Graph Ent

Version β, Release 1.1

NICHOLAS M GLYKOS, 2002

Please send comments, suggestions and bug reports to `glykos@mbg.duth.gr`

This document was prepared with LaTeX2e. The graphs were prepared either with locally written programs (*ps* or *Conv2D*), or with *pluto* and *pltdev* from the CCP4 suite of programs.

The latest version of the program is available from the following www address :

`http://www.mbg.duth.gr/~glykos/`

"... Most of the current confusion is, in the writer's opinion, the direct result of failure to define the problem explicitly enough. Today, programming and running a computer is much easier than actually thinking about a problem, so one may program an algorithm appropriate to one kind of problem, and then feed the data of an entirely different problem. If the result is unsatisfactory, there is an understandable tendency to blame the algorithm and the method that produced it, rather than the faulty application."

Edwin T. Jaynes
"On the Rationale of Maximum-Entropy Methods", Proceedings of the IEEE, 1982, Vol.70, No.9, pages 939–952.

Contents

1 Recent additions

1.1 Version β, Release 1.1

- Graphical user interface added to allow selection of .mtz columns for the calculation (actually a wrapper for CCP4's sftools program).
- Added support for .mtz GLGL column-type combination.
- Support for threaded FFTW execution on parallel machines.
- Changed the way the FOM-weighted syntheses are calculated.
- Grid lines are drawn on the plot every 0.250 fractional units.
- Updated documenation (postscript and html and man).

1.2 Version β, Release 1.0

- When the calculation reaches convergence, *GraphEnt* will plot the final map using the mean and rmsd of the whole map (and not of the given section as happens during normal operation). The implication is that you may notice a significant change on the appearance of the *GraphEnt* map upon completion of the calculation.
- Keyword `GRANsections` added to allow plotting a projection of a stack of sections (projection calculated using the maximum function and not the average).

1.3 Version α, Release 0.9

- The program detects instabilities in the calculation by monitoring the value of entropy and confirming that it is monotonically decreasing. If not, diagnostic messages are written out.
- When the user issues a TSTP signal (CTRL-Z) from the terminal, the program will behave as if convergence was achieved, write the current map and gracefully exit.
- When the program is called as "graphent" (normally through a symbolic link to "GraphEnt"), it enters a very quiet mode in which only error and diagnostic messages are written out.

1.4 Version α, Release 0.8

- Limit on number of reflections expanded to 120000.
- Fix an out-of-bounds error in the calculation of the normal probability plot (which would gracefully dump core if the total number of reflections for a difference Patterson function exceeded the maximum number of reflections).
- For a Patterson function calculation without an explicit definition of the F_{000} term, the program will use $F_{000} = 2\max(F)$ [instead of the previously adopted $F_{000} = \max(F)$].
- **Year-2001 enhancements :**
  - At last : Support for ReGIS graphics added (keyword `VT125`). You can now undust this good-old VT330+ console and put it back to work again. Thanks to PGPLOT.
  - Support for 1D data added (keyword `ONEDimensional`). It is now possible to calculate this long-sought $[0v0]$ Patterson projection function for your 2 MDalton multi-(protein-nucleic-acid) complex.
- Updated documentation (postscript and html), added man page.
- Tidy-up in the hope that this will be a long-lived release.

1.5 Version α, Releases 0.6, 0.7

- Minor correction for .mtz file handling.
- Keyword `GRAMaxContours` added to allow an explicit definition of the maximum number of contour lines that the program will draw (useful for avoiding wasting CPU time for drawing contours in the Patterson origin peak).

1.6 Version α, Release 0.5

- Keyword `PHASeless` added to allow 'free' phase refinement of reflections with low FOM.
- Keyword `SWITch` added to allow switching of a fom-weighted calculation to the phase-less mode.
- Keyword `PSOUt` added. When this keyword is present (and *GraphEnt* was compiled with PGPLOT support), the program will write out two postscript files (`CONVENTIONAL.ps` and `GRAPHENT.ps`) containing contour diagrams of the conventional and *GraphEnt* map sections that were displayed during the run.
- Keyword `SHOW` added to show evolution of map entropy during calculation.
- New default for Patterson function calculation : `LIMIT 0.5`.
- The F_{000} is now squared internally for a Patterson function calculation.
- Unconvincing attempt to introduce an "informative" prior for Patterson function calculations (a map with an origin peak, keyword `PRIOr`). Not very useful (and very broken for non-primitive lattices).

1.7 Version α, Release 0.4

- Keyword `TIME` added. This allows an explicit definition of how many minutes you are prepared to wait for *GraphEnt* to finish the calculation. After the specified period elapses, *GraphEnt* will write out the current map and will exit gracefully.
- Adjust frequency of writing out info during a calculation performed with the `VERBose` flag turned off (which is the default).
- Keywords `GRAFirst` and `GRALevel` added to allow explicit definition of the contouring levels for the graphics windows.
- The keyworded input files can now contain comment lines (whose first character must be `!`, `#` or `*`).
- More documentation written.

1.8 Version α, Release 0.3

- The `TARGet` keyword will now allow you to do the calculation even if the uniform prior is consistent with your data. Useful for anomalous Patterson map calculation.
- Increase reflection limit.
- More documentation.
- Keep on adding comments to the code.
- `VERBose` is no longer the default.

1.9 Version α, Release 0.2

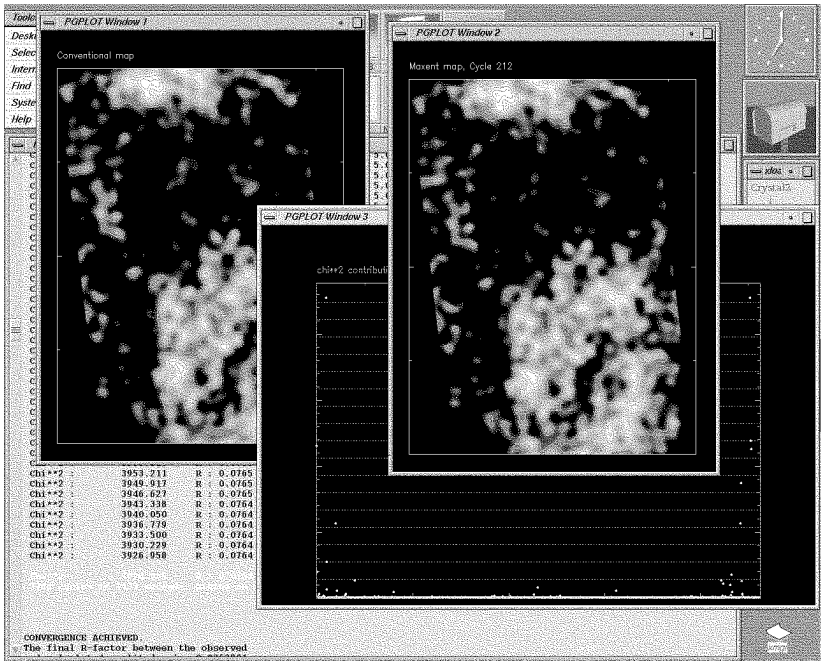
- Corrected error in the calculation of the standard deviation of isomorphous difference Patterson function coefficients.
- Keyword `LIMIt_sigmas` added to allow rejection of small isomorphous differences for the case isomorphous difference Patterson calculations.
- Keyword `F000` added to allow explicit definition of the F_{000}.
- When the F_{000} is not given, instead of making it dependent on the number of grid points in the map, it is explicitly approximated either through the volume of the unit cell (for phased syntheses), or through the largest F observed (Patterson syntheses).
- Some additional information is written out before scaling and exporting the map files.
- Few minor corrections, mainly about the messages written out during execution.
- More documentation written.
- Started (slowly and painfully) adding comments to the code.

Graphical MaxEnt

A maximum entropy program with graphics capabilities.

2 Overview

GraphEnt is an attempt to provide an automatic way for calculating the maximum entropy map that corresponds to a set of observations, while offering a useful graphical output of the current stage of the calculations. Here is a screenshot from a SGI workstation running *GraphEnt* :



The automation is mainly directed towards the macromolecular crystallographic community, with direct support for CCP4 mtz reflection files. Doing the calculation is as simple as `graphent 15 3.5 my_file.mtz` (for a GUI-based selection of the type of synthesis you want to calculate), `GraphEnt 15 4 my_file.mtz` (for a fully automated run using all data between 15 and 4Å resolution), or `GraphEnt h0l 10.0 3.5 my_file.mtz` (for calculating the *GraphEnt* map corresponding to the [010] projection using only data between 10.0 and 3.5Å resolution).

Having said that, it is still quite easy for the general crystallographic community to use the program. All is needed is an ASCII file containing unit cell dimensions and a list of reflections expanded to *P*1 :

```
AUTO  30.40 42.10 81.40 90.00 90.00 90.00  20
  -7  -5   1  559.200    8.700  138.578  0.4500
.........................................
   8   0   0  415.500    9.500  180.000  0.4000
```

Clearly, this black-box approach leads to loss of flexibility[1]. For this reason, *GraphEnt* also supports a more detailed form of input files, like this one[2] :

```
CELL            30.40 42.10 81.40 90.00 90.00 90.00
SPACEGROUP      20
GRID            128 64 64
PERM            3 1 2
FOM
VERBOSE
GRACYCLES       5
GRASECTION      14
GRAGRAYSCALE
LAMBDA          1.0
MAP_FORMAT      ccp4
REFL
  -7  -5   1  559.200    8.700  138.578  0.4500
  -7  -5   2  276.600   11.400  220.410  0.3500
 ..........................................
   7   5   3  532.700    8.900  349.320  0.4400
   8   0   0  415.500    9.500  180.000  0.4000
```

This document is organised as follows : Section 4 is for the impatient who just want to give it a try (and hopefully *not* get what they deserve). Section 5 gives a quick overview of the algorithm and a pictorial introduction to the advantages of the method. Section 6.1 discusses how to install and test the program. Sections 6.2, 6.3 and 6.4 present the supported modes of operation and the automated modes of execution. Section 6.5 presents the files produced by *GraphEnt*, and section 6.7 discusses the use of the normal probability plot produced by the program. Finally, section 7 contains the list of keywords recognised by *GraphEnt*.

3 A word of caution

I should warn you right from the beginning that the algorithm used in *GraphEnt* is neither the most stable nor the most efficient of those published[3] (but it is the one that is the easiest to code). Additionally, for those cases where the calculation includes a known figure-of-merit for the phase angles, *GraphEnt* is performing additional approximations which although I hope that are generally safe, they do not represent the best that can be achieved with the data.

