Welcome to **d*TREK**, a flexible, customizable, device-independent software suite for the visualization and processing of single crystal diffraction images. This document describes how to use **d*TREK** and answers many frequently asked questions. For even more help, please send an e-mail with a description of your problem to Dr. Jim Pflugrath jim.pflugrath@rigaku.com.

Most of **d*TREK** can be used through two graphical user interfaces: **dtprocess** and **dtdisplay**, but one can also use scripts if desired. Please read on for a short description of how to process your images with **d*TREK**. **d*TREK** can do even more than what's described in this document, just ask! The **Frequently Asked Questions** section begins on page 8.

DISPLAYING IMAGES

D1. Check with your d*TREK manager that all site customizations have been made to the `Dtprocess` and `Dtdisplay` resource files **AND** you have installed a valid d*TREK license.
Note: Any "position out of range" message normally indicates a license problem.
A **dtdisplay** core dump indicates the Dtdisplay resource file is not available to the X server.
It is suggested to use the *tcsh* or *csh* shell with d*TREK. *Bourne* or *bash* shell users may wish to source `$DTREK_ROOT/bin/DTREK_LOGIN.sh`. The DISPLAY environment variable must also be defined. On some systems you may need to use the IP address and not the host name. You must also be able to create and execute files in the current working directory.

D2. Display `your_image_file` by typing on the command line:

```
dtdisplay your_image_file &
```

There are now two ways to zoom: (1) by clicking on the **Zoom** toggle button (upper left, below the File Edit … menubar) and dragging out a zoom box with the **left_mouse button**, *or* (2) by using your **middle** finger to press and drag the **middle_mouse button** to create a rectangle cursor. To move the cursor without changing its size, press and hold the **left_mouse button** with your index finger while moving the mouse. Practice this two-button operation until it is natural. Hint: do not use your index finger on the middle_mouse button -- you will have difficulty panning the cursor. To unzoom, single-click on the image with the middle-mouse button without moving the mouse, or toggle on and off the **Zoom** toggle button.

How to use the dtdisplay cursors

The toggle buttons in the upper left below the **File Edit …** menubar activate cursor modes for the **LEFT** mouse button only. When active, they control the action that occurs when the **LEFT mouse button** is released. If you do not wish to use the middle_mouse button to zoom, click on the **Zoom** toggle button and drag out a rectangle with the **left mouse button.** To edit the direct beam position click on the **Beam circle** toggle button then drag out a circle with the **LEFT mouse button** and position the circle center at the desired beam position. Before you release the **LEFT mouse button**, you may reposition the cursor by also pressing the **Ctrl**, **Alt**, or **Shift** key, then moving the mouse. Or you may keep the **LEFT mouse button** held down and also press the middle mouse button and move the mouse. Release **only** the middle mouse button to stop panning; release the **LEFT mouse button** to activate the cursor function (Beam circle, spot size, measure in Ångstrom).

If you set the **Edit / Erase (edit mask) mode** to draw or erase the beamstop shadow, then two new cursor functions are available via the toggle buttons: **Erase circle** and **Erase line**. To erase a beamstop circle, set the **Erase circle** toggle button and then drag out a circle to erase with the

LEFT mouse button and release. All cursors may be panned (moved around) from their initial starting position by keeping the **LEFT mouse button** held down and also pressing the **Ctrl**, **Alt**, or **shift** key, then moving the mouse. Or you may keep the **LEFT mouse button** held down and also press the middle mouse button and move the mouse. Release the **LEFT mouse button** to activate the cursor function.

D3. Make sure the detector distance, swing angle and wavelength are correct. If not, enter the correct values in the menu on the left side. Examine the image and especially the direct beam position as indicated by the red + mark. If the beam position is not visible, click on **View / Resolution arcs**. If the beam position is not in the correct place, then use the **Beam circle** toggle button and cursor to set the direct beam position to the correct place (or you can set the beam position with **dtprocess**, see below). Remember that ice rings can occur at 3.92 Å and 3.68 Å resolution (among others) which can help confirm both the beam center and detector distance. Consider creating a mask of bad or blocked pixels for use later (see step P4 below, highly recommended!).

PROCESSING IMAGES

P0. Start dtprocess by selecting **Process / dtprocess** in the dtdisplay menu bar. If desired, select **File / Open...** in the dtprocess menu bar to load different default input values that were saved in a previous session. Remember now, there should be two windows open: one for dtprocess and one for dtdisplay. One may also start dtprocess by typing `dtprocess` on a shell command line.

One may run **dtprocess** in an automatic mode by using steps P1 and P2 or in a manual mode by using steps P3 and on. Whether you use either manual or automatic mode please consult how to integrate in steps P11 and on.

dtprocess: Automated one-button Strategy for first-timers (automated Find, Index, Refine, Predict, Strategy)

P1. Select the dtprocess window, set **Flow chart mode** to **Auto strategy** if it is not already set. Double-check that the detector distance, swing angle (2θ), and beam position in the **Setup** menu are correct and edit them if necessary. It is also recommended that you specify a Mask file (i.e. beam.mask) to designate shadows and/or bad pixels.

P2. Select **Write dtprocess.head**. This automatically finds spots, indexes, refines and determines a strategy from the current menu settings. You can save current dtprocess menu settings with **File / Save Header As...** for later use. Please read about what happens by consulting steps P3 and on below.

If you have set **User chooses solution** in the **Setup** or **Index** menu of **dtprocess**, then you will have to pick an indexing solution from a list presented to you (see step P6 below).

dtprocess: Manual Strategy for advanced first-timers (manual Find, Index, Refine, Predict, Strategy)

P3. Perform steps D1 to P0 above or type **`dtprocess your_image_file &`**.
Set **Flow chart mode** to **Manual**.

