a length $1/\lambda$ or 1 if dimensionless units are used yields the scattered beam wavevector \mathbf{s} . Adding this the source vector \mathbf{s}_0 yields the reciprocal lattice coordinate \mathbf{r} of the reflection at the diffracting condition. In order to place all the reflections in the same reference frame, the vector \mathbf{r} must be unrotated by the observed rotation angle by multiplying by the inverse crystal rotation matrix \mathbf{G} and then the inverse of the crystal goniometer matrix \mathbf{R} .

$$\begin{aligned} t\mathbf{s} &= \mathbf{D} \mathbf{v} \\ \mathbf{s} &= t\mathbf{s} / |t\mathbf{s}| \\ \mathbf{r} &= \mathbf{s}_0 - \mathbf{s} \\ \mathbf{x} &= \mathbf{G}^{-1}\mathbf{R}^{-1}\mathbf{r} \end{aligned}$$



When a list of observed reflections is transformed the result is a list of reciprocal lattice coordinates that can be used for autoindexing. Recall that the vectors \mathbf{x} result from multiplying the Miller index of the reflection \mathbf{h} by the crystal setting matrix \mathbf{B} and the crystal orienting matrix \mathbf{C} :

$$\mathbf{x}_i = \mathbf{CBh}_i$$

The purpose of autoindexing is to elucidate the values of \mathbf{C} , \mathbf{B} , and \mathbf{h}_i given the values of \mathbf{x}_i and the knowledge that \mathbf{h}_i are integer triplets. Let $\mathbf{A} = \mathbf{CB}$ and after selecting 3 initial \mathbf{x} vectors \mathbf{x}_1 , \mathbf{x}_2 , \mathbf{x}_3 , we can make the following transformations:

$$\begin{aligned} \mathbf{x}_1 &= \mathbf{A}\mathbf{h}_1 \\ \mathbf{x}_2 &= \mathbf{A}\mathbf{h}_2 \\ \mathbf{x}_3 &= \mathbf{A}\mathbf{h}_3 \end{aligned} \quad (1)$$

$$\mathbf{H} = \mathbf{h}_1 \mathbf{h}_2 \mathbf{h}_3 \quad (2)$$

$$\mathbf{X} = \mathbf{x}_1 \mathbf{x}_2 \mathbf{x}_3 \quad (3)$$

$$\mathbf{X} = \mathbf{A}\mathbf{H} \quad (4)$$

$$\mathbf{A}^{-1}\mathbf{X} = \mathbf{H} \quad (5)$$

$$\mathbf{A}^{-1} = \mathbf{H}\mathbf{X}^{-1} \quad (6)$$

$$\mathbf{A}^{-1} = \begin{matrix} \mathbf{a}_1 \\ \mathbf{a}_2 \\ \mathbf{a}_3 \end{matrix} \quad (7)$$

From equation 5, we have a row in \mathbf{A}^{-1} times a column in \mathbf{X} is an integer. But recall that \mathbf{X} is composed of three of the reciprocal lattice vectors, but any three could be chosen. This means that any of the reciprocal lattice vectors times a row in \mathbf{A}^{-1} should be an integer. The algorithm used calculates \mathbf{X}^{-1} from three of the observed reciprocal lattice vectors, then loops over small integer guesses for a row in \mathbf{H} and uses equation 6 to give a trial value for rows in \mathbf{A}^{-1} . These in turn are multiplied by the all the original reciprocal lattice vectors to see if an integer is a result for all of them with the following residual function:

$$r = (1.0 + (\sum \cos 2\pi(\mathbf{a} \cdot \mathbf{x}_i) / n)) / 2$$

If r is > 0.95 , then the guess for a row in \mathbf{H} is a solution to the indexing problem. The best guesses for rows in \mathbf{H} are kept in order to build the \mathbf{A}^{-1} matrix, which is then inverted to give the \mathbf{CB} matrix. But since \mathbf{C} is an orthogonal rotation matrix, it follows that the unit cell constants can be derived from $(\mathbf{CB})^{-1}$:

$$\begin{aligned} a &= |\mathbf{a}_1| \\ b &= |\mathbf{a}_2| \\ c &= |\mathbf{a}_3| \\ \cos \alpha &= \mathbf{a}_2 \cdot \mathbf{a}_3 / (|\mathbf{a}_2| |\mathbf{a}_3|) \end{aligned}$$

$$\begin{aligned}\cos \beta &= \mathbf{a}_1 \cdot \mathbf{a}_3 / |\mathbf{a}_1| |\mathbf{a}_3| \\ \cos \gamma &= \mathbf{a}_1 \cdot \mathbf{a}_2 / |\mathbf{a}_1| |\mathbf{a}_2|\end{aligned}$$

Matrix **B** can be calculated from the cell parameters and used to derive matrix **C**. Finally, matrix **C** is decomposed into three rotations. The result is the unit cell parameters and the crystal rotation angles. These can be used as input to tCrefine and refined.

Further refinements of this algorithm are made by using differences between the observed reciprocal lattice coordinates. This helps in two ways: (1) it reduces the error in the calculated coordinates due to incorrect detector position and (2) it creates multiple observations for small reciprocal lattice distances. Difference vectors that overlap can be averaged together to yield a better estimate of the vectors.

3 Input image

The tCindex requires one image as input. It reads the header of the input image to get the initial values of the source, crystal goniometer, and detector properties. It uses the description of the crystal goniometer, crystal rotation, and detector goniometer found in the header when calculating reflection positions for a given Miller index. tCindex does not read the binary pixel data in the image which may be even be missing (i.e. the image is dimensioned 0 by 0). If the detector spatial distortion and non-uniformity of response information in a header requires additional images to be read in, then tCindex does so automatically. A typical image header is shown below. The keywords required by tCindex are the same as those required by dtpredict, so please refer to that documentation for their descriptions. (The following header matches the description of the detector and source goniometers at the National Synchrotron Light Source beamline X8C.)

```
1 {
2   HEADER_BYTES= 2048;
3   TYPE=mad;
4
5   SIZE1=512;
6   SIZE2=512;
7   CRYSTAL_GONIO_VALUES=0.0 0.0 0.0;
8   COMMENT=Header edited by dtheadereedit;
9   SCAN_TITLE=start end inc time nOsc nDark nDup nDlim nDC nDCup;
1  SCAN_ROTATION=0.0 12.0 0.2 4 0 1 0 100 1 0;
2  SCAN_ROTATION_AXIS_NAME=omega;
3  SCAN_ROTATION_VECTOR=1.0 0.0 0.0;
4  SCAN_TEMPLATE=lyso3.####.ss;
5  SCAN_SEQ_INFO=0 2 0;
6  ROTATION=0.0 0.2 0.2 4 0 1 0 100 1 0;
7  ROTATION_VECTOR=1.0 0.0 0.0;
```

```
8  ROTATION_AXIS_NAME=omega;
9  CRYSTAL_UNIT_CELL=88.29 88.29 103.65 90.00 90.0 120.0;
10 CRYSTAL_MOSAICSPREAD=0.145;
11 CRYSTAL_DESCRIPTION=Test of dtpredict;
12 CRYSTAL_SPACEGROUP=175;
13 CRYSTAL_REFINE_FLAGS=0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
14 CRYSTAL_ORIENT_ANGLES=10.04 -26.37 31.81;
15 CRYSTAL_ORIENT_VECTORS=1 0 0 0 1 0 0 0 1;
16 SOURCE_VECTORS=0.0 0.0 1.0 0 1 0 1 0 0;
17 SOURCE_POLARZ=0.5 1.0 0.0 0.0;
18 SOURCE_SPECTRAL_DISPERSION=0.0002 0.0002;
19 SOURCE_SIZE=0.0 0.0 0.0 0.0;
20 SOURCE_CROSSFIRE=0.0 0.0 0.0 0.0;
21 SOURCE_WAVELENGTH=1 1.54178;
22 SOURCE_REFINE_FLAGS=0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
23 DETECTOR_NUMBER=1;
24 DETECTOR_NAMES=D0_;
25 D0_NONUNF_TYPE=Dark_only;
26 D0_NONUNF_INFO=nonunf.img dark.img;
27 D0_SPATIAL_DISTORTION_TYPE=Interp_spatial;
28 D0_SPATIAL_DISTORTION_INFO=distor;
29 D0_DETECTOR_DIMENSIONS=512 512;
30 D0_DETECTOR_SIZE=50.0 50.0;
31 D0_DETECTOR_VECTORS=1 0 0 0 1 0;
32 D0_DETECTOR_DESCRIPTION=ANL-SBC gold detector single module;
33 D0_DETECTOR_REFINE_FLAGS=0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
34 D0_GONIO_NUM_VALUES=6;
35 D0_GONIO_NAMES=RotZ RotX RotY TransX TransY TransZ;
36 D0_GONIO_UNITS=deg deg deg mm mm mm;
37 D0_GONIO_VECTORS=0 0 1 1 0 0 0 1 0 1 0 0 0 1 0 0 0 -1;
38 D0_GONIO_VALUES=1.0 2.0 0.0 -.5 -.8 102.3;
39 D0_GONIO_DESCRIPTION=A simple detector goniostat;
40 CRYSTAL_GONIO_NUM_VALUES=3;
41 CRYSTAL_GONIO_NAMES=Omega Chi Phi;
42 CRYSTAL_GONIO_UNITS=deg deg deg;
43 CRYSTAL_GONIO_VECTORS=1 0 0 0 1 0 1 0 0;
44 CRYSTAL_GONIO_DESCRIPTION=A 3-circle eulerian crystal goniostat;
45 FILENAME=hex2.img;
46 BYTE_ORDER=big_endian;
47 Data_type=short int;
48 COMPRESSION=None;
49 }
```

