CRUNCH, The Manual

Version number 1.2

R. de Gelder, R.A.G. de Graaff, A.J. Kinneging & W.J. Vermin

X-ray Department Gorlaeus Laboratories
PO Box 9502, 2300 RA Leiden
The Netherlands

February 23, 2007

Inquiries:

Department of Structural Chemistry, Leiden Institute of Chemistry, Gorlaeus Laboratories, University of Leiden, PO Box 9502, 2300 RA Leiden, The Netherlands.

Email:

rag@chem.leidenuniv.nl (R.A.G. de Graaff)
rdg@sci.kun.nl (R. de Gelder)

Disclaimer:

The software comprising the program system Crunch is provided in good faith. The software is maintained regularly. However, no guarantee whatsoever is given, neither with respect to the system's performance, nor concerning the results obtained using the software. At no time the authors or their employers can be held responsible for any damage or hardship incurred, directly or indirectly, by the use of the software.

Contents

1 General Information

Crunch is a direct method program developed for solving difficult small and medium- sized structures *ab initio*. Datasets should be of atomic resolution, better than 1.1. The system has been designed to run under the Unix operating system. So far successful installation has been effected on the Silicon Graphics, True 64, IBM Risc 6000, HP UX, SUN, OS X and Linux platforms, both Intel and Alpha. Although in principle aimed at difficult equal-atom problems, Crunch may be used effectively for routine structure determination, of heavy-atom structures too. The program needs about 20 Mb disk space. A minimum of 256 Mb of Ram is recommended for using Crunch. Usually more is better here!

The system consists of two main sections and assorted utilities. The first section, Deter, determines the phases. The second section, Autofour, evaluates the results of each Deter cycle, trying to find the complete model based on the results obtained by Deter. The contents of the unit cell should be given as accurately as possible. Especially the second principal program in the system, the section which tries to find the complete model based on the results of the first section, depends on the availability of good estimates of B-overall and the scale. Most matrix and vector manipulations are done using the Blas and Lapack routines. It is of great advantage to use processor specific optimized versions of this software. A makefile for doing this is supplied for the program Deter. An include file 'flags' which defines the flags used in the makefiles is prepared during installation. This file may be adapted to the local situation.

Crunch supports just about any input of structure factors you can think of as long as the files contain hkl, F or F**2 and sigma(F or F**2), one reflection pro record and no records containing other items. The entries in the file must be separated by blanks, comma's or plus or minus signs. A record such as 12 3 4 1.2340.566 will not do. Crunch_1.1 or higher no longer requires the presence of the Dirdif system (inquiries Dirdif to: ptb@sci.kun.nl or rdg@sci.kun.nl).

In earlier versions of Crunch Dirdif was used to handle the crystallographic data such as cell contents, unit cell dimensions, symmetry etc. Crunch uses a reflection file containing the asymmetric unit in reciprocal space ONLY. However, your reflection file is cleaned of redundancies automatically. Atomic coordinates are produced in the .pdb and .spf formats. Visual inspection of the results is therefore straightforward, using the Pluton/Platon graphic suite written by Ton Spek (email: a.l.spek@chem.uu.nl). Unix scripts are written to be compatible with the Bourne/Korn shells. The scripts provided do not allow the running of more than one crunch-job from within one directory. However, multiple runs of Crunch on one computer from different directories and/or by different users are allowed.

Crunch was developed for the *ab initio* solution of the phase problem, which also covers the situation where a crystallographer doesn't know anything about his compound except that it is organic and its approximate atomic weight. This has proven to be extremely useful for the identification of unknown natural compounds.

If you've used Crunch in any resulting paper please refer to:
Automatic Determination of Crystal Structures using Karle-Hauptman Matrices.
Acta Crystallographica A49 (1993), 287-293,
R. de Gelder, R.A.G. De Graaff & H. Schenk.
X-ray Department Gorlaeus Laboratories
PO Box 9502, 2300 RA Leiden
The Netherlands

2 Installation

Copy crunch.tar.gz and Crunchmake.sh into the directory where you would like to have Crunch installed. If necessary, make Crunchmake.sh executable.