P4. In the **Setup** menu, check that the detector position is correct (**Det dist**, **Det swing**, **Direct beam**). Use dtdisplay (**Edit / Erase (edit mask) mode**) to mask the pixels in the beam stop shadow by using the **Erase circle** and **Erase line** cursors (see step D2), then (in dtdisplay) **File / Save Mask or Image As...**. Back in the dtprocess **Setup** menu, change **Mask/Non-uniformity type** to **Simple mask** and enter the saved mask file. A mask is virtually required when processing CCD images since all CCDs have bad pixels. Set the **Master resolution** and requested **Spacegroup** if desired (you may override these settings later). Select **Write dtprocess.head**. (Note: the **Setup** menu has a **d*TREK output file prefix** field to add a prefix to default output file names. This is **very useful** when processing multiple scans of images.)

P5. Select **Find** in the flowchart. Select the image or images to search for spots from the **Images** list. Set **Sigma**, **Minimum**, and possibly other spot finding criteria (box size), then select **Run find**. Review the found spots in dtdisplay.

P6. Select **Index** in the flowchart. Select **User chooses solution**. If you know your spacegroup, set it in **Spacegroup num**. Select **Run index**. Enter desired lattice solution number in the **Input:** field, then view the different orientations and select one. You can click on **<*CR>** next to the **Input:** field to select the default answer. If the desired lattice is not listed, select **Abort**, double check detector distance and beam position, change **Resolution** and/or **I/sigI** values and re-run. Important: double check detector distance, swing angle (should it change sign?), direct beam position, or increase the **Max cell length**. A feature is to check the position of the beam by searching in a small (~10 pixel radius) near the input beam position. You may change the radius by specifying **–beamcheck radius+2 radius** (substitute an integer like 15 for **radius**) on the command line and pressing **<enter>**. The radius should be smaller than the shortest distance between spots or you may end up mis-indexed by one lattice spacing. You may also disable this check by specifying **–nobeamcheck** on the command line. You may wish to limit the reflections used in indexing with the **Resolution** and **I/sigI cutoff** values. Sometimes using difference vectors is helpful (click on **Show advanced options**, then **Use diff vecs, not direct vecs**). Since dtindex picks the lowest spacegroup number consistent with the Bravais lattice, you may wish to change the spacegroup number. Select **Edit / Edit header items ... / Crystal properties** to change the spacegroup. The program **dtcell** may be used to perform unit cell transformation, too. See step P23 below.

P7. Select **Refine** in the flowchart. Select one of the desired **Macros**, such as **Fit Most** or select **Interactive dialog** to manually select items to refine. Select **Run refine**. After refining with the dtfind.ref reflection list, it is best to refine from images. Dismiss the log file, then in the **Refine** menu select **Reflns: Image sequence** at the top and select the images used to predict, find and refine reflections. Select **Run Refine** again**.** Usually the choice of the first image or first few images is good, but sometimes you may wishes to use additional images such as those 90 degrees away (e.g. 85-95). At this point, it is a good idea to refine **ALL** parameters (fix **Source Rots** if there is a problem, this is the default). Repeat until satisfied. ***Note: crystal mosaicity can now be refined even with reflections from a single image with a few exceptions. See discussion on crystal mosaicity near the end of this document!***

P8. Select **Predict** in the flowchart. Select the image to predict spots for. Adjust the **Crystal mosaicity** if desired. Select **Run predict**. Examine the predictions in dtdisplay, if they don't match consider returning to step P7. To predict more (less) reflections, increase (decrease) Crystal mosaicity and re-predict. It may be wise to predict reflections for the second and third images in a scan to confirm that the rotation axis vector is properly specified. This is especially true if the images were created at a beamline. If they do not match, see the FAQ on page 8 below. Be aware incorrect refined values can be masked by an incorrectly large mosaicity value.

P9. Select **Strategy** in the flowchart. For very fast, but approximate results, leave the **Cell length scale** factor set to 0.3. Set the desired **Resolution** range (default of 0, 0 means go to edge of detector). Select **Run strategy**. (Note: contact Rigaku for info on multi-scan strategies.)

dtprocess: Automated Integration and Scaling for first-timers

P10. Start dtprocess as above. Set **Flow chart mode** to **Auto run next**. See step P14 below to see if you should set the integration **Box size** automatically or manually. Select **Write dtprocess.head** in the **Setup** menu.

dtprocess: Manual Integration and Scaling for first-timers

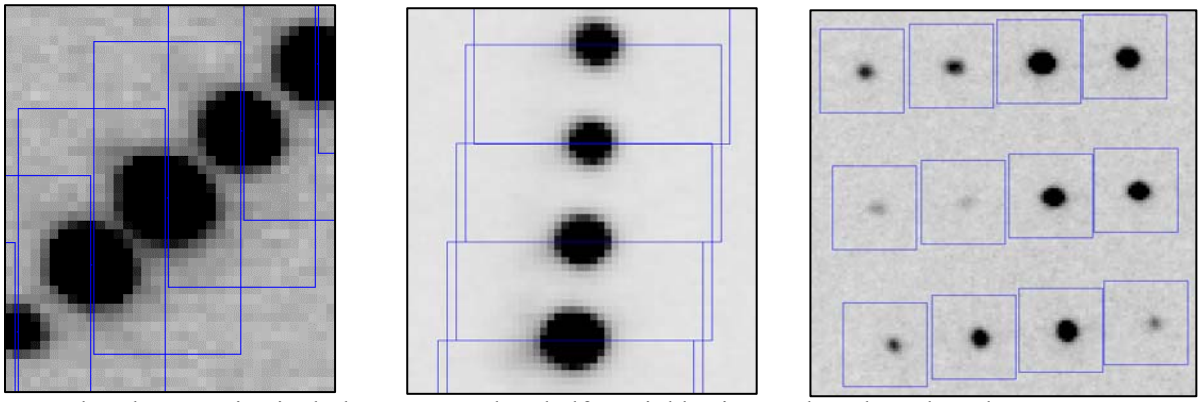
P11. If the first image to integrate is not the same one used above, then it is safest to repeat the find, index, and refine steps P1-P8. Select **Predict** in the flowchart. Confirm the **Crystal mosaicity**, then **Run predict**. Confirm that the predicted reflections superimpose on the spots in the image. Select **Integrate** in the flow chart.