4 Running tCindex

After tCindex has been installed and placed in your PATH, just enter tCindex. This version takes no command line options, but instead prompts for an image file name, the name of a reflection file, a maximum cell length and the output solution file. It reads the image header along with any spatial distortion files, non-uniformity files and then the

reflection file. Messages are written to stdout and stderr as required. The syntax for running tCindex is:

```
tCindex
```

4.1 Example

```
1  tCindex
2  Enter the name of image with info to construct things from: hex2.img
3  File hex2.img successfully opened.
4  Header of file hex2.img successfully read.
5  Image is 512 by 512 pixels.
6  Data_type in header is short int.
7  Compression_type is None.
8  Byte_order is big_endian.
9  Enter the name of the input reflection list file: hex2.ref
10 Creflnlist::constructor called with filename: hex2.ref
11 Creflnlist::nRead called.
12 Names of reflection fields:
13 H (int)
14 K (int)
15 L (int)
16 Detector_number (int)
17 Nonunf_flag (int)
18 Intensity (float)
19 SigmaI (float)
20 Obs_pixell1 (float)
21 Obs_pixel2 (float)
22 Calc_1mm (float)
23 Calc_2mm (float)
24 Calc_rot_start (float)
25 Calc_rot_end (float)
26 Obs_rot_mid (float)
27 Obs_rot_width (float)
28 Calc_polarz (float)
29 Calc_lorentz (float)
30 Calc_oblique (float)
31 Calc_partial (float)
32 Resolution (float)
33 Calc_recip1 (float)
34 Calc_recip2 (float)
35 Calc_recip3 (float)
36 INFO in Creflnlist::nRead, EOF after 87 reflections read in (87
37 total now in list).
38 File distor.calpar successfully opened.
39 Warning in Cimage_header::nRead, header does not have proper ending!
40 Info in Cimage_header::nRead, able to repair ending!
41
42 File distor.x_int successfully opened.
43 Warning in Cimage_header::nRead, header does not have proper ending!
44 Info in Cimage_header::nRead, able to repair ending!
```

```
34 Header of file distor.x_int successfully read.
35 Image is 256 by 256 pixels.
36 Byte_order is big_endian.

37 File distor.y_int successfully opened.
38 Warning in Cimage_header::nRead, header does not have proper ending!
39 Info in Cimage_header::nRead, able to repair ending!
40 Header of file distor.y_int successfully read.
41 Image is 256 by 256 pixels.
42 Byte_order is big_endian.

43 File distor.inv_x_int successfully opened.
44 Warning in Cimage_header::nRead, header does not have proper ending!
45 Info in Cimage_header::nRead, able to repair ending!
46 Header of file distor.inv_x_int successfully read.
47 Image is 256 by 256 pixels.
48 Byte_order is big_endian.

49 File distor.inv_y_int successfully opened.
50 Warning in Cimage_header::nRead, header does not have proper ending!
51 Info in Cimage_header::nRead, able to repair ending!
52 Header of file distor.inv_y_int successfully read.
53 Image is 256 by 256 pixels.
54 Byte_order is big_endian.
55 D0_NONUNF_TYPE: >>Dark_only<<
56 Cnonunf::nInitValues called

57 File nonunf.img successfully opened.
58 Header of file nonunf.img successfully read.
59 Image is 512 by 512 pixels.
60 Data_type in header is short int.
61 Compression_type is None.
62 Byte_order is big_endian.

63 File dark.img successfully opened.
64 Header of file dark.img successfully read.
65 Image is 512 by 512 pixels.
66 Data_type in header is short int.
67 Compression_type is None.
68 Byte_order is big_endian.
69 There were 262144 pixels, with 0 bad pixels in the nonunf file.
70 Enter max cell length: 200
71 Enter the name of the solution file: hex2.answers
```