Mind that sufficient (> 20Mb) disk space is available before you start the installation. Specify enough memory (>= 256Mb) for the program to run in, the script asks for this information. You need to have gunzip, the gnu unzipper, available on your system. Running the Crunchmake script takes care of the complete installation of Crunch. In the directory you've chosen for Crunch to be installed into, a new directory 'crunch_1.2' should now be present. This directory should contain the following: A directory manual containing this manual - in the form of the files manual.aux, manual.toc and manual.tex in the directory manual, the files crunch and flags and the directories drivers, programs, source and rn001. Crunch is a dynamic link to the script which actually runs the program. Move this to some suitable directory in your PATH, such as /usr/local/bin. Copy the directory rn001 into the directory where you usually solve your structures. You may have to close and reopen your X-terminal to activate the link 'crunch'.

You are now ready to proceed.

After you've tested the success of the installation - see next section - you might like to recompile everything, using levels of optimization different from the standard provided by the script and/or different libraries or compilers. To accomplish this, go to the directory crunch_1.2. In the file flags change the value of the variable FFLAGS into the desired level of optmisation. Other compiler options may be inserted here as well. If you want to use a different compiler change the value of FF. If you want to try some liberary or other to use Lapack and Blas change the variable LDFLAGS to the appropiate value. Next change the value of LIB to 'makefile'. After you have changed everything to your satisfaction go to the directory source and type `'make clean'` Next copy the 2 lines of Crunchmake.sh where the value of the environment value CRUNCH is set and exported and execute these shell commands. If you like you can put these lines into your .bashrc file. Then opening a terminal will set the environment value CRUNCH and export it, meaning you don't have to do this ever again. Finally, typing `'make CRUNCH'` will reinstall the system using the options you have chosen.

Crunchmake.sh determines the type of machine it is going to install Crunch on. On OS X machines the Lapack and Blas libraries provided by Apple are used by default. On Linux Alpha machines the presence of Compaq's cxml library (http://www.digital.com for details, search for cxml) is detected automatically if the standard rpm's have been used to install it. If you have Lapack and Blas installed on any other system change the value of the variable LIB in the file flags as described above. Change LDFLAGS too in order to reflect the location of these libraries.

The script checks for the presence of a Fortran compiler on your system. The prefered option on Linux and OS X systems is GNU's gfortran. If this is not present I strongly advise you to get it installed on your system. For me this compiler works on Intel Macs too even though the current version is experimental. The g77 system may be used as well, however, there is a serious performance bonus in using gfortran. Especially g77 without a numeric library such as Atlas for Deter to use, gets you a slow program.

NB Sometimes lots of things go wrong and the script still tells you everything is fine. Do not be fooled, check your installation as is described above as well as by running the test problem rn001 provided.

3 Starting the first time

Open the directory rn001 which you have just created. Type `'crunch rn001 clear'`. This will leave you with just the files rn001.crysin, rn001.frefa and crunch.log. Now run Crunch by typing `'crunch rn001 try 1 5'`. Crunch will now proceed to solve the structure. On a modern computer this should take a few seconds, however, if you are using g77 without a numeric library we are talking minutes here. The space group is P21, N=48, it is a simple steroid, an equal-atom structure.

When asked for input, choose default options by giving a return. When the program is finished your directory rn001 should contain the files rn001.report, rn001.pdb and rn001.spf, among others. Read the report file. If you wish to do so, plot the result using Ton Spek's program Platon, using the rn001.spf file. Compare the results, checking against the files present in the rn001 directory supplied. If you meet with any problems please contact the authors.

4 Solving a structure of your own

Open the directory where you would like to solve your structure. In the following 'code' stands for the name of your compound. This may be any suitable abbreviation, of course. If you already have a file in there 'code.crysin', containing the crystallographic data in the Dirdif format, so much the better. If not, Crunch will prompt you for the information required. The file 'code.crysin' will be created.