P12. Either set the image start and end sequence number in the **Image sequence:** fields or select the images to integrate in the **Images** list (i.e. select **Select all**). Integrate each set of contiguous images in a separate run with a different **Batch Prefix** (see step P18).

P13. Set the **Resolution range** to integrate. If 0,0 is used then reflections on the entire area of the detector will be integrated.

P14. Set the **Max box size**, a first guess to use is about 3-4 times the spot size by examining the spots in dtdisplay or leave at 0,0 to let the program decide. This sets the maximum box size used for integration. **dtintegrate** will choose spot shapes automatically and may adjust the box size smaller. Do not select a box size that includes more than half of a neighboring spot. Do not make a box size too small…you must leave a border around the spot for background. If you make the box size too big, then more memory might be used and processing will be slower. In dtdisplay, change the reflection symbol size by clicking on **Edit / Refln view props…**.

Examples of integration Box size:



Note that the Box size includes no more than half a neighboring peak and need not be square.

P15. **Pad & MosModel:** Set **Pad** (the first number) to 0 or 1 for wide-sliced images and to 2 or 3 for fine-sliced images. If the crystal mosaicity is much less than the image rotation angle width (i.e 0.3° mosaicity versus 1° image rotation angle width), then you may wish to set padding to 0.

MosaicityModel: One can also have the refined mosaicity modified by the equation: MosaicityUsed = (MosRefined * MosMul) + MosAdd. For example, to double the refined mosaicity use **2 0**. To fix the mosaicity to 0.5, use **0 0.5**. To add 0.2 to the refined mosaicity use **1 0.2**. Remember that the actual mosaicity used for prediction is not so critical since **dtintegrate** will determine the spot shape in 3 dimensions regardless of the mosaicity used. Set **Prerefine** to **2 refine batches** for most situations. For wide-sliced images, consider Pad 0, MosaicityModel 0 2.

P16. Set the **Profile parameters** to 50 7 for profile-fitting macromolecule crystals, perhaps 20 20 or 10 10 for small molecule crystals. For very large unit cells **and** a large detector, some CPU memory can be saved by profile fitting sequentially after integration is finished. Add an " **s**" to do sequential profile-fitting which saves memory but uses much more disk space for the scratch files.

P17. Set **Images per batch** to 1 2 for viruses, 1 4 for most proteins, and 20 20 for small molecules. With version 7 there are **two** parameters: the first is images_per_**scaling_batch**, the second is images_per_**refinement_batch**. If only one number is given, then it will be used for both parameters. At the end of every refinement batch (i.e. the second number), **dtintegrate** will do a refinement, re-prediction and integration. You may need to use a larger images_per_**refinement_batch** if (1) you are doing thin or fine slice data collection, (2) your refinements are not stable, or (3) you have too few reflections per refinement. Tip: if **dtintegrate** is using too much memory, reduce images_per_**refinement_batch** or profile fit sequentially (see P16).

Reflections will be divided into scaling *batches* by **dtintegrate**. Scaling batches will be every image_per_**scaling_batch** (the *first* parameter). Sometimes the first or last scaling batch will not have enough overlaps during scaling and will not scale well. In this case, use the –batchrestrain option (see step P22) or the **dtreflnmerge** … **-rebatch** … command to rebatch later if you like. Tip: a small molecule crystal which is entirely bathed in the X-ray beam should not need batch scaling (see below).

P18. If you plan on combining reflections from more than a single scan, use a different **Batch prefix** for each scan. Set it to a single digit (0-9). A blank is treated as no-prefix and is not the same as a 0. This is required in scaling to designate reflections from different scans. If you forget to do this, you can always run **dtreflnmerge** to add a batch prefix later (i.e. `dtreflnmerge dtprofit.ref -sBatch+=2 2_dtprofit.ref`).

P19. If you want to automatically scale the reflections after integration, set the **Flow chart mode** to **Auto run next**.

P20. Select **Run integrate**. As integration proceeds, check that refinement is good by the comment that is output with the **rms** values, that the parameter shifts are essentially zero, and that few reflections are rejected. The dtdisplay program should have spots in the circles and no spots in the squares. You can turn off the squares by selecting **dtdisplay/Edit/Refln view props…/Symbol/o and no []**. At end of integration, consult the **Summary** table to ascertain if problems occurred. Also examine the list of axial reflections (zone.ref) for systematic absences. Plot the period refinement results by clicking on **Utils/PlotStats…** or with

```
jdtplot dtintegrate.log
```

If parameters drift too much, you may wish to fix them and re-integrate. This step produces a **dtprofit.ref** file than can be used for scaling and averaging.

[P20.5] The **Merge** or **Combine refln files** dialog is not necessary if you have a single `dtprofit.ref` file from the integration of a single scan. If you have more than one

`dtprofit.ref` file to scale, merge them by selecting them in the **Reflnlists** list (use ctrl-left mouse to add to the selection) and click on **Run merge**. Read also some of the FAQs below.