I should also warn you that the amount of time that the calculation may require depends on the input data quality and there is no *a priori* guarantee that the given algorithm will converge even if given enough time. Having said that, a 262,144 (=128x64x32) pixels *GraphEnt* $mF_o \exp(i\phi_{best})$ map corresponding to a reasonably accurate data set could be calculated in less than 8 minutes of CPU time on a DEC Alpha 1200, a 524,288 (=128x128x32) pixels *GraphEnt* difference Patterson map for a loosy derivative (which makes the calculation easy) took only 46 seconds on the same machine, and a 2Å $(2mF_o - DF_c)\exp(i\phi_c)$ synthesis with 3,072,000 (=160x160x120) pixels took ≈40min. Finally, a 2-0.8Å $(2F_o - F_c)\exp(i\phi_c)$ synthesis with 9,437,184 (=192x256x192) pixels for a 4-α-helical bundle protein took only ≈13 minutes on a Pentium 800MHz (but the data were, of course, rather weak at this resolution range).

[1] For example, this input file with 7 columns of which the last one is always less or equal to 1.0, will be interpreted by *GraphEnt* as a request for a phased figure-of-merit-weighted Fourier synthesis, and there is very little you can do to change this in the automatic mode.

[2] Actually, *GraphEnt* only understands this detailed form of input. The automatic modes are wrappers which prepare an input file that the core program can interpret.

[3] See for example *Maximum Entropy and Bayesian Methods in Inverse Problems* (1985), edited by Smith, C.R. & Grandy, W.T., Jr., Dordrecht : Reidel.

4 A doer's guide.

4.1 Scenario A : Non-centrosymmetric space group - CCP4 available.

If you have the CCP4 suite of programs up-and-running on your machine and if the space group of your crystals is not centrosymmetric, all you need is a .mtz file containing your data. Then give something like : `graphent mydata.mtz` (to use all data), or, `graphent 15 3.5 mydata.mtz` (to define resolution limits), or, `graphent h0l 15 3.5 mydata.mtz` (to use only the *h*0*l* terms), and you will be presented with a smallish window from which you can choose the columns you wish to use for your intended synthesis. If the symbol `graphent` is not defined, please read the installation section of this manual (6.1).

Now, the difficult bit. The order with which you click on the columns to select them is important : you must select first the amplitude terms (with their standard deviation), followed (possibly) by phase angles, and finally figure-of-merit. If you choose columns with the wrong order the program will not cooparate. To see what column type combinations (and in what order) are supported by the program click on "HELP" in the dialog window.

Please note that not all conceivable types of syntheses are supported by the program. If your intended synthesis is not listed in the "HELP" screen of the dialog, you will have to use sftools to reduce the synthesis you have in mind to a recognisable (by *GraphEnt*) combination.

If all goes well, the dialog window will disapper (possibly after asking you to confirm a setting by just pressing 'y' or 'n'), and then you should see a section from the conventional map, followed (after some time) by a window showing the same section from the *GraphEnt* map. That second window will be updated as the calculation progresses, until convergence is achieved. When the calculation is over, a third window will appear containing a graph of the contributions of the various reflections to the χ^2 test. If you are calculating a difference Patterson synthesis, the first window you will see is the normal probability plot for your data.

Hit <RETURN> in the unix shell (where you started the calculation from) to exit the program. The directory should contain (in-between other files) the basic output from the program, namely the *GraphEnt* map : `maxent.map`.

4.2 Scenario B : Centrosymmetric space group - CCP4 available.

Unfortunately, the current version of the program sftools (from the CCP4) refuses to cooperate if the space group is centrosymmetric. The result is that *GraphEnt*'s GUI will not work. The way to proceed is to use CAD to select the columns you need for the calculation. Using the program in this mode is described in detail in section 6.3.

4.3 Scenario C : CCP4 not available

A detailed description of how to proceed is given in section 6.4. In summary, all you need is an ASCII file containing your data expanded to *P*1. Unfortunately, *GraphEnt* can not at present do the expansion for you.

PLEASE NOTE THAT IRRESPECTIVELY OF HOW YOU STARTED THE CALCULATION, WHAT THE PROGRAM REALLY READS AS INPUT IS A KEYWORDED ASCII FILE WITH THE NAME `MAXENT_AUTO.IN` WHICH IS LEFT BEHIND AFTER THE CALCULATION FINISHES.

IF YOU WANT TO MAKE USE OF ANY OF THE ADDITIONAL CAPABILITIES OF THE PROGRAM (AS DEFINED BY THE VARIOUS KEYWORDS, SEE PAGE 30), YOU WILL HAVE TO EDIT THIS FILE, MAKE THE CHANGES YOU WISH TO MAKE, AND RE-RUN *GraphEnt* WITH '`GraphEnt MAXENT_AUTO.IN`' OR '`graphent MAXENT_AUTO.IN`'

5 Methods, algorithms and a few examples

Ab initio determination of crystal structures based on a maximum entropy formalism has produced a wealth of papers debating the utility of the method, but very few actual determinations. On the other hand, a maximum entropy formalism aiming at the production of a "maximally non-committal" map is an almost standard method in fields of science like radioastronomy, but a rare exception in both X-ray crystallography and electron microscopy (or crystallography).

The calculation of a maximum entropy map when an atomic model can be built in a conventional $\mathrm{F}_o \exp(i\phi_c)$ (or $\mathrm{F}_o \exp(i\phi_o)$) synthesis, is probably a waste of CPU time. But when the map is the end product (low resolution electron or potential density maps, Patterson functions, etc.), the calculation of the MAXENT map can be more than useful (see section 5.2 for few examples).

5.1 The method

The question is : Given a set of incomplete and noisy data (say, $F_{o,\mathbf{h}}$ with its $\sigma(F_\mathbf{h})$ and $\phi_\mathbf{h}$), which map (of a large number of maps consistent with the observed data) is the one that will minimise the probability of misinterpreting it ? Stating the same problem in a different way, we could ask (i) which map (of the set of admissible maps) will only show features for which there is evidence in the data, or, (ii) which map makes the least assumptions about the data (especially the missing data, but also the distribution of errors in the observed).

Clearly, the $F_{o,\mathbf{h}} \exp(i\phi_\mathbf{h})$ synthesis is not the map we want : Not only we assume that all missing data have $F = 0$ (a rather improbable event), but also that $F_\mathbf{h} = F_{o,\mathbf{h}}, \forall \mathbf{h}$. Gull, S.F. & Daniell, G.J.[4], suggested that the map we need is the one for which the configurational entropy $-\sum_j m_j \log m_j$, where m_j is the density at the grid point j of the map, reaches a maximum. It is easy to see that $-\sum_j m_j \log m_j$ reaches a maximum when $m_j = e^{-1}, \forall j$, that is, when the map has a uniform density, and thus, contain no information. Maximising $-\sum_j m_j \log m_j$ subject to the constraint that the map is consistent with the observed data, gives the MAXENT map.

The consistency with the observed data is described in terms of the difference between the observed data and those calculated from a trial map, *weighted* by the standard deviation of the measurement. If $F_{c,\mathbf{h}}$ is the calculated value of the datum $\mathbf{h}$, $F_{o,\mathbf{h}}$ its observed value and $\sigma(F_\mathbf{h})$ the standard deviation of the observation, then the statistic

$$\sum_\mathbf{h} \frac{\mid F_{c,\mathbf{h}} - F_{o,\mathbf{h}} \mid^2}{\sigma(F_\mathbf{h})^2}$$

possesses a χ^2 distribution with an expected value equal to the number of data points. Maximising $-\sum_j m_j \log m_j$ subject to the constraint $\sum_\mathbf{h} \mid F_{c,\mathbf{h}} - F_{o,\mathbf{h}} \mid^2 / \sigma(F_\mathbf{h})^2 = n$, where n is the number of data points, gives the basic iteration formula :

$$m_\mathbf{j} = \exp\{-1 + \lambda \sum_\mathbf{h} \frac{F_{o,\mathbf{h}} - F_{c,\mathbf{h}}}{\sigma(F_\mathbf{h})^2} \exp(2\pi i \mathbf{jh})\}$$

Given $F_{o,\mathbf{h}}$, $\sigma(F_\mathbf{h})$ and an positive multiplier λ, this equation can determine the densities $m_\mathbf{j}$ on a map. The program *GraphEnt* applies this formula iteratively (starting from a uniform map) until convergence (as judged by the value of χ^2) is achieved. Although this algorithm is neither the most efficient nor the most stable, it is relatively easy to code and it leads (at least in the case of Patterson functions), to the same results as other, more complex algorithms[5].

5.2 Examples

The following examples illustrate some of the properties of the *GraphEnt* maps that I thought it would be worthwhile mentioning explicitly[6]. To further emphasize the generality of the method, I have included examples ranging from one-dimensional hypothetical structures giving 18 reflections in total, to a 0.8Å resolution synthesis for a small protein (with approximately 50000 unique reflections).

[4] Gull, S.F. & Daniell, G.J., (1978), *Nature*, **272**, 686–690.

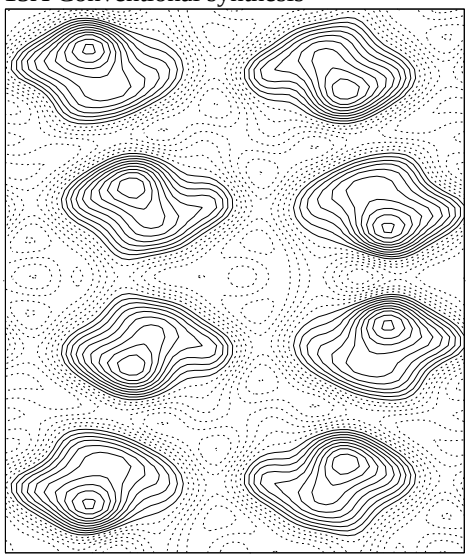
[5] Skilling, J. & Bryan, R.K., (1984), *Mon. Not. R. astr. Soc.*, **211**, 111–124.

[6] I would have hoped that with so much literature on the subject of maximum entropy, it would not have been necessary to illustrate the advantages of the method. Unfortunately, my experience is that maximum entropy maps are still being treated with some scepticism (if not scorn) by the community. So, there you go.