The file 'code.crysin' contains the usual crystallographic data such as cell constants, wavelength used, cell contents etc. It is important to give the cell contents as accurately as is possible. Next, you need a file containing h,k,l, F and sig(F), one reflection for each record, with proper separation between numbers (spaces, comma's or a plus or minus sign are required). The values for h,k,l should be consecutive on your file. However, the other variables may occur in any order. E.g. the hkl may be the last three numbers etc. In the record other numbers may be present, the program will skip them if required. Before you do anything else your data file must be converted into a 'code.frefa' file. Type `'crunch code conhkl 'filename'`. Here 'filename' is the full name of the file - NOT the full path - containing your reflections. Crunch will ask you a few questions and next the file will be converted to a Crunch-friendly one. The whole procedure is self explanatory. Now type `'crunch code try 1 5'`. Provide the information you are asked for.

Use the defaults indicated. Crunch determines automatically whether to treat the problem as a heavy atom structure or an equal atom one. In some cases you are given the option to treat for instance an organic compound containing chlorine as an equal atom problem. Basically you just give return a few times, Crunch will use sensible defaults and your structure will be solved. Try larger values of the last number given - 5 in this case - if you expect the solution to be difficult. Your structure should come out if all goes well. Crunch is a multi-solution method. In the example given, 5 random starts will be used to find the solution.

If you suspect your data is not truly of optimal quality, Crunch works best if you calculate E-values from your reflections using the option 'blessing'. See for details the section on syntax(Nr 6).

5 A few helpful hints

1. Sometimes heavy atoms are disordered, for instance the chlorine in a perchlorate may be quite 'mobile'. If you expect something like that treat your structure as an equal atom one. Sometimes changing the cell contents - in 'code.crysin' - is needed in order to do this: a) If Crunch does not give you the option or b) you have a largeish organic molecule containing one or two sulfur or chlorine atoms. If Crunch has problems finding a solution, even if you told the system it is dealing with an equal atom problem, change the cell contents in your code.crysin file, replacing the heavier atoms by an equivalent number of oxygen atoms.

2. Atoms such as P, S and Cl are heavy in the context of Crunch. That is compared to first row elements such as C and O etc.

3. If you have a transition metal or an even heavier element in your compound DO NOT specify Cl and the like as heavy atoms.

4. If something has gone wrong, e.g. you've started Crunch while you didn't mean to and next interrupted the proceedings by 'Control c', ALWAYS start again by using the Crunch option 'clear': Type `'crunch code clear'`. This will clean up your directory, preventing otherwise inexplicable failures.

NB using the option 'clear' will delete the E-values you may have calculated using the option 'blessing'

6 Crunch syntax

You may use any one of the options given below running the program. Always type your alias or the name of the dynamic link to the Crunch-script, followed by the complete specification of the run. The general syntax of a Crunch command line:

```
crunch code run [start end] [first n, last n, all]
```

Below the variables 'crunch', 'code' and 'run' have the following meaning throughout:

crunch:	The name suggested for the link to or the alias of the script crunch.uni.
code:	compound name (e.g. rn001 in the test structure).
run:	Defines the type of run which will be executed.
start:	The number of the first trial do be done.
end:	The number of the last attempt.

The last two values must be specified if one of the options `'first'` or `'last'` is to be used.

`conhkl filename`	This type of run has a syntax different from the others. Use this run to convert your local structure factor file into a crunch-friendly (.frefa) format. The filename should be given with extension(s). See for more information the section Examples.
`clear`	Most files created by earlier runs of Crunch are removed. Only the .crysin, deter.use, autofour.use and .frefa files are kept.
`blessing`	E-values are calculated and saved for further use. The routines from the DREAR-suite, Levy and Eval are used. Reference: Blessing, R.H. & Smith, G.D. (1999). "Difference structure factor normalization for heavy-atom or anomalous scattering substructure determinations" J. Appl. Cryst. 32, 664-670. Until the option 'clear' is used to clean up, these E-values will be used rather

than the ones calculated by Normal.

`pmf` If you have first calculated E's using the option blessing this option may be used to create a file 'coordinates' containing partial structures derived from the patterson function. These will be used as starts to the phasing process instead of random phases which are used normally.
Values of start and end must be given to determine the number of starts to be generated.