P21. Consider using the **dtcell** program to help determine the Laue class and spacegroup. If you use **dtcell** after **dtscaleaverage**, you may need to re-run **dtscaleaverage** with the selected spacegroup. If you use **dtcell** before **dtscaleaverage**, the reflections may not be scaled well enough for **dtcell** to give good results. **dtcell** can also be used to interchange axes and re-index reflections. Do **not** run **dtcell** from the **dtprocess** command line. Use a terminal window or shell. For example:

```
dtcell dtintegrate.head dtprofit.ref -reduce
```

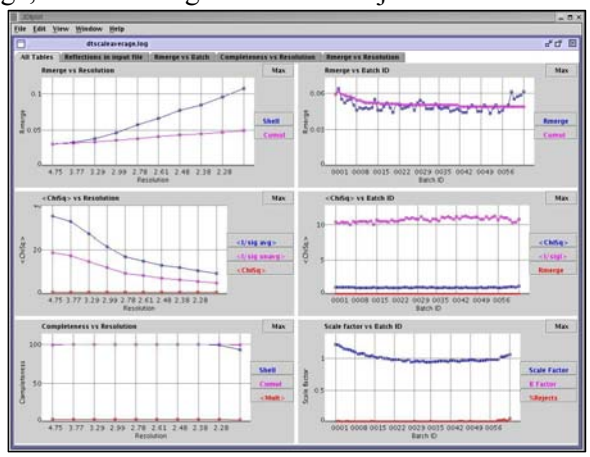
Consult the **dtcell** documentation for more information.

P22. Select **Scale/Average** in the flowchart. Select the reflnlist file to scale. Generally this is **dtprofit.ref** from the integration, but may be another reflnlist that you have merged from other lists. Select the type of absorption correction to apply. Usually **Batch+4th 3D** spherical harmonics is best for macromolecule scans, while **4th 3D +4th 2D** is best for small molecule scans. Use **Batch only** if the reflections do not come from a single crystal. To automatically determine an error model set ErrMul (Error multiplier) and ErrAdd (Error addend) to a blank or negative number. You may also set ErrMul and leave ErrAdd blank or negative. Set the **Reject(s)** to delete a fraction of the data (i.e. **0.01**, this is simplest) or a sigma (i.e. **3**) or both (i.e. **3 0.01**). To have $|\chi^2|$ set near a specific number (i.e. 1), enter that number as a second ErrAdd number. By default, ErrMul and ErrAdd will be adjusted to try to get $|\chi^2|$ close to 1. Add **–nochisq** to the command line to prevent this. Or add **–chisq #** to try to get $|\chi^2|$ close to #. Add **–batchrestrain …** to the command line to restrain batch scale factors (See **dtscaleaverage –h** for help). Select **Run scale**. ***Please consult dtscaleaverage.doc for more detailed information, other possibilities, and many other helpful suggestions.***

P23. Examine your scaling results with jdtplot by clicking on **Utils/PlotStats…** or with

```
jdtplot dtscaleaverage.log
```

You want any batch scale factors to vary smoothly within a scan. You want to reject batches that have bogus scale factors, high $|\chi^2|$, high Rmerge, and/or a large number of rejects. Batches are



identified by a BatchID which usually is a single number of 4 or more digits (e.g. 0001, 0100, 10001, 20190). The last 4 digits of the BatchID are usually the same as the image sequence number. The preceding digits are the Batch Prefix or ScanID (see step P18 above). Use the **dtprocess Scale/Average** menu to re-run **dtscaleaverage** with the batches to reject in the **Batches to reject** field (comma separated, wildcards allowed, no whitespace; i.e. `'0001,100?,2001-2003,4*'` use quotes if you use wildcards). Sometimes the first or last

scaling batch will not have enough overlaps during scaling and will not scale well. Either reject these batches or rebatch. Use the **dtrebatch** program or the **dtreflnmerge** … **-rebatch** … command to rebatch (or **dtscaleaverage … -rebatch bookends …**) and re-run scaling. After examining the plots, you may choose to reject more batches or images. I recommend that you cutoff the resolution at the point that the unaveraged <I/sigI> is 2 (see the FAQs below).

Note the jdtplot above on the left has batches (i.e. images) with inconsistent scale factors, high Rmerge values, high reject rates, and high $|\chi^2|$ values. When these batches are deleted from scaling, the result is the jdtplot on the right. The bad batches were the result of a goniometer problem.

The following pages contain some further tips on using d*TREK as well as many frequently asked questions along with their answers. These pages are a good place to look for help. If you can not find the answer to your question, please do not hesitate to send an e-mail to Dr. Jim Pflugrath <jim.pflugrath@rigaku.com>.

Versioning of output files

Generally, d*TREK programs will not overwrite an existing file. Instead, existing files will be renamed by appending a version number to them. Thus, the most recent version of a file has no version number, while the oldest version is 1. Example:

dtintegrate.log	newest
dtintegrate.log.1	oldest
dtintegrate.log.2	created AFTER dtintegrate.log.1, but BEFORE dtintegrate.log

When you are satisfied with your processing, you may wish to delete earlier versions of your files. A unix command like `rm dtintegrate.log.?` would not delete dtintegrate.log, but would delete both the dtintegrate.log.1 and dtintegrate.log.2 files.

Prefixing of output files

Generally, d*TREK will create output files with obvious default names. For example **dtprocess** will create dtrefine.scom and dtrefine.log when it runs the **dtrefine** program, while **dtrefine** itself will create the dtrefine.head file. A prefix can be prepended to default filenames. The prefix is defined by the DTREK_PREFIX environment variable. For example, if

setenv DTREK_PREFIX peak_

was set, then the just mentioned **dtprocess** and **dtrefine** files would have been named peak_dtrefine.scom, peak_dtrefine.log, and peak_dtrefine.head. The DTREK_PREFIX environment variable is a useful tool when testing d*TREK strategies within the same directory. If DTREK_PREFIX is not set (i.e. unsetenv DTREK_PREFIX), then no prefix is prepended to default filenames. The **dtprocess/Setup** menu has a field to enter the d*TREK prefix, too.