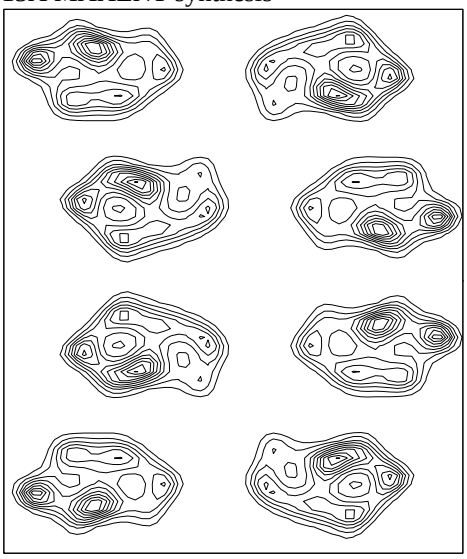
5.2.1 Positivity and improved resolution

Given that the maximum entropy map is the most uniform map consistent with the data, it is surprising how much more informative can be from the conventional Fourier synthesis, especially when accurate data are available. The following figure illustrates this point by comparing a conventional 15Å low resolution projection map with the corresponding *GraphEnt* map, and with two conventional maps calculated at higher (13 and 11Å) resolution[7].

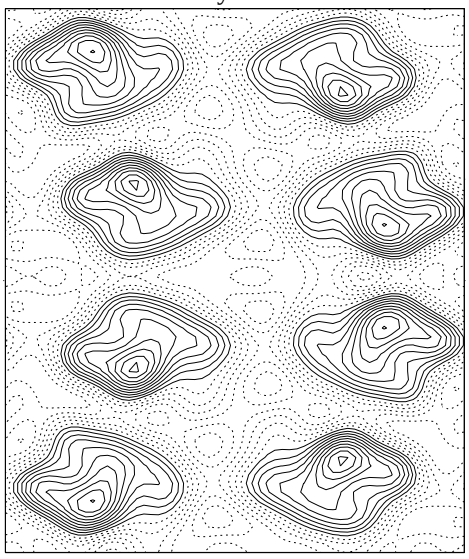
15A Conventional synthesis



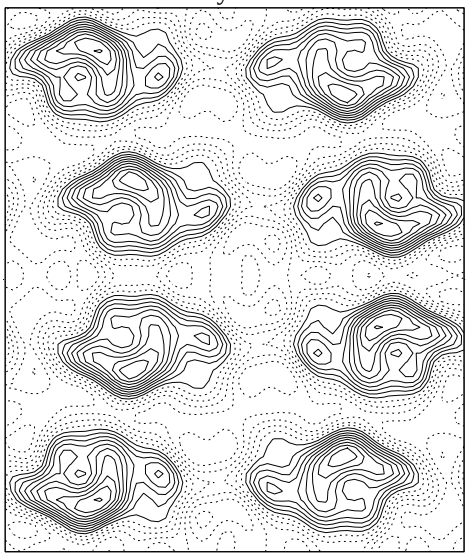
15A MAXENT synthesis



13A Conventional synthesis



11A Conventional synthesis



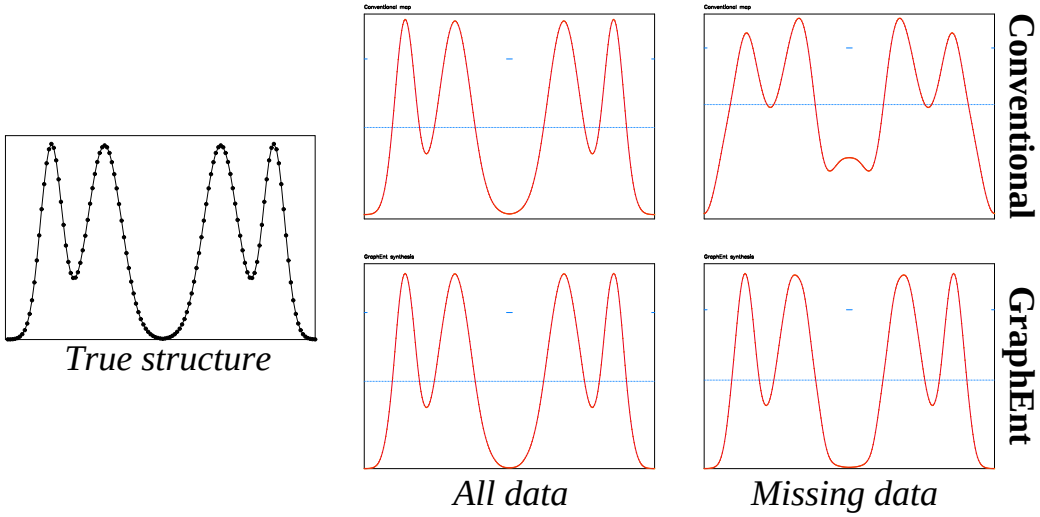
There are several things to note in these maps : The first is that the *GraphEnt* map is always positive with almost no detail in the background. This is clearly not the case with the Fourier syntheses, which have negative regions (dashed lines) and fine detail in the background which arise not only from the absence of the F_{000} term, but also from the series termination errors[8]. The second observation is that the peaks on the MAXENT map are better resolved, even when compared with the 11Å Fourier synthesis. This is not too surprising given that the Fourier transform of the maximum entropy map has non-zero amplitudes all the way to physical limits of the transform.

[7] The asymmetric unit of the pmg plane group consists of the projections of two lysozyme molecules related by a simple translation.

[8] In all maps shown, contours are plotted every 10% of the maximum density.

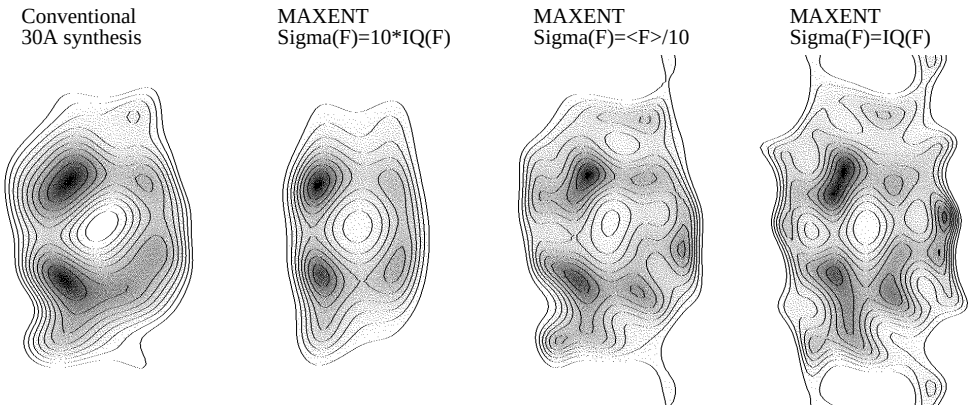
5.2.2 Insensitivity to missing data

The second example illustrates the behaviour of the method with respect to missing data. This example was constructed as follows : one-dimensional data were calculated from a hypothetical (1D) structure containing two Gaussians in the asymmetric unit of the $\wp m$ cell (where $\wp$ denotes the one-dimensional lattice). This hypothetical structure is shown on the far left panel in the figure below, and the data calculated by Fourier transforming this structure only included 18 strong reflections. The middle column of graphs shows the conventional and *GraphEnt* syntheses that were obtained when all these 18 reflections were included in the calculation (and, of course, both are essentially identical with the starting structure). When the calculation was repeated with 6 reflections missing from the data set, the conventional map (top, right-hand corner graph) was far from ideal : a new peak appears at $x = 0.5$, and the relative heights of the two Gaussians are no longer the same. In sharp contrast, the *GraphEnt* map (lower, right-hand side graph) is almost identical with the synthesis calculated with all data (and with the correct structure).



5.2.3 Sensitivity to the accuracy of the estimated standard deviations

This example is based on real (electron microscopy) data and shows the importance of having reasonably accurate estimates of the errors present in the data. The left-hand-side panel on the figure that follows is the conventional 30Å projection of photosystem II (courtesy Dr Andreas Holzenburg). The other three panels show *GraphEnt* maps which were calculated with standard deviations ranging from grossly overestimated (second from the left) to seriously underestimated (right hand side panel). Clearly, overestimating the standard deviations is no harm : although the final map will not be the best that can be done with the data, it will not be possible to misinterpret it. Underestimating the standard deviations, on the other hand, can lead to serious problems : the MAXENT algorithm will be “fitting” noise instead of real signal and the final map will contain fine structure not required by the data. Misinterpreting such a *GraphEnt* map should present no problems.

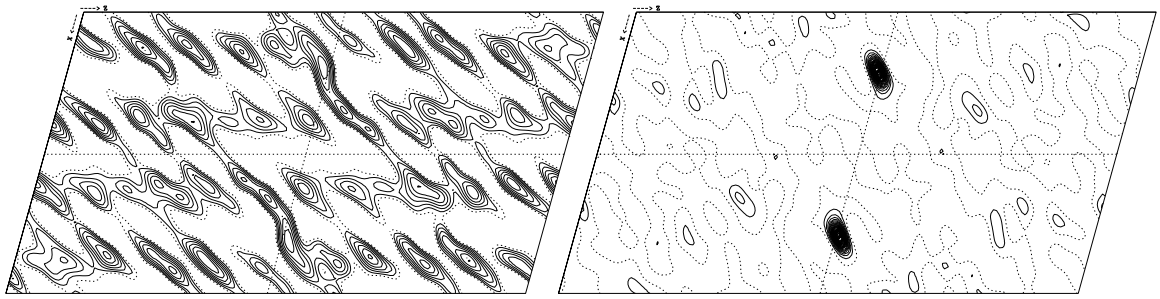


It is worth mentioning on passing that most data processing programs will produce raw data with underestimated standard deviations (especially for weak reflections). The solution is, of course, to calculate a normal probability plot of the form $(I_{obs} - < I >)/\sigma(I)$ and confirm that it has mean 0 and variance 1. I should also mention here that the greatest problem with incorrectly estimated standard deviations appears to come from the electron microscopy field : the majority of the data sets that I have come across tend to have almost constant average values of $F/\sigma(F)$ throughout the resolution range. An example of what *GraphEnt* would do in such cases is presented in section 9.8.

5.2.4 Insensitivity to outliers

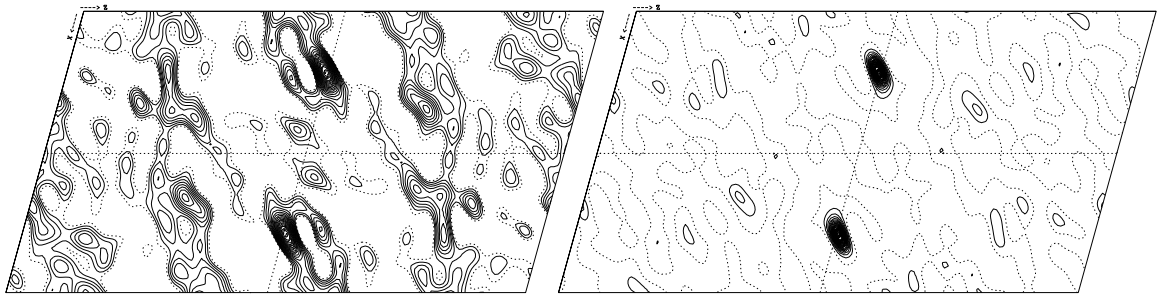
This example shows results from an anomalous Patterson function calculation using data collected from a horse heart myoglobin crystal. The data were collected with CuK_α radiation and the anomalous signal comes from the iron atom of heme (with $\Delta f''_{\mathrm{Fe,CuK}\alpha} = 3.2e^-$). To make the example more realistic we only used data between 20 and 3Å resolution, and we simulated the presence of outliers in the data by multiplying the amplitude (ΔF_{ano}) and standard deviation ($\sigma(\Delta F_{ano})$) of three randomly chosen strong reflections by a factor of 3.0.