Explanation

Line 1 tCindex is started.

Line 2 The input image file is requested and the name is typed in.

- Lines 3-8 Information about the image file is listed.
- Line 9 The input reflection file is requested and the name is typed in.
- Lines 10-36 The reflection file is read in. The fields in the file are listed. 87 reflections were read in.
- Lines 37-63 The detector spatial distortion correction requires that an interpolation table fileset be read in. These lines show these files have been successfully read.
- Lines 64-78 The detector non-uniformity correction requires that non-uniformity image and a dark image be read in. These lines show these files have been successfully read.
- Line 79 The maximum cell length is requested and is typed in.
- Line 80 Autoindexing is finished. The output file name is requested and is typed in.

4.2 Results

The results of autoindexing are presented in a reflection file. Since all reflection files must have H, K, L, Intensity and SigmaI columns, tCindex the three index values of the solution, the integerness residual, and the cell volume in these labeled fields. The cell is parameters and the crystal rotations angles are listed next. They are sorted on increasing residual so the last group is the best solution. Equivalent solutions are also listed. In the future, the results will be output to an image header.

```
1 3 11 0
2 H
3 K
4 L
5 Intensity
6 SigmaI
7 A
8 B
9 C
1 Alpha
2 Beta
3 Gamma
4 Rot1
5 Rot2
6 Rot3
1 486 483 0.920168 26333.4 88.2752 103.665 88.2865 89.9963 119.996 89.9826 -109.162 -20.9355 97.7389
2 481 484 0.920168 26333.4 88.2752 103.665 88.2865 89.9963 60.0041 90.0174 70.8382 -20.9355 97.7389
3 497 484 0.920168 26333.4 103.665 88.2752 88.2865 60.0041 90.0037 89.9826 49.3443 61.9302 53.5071
4 484 486 0.920168 26333.4 88.2752 88.2865 103.665 90.0037 89.9826 60.0041 160.842 -20.9355 97.7389
5 492 483 0.920168 26333.4 103.665 88.2752 88.2865 60.0041 89.9963 90.0174 -130.656 61.9302 53.5071
6 483 481 0.920168 26333.4 88.2752 88.2865 103.665 90.0037 90.0174 119.996 -19.1581 -20.9355 97.7389
```