Multiple **dtdisplay** programs can be running on the same graphics workstation if the DTDISPLAY_PREFIX environment variable is set differently for each one. Only those jobs with the same value for DTDISPLAY_PREFIX will use that particular **dtdisplay**.

Purging of files

With **dtprocess**, one may select **Utilities / Purge files** which will delete the scratch and some other non-essential files created by d*TREK.

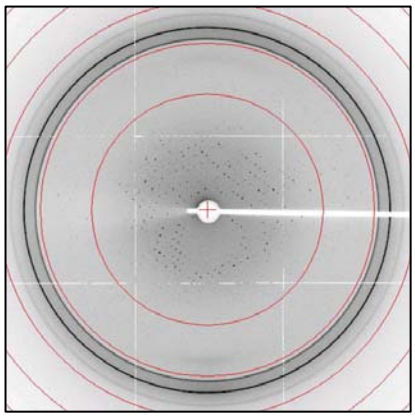
FREQUENTLY ASKED QUESTIONS

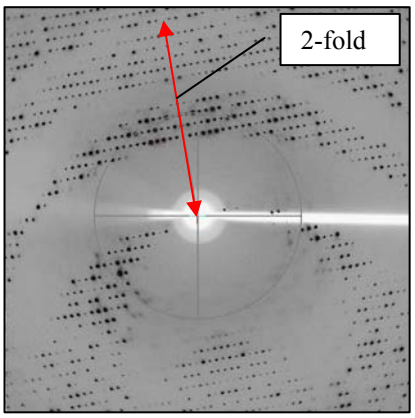
The beam center is close, but not quite right in my synchrotron images, so I cannot index.

Always determine the beam center in ***PIXELS*** on the detector because then you don't need to worry about the detector orientation and any world coordinate conventions. Different beamlines use different conventions even for the same kind of detector, so be wary! If you get the millimeter position of the direct beam from some other program, you may need to switch X and Y and/or subtract X and/or Y from the maximum pixel dimension. In **dtprocess / Setup** you can put **mm** after a direct beam position value and it will convert to pixels for you. Remember: it is wise to expose a direct beam image or a powder sample *at the distance* you will collect images and keep it with your images for use later. Also make use of **dtdisplay** options, the **dtfind … -beamcenter** option, and the **dtindex … -beamcheck** option. See also the next question.

My image doesn't index. Why not? How do I determine the direct beam position correctly?

The most frequent reason the image doesn't index is that the direct beam position is wrong. Use the **Beam circle** cursor in **dtdisplay** to set the direct beam position (see step D3 above). Make sure that position appears in the **dtprocess** Setup menu (see step P4 above). The direct beam position will be at the center of any ice or powder rings that are in the image (see figure on left below). The direct beam position will be on any symmetry axes (see figure on right below). The direct beam position should be somewhere in the beamstop shadow (see figure on right below). The direct beam position will be an **integral** number of d*-spacings along a symmetry axis, so you can make the beam circle cursor have a radius that is an integral number of d*-spacings and move the whole circle (see step D2 above) until the perimeter intersects a Bragg reflection and the center is at the beam position.





I've clicked on Run index **and the dtindex.log file appears, but it seems stuck and the cursor is changed to some funny shape. What's going on?**

When the cursor is a funny shape (it's a desk clock actually), then **dtprocess** is waiting for you to enter input in the **Input:** field below the log file. You can type something in the field or paste in it. The buttons **Yes** and **No** will paste a yes or no in and add a <carriage_return>. The button ***<cr>** will add a <carriage_return> to whatever is in the input field sending the text to the waiting program. Most programs expecting input will have a default answer, so clicking on the ***<cr>** is often the only action needed. Check the output log file (in the window) to see what the default would be. If you see the funny cursor, but the **Input:** input field is not visible, click on **View/Logfile…** then select a log file to view.

I indexed from the first image and the predictions look great, but when I predict for the second image, they are off.

It is likely the rotation range of the second image is not consecutive (i.e. first is 0 to 1°, second is expected to be 1 to 2°, but is actually 90 to 91°) or the rotation axis direction is wrong. Check the rotation start and end of the second image. If that is OK, try changing the axis direction by editing the header in dtprocess: Click on **Edit/Edit header items…/All properties** then on keyword replace CRYSTAL_GONIO_VECTORS and negate the the first vector (1 0 0 goes to -1 0 0), then **Replace**. If that does not work, then edit the header again, but this time check SCAN_ROTATION_AXIS_NAME and change the name to the axis that is the scan axis which should match a name in the CRYSTAL_GONIO_NAMES keyword. Then click on **OK** to save the modified header. You should repeat the Index and Refine steps after making these changes. It is also possible that the first and second images are from different scans and that's why predictions were off originally. You may also try to change the direction of the rotation axis vector in other ways, such as from along X (1 0 0) to along Y (0 1 0).

Finally, some beamlines write images with incorrect image headers. BioCARS at the APS is the biggest offender and deserves special mention here. Contact Jim Pflugrath for more details.

I've indexed and processed a scan of images. I want to use the same information for a different scan of the same crystal at another wavelength, detector position, or crystal goniometer.

This is changed from earlier versions. One can predict spots for the different scan using the **dtprocess Predict** dialog. Add `–differentscan` (same detector position) or `–differentdet` (different detector position) to the command line option. The output header produced is suitable for input to an integrate run. The new header file usually called **dtpredict.head** will be created.

Another possibility is to use the `–differentscan` (same detector position) or `–differentdet` (different detector position) as a **dtintegrate** command line option:

```
dtintegrate dtrefine.head –seq 201 300 –differentscan –prefind 2 …
```

Another possibility is to use the **Special scans** option in the **dtprocess Integrate** menu. Click on the **Sequence num** toggle button and select **Special scans**, then select the rightmost toggle button with a scan template. For this to work, all the images from the different scans must be in the same directory and have the same scan template, but difference sequence numbers. You may need to add an extra ? to the scan template in the **Setup** menu first to see all the images. Select the images you wish to process, then **Run integrate**. The program will automatically build and run a script to process each scan and merge the results.