A comparison of the Harker sections ($v = 1/2$) from the conventional and *GraphEnt* maps (shown in the figure that follows) is rather striking : The presence of outliers in the data has completely wiped-out the signal from the conventional map (left-hand-side panel), leaving behind the only too familiar to macromolecular crystallographers chequer-board appearance. In sharp contrast, the *GraphEnt* map resembles more a map calculated with hypothetical error-free data than an anomalous Patterson function calculated with real data (both maps are contoured with the dashed contour at the mean, and then every 0.5 rmsd of the whole map).



5.2.5 Insensitivity to noise

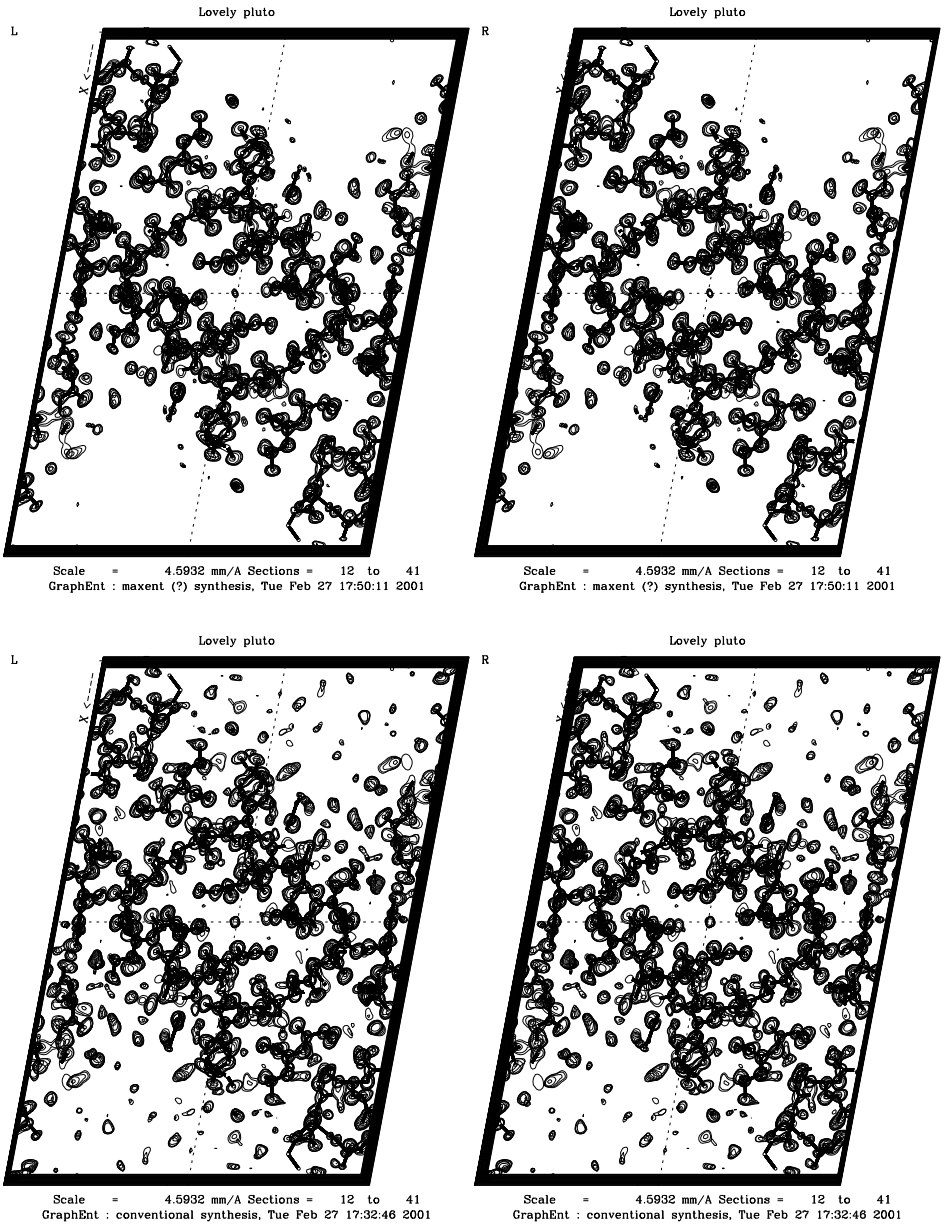
The following figure compares the conventional (left) and *GraphEnt* (right) map at the section $v = 1/2$ of the same 20–3Å anomalous Patterson function as for the previous example, but this time after the outliers have been removed from the data set (but only for the calculation of the conventional synthesis, the *GraphEnt* map still includes the outliers). As it is obvious, the *GraphEnt* map is rather insensitive to the presence of random (white) noise in the data, but this is definitely not so for the conventional Fourier synthesis.



5.2.6 Insensitivity to series termination errors & noise.

The following two figures compare a 4.6Å thick stack of sections from a 4.0-0.8Å $(2F_o - F_c)\exp(i\phi_c)$ *GraphEnt* (top) and conventional (lower) maps for a 4-α-helical bundle protein at the final stage of its refinement ($R = 0.102$).

Both maps are contoured at 1.3σ above their respective mean densities.



As it is obvious, the conventional map suffers from quite appreciable series termination errors and contains a number of peaks (and continuous features) that are not required by the data. If you think that this is understandable (given that the low resolution cutoff was only 4Å), re-consider : the low resolution data that have been excluded from the calculation are only 204 reflections out of a total of 48774 reflections (ie less than 0.4% of the total number of reflections).

5.2.7 Sensitivity to the value of the F000 term.

This is a rather important subject (and the most common source of problems with the program). For this reason, a separate section has been devoted to discussing the matter in detail (section 8.1, page 37).

6 Using the program

6.1 Installation guide

GraphEnt is distributed both as pre-build executables for SGI, Linux, Solaris & DEC Alpha OSF machines, and as source code for the other machines.

6.1.1 Using the pre-compiled executables

If you have root previleges, you can simply find the correct (for your architecture) executable, move it to a directory in the users' PATH and give the name `GraphEnt`. Do not forget to make a symbolic link with the name `graphent` pointing to the same executable. If you do not have PGPLOT installed (obtainable from `http://-astro.caltech.edu/~tjp/pgplot/`), then the users should also define an environmental variable `PGPLOT_DIR` pointing to the directory that contains the two essential PGPLOT files that I also include (`grfont.dat` and `pgxwin_server`).

If you want to install the program in your area, then go to the `GraphEnt/bin/my_arch` directory, uncompress a suitable executable, make a symbolic link with `ln -s ./myexecutable ./graphent` and add something analogous to the following three lines in your .cshrc file (assuming that you unpacked *GraphEnt* in your top directory) :

```
alias GraphEnt /usr/people/<username>/GraphEnt/bin/my_arch/<my_uncompressed_executable>
alias graphent /usr/people/<username>/GraphEnt/bin/my_arch/graphent
setenv PGPLOT_DIR /usr/people/<username>/GraphEnt/bin/my_arch
```

6.1.2 Building from source

The minimum requirement for building *GraphEnt* is that you have a C compiler and the the FFTW libraries compiled in the single precision mode (FFTW can be obtained from `http://www.fftw.org/`). If you want .mtz support you also need a fortran compiler and the CCP4 library, and for graphics support you need the PGPLOT library (from `http://astro.caltech.edu/~tjp/pgplot/`). To build an executable without CCP4 or graphics support, the following should suffice (assuming you are in the `GraphEnt/src/` directory) :

```
cc <my_optimisation_flags> -DNOCCP4 -DNOPGPLOT GraphEnt.c -o GraphEnt -lsrfftw -lsfftw -lm
```

which assumes that you had built FFTW with the `--enable-float --enable-type-prefix` options, and that the libraries are in the `ld`'s search path (add a `-L/my_lib/dir/fftw/` flag to `cc` if they are not).

If you have the CCP4 and PGPLOT libraries, and these are located somewhere in `ld`'s search path, you can build the *GraphEnt* executable by giving :

```
cc <my_optimisation_flags> -c GraphEnt.c
f77 <my_optimisation_flags> GraphEnt.o -lsrfftw -lsfftw -lccp4 -lcpgplot -lpgplot -lX11 -lm
mv a.out GraphEnt
```

where `<my_optimisation_flags>` correspond to the highest safe level of optimisation supported by your compilers (you can always check that everything is OK by comparing the results from the test files included in the `examples` directory). When you built PGPLOT do not forget to 'make cpg' as well (this will build the C wrapper library for PGPLOT).

For specific examples as to how the executables were built on the various supported machines, see the `OREADME` files in the various `GraphEnt/bin` subdirectories.

This way of linking *GraphEnt* (ie with separate calls to cc & f77) will not work on DEC (Compaq) machines. The way to compile and link (in one step) is described in the file `GraphEnt/bin/OSF/OREADME`.

Building a useable executable on non-Unix machines is expected to be rather more challenging, with the first challenge being to build the FFTW library. If you manage to build FFTW, then you might as well give it a try with *GraphEnt* but I believe that you should aim for a "simple" executable (no CCP4 or PGPLOT). If you are getting error messages during compilation, try adding a `-DVMS` flag in the `cc` step above. If the problems persist, please do send me a mail message, but do not expect too much.