`try` Crunch uses random starts from 'start' until a solution is found or 'end' is reached. If the structure is found various files such as code.pdb, code.spf and code.report are created. Some information on all trials done is given in the file 'hits'.

`collect` Scan all crunch-attempts from 'start' to 'end'. The results are given in the file 'hits'.

`deter` Only the section calculating the phases - Deter - is executed. The results are saved in files 'code.phi1...n'. As above, the calculations are done for the trials 'start' to 'end'.

`peaks` The phases generated using the option 'deter' are converted into possible fragments, using Exfft. This again is done for the trials 'start' through to 'end'.

`recycle` This option implements a sort of 'shake and bake' strategy. The trial 'start' is run in the normal way. If a solution is obtained then the program stops as usual. If not, the best model obtained in the Autofour section is used to calculate phases to be refined using Deter. A new model is found etc. This process is continued until the structure is solved or 'end-start+1' cycles are completed.

`autofour` This option may be used in two completely different ways:

1. You may have found a heavy atom or a small part of the structure by using some other method. If you then prepare a .frefa file as was described above, followed by creating a file 'code.pek' containing the known part of your structure, you can try to extend the model to the complete structure by just typing `'crunch code autofour'`, not giving the values start and end. A description of the code.pek file is given below.

2. You have created files 'code.pek1...n' by using the peaks option. Then type `'crunch code autofour start end'` to try to extend one or more of the models obtained.

The values `'start'` and `'end'` are optional except if you use `'recycle', 'deter', 'pmf' or 'peaks'`. The default values are 1 and 10 respectively.

Using the option `'try'`, which is normal when you are trying to solve a structure, usually the second step, model extension using Autofour, is started with the largest consistent fragment present in the map. However, following the value of `'end'` you may specify the options `'first'` or `'last'`, followed by a number 'n', or `'all'`. The effect is that the first 'n', the last 'n' or all peaks found in the map, are used as input to Autofour. Remember to specify the values `'start'` and `'end'` if you'd like to use the options just described. The script gets confused if you specify just the parameters `'first'` or `'last'`, interpreting them as values for `'start'` and `'end'`.

An example run: `'crunch rn001 collect 27 74'` means crunch rn001 and collect 'hits' from sets 27 to 74. Hits contains a brief report on this run. If you are using the option 'try' the results of the first successful run are summarized in a file code.report. Should you want to look at the results of a particular successful run collected in the file hits, use the command `'crunch code try n n'`. Further examples may be found in the designated section.

7 Required before running Crunch

You just need a structure factor file, the usual crystallographic data such as cell constants, cell contents, the wavelength used etc. Any structure factor file containing one reflection pro record will do, as long as h,k,l, Fobs (or F**2) and sigma(F) (or sig(F**2)) are there. Remove any records which contain "non intensity" information from the file. Convert the file to a 'code.frefa' file using the option conhkl. Do use an alias or the dynamic link provided to run the script crunch.uni. In the special case you want to use the syntax `'crunch code autofour'`, that is, you want to use crunch just to extend a model found some other way, you need to prepare one more file, 'code.pek'.

8 What to do in the case of problems?

In the case of failure of a standard run of Crunch - the structure does not come out - try changing some input parameters to Deter. Look in the file deter.asc which should be present in the Crunch subdirectory 'drivers'. Create a file called deter.use in the directory you are trying to solve your structure from. This file should be of the same format as deter.asc, however you need to specify only the items you wish to change.

1. Increase the number of trials.

2. Use the option 'blessing' to calculate E-values and try to solve the structure again. This is particularly useful if the quality of your data is not completely up to scratch. In one test, collecting the results of 100 trials, using Blessing's E's six solutions were found, instead of two obtained using Normal80's E's.