I've clicked on Run dt**** **and a message box that says "Abort not possible" appears. What's going on?**

dtprocess could not figure out the process id of the subprocess that is running, so it cannot kill the subprocess if you click on the **Abort** button. There are two common situations that cause this: 1) the subprocess has finished in less than 1 second and 2) you have started **dtprocess** from **dtdisplay**, but then exited from **dtdisplay** leaving **dtprocess** running. If you do wish to abort the subprocess, you will have to use another window to run **`ps`** and then **`kill`** commands.

I've started with image 10, and want to start processing from image 1, but I get an "Invalid starting seq num" error.

d*TREK assumes that image 1 is from a different scan, so it won't let you do that. You should dtdisplay the first image you intend to process. If you don't want to find, index, and refine from that image, it's OK. Just select image 10 after *starting* with image 1. This error can also occur if there are no images that match your image scan template.

The program complains about a missing mask or non-uniformity file. What's that?

This file is used to flag bad pixels. If the Mask/non-uniformity type is set to *Simple mask* in the **Setup** menu, then a mask file must be supplied. The mask file is like an image file. Pixels with a value of 0 in the mask file are considered to be bad pixels. An easy way to create this file is described in step P4 above. If a CCD image has a pedestal or pixel offset, then bad pixels may have a value other than 0 in the images. To make such images usable by d*TREK, use the dtnonunfedit command. Suppose the pedestal is 10 and all pixels below a value of 11 are bad, then use a command like:

```
dtnonunfedit input_mask_image input_mask_edit_image 11 output_mask_image
```

Note that input_mask_image, input_mask_edit_image and output_mask_image may all be the same name:

```
dtnonunfedit marccd.mask marccd.mask 11 marccd.mask
```

With the above command, the file versioning feature of d*TREK will create a new `marccd.mask` file. See `dtnonunfedit -h` for additional help.

I have raw CCD images that have not been corrected for dark current, DC offset, non-uniformity of response, and spatial distortion. How do I process these images?

d*TREK can process these images if you tell it about the dark current, DC offset, non-uniformity of response and spatial distortion. You will need calibration files from the detector manager or manufactor or beamline. These files usually include the DARK file and the TRANSFORM file (best), or the DISTOR files and the NONUNF file. For some raw images from some detectors a dark file can be made with the dtsbcdarkmask and dtsbcdark programs. With the appropriate calibration files and software, you can transform raw images into corrected images. Please read the document $DTREK_ROOT/doc/MSWORD/sbcinfo.doc. Contact us for more information.

I have a monoclinic cell (e.g. spacegroup P2 or C2), but dtindex does not give me the beta angle that I want. How do I get dtindex to work for me?

There are 44 lattice characters of which many are monoclinic. Of the possibilities, **dtindex** only shows the one with the best least squares residue or, if the residuals for two or more characters in a class are under 2%, the one with the lowest beta angle above 90°. To get a different cell, select the triclinic solution (spacegroup P1), perform a refinement with **dtrefine**, then use **dtcell** to reduce the cell:

```
dtcell dtrefine.head –reduce
```

Then you can enter the letter L at the **dtcell** prompt to see all 44 lattice characters and L# to select the monoclinic cell that you desire.

I have processed data from two crystals and they don't merge well together. What happened?

It is possible that the two datasets have different settings or hand if the spacegroup is a polar one like P3 or R3. One of the datasets needs to be re-indexed to match the other. This is easily done with the command:

```
dtcell dtprofit1.ref –transform reference_dtprofit.ref
```

The above command will determine and apply the transformation matrix to `dtprofit1.ref` that gives the lowest Rmerge with `reference_dtprofit.ref`.

When dtdisplay displays reflnlists, I don't understand the symbols or colors.

In dtdisplay, click on **Help/Refln colors** and read the answer in the pop-up window.

How do I delete or add spots to the reflnlist found by dtfind?

In **dtdisplay**, click on the spot you want to delete, then press the **-** or **Delete** key. Or press the **+** or **Insert** key to add a spot. You can also click on **Edit/Del 10% lowest reflns** which deletes the 10% lowest intensity reflections. After all edits, be sure to **File/Write refln list…** to write a new reflnlist file.

How do I turn off (on) the squares around spots with I display a reflnlist in dtdisplay, especially when integrating?

Click on **Edit/Refln view props…** and change **Symbol** to **o and no []** (**o and []**).

When I change the window size in Integrate or Find, the refln symbols in dtdisplay do not change size. How can I make the displayed box size the same as the window size in dtintegrate?

In **dtdisplay**, click on **Edit / Refln view props …** and change the symbol shape and size in the dialog that pops up. Since **dtintegrate** does not tell dtdisplay the integration window size, you have to enter the numbers in the dialog. Also note that each reflection can have a different window and spot size.

How do I convert a d*TREK reflnlist file to a CCP4 MTZ file? A CNS file?

Use the **dtrek2mtz** program supplied with the CCP4 distribution (source code is found in $DTREK_ROOT/bin/dtrek2mtz-new.f). Use **to_cns** in CNS. These programs only know about the ASCII or text version of d*TREK reflnlist files. To convert a binary d*TREK reflnlist file to text format use dtreflnmerge: `dtreflnmerge binary.ref text.ref –text`

Examples:

```
to_cns –f dtrek –nomean -.ref -.hkl
dtrek2mtz hklin dtscale.ref hklout dtscale.mtz << EOF
END
EOF
```

I have 2 (or more) reflnlists from different scans that I want to merge and scale together.