6.1.3 Testing the executable

Assuming that the executable is in your PATH, and that its name is *GraphEnt*, go to the `/my_dirs/GraphEnt/-examples/` directory, make your window at least 125 characters wide, and type `GraphEnt Myoglobin_anom_-Patt.in`. What you should see on your terminal should be similar to this :

```
-------------------------------------------------------------------------------------------------------------------------

                                  ###   ###                  #######           #
                                   ##   ##                    #    #           #
                                   # # # #   ####   ### ###    #    # ## ###   ######
                                   # # # #  #    #   #   #     #  #    ##   #    #
                                   #  #  #       #    # #      ####    #    #    #
                                   #  #  #   #####     #       #  #    #    #    #
                                   #     #  #    #    # #      #    #  #    #    #
                                   #     #  #    #   #   #     #    #  #    #    #   #
                                  ###   ###  #### # ### ###   ####### ###  ###    ###

                                              Beta, Release 1.1

                                       Gull, S.F. & Daniell, G.J. (1978), Nature, 272, 686-690
                                             Collins, D.M. (1982), Nature, 298, 49-51

                                                          NMG

-------------------------------------------------------------------------------------------------------------------------

Keyword             CELL : Cell dimensions set to  35.84  28.89  63.57  90.00 105.47  90.00
Keyword             F000 : F000 known and set to 30.000000.
Keyword         GRAFIRST : first contour at mean + 1.000000 rmsd.
Keyword         GRALEVEL : contouring interval set to 1.000000 rmsd.
Keyword           TARGET : target value for chi^2 set to 100.00
Keyword           LAMBDA : initial value for lambda set to 40000.000000
Keyword          VERBOSE : verbose mode set.
Keyword       GRASECTION : Plot section 0.500000 (fractional coordinates).
Keyword            LIMIT : exclude reflections with |F|/sig(|F|)<0.500000.
Keyword       SPACEGROUP : space group number set to 1
Keyword       MAP_FORMAT : CCP4 map file selected.
Keyword        PATTERSON : Patterson map run [h k l F sig(F)].
Keyword      PERMUTATION : Permutation set to 3 1 2
Keyword             GRID : Grid set to   160    96    56
Keyword        GRACYCLES : Plot every 5 cycles.
Keyword    GRATWOWINDOWS : Will keep conventional map plot.
Keyword      REFLECTIONS : start reading reflections.
2274 reflections rejected due to sigma cutoff (Keyword LIMIt)
-------------------------------------------------------------------------------------------------------------------------

   Added reflection   -2   +6   +0        +42.336       +40.730     -0.0
   Added reflection   -2   -6   +0        +42.336       +40.730     -0.0
   Added reflection   -3   +8   +0       +229.620      +406.417     -0.0
   Added reflection   -3   +7   +0        +23.072       +45.970     -0.0
   Added reflection   -3   -7   +0        +23.072       +45.970     -0.0
   Added reflection   -3   -8   +0       +229.620      +406.417     -0.0
   Added reflection   -4   +7   +0       +341.880      +424.920     -0.0
   Added reflection   -4   +6   +0       +152.727      +194.618     -0.0
   Added reflection   -4   -6   +0       +152.727      +194.618     -0.0
   Added reflection   -4   -7   +0       +341.880      +424.920     -0.0
   Added reflection   -5   +6   +0        +37.322       +72.535     -0.0
   Added reflection   -5   -6   +0        +37.322       +72.535     -0.0
   Added reflection   -8   +6   +0        +64.509       +93.693     -0.0
   Added reflection   -8   -6   +0        +64.509       +93.693     -0.0
   Added reflection  -10   +4   +0       +113.128      +167.607     -0.0
   Added reflection  -10   -4   +0       +113.128      +167.607     -0.0

 - Square F000 for Patterson calculation, F000=900.000000.
 - Initial lambda is 40000.000
 - Verbose output requested.
 - Lambda linearly depended on number of cycles.
 - Grid (along fast, medium and slow)   160     96     56
 - Axes permutation is fast-Z, medium-X, slow-Y


 - Data structures require 19.9 MBytes of memory.
 - About to allocate memory.
 - FFTW is learning how to do FFTs  ...
 - FFTW learned how to do FFTs.
 - Saving FFTW's wisdom file ...
     There are 636 observations with <F>=    140.28
 - Loading data on arrays ...
 - Do not panic yet ...
 - About to calculate conventional Fourier transform of data
 - Write out conventional Fourier transform (file conventional.map)
 - Sum and average of densities are     -19.0625 and      -0.0000.
 - Scale factor applied :      0.00574329.
 - Map entropy is undefined (negative values are present).


  (Q)QOPEN: file opened on unit  1      Status: UNKNOWN
 Logical Name: conventional.map      Filename: conventional.map


   File name for output map file on unit   1 : conventional.map
      logical name conventional.map


 FORMATTED      OLD     file opened on unit  24
 Logical name: SYMOP, Filename: /usr/local/ccp4/lib/data/symop.lib


   Minimum density in map  =     -262.72626   Maximum density         =      999.00000
   Mean density            =        0.00000
   Rms deviation from mean =       58.00461
```

```
------------------------------------------------------------------------------------------------------------------------

 - MAXENT starts here


 Chi**2 :         492.602     R : 1.0000       Lambda :  40000.00000       Nobs :     620
 Chi**2 :         485.034     R : 0.9972       Lambda :  40000.00000       Nobs :     620
 Chi**2 :         477.884     R : 0.9942       Lambda :  40800.00000       Nobs :     620
 Chi**2 :         470.680     R : 0.9911       Lambda :  41616.00000       Nobs :     620
 Chi**2 :         463.181     R : 0.9877       Lambda :  42448.32000       Nobs :     620
 Chi**2 :         455.133     R : 0.9837       Lambda :  43297.28640       Nobs :     620
 Chi**2 :         446.211     R : 0.9791       Lambda :  44163.23213       Nobs :     620
 Chi**2 :         435.969     R : 0.9735       Lambda :  45046.49677       Nobs :     620
 Chi**2 :         423.800     R : 0.9663       Lambda :  45947.42671       Nobs :     620
 Chi**2 :         408.912     R : 0.9568       Lambda :  46866.37524       Nobs :     620
 Chi**2 :         390.377     R : 0.9441       Lambda :  47803.70274       Nobs :     620
 Chi**2 :         367.352     R : 0.9270       Lambda :  48759.77680       Nobs :     620
 Chi**2 :         339.573     R : 0.9045       Lambda :  49734.97234       Nobs :     620
 Chi**2 :         308.114     R : 0.8762       Lambda :  50729.67178       Nobs :     620
 Chi**2 :         275.884     R : 0.8436       Lambda :  51744.26522       Nobs :     620
 Chi**2 :         246.902     R : 0.8105       Lambda :  52779.15052       Nobs :     620
 Chi**2 :         224.139     R : 0.7802       Lambda :  53834.73353       Nobs :     620
 Chi**2 :         207.880     R : 0.7553       Lambda :  54911.42820       Nobs :     620
 Chi**2 :         196.440     R : 0.7363       Lambda :  56009.65677       Nobs :     620
 Chi**2 :         187.910     R : 0.7225       Lambda :  57129.84990       Nobs :     620
 Chi**2 :         181.006     R : 0.7118       Lambda :  58272.44690       Nobs :     620
 Chi**2 :         175.019     R : 0.7033       Lambda :  59437.89584       Nobs :     620
 Chi**2 :         169.590     R : 0.6960       Lambda :  60626.65376       Nobs :     620
 Chi**2 :         164.540     R : 0.6897       Lambda :  61839.18683       Nobs :     620
 Chi**2 :         159.778     R : 0.6839       Lambda :  63075.97057       Nobs :     620
 Chi**2 :         155.255     R : 0.6784       Lambda :  64337.48998       Nobs :     620
 Chi**2 :         150.940     R : 0.6732       Lambda :  65624.23978       Nobs :     620
 Chi**2 :         146.815     R : 0.6682       Lambda :  66936.72457       Nobs :     620
 Chi**2 :         142.864     R : 0.6634       Lambda :  68275.45907       Nobs :     620
 Chi**2 :         139.076     R : 0.6586       Lambda :  69640.96825       Nobs :     620
 Chi**2 :         135.441     R : 0.6540       Lambda :  71033.78761       Nobs :     620
 Chi**2 :         131.950     R : 0.6494       Lambda :  72454.46336       Nobs :     620
 Chi**2 :         128.597     R : 0.6450       Lambda :  73903.55263       Nobs :     620
 Chi**2 :         125.375     R : 0.6406       Lambda :  75381.62368       Nobs :     620
 Chi**2 :         122.279     R : 0.6363       Lambda :  76889.25616       Nobs :     620
 Chi**2 :         119.304     R : 0.6321       Lambda :  78427.04128       Nobs :     620
 Chi**2 :         116.444     R : 0.6279       Lambda :  79995.58211       Nobs :     620
 Chi**2 :         113.695     R : 0.6239       Lambda :  81595.49375       Nobs :     620
 Chi**2 :         111.051     R : 0.6200       Lambda :  83227.40362       Nobs :     620
 Chi**2 :         108.508     R : 0.6161       Lambda :  84891.95170       Nobs :     620
 Chi**2 :         106.060     R : 0.6122       Lambda :  86589.79073       Nobs :     620
 Chi**2 :         103.702     R : 0.6084       Lambda :  88321.58654       Nobs :     620
 Chi**2 :         101.429     R : 0.6047       Lambda :  90088.01828       Nobs :     620
 Chi**2 :          99.237     R : 0.6010       Lambda :  91889.77864       Nobs :     620

 Chi**2 :          99.237     R : 0.6010       Lambda :  91889.77864       Nobs :     620
 43 cycles in 36 seconds, giving an average of 0.837 seconds per cycle.
------------------------------------------------------------------------------------------------------------------------



CONVERGENCE ACHIEVED.
The final R-factor between the observed
and calculated amplitudes is  0.6010417


 - Write out maxent.map map file.
 - Sum and average of densities are    1101.5133 and       0.0013.
 - Scale factor applied :    173.43777466.
 - Map entropy is 13.157804.

 (Q)QOPEN status changed from NEW to UNKNOWN for maxent.map

  (Q)QOPEN: file opened on unit  1      Status: UNKNOWN
 Logical Name: maxent.map      Filename: maxent.map


   File name for output map file on unit   1 : maxent.map
      logical name maxent.map


 FORMATTED      OLD     file opened on unit  24
 Logical name: SYMOP, Filename: /usr/local/ccp4/lib/data/symop.lib


   Minimum density in map  =        0.00242   Maximum density         =      998.99994
   Mean density            =        0.22210
   Rms deviation from mean =        2.12448



 - Calculating table of largest contributions to chi**2
 - Will write out all reflections contributing to chi^2 by more than 10.000000 times the rmsd of all contributions.



Normal termination ? (45 seconds)
```

At this point, and if you have graphics support, you should also have three graphics windows on your monitor, one on top of the other (but you can move them of course). Hit <RETURN> in the terminal window to finish with the run and close the graphics windows.

Note that the exect numerical results you are getting depend on your computer's hardware and the compiler flags used, and so, you should not be expecting to see exactly the numbers shown in the example above.