3. In the subdirectory 'drivers' of the Crunch system a file 'deter.asc' is present. In the directory were you are trying to solve your structure create a file 'deter.use', of a similar format. You need to supply only those parameters you would like to give non-default values to. It is best not to change NNRM, INMX and NINMX in this way, as Crunch allows you to specify the number of matrices used and their orders when you start your efforts or after you've cleaned up the directory. Try different values for the following parameters:

   IBK Default this value is calculated by Deter, based on the number of 4-byte words the program has available. This number is calculated on installation, the value you've given as the total amount of Mb present in the computer is used. If you make the value of IBK too large Crunch will run out of memory. The value you specify here must be smaller than the square root of the number of words available in Deter(c.f. the file deter.cout). However, sometimes smaller values give better results. Try something like 1500.

   NRL Increase or decrease the value a bit.

   ICRA You might like to try the value 2 here.

   NOTF This parameter defines the weight of forbidden reflections in the matrix construction process. The default is one, i.e. they are treated as valuable compared to very weak reflections. Sometimes treating them as statistically very weak works better: Change the value into zero. Changing the value into two is another option, increasing their importance.

   Try all this using the option `'crunch code deter 1 1'`. Inspect deter.cout and try to maximize the average E-value in the matrices and/or the number of strong, independent reflections.

4. For larger structures it is sometimes useful to use matrices of smaller dimensions than the default chosen by Crunch. Type `'crunch code clear'`. Next, type `'crunch code deter 1 1'` and now, when the system asks if you would like to use default values for the matrix construction, type `'u'`.

   Choose a different, preferably lower, number of matrices and/or a different order for the matrices. The minimum number allowed is 1. The number of matrices you may use is prescribed, not just any positive number will do. If you type a number here which is unacceptable Crunch will substitute the nearest acceptable one. The order of the matrices you may choose to be any odd number. Try to choose your parameters to fit the requirements. E-average should be high. The number of strong, independent reflections should be at least about 2.5 times the number of independent atoms you are looking for. A significant number of symmetry equivalents - including Friedels - should be present. If you succeed in finding a set of matrices which looks promising, - high average E, enough strong independent reflections while containing a sufficient amount of redundancy - try to solve the structure again using the options `'try'` or `'collect'`.

5. Generate about a hundred solutions. Look in the file 'hits' for the solution which yields the lowest value for R2. Type: `'crunch code recycle n n+10'` or 20. n is the number of the trial chosen. If this does not work and other trials looking better than most others exist, try them in the same way.

6. As there is a file deter.asc, so a file autofour.asc is present in the directory 'drivers'. The format is the same as is its use: You may override the values given here by creating a file autofour.use in your working directory, specifying different values therein. If you suspect you reflection data is not of terribly good quality, try a different value for STOPR, something like 40. STOPR is the threshold value of R_2 in percentage points. For values of R_2 below STOPR Autofour considers the structure to be solved. Should your file contain reasonable standard deviations, try using IWEIGHT=1. Or, if you're trying to solve a structure containing one or two relatively heavy atoms such as Cl or S, try giving IVAL some value, say 10 or 20. The latter is a good idea too if you use Crunch just to extend a small model based on a few (heavy) atoms.

If all this does not work give up. Crunch will not solve this structure, at least not for you. If you like, contact the authors, giving full details of what you have done so far.

9 The file code.pek

For each atom in your input model the file should contain a record as defined below.

1 - 10		May be used for atom name	Format: 10X
11 - 20	FI	Table number belonging to this atom	Format: F10.0
21 - 30	X	The fractional x-coordinate	Format: F10.3
31 - 40	Y	The fractional y-coordinate	Format: F10.3
41 - 50	Z	The fractional z-coordinate	Format: F10.3

Note: The records specify the format of the code.pek file, needed if you want to run autofour with a model found NOT using Crunch. Pay attention to the parameter

FI, its value should tally with the input you have specified in your file 'code.crysin' For instance, you've found a Cl-atom and you would like to use this to find the other atoms. You specified Cl as the second atom type in your crysin file. Use FI=2.0 in this case.