See the `$DTREK_ROOT/doc/MSWORD/dtscaleaverage.doc` document for how to do this. In a nutshell, the two scans should have different batch prefixes (see step P18 above). You can use (items in [] are options):

```
dtreflnmerge 1_dtprofit.ref [-sBatch+=1] \
             2_dtprofit.ref [-sBatch+=2]   1+2_dtprofit.ref
```

command to combine the two dtprofit files, then scale in the usual way (see step P22 above).

I have to exclude some images from scaling. How do I do that?

In the **dtprocess Scale** dialog, enter the batch IDs for those images in the **Batches to reject** field and click **Run scale**. Or add the option `–rejectbatch id sBatchTemplate` to the **dtscaleaverage** command line. `sBatchTemplate` is a string with sBatch IDs (separated by commas, wildcards * and ? allowed, ranges allowed) to delete from the scaling process with no whitespace. Example of `sBatchTemplate`: 0001,0010-0011,200?,3*

I have problems reading compressed images with the .Z and .gz extensions. Can you help?

Currently, uncompressing is done to a temporary file in the same directory as the compressed images, so if you do not have write permission in that directory, you will have problems. To overcome this, set the environment variable TMPDIR to a directory that you have write permission for:

```
setenv TMPDIR /usr/tmp
```

By the way, using TMPDIR on a local disk different from the image disk will generally speed up processing of compressed images.

How do I determine the resolution cutoff for my data?

Look in the dtscaleaverage output at the "Rmerge vs Resolution" table. Make sure that the reduced ChiSq is near 1 otherwise your sigmas may not be reasonable. Then look at the **I/sig unavg** column. I recommend cutting the resolution at the point where the unaveraged I/sig falls to 2. This is

usually the same point where the Rmerge is between 30% and 40%. Others would recommend a higher resolution cutoff, but this does not take into consideration that there are errors other than random errors that are normally distributed.

|ChiSq|s are close to 1

Rmerge vs Resolution

Resolution range	Average counts	Num rejs	Num mults	I/sig unavg	I/sig avg	Rducd ChiSq	Model Eadd*	Rmerge shell	Rmerge cumul
19.77 - 3.01	7340	117	1190	15.5	22.7	**0.94**	0.05	0.031	0.031
3.01 - 2.39	2924	31	1249	9.1	13.5	**0.82**	0.08	0.049	0.036
2.39 - 2.09	1992	29	1259	7.0	10.4	**0.82**	0.10	0.063	0.041
2.09 - 1.90	1276	19	1174	5.1	7.3	**0.88**	0.16	0.093	0.045
1.90 - 1.76	714	15	949	3.6	5.1	**0.95**	0.20	0.133	0.049
1.76 - 1.66	462	13	749	2.8	3.8	**0.99**	0.27	0.185	0.052
1.66 - 1.58	352	10	672	2.5	3.3	**1.07**	0.29	0.225	0.054
1.58 - **1.51**	284	9	636	**2.0**	2.8	**1.09**	0.35	0.279	0.056
1.51 - 1.45	224	10	596	1.7	2.2	**1.18**	0.42	0.339	0.059
1.45 - 1.40	199	10	363	1.7	2.3	**1.22**	0.38	0.361	0.060
19.77 - 1.40	1951	263	8837	6.0	8.6	0.95	0.03	0.060	0.060

I/sig unavg is the mean I/sig for the unaveraged reflections in the input
I/sig avg is the mean I/sig for the unique reflections in the output
* When EMul == 1.96

Cutoff at 1.5 Å resolution

OK, I know the Bravais lattice from indexing and refinement, but how do I determine the exact spacegroup?

You need to determine the symmetry in the observed diffraction pattern (2-folds, 3-folds, 4-folds, 6-folds, screw axes, systematic absences, etc.). The **dtcell** program does this automatically for you:

```
dtcell dtintegrate.head dtprofit.ref
```

Type 'dtcell' for help on using this utility. You may also plot the axial reflections (h00, 0k0, 00l) found in zone.ref directly onto overlaid images with dtdisplay (File/Read reflnlist…) in order to examine the actual diffraction data for systematic absences along the reciprocal cell axes a*, b*, and c*. Intensities for the *h00*, *0k0*, and *00l* reflections are listed also near the end of the log file created by dtintegrate and/or dtprofit.

In the end, only **you** can judge whether the software made the correct choice, so take the time to learn some crystallography if you have not already done so.

I integrated the images with spacegroup P222 (or P2), but now that I see the systematic absences, I have determined that the spacegroup is $P2_12_12_1$ (or $P2_1$). Do I need to re-integrate the images before I do scaling?

No.

Do I need to re-integrate if I integrate in spacegroup P1 and the true spacegroup is C2 (or $P2_12_12_1$)?

One can use dtcell to reindex the output of dtintegrate to the chosen spacegroup, so one should not need to re-integrate. However, there may be some advantages integrating again with the proper spacegroup because the reflection predictions may be better if the unit cell constraints are applied in the reflection position refinement.

I am told that the CCD detector I used at the synchrotron has a pixel saturation value 50000, is that a problem?

No, that is not a problem, but you must tell d*TREK about this lower value. The default pixel saturated value is 65535 for most CCDs. To change this, you must set an environment variable before using d*TREK:

```
setenv SATURATED_VALUE 50000
```

What about the source polarization at synchrotrons? How is that set?

It is set in the .head input files in the SOURCE_POLARZ keyword. To change or check it in dtprocess, select Edit/Edit header items/All props and keyword SOURCE_POLARZ. For horizontal axes, the value should be 0.99 0 1 0. For vertical axes, the value should be 0.99 1 0 0. Then click Replace_value and OK. If you have any doubts, please contact Jim Pflugrath by e-mail.

Why is my mosaicity value different from other packages?