6.2 Supported crystallographic calculations

GraphEnt automatically recognises the following types of syntheses :

- F^2 Patterson synthesis, defined by $h, k, l, F, \sigma(F)$
- ΔF^2 difference Patterson synthesis, defined by $h, k, l, F_1, \sigma(F_1), F_2, \sigma(F_2)$. The function calculated is the one commonly used for isomorphous difference Patterson maps with an amplitude of $F = (F_1 - F_2)^2$ and a standard deviation $\sigma(F) = 2\sqrt{F}\sqrt{\sigma(F_1)^2 + \sigma(F_2)^2} + \sigma(F_1)^2 + \sigma(F_2)^2$.
- Phased Fourier synthesis without FOM, defined by $h, k, l, F, \sigma(F), \phi$
- Phased Fourier synthesis with FOM, defined by $h, k, l, F, \sigma(F), \phi, FOM$

All other types of syntheses that can be reduced to any of the above are also supported but the reduction step is a user's responsibility. For example, to calculate a $(2F_o - F_c)\exp(i\phi_c)$ map you would have to prepare a column containing the $(2F_o - F_c)$ term by yourself, and give the program six columns of the type $h, k, l, (2F_o - F_c), \sigma(2F_o - F_c), \phi_c$. For CCP4 users this is easily (and interactively) achieved with the program `sftools`[9].

6.3 The `.mtz` wrapper

GraphEnt offers limited capabilities for a completely automated run with only input the name of a .mtz file. The major limitation is that you do not assign columns or chose a type of calculation. What happens is that *GraphEnt* will open your .mtz file and read the list of column types. If it finds a recognisable set of column types it will simply go ahead and do the calculation. If what *GraphEnt* decides to do is not what you wanted, simply use `mtzutils` or `sftools` to select, or create the column types that *GraphEnt* expects for your type of calculation.

This is the list of column types (**order is important**) and corresponding calculation that *GraphEnt* will perform :

Column types	Action performed
HHHFQPW	Assumed to be $h, k, l, F, \sigma(F), \phi, FOM$. Synthesis will be $mF\exp(i\phi)$
HHHFQP	Assumed to be $h, k, l, F, \sigma(F), \phi$. Synthesis will be $F\exp(i\phi)$
HHHFQFQDQ	Assumed to be $h, k, l, F_P, \sigma(F_P), F_{PH}, \sigma(F_{PH}), \Delta F_{ano}, \sigma(\Delta F_{ano})$ for a derivative. Synthesis will be a $(F_P - F_{PH})^2$ isomorphous difference Patterson function. An input file for the anomalous synthesis will also be prepared (which can be used as input for a second run).
HHHFQFQ	Same as the previous one, but no additional input file for the anomalous part is prepared.
HHHDQ	Assumed to be $h, k, l, \Delta F_{ano}, \sigma(\Delta F_{ano})$. Synthesis will be an anomalous difference Patterson function (ΔF^2_{ano}).
HHHFQ	Assumed to be $h, k, l, F, \sigma(F)$. Synthesis will be the Patterson function.

Please note that *GraphEnt* will only check the column types immediately after the indeces, and if a match is found the rest of the columns will be ignored. Furthermore, the search is performed in the order shown in the table and the calculation performed is the first matching. What this means is that if your column types are HHHFQP, *GraphEnt* will go for a *phased* synthesis no matter what you may wanted to do. If your intention was to calculate a Patterson function with Fs and $\sigma(F)$s, you will have to use `mtzutils` or `sftools` to remove the column containing the phases.

If the column types of your .mtz file are in the right order, just give `GraphEnt <my_file.mtz>` for a run that will use all data present in the file, or `GraphEnt 15.0 3.5 <my_file.mtz>` to use only data between 15 and 3.5Å resolution. If what you are calculating is isomorphous difference Patterson functions for a derivative, and your space group has centric zones of the type $h0l, hk0, 0kl$, you may as well try something like `GraphEnt h0l 15.0 3.5 <my_file.mtz>` to calculate the [010] Patterson projection. *GraphEnt* will also recognise and use all zone selections recognised by mtzutils (ie H00, 0K0, 00L, HH0, -HH0, HHH, HK0, 0KL, H0L and HHL).

[9] I should add here that for some of the more complex syntheses, the most difficult part of setting-up the calculation appears to be the propagation of errors, ie calculating correctly the standard deviation of the amplitude terms.

GraphEnt is always performing the calculation in space group *P*1. To avoid unnecessary repetition, the program calls `CAD` (and possibly `MTZUTILS`) from the CCP4 distribution to expand the data to *P*1. This means that *GraphEnt* will fail if CCP4 is not correctly set-up or if the various symbols are not defined (especially the `CLIBD` variable, you can check its presence with `setenv | grep 'CLIBD'`).

NOTE WELL : Maximum entropy maps may well predict non-zero amplitudes for data beyond the high resolution limit of your input data set (thus giving —for good data— a degree of "super-resolution"[10]). For this reason the grid size that *GraphEnt* uses is significantly larger than that used by the conventional FFT (and even this may not be large enough). The implication is that if you want to do a calculation using all your data to 2Å resolution, you are better-off submitting a batch job for the night instead of trying to do it interactively. In addition, and because *GraphEnt* is doing the calculation in *P*1, the higher the symmetry, the larger the grid, the slower *GraphEnt* will be. As a last precaution, I should add that I have never performed a calculation with more than 9437184 (=192x256x192) pixels.

6.4 The `AUTO` wrapper.

This is a wrapper to minimise the amount of input to the program. You simply create an ASCII input file like this :

```
AUTO       94.14900        24.17000        64.31901        90.00000       130.36700       90.00000  1
  -22     0     6          50.32293         2.43270        67.66310         2.33278
  -22     0     7         148.78082         3.24875       139.47423         2.21939
  -22     0     8         189.90724         3.95280       210.31883         2.26664
  -22     0     9         104.71589         2.51189       130.89922         1.64204
  -22     0    10         226.21014         4.65611       245.38899         2.75075
  -22     0    11          72.67593         2.26654        82.38221         1.58337
  -22     0    12         337.86499         6.89991       350.74295         3.45005
  -22     0    13          19.86930         6.24350        48.52971         1.82860
  -22     0    14         173.24773         3.65465       163.61771         2.30839
  ..........................................................................
   14     0     3         250.23837         4.61909       307.68475         4.29308
   16     0     0         760.94031        15.55367       687.91992         8.57631
   16     0     1         350.09189         6.37660       328.42670         3.76660
```

and you give `GraphEnt <my_file.in>`. For this wrapper to work, the first line must begin with the keyword `AUTO` followed by unit cell dimensions and space group number, and then a list of reflections. Depending on the number of columns in the file, *GraphEnt* will perform one of the following calculations :

Columns	Action performed
5	Assumed to be $h, k, l, F, \sigma(F)$. Synthesis will be the Patterson function.
6	Assumed to be $h, k, l, F, \sigma(F), \phi$. Synthesis will be $F \exp(i\phi)$
7	If all values of the the last column is less or equal to 1.0, it is assumed that the columns are $h, k, l, F, \sigma(F), \phi, FOM$. Synthesis will be $mF \exp(i\phi)$
7	If the last column contains values greater than 1.0, it is assumed that the columns are $h, k, l, F_P, \sigma(F_P), F_{PH}, \sigma(F_{PH})$ for a derivative. Synthesis will be a $(F_P - F_{PH})^2$ isomorphous difference Patterson function.

NOTE (1) : The input file must contain your data expanded to *P*1. *GraphEnt* will not expand the data for you.

NOTE (2) : In the case of AUTO-labelled files you can not specify resolution limits or projections.

6.5 Program output

Depending on the type of calculation performed, *GraphEnt* can create up to four graphics windows and leave behind (after the calculation is over) up to 10 files. The number of files created depends not only on the type of calculation performed but also on the availability of PGPLOT with your executable[11].

[10] Hopefully, and if the F_{000} and standard deviations of your data are correctly estimated, the maxent map should only show you the degree of resolution that is required by your data. The fact that it may look sharper than the conventional map is not due to a "super-resolution" effect arising from the maxent algorithm, but because the conventional transform is not optimal for your problem. To make this clear, if you had a 100% complete and noise-free data set extending to infinite resolution, then the conventional and maxent map would be identical. Actually, if you start using maxent systematically, you will note that the conventional and maxent maps start looking quite similar under considerably less stringent conditions.

[11] The reason is that with PGPLOT it is relatively easy to save the contents of a graphics window as a postscript file. *GraphEnt* is using this facility to save postscript versions of the normal probability plot and of the graph showing the contributions of each reflection to χ^2.

The four possible graphics windows are :

1. A window showing a section from the conventional Fourier syntesis (the section plotted can be selected with the keyword GRASection, see page 32).
2. A window showing the same section from the MaxEnt map.
3. A normal probability plot of the input data (calculated only in the case of a difference Patterson calculation).
4. A scatter plot of the contribution of each reflection to the final value of χ^2.

A directory devoted to running *GraphEnt* may contain all these files after the end of the calculation (assuming that you started with just the file myfile.mtz) :

```
total 2440
-rw-r--r--   1 glykos    sys           68 Mar 27 18:19 CHIcontributions.dat
-rw-r--r--   1 glykos    sys        39223 Mar 27 18:19 CHIcontributions.ps
-rw-r--r--   1 glykos    sys        63767 Mar 27 18:14 CONVENTIONAL.ps
-rw-r--r--   1 glykos    sys        65022 Mar 27 18:18 GRAPHENT.ps
-rw-r--r--   1 glykos    sys        75097 Mar 27 18:14 MAXENT_AUTO.IN
-rw-r--r--   1 glykos    sys        73771 Mar 27 18:14 MAXENT_FROM_MTZ.in
-rw-r--r--   1 glykos    sys        38583 Mar 27 18:14 MAXENT_FROM_MTZ_ANOMALOUS.in
-rw-r--r--   1 glykos    sys        50092 Mar 27 18:14 MAXENT_SUM_PATT.in
-rw-r--r--   1 glykos    sys        16754 Mar 27 18:14 Normal_probability.ps
-rw-r--r--   1 glykos    sys         3740 Mar 27 18:14 Normplot_tails.dat
-rw-r--r--   1 glykos    sys       861264 Mar 27 18:14 conventional.map
-rw-r--r--   1 glykos    sys       861264 Mar 27 18:18 maxent.map
-rw-r--r--   1 glykos    sys       317920 Mar 27 18:14 myfile.mtz
```

The file myfile.mtz is where you started from, and the files conventional.map and maxent.map are the conventional and maxent map files containing the syntheses corresponding to your data (see below for a description of the supported map formats).

The files CONVENTIONAL.ps and GRAPHENT.ps are only produced if you have graphics support and contain postscript images of the conventional and *GraphEnt* map sections that were plotted on your monitor.

The files Normal_probability.ps and Normplot_tails.dat are produced only when you are performing a difference Patterson calculation. See page 23 for a more detailed discussion of the normal probability plot facility.