7	486	497	0.920168	26333.4	88.2865	103.665	88.2752	89.9826	60.0041	90.0037	-62.3429	-4.98073	-27.0563
8	484	492	0.920168	26333.4	103.665	88.2865	88.2752	119.996	90.0174	90.0037	169.34	61.9302	53.5071
9	483	497	0.920168	26333.4	103.665	88.2865	88.2752	119.996	89.9826	89.9963	-10.6599	61.9302	53.5071
10	486	484	0.920168	26333.4	88.2752	103.665	88.2865	90.0037	119.996	90.0174	-70.8382	20.9355	-82.2611
11	481	492	0.920168	26333.4	88.2865	103.665	88.2752	89.9826	119.996	89.9963	117.657	-4.98072	-27.0563
12	497	483	0.920168	26333.4	103.665	88.2752	88.2865	119.996	90.0037	90.0174	130.656	-61.9302	-126.493
13	492	484	0.920168	26333.4	103.665	88.2752	88.2865	119.996	89.9963	89.9826	-49.3443	-61.9302	-126.493
14	481	483	0.920168	26333.4	88.2752	103.665	88.2865	90.0037	60.0041	89.9826	109.162	20.9355	-82.2611
15	497	481	0.920168	26333.4	88.2865	88.2752	103.665	90.0174	89.9963	60.0041	27.6745	-4.98073	-27.0563
16	484	497	0.920168	26333.4	103.665	88.2865	88.2752	60.0041	90.0174	89.9963	10.6599	-61.9302	-126.493
17	484	481	0.920168	26333.4	88.2752	88.2865	103.665	89.9963	89.9826	119.996	19.1581	20.9355	-82.2611
18	483	492	0.920168	26333.4	103.665	88.2865	88.2752	60.0041	89.9826	90.0037	-169.34	-61.9302	-126.493
19	483	486	0.920168	26333.4	88.2752	88.2865	103.665	89.9963	90.0174	60.0041	-160.842	20.9355	-82.2611
20	492	486	0.920168	26333.4	88.2865	88.2752	103.665	90.0174	90.0037	119.996	-152.326	-4.98072	-27.0563
21	486	492	0.920168	26333.4	88.2865	103.665	88.2752	90.0174	60.0041	89.9963	-117.657	4.98073	152.944
22	481	497	0.920168	26333.4	88.2865	103.665	88.2752	90.0174	119.996	90.0037	62.3429	4.98072	152.944
23	497	486	0.920168	26333.4	88.2865	88.2752	103.665	89.9826	89.9963	119.996	152.326	4.98072	152.944
24	492	481	0.920168	26333.4	88.2865	88.2752	103.665	89.9826	90.0037	60.0041	-27.6745	4.98073	152.944

4.3 Overcoming difficulties

Since tCindex gets all of its input information from an image header, difficulties may arise when the header is incomplete. Be sure that the detector spatial distortion information and non-uniformity information is correct and the files exist in the current directory or the filenames in the header include the path or directory.

The output of tCindex clearly shows when any files have not been read properly. Examine the output to ensure that the initial source, crystal, goniometers, detector and rotation properties are correct. If these are incorrect, then the image header is incorrect and should be modified.

In these cases, add or correct the information (use dtheadredit, see section 5 of the dtpredict manual) and run tCindex again.

Unlike the autoindexing in MADNES, tCindex automatically tries multiple combinations of vectors until a solution is found. It uses a different residual and a different method of grouping difference vectors. It also rejects reflections which are too close to neighboring reflections and too close to the crystal rotation axis. If you still have difficulties, you can only change the maximum cell length allowed. If this is too long and unreasonable, then many solutions will give good integer residuals because reflections will be predicted to overlap. Choose a legitimate maximum cell length that can still have separated reflections on the detector focal plane.

If you still have difficulty, input a wider rotation range of observed reflections.

If you still have difficulty, then the crystal is cracked, twinned, or slipping during the acquisition of the input observations.

5 File formats

Image file formats are discussed in the `calibrate` and `dtpredict` documentation. An example of an image header is given in Section 3.