There are many models for crystal mosaicity that try to model the observed rocking curve of Bragg reflections. We have found that no model is correct in all cases. Models that have been used are gaussian, double gaussian, Lorentzian, Cauchy, or a combination of distributions. d*TREK empirically finds the rocking curve widths of the Bragg reflections and integrates the entire spot. It is not concerned with full-width at half maximum because long tails will not be accounted for. At the present time, d*TREK and other programs do an excellent job with sharp reflections. However, with weakly diffracting crystals or with spots that have weak tails, it may be best to use **dtpredict** and **dtdisplay** to judge the effective mosaicity by eye (i.e. trial and error) and then not refine it (i.e fix it). The problem is that the weak tails of strong reflections may not be that weak when compared to nearby weak reflections. These cases must be flagged as overlapped and treated appropriately. d*TREK does not refine beam crossfire or divergence separately, thus these values are subsumed into the value of the effective mosaicity refined by d*TREK.

We have found also that mosaicity can be artificially increased by poor orientation matrix refinement. Consider this, if the crystal orientation is slightly off, then a larger mosaicity value is required to explain the observed reflections on the image.

Since version 7.3, a **Mosaicity Model** has been implemented in **dtintegrate**. The mosaicity value used (**MosUsed**) is defined as:

MosUsed = (MosRefined * MosMul) + MosAdd

where MosMul (default 1.0) is a multiplier for the refined mosaicity and MosAdd (default 0) is a number added. Your experience will tell you what numbers to use. If you want **dtintegrate** to use an overestimate of the refined mosaicity, you can set MosMul and MosAdd appropriately (see P15 above). Remember that even if you underestimate the mosaicity, that a non-zero **Pad** value may still save the day.

How do I make dtintegrate use less memory?

Since **dtintegrate** uses a 3D method for integration it must buffer pixel areas from several images in memory for all active reflections. There are some specific things one can do to use less memory. First, one can explicitly set a smaller box or window size with the -window option. Use a size at least 3 times the spot size (see P14 above), but also check the "Strong peak info listing" in the **dtintegrate** log file which shows the spot ellipse sizes (Major Axis, Minor Axis) to make sure they are less than one-third the box size. Second, one can use an images_per_refinement_batch of 2 (see P17 above). This will perform refinement and integration every 2 images and flush the filled buffers out of memory more often. Third, use a pad of 0, but be sure to have mosaicity correct or slightly overestimated. Fourth, do not use the -display option. You will have to edit the command line to remove this option. Fifth, process images in two or more resolution ranges. For example, instead of processing from 100 to 1.2 Å resolution in a single run, process in two separate runs from 100 to 2 Å resolution, then from 2 to 1.2 Å resolution.

The dtprocess File/Print **button doesn't seem to print. How can I get it to print?**

The print command used by this button is set by the initial value of the X resource `*dtprocess*tfLogfileSearch.value in the` $DTREK_ROOT/bin$DTREK_BINSUFFIX/X/Dtprocess resource file. The default is `lpr`, but should be changed to the print command on your system (e.g. `lpr -Ptext`).

I have a dtdisplay window open to look at images, but the dtintegrate job running in the background keeps updating dtdisplay. How can I stop that?

In dtdisplay click on File/Respond_to_updates/dtintegrate in order to toggle whether dtdisplay responds to image updates received from dtintegrate. You can set the environment variable DTDISPLAY_PREFIX to a prefix string (i.e. setenv DTDISPLAY_PREFIX 1_) before launching dtdisplay and it will respond only to those programs that have the same DTDISPLAY_PREFIX value (i.e. those launched after launching dtdisplay).

OK, the GUIs (graphical user interfaces, pronounced gooo-eee's) are great, but I'd rather use a shell script so I don't have to click any buttons. Can I do that?

Yes, an example shell script is found in `$DTREK_ROOT/bin/sample_script.com`. Also, the **dtprocess** GUI is building scripts and running them. Just look for dt*.scom in your working directory.

What files do I need to keep?

dtintegrate.log	The integration log file
dtscaleaverage.log	The scale/average log file
dtprofit.ref	The profile-fitted integrated intensity reflection file (output by dtintegrate, input to dtscaleaverage.
dtscale.ref	The unique reflections after scaling and averaging.
dtintegrate.head	A .head file which can be used to get going again.
Your image files	The raw data is always good to keep.

There are other log files, reflection files, and the script files which you do not need to keep as they are easily recreated. If you created an unmerged, scaled reflection list file, then you will want to keep that.

d*TREK is a collection of programs that are run separately by the **dtprocess** graphical user interface. Files are used to transfer information from one program to another. For example, the **dtfind** program read an input.head file to get information and writes out a dtfind.head file with the spot shapes and a dtfind.ref file with the spot positions. Generally the output of one program is used as input to the next program. The output filenames are usually based on the name of the program (e.g., dtfind.head, dtfind.ref, and dtfind.log are created by running dtfind).

I have changed the beam position in the dtprocess Setup menu, but I do not see the red + change position in the dtdisplay image.

Initially, **dtdisplay** uses the beam center found in the image file. You must select **Write dtprocess.head** for the edited beam position to be communicated to the **dtdisplay** process. You may also run other *action* commands in other menus to communicate the beam center: Run find, Run refine, Run integrate will update the beam center, too.

I'm starting to publish lots of structures solved with data from d*TREK. What's the proper reference for d*TREK?

Pflugrath, JW (1999) *Acta Cryst.* D**55**, 1718-1725.

Although that's the official publication, Thad Niemeyer of Rigaku/MSC has re-written the bulk of the crystallographic code in d*TREK. Robert Bolotovsky of Rigaku has made numerous contributions as

well. The reqab absorption correction algorithm is written by Robert Jacobson of Iowa State University. We also want to acknowledge the published works of M. Rossmann, W. Kabsch, A. Leslie, Z. Otwinowski, W. Minor, T. Higashi and G. Bricogne. They, their colleagues, and many others have helped us put this thing together.