The file CHIcontributions.ps is only produced if you have graphics support and it contains the contribution of each reflection to the final value of χ^2. The ASCII file CHIcontributions.dat contains a list of reflections whose contribution to χ^2 is higher than 10σ above the mean contribution of all reflections. Depending on the case, these reflections may also be flagged as "suspect" (the 10σ limit can be changed with the keyword CHILimit, page 35). In conjunction with the keyword REJEct, some or all of these reflections can be excluded from a subsequent calculation (page 33).

Finally, the files that start with MAXENT_ are intermediate files produced during the interpretation of the .mtz file, or, in the case of MAXENT_FROM_MTZ_ANOMALOUS.in and MAXENT_SUM_PATT.in, files that can be used as input to *GraphEnt* for calculating an anomalous differences Patterson function, and a so-called "summed" difference Patterson (based on $F_H \approx \sqrt{\Delta F_{iso}^2 + \Delta F_{ano}^2}$). Please note that I have not used the MAXENT_SUM_PATT.in file extensively, so there might as well be some bugs lurking in there.

6.6 Map formats

The basic program output consists of two maps : the conventional synthesis (file conventional.map) and the MaxEnt synthesis (file maxent.map). The program can output maps in three formats : binary CCP4 map files, ascii CCP4 map files (NA4) which can be converted to binary files with the CCP4 program maptona4, and simple ASCII files for the non-CCP4 users. This last "format" is purposedly as structure-less as possible : the map is given as successive sections separated by a blank line with individual values scaled to a maximum of 999.0. Hopefully, this can be transported to numerous other programs with minimal user intervention.

```
-101.356  -93.902  -73.867  -45.791  -12.287 ...........   26.791  -12.532  -45.979  -73.995  -93.967
 -95.929  -85.706  -59.859  -28.781   -0.416 ...........   24.285   -0.661  -28.970  -59.988  -85.771
 -58.698  -48.410  -23.921    0.885   15.397 ...........   19.480   15.152    0.696  -24.049  -48.475
 ..........................................................................................
 ..........................................................................................
  58.074   39.867   -5.397  -53.678  -77.325 ........... -58.833  -77.571  -53.867   -5.526   39.803
  -7.705  -15.480  -33.664  -48.621  -44.073 ...........  -9.892  -44.319  -48.810  -33.792  -15.545
 -68.427  -67.131  -62.155  -49.809  -24.415 ...........  17.952  -24.661  -49.998  -62.283  -67.196


-103.449 -119.073 -102.959  -58.338   -2.439 ...........  11.988  -10.371  -21.613  -39.107  -69.976
 -96.769 -118.809 -104.169  -59.655   -7.288 ...........  15.538   12.007    9.218  -11.334  -52.477
 -59.809  -87.093  -77.942  -40.976   -0.372 ...........  10.221   30.511   44.319   30.797  -11.023
 ..........................................................................................
 ..........................................................................................
  45.125   35.353   12.693   -4.512   -4.562 ........... -102.067 -116.048  -80.695  -22.341   26.209
 -16.466  -24.609  -27.901  -16.405   10.548 ........... -46.626  -72.990  -62.263  -36.176  -17.877
 -73.315  -83.683  -73.321  -39.784    7.460 ...........  -7.622  -37.973  -44.584  -45.890  -55.999


-121.906 -163.951 -138.990  -60.994   19.728 ...........   8.199   19.824   35.737   18.103  -44.381
-107.299 -166.345 -154.607  -83.936   -5.668 ...........   0.903   35.736   68.483   55.700  -14.293
 -71.070 -141.078 -141.586  -81.543  -11.144 ........... -21.129   39.113   93.862   94.527   27.649
 ..........................................................................................
```

Map format defaults : Please note the following about format availability : the ASCII and NA4 map formats are always available (even if you compiled the program without the CCP4 library). The binary CCP4 map files are only available for executables that were prepared with CCP4 support (ie linked with the CCP4 library).

If *GraphEnt* was compiled without the CCP4 library, then the default output map format is ASCII. The way to produce a NA4 map file (which can then be converted to binary CCP4 map files), is to let *GraphEnt* start the actual calculation, then stop it (with <CTRL-C>), edit the `MAXENT_AUTO.IN` file and change the `MAP` keyword from `ASCII` to `NA4`. Then run *GraphEnt* again by giving `GraphEnt MAXENT_AUTO.IN`

If *GraphEnt* was compiled without the CCP4 library but CCP4 output is requested, the program will write out a NA4 map file.

6.7 The normal probability plot & how to use it.

When an isomorphous difference Patterson is calculated, *GraphEnt* will plot the normal probability diagram of the input data, together with a reference dotted line of gradient 1.0 and zero intercept[12]. The usage of the normal probability plots for accessing the usefulness (or otherwise) of a putative derivative is well documented and will not be discussed here (see Howell, P.L. & Smith, G.D. (1992), *J. Appl. Cryst.*, **25**, 81–86, and Abrahams, S.C. & Keve, E.T. (1971), *Acta Cryst.*, A**27**, 157–165). If you scaled your (macromolecular) data using the program `scaleit` from the CCP4 suite, then although you have not seen the plot, you have seen the variation of its gradient and intercept versus resolution (using the program `xloggraph` on the .log file written by `scaleit`). The reason for repeating the calculation here, is that the normal probability plot can also be used to select suspect data that do not fit an otherwise linear trend. The important thing is that the selection is not performed on the basis of just the magnitude of the difference (ie $||F_{PH}| - |F_P||$, as happens in `scaleit`), but on the basis of both the observed amplitudes and their standard deviations. The normal probability plot together with the "large contributions to χ^2" table (files `CHIcontributions.dat` and `CHIcontributions.ps`), which is produced after the calculation is over, should allow you to justifiably select outliers[13].

This is achieved as follows : *GraphEnt* will write out an ASCII file (named `Normplot_tails.dat` which contains the *hkl* indeces for all reflections that comprise the tails of the plot. These points are shown in the graphics window with a different colour. If some of these points deviate significantly from the rest of the plot,

[12] Plotting will be performed only if *GraphEnt* was compiled with graphics support (ie with PGPLOT, see section 6.1)). Even in the absence of PGPLOT, the normal probability plot will still be calculated, and the numbers will be written to an ASCII file which can be used as input to almost any plotting program (file `MAXENT_normal_prob_plot.dat`).

[13] As I understand it, the choice to treat as suspect (or even to reject) all reflections that give values of $||F_{PH}| - |F_P||$ more than `something` times the rmsd of isomorphous differences, is due to the inability of the conventional Fourier synthesis to take into account the standard deviations of the measurements. Let me give an example : suppose that for the 312 reflection, $|F_{P_{312}}| = 103$, $\sigma(|F_{P_{312}}|) = 14$, $|F_{PH_{312}}| = 183$, $\sigma(|F_{PH_{312}}|) = 120$, and assume for the sake of argument that the rmsd of the observed differences is 20 e^-. Then, we can ignore the fact that the measurement of $|F_{PH_{312}}|$ is loosy, and reject this reflection as "highly improbable". This is of course nonsense : the standard deviation of the difference $|F_{PH_{312}}| - |F_{P_{312}}|$ is 134 e^-, which means that with the observed difference of 80 e^- we can not even say at the 50% significance level that the amplitudes $|F_{P_{312}}|$ and $|F_{PH_{312}}|$ are indeed different. The trouble is that if you include the reflection in your Fourier synthesis, it will probably make a mess out of your map because in the case of the conventional synthesis you treat all differences as if having zero standard deviation. Needless to say that the maxent map not only is insensitive to such differences, but that you *should* actually avoid rejecting anothing until you are certain that for some reason the standard deviations are wrong.

then they are *candidates* for rejection (note that some deviation from linearity will always be present near the tails. What you are looking for is an outstanding deviation.)

You can then match what you see in the plot with what is written in the `Normplot_tails.dat`, decide which reflections to exclude, write their indeces in an ASCII file with the name `REJECT.HKL`, and then re-run the program using the `MAXENT_AUTO.IN` file after adding the keyword `REJECT` (see page 33). Because this sounds quite complicated, I will now give a detailed example to show how it works :

We start with just one .mtz file containing data for a putative derivative :

```
crystal2 ~/test
crystal2 ~/test d
total 260
-rw-r--r--   1 glykos   sys        262300 Dec 16 15:45 from_scaleit.mtz
crystal2 ~/test
crystal2 ~/test mtzdump hklin from_scaleit.mtz

 ###########################################################
 ###########################################################
 ###########################################################
 ### CCP PROGRAM SUITE: MTZDUMP     VERSION 3.5: 18/06/98##
 ###########################################################

.............

 OVERALL FILE STATISTICS for resolution range   0.001 -   0.245
 =====================
 Col Sort    Min    Max    Num      %     Mean     Mean   Resolution   Type Column
 num order              Missing complete          abs.   Low    High       label

   1 ASC    -46      35      0  100.00    -11.3     18.0  35.81   2.02   H  H
   2 NONE     0      11      0  100.00      4.0      4.0  35.81   2.02   H  K
   3 NONE     0      31      0  100.00     12.3     12.3  35.81   2.02   H  L
   4 NONE    4.4   902.0     3   99.96    92.65    92.65  35.81   2.02   F  FP
   5 NONE    0.6    26.2     3   99.96     3.34     3.34  35.81   2.02   Q  SIGFP
   6 NONE    8.7   956.3  3500   51.49   137.07   137.07  18.78   2.50   F  FPH
   7 NONE    1.2    41.5  3500   51.49     7.95     7.95  18.78   2.50   Q  SIGPH
   8 NONE  -73.2    72.3  3718   48.47     0.29     7.29  18.78   2.51   D  DPH
   9 NONE    0.0    66.8  3718   48.47    11.85    11.85  18.78   2.51   Q  SIGDPH

 No. of reflections used in FILE STATISTICS     7215

 LIST OF REFLECTIONS
 ===================

.............

 MTZDUMP:   Normal termination of mtzdump
 Times: User:       0.2s System:    0.1s Elapsed:    0:03
crystal2 ~/test
crystal2 ~/test
```