10 Some useful files

flags	This file contains the compiler and linking options during used during installation of the system. The adventurous user might want to play with this.
code.frefa	The reflection file which must be present. However, you may convert a reflection file in a local format using the option provided. The .frefa file may be used in the Dirdif system as well.
code.crysin	The Dirdif system file which contains the crystallographic data on your compound. This may be prepared interactively.
hits	For each random start this gives the determinant value obtained as well as the values of R_2 and R_2 expected found by Autofour.
crunch.log	Contains a log of all Crunch's runs submitted from this directory.
deter.use	Similar to 'drivers/deter.asc'. Values specified will override defaults in 'drivers/deter.asc'.
autofour.use	As 'deter.use' but defining the input to Autofour.
deter.cout	The output of the program Deter. Only the output of the last run is kept.
autofour.cout	The output of Autofour's last attempt.

The last two files are not present if the Crunch run was successful. The following files are present in the case of success only.

code.report	This file contains relevant information on the proceedings leading up to the result.
code.spf	The coordinates in spf format. This file may be used to prepare plots of the molecule(s) found using Pluton or Platon.
output	The file contains all output generated during the last successful attempt.
code.pdb	The coordinates in Protein data bank format.

11 Examples

1. Converting a local structure factor file named 'xtal.dat'

   In this example the user typed 'crunch hornef conhkl xtal.dat'. The following information appeared on the screen - the bit in [] was provided by the user - :

   ```
   ==================================
   ** CONHKL conversion of reflection files
   ==================================
   ** The first record on your reflection-file is:
   HKL 0 -2 0 66.23596 1.00171 1 SKIP
   == > You must now specify your record type using keywords
   ** Example:  hkl skip fobs sigma
   ** This means:
   ** The first 3 numbers are h,k,l, the 4th should be skipped,
   ** the next one is Fobs and the last one is sigma(Fobs)
   ** Typing f2obs instead of fobs tells the program your file
   ** contains F**2 rather than F
   == > Specify your record:  [skip hkl fobs sigma skip]
   ==================================
   ** conhkl ended
   ```

   Users of the package XTAL will recognize the record shown as typical input of ADDREF, the program in XTAL which reads in structure factors. Hornef is the name of the compound. The programs shows the first record of the file, followed by an explanation on how to proceed. Here the records consist of a bit of text, h,k,l, Fobs, sigma(Fobs) and some other information which Crunch does not need. By typing the record between brackets the user tells Crunch to skip the first item and to consider the next five numbers as h,k,l, Fobs and sigma respectively. The rest of the record is skipped again. Use lower case characters here! At the end of the run the file 'hornef.frefa' was prepared, ready for use by Crunch.

2. Trying to solve a structure the first time.

   The user next typed `'crunch hornef try'`. On the screen appeared:

   ```
   Crunch interactive, always give return to choose defaults
   CELLCO C 92 H 72 O 20
   The structure is an Equal_atom structure
   [D]efault or [U]ser specified matrix construction?
   ```

   The cell contents are shown on screen. Crunch has decided that hornef is an equal atom structure. The user gave a return on the next question as he wanted to try a default run of Crunch. In this case a maximum of 10 random starting sets is tried. If you next run Crunch you will not be asked these questions again. Should you want to change anything, for instance the way the Karle–Hauptman matrices are constructed, you will have to remove the file 'code.par', in this case hornef.par. You may choose to use the command `'crunch code clear'` instead.

3. The user wants to collect the results of a number of trials, to see whether there are any (non)solutions which look more promising than the others. See the section 'What to do in the case of problems?'. He typed: `'crunch hornef collect 1 25'`. After a few moments the Unix prompt reappears.

4. On completion the file 'hits' is inspected. Solution number 9 looks better than the others: A higher value of the determinant and a lower final value of R_2 are the criteria to look for. The user now typed: `'crunch hornef recycle 9 15'`. Again after a few seconds the Unix prompt reappears. Crunch will do the calculation for trial number 9 again. However, instead of proceeding to trial 10 in the case of failure, the best model obtained is used to calculate phases which are used as starting values for the maximization of the determinants. The refined phases are used to calculate a new start for the second section of Crunch. In this case this is done six times (15-9). Of course Crunch stops iterating if a satisfactory solution is obtained. Note that it is not necessary to use the options `'try'` or `'collect'` before using `'recycle'`. It is perfectly valid to try to solve your structure using a run such as: `'crunch hornef recycle 1 10'`.