Then, we run *GraphEnt* on the centrosymmetric [010] projection :

```
crystal2 ~/test
crystal2 ~/test GraphEnt h0l 10 3 from_scaleit.mtz

-------------------------------------------------------------------------------------------------------------------------------

                              ###   ###                  #######           #
                               ##   ##                    #    #            #
                               # # # #   ####   ### ###    #    # ## ###   ######
                               # # # #  #    #   #   #     #  #    ##   #    #
                               #  #  #       #    # #      ####    #    #    #
                               #  #  #   #####     #       #  #    #    #    #
                               #     #  #    #    # #      #    #  #    #    #
                               #     #  #    #   #   #     #    #  #    #    #   #
                              ###   ###  #### # ### ###   ####### ###  ###    ###

                             Gull, S.F. & Daniell, G.J. (1978), Nature, 272, 686-690
                                   Collins, D.M. (1982), Nature, 298, 49-51

                                                NMG

-------------------------------------------------------------------------------------------------------------------------------

 - Assuming that input is a .mtz file. Interpreting ...
.............................................

 - Now trying lambda = 0.010000
 .........................................
 - Initial value for lambda set to 1000.000000

-------------------------------------------------------------------------------------------------------------------------------

 - MAXENT starts here

 Chi**2 :        1593.822     R : 1.0000       Lambda :   1000.00000       Nobs :     366
 Chi**2 :        1588.187     R : 0.9992       Lambda :   1000.00000       Nobs :     366
 ..................................................................................
 Chi**2 :         365.790     R : 0.5621       Lambda :    945.19320       Nobs :     366
 803 cycles in 74 seconds, giving an average of 0.092 seconds per cycle.
-------------------------------------------------------------------------------------------------------------------------------


CONVERGENCE ACHIEVED.
The final R-factor between the observed
and calculated amplitudes is  0.5621040

.....................................

Normal termination ? (100 seconds)
```

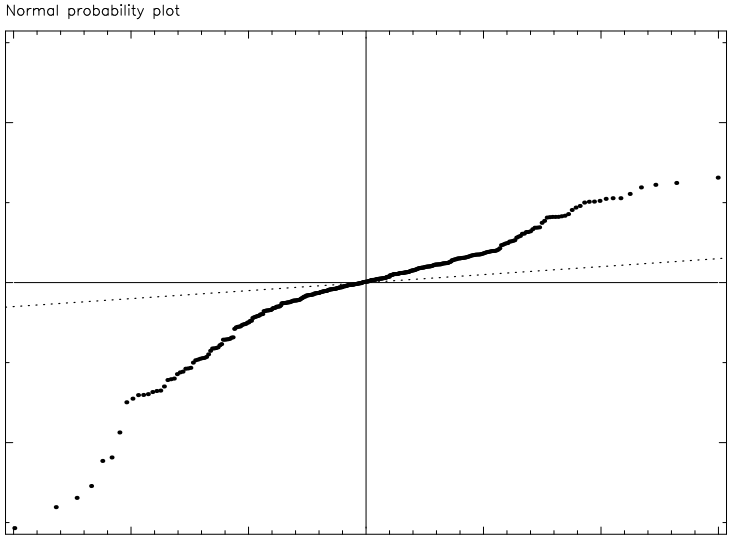
Now we have all these files :

```
crystal2 ~/test d
total 652
-rw-r--r--   1 glykos   sys            68 Dec 16 15:51 CHIcontributions.dat
-rw-r--r--   1 glykos   sys         37224 Dec 16 15:51 CHIcontributions.ps
-rw-r--r--   1 glykos   sys         31101 Dec 16 15:49 MAXENT_AUTO.IN
-rw-r--r--   1 glykos   sys         30595 Dec 16 15:48 MAXENT_FROM_MTZ.in
-rw-r--r--   1 glykos   sys           103 Dec 16 15:48 MAXENT_FROM_MTZ_ANOMALOUS.in
-rw-r--r--   1 glykos   sys         10365 Dec 16 15:48 Normal_probability.ps
-rw-r--r--   1 glykos   sys           825 Dec 16 15:48 Normplot_tails.dat
-rw-r--r--   1 glykos   sys        132176 Dec 16 15:50 conventional.map
-rw-r--r--   1 glykos   sys        262300 Dec 16 15:45 from_scaleit.mtz
-rw-r--r--   1 glykos   sys        132176 Dec 16 15:51 maxent.map
crystal2 ~/test
```

Both CHIcontributions.dat and Normplot_tails.dat point to problems with reflections 0,0,11 and -12,0,8 :

```
crystal2 ~/test
crystal2 ~/test
crystal2 ~/test more CHIcontributions.dat
    0    0   11          55.19882
  -12    0    8          59.04416
crystal2 ~/test
crystal2 ~/test
crystal2 ~/test more Normplot_tails.dat
     0     0     6            -2.99385       -30.69588
     2     0     4            -2.64107       -28.07780
   -12     0     8            -2.46310       -26.91301
     0     0    11            -2.34000       -25.43124
    -4     0     8            -2.24461       -22.29077
     0     0     7            -2.16611       -21.85669
     4     0    10            -2.09905       -18.73302
   -16     0     5            +2.04028       +10.47118
     8     0     6            +2.09905       +10.55087
     4     0     4            +2.16611       +10.55754
    -8     0     9            +2.24461       +11.08962
     6     0     3            +2.34000       +11.90654
     4     0     6            +2.46310       +12.23197
   -16     0    10            +2.64107       +12.45890
     2     0     6            +2.99385       +13.12762
crystal2 ~/test
crystal2 ~/test
```

The normal probability plot suggests that all seven reflections in the lower left-hand side corner are suspect. Its somewhat sigmoidal shape suggests the presence of non-normally distributed (systematic) errors :



Let's repeat the calculation but with these seven reflections excluded from the calculation. The first step is to create a file with the name REJECT.HKL whose first three columns contain the indeces of the reflections to be excluded :

```
crystal2 ~/test
crystal2 ~/test cp Normplot_tails.dat REJECT.HKL
crystal2 ~/test ed REJECT.HKL
crystal2 ~/test more REJECT.HKL
     0     0     6            -2.99385       -30.69588
     2     0     4            -2.64107       -28.07780
   -12     0     8            -2.46310       -26.91301
     0     0    11            -2.34000       -25.43124
    -4     0     8            -2.24461       -22.29077
     0     0     7            -2.16611       -21.85669
     4     0    10            -2.09905       -18.73302
crystal2 ~/test
crystal2 ~/test
```

Then, we edit the file `MAXENT_AUTO.IN` **and we add the keyword REJECT :**

```
crystal2 ~/test
crystal2 ~/test ed MAXENT_AUTO.IN
crystal2 ~/test more -20 MAXENT_AUTO.IN
REJECT
CELL               94.14900        24.17000        64.31901        90.00000       130.36700        90.00000
SPACEGROUP   1
MAP_FORMAT   CCP4
DIFF_PATT
PERMUTATION  3 1 2
GRID           128    256      1
GRACYCLES    80
GRATWOWINDOWS
REFLECTIONS
   -30     0     9          89.88602         3.43968       123.75751        12.84017
   -30     0    10         126.17858         3.93975       110.84611        10.25688
   -30     0    11          38.71215         5.14720        36.43570        15.66436
   -30     0    12         165.68549         4.99690       154.67838         7.42726
   -30     0    13          38.65771         4.30664        43.59790        16.74030
   -30     0    14         158.72888         4.75254       159.23166         5.49528
   -30     0    15          86.40644         3.25414        84.79811        15.51947
   -30     0    16         150.11438         4.57498       146.66685         5.11194
   -30     0    17         132.07582         4.08662       164.78131         5.11169
   -30     0    18          21.89952         8.06613        23.18951        10.87039
   ..............................................................................
crystal2 ~/test
crystal2 ~/test
```

... and we run it again, but this time giving as input the `MAXENT_AUTO.IN` file :

```
crystal2 ~/test
crystal2 ~/test GraphEnt MAXENT_AUTO.IN

-------------------------------------------------------------------------------------------------------------------------

                            ###   ###                  #######           #
                             ##   ##                     #    #           #
                             # # # #   ####   ### ###    #    # ## ###   ######
                             # # # #  #    #   #   #     #  #    ##   #    #
                             #  #  #       #    # #      ####    #    #    #
                             #  #  #   #####     #       #  #    #    #    #
                             #     #  #    #    # #      #    #  #    #    #
                             #     #  #    #   #   #     #    #  #    #    #   #
                            ###   ###  #### # ### ###   ####### ###  ###    ###

                           Gull, S.F. & Daniell, G.J. (1978), Nature, 272, 686-690
                                 Collins, D.M. (1982), Nature, 298, 49-51

                                                 NMG

-------------------------------------------------------------------------------------------------------------------------

Keyword           REJECT : 7 reflections specified in REJECT.HKL.
Keyword             CELL : Cell dimensions set to  94.15  24.17  64.32  90.00 130.37  90.00
Keyword       SPACEGROUP : space group number set to 1
Keyword       MAP_FORMAT : CCP4 map file selected.
Keyword        DIFF_PATT : Difference Patterson map run [h k l FP sig(FP) FPH sig(FPH)].
Keyword      PERMUTATION : Permutation set to 3 1 2
Keyword             GRID : Grid set to   128   256     1
Keyword        GRACYCLES : Plot every 80 cycles.
Keyword    GRATWOWINDOWS : Will keep conventional map plot.
Keyword      REFLECTIONS : start reading reflections.
Reflection rejected :   -12     0     8
Reflection rejected :    -4     0     8
Reflection rejected :     0     0     6
Reflection rejected :     0     0     7
Reflection rejected :     0     0    11
Reflection rejected :     2     0     4
Reflection rejected :     4     0    10
-------------------------------------------------------------------------------------------------------------------------

.........................................................................................................................

Normal termination ? (32 seconds)
```

NOTE WELL : Because the normal probability plot is calculated with data expanded to *P*1, each point on the plot may actually correspond to a superposition of several symmetry-equivalent reflections. When you reject data, you MUST reject all symmetry equivalent reflections that are present in your *P*1 data set. Failure to do so will show-up in your maps as absence of the expected symmetry elements. Now : under normal circumstances the `Normplot_tails.dat` file will contain all symmetry equivalent reflections, except if these are near the assumed linear part of the plot. In this case, I'm afraid that you will have to manually add the indeces of the missing equivalents in the REJECT.HKL file (sorry).

6.8 Working with X-PLOR and CNS

I can see at least three possible basic strategies when it comes to interfacing X-PLOR and *GraphEnt.* The first is to tell X-PLOR to (i) expand your reflections to *P*1, and (ii) to write them out in an ASCII X-PLOR reflection format. Then (and assuming that your text editor can record macros), edit this ASCII file and convert it to the format that *GraphEnt* expects. I wouldn't suggest this method for any but the simplest cases.

A second possible way is to use `xdlmapman` to convert an X-PLOR reflection file to a .mtz file. This may not work very well if your X-PLOR file contains weights and/or a figure-of-merit column.

A third, more reproducible way to do the trick, is to use f2mtz to convert the X-PLOR reflection file to .mtz, and then use the .mtz wrapper of *GraphEnt* to do the calculation. I will illustrate this with an example based on the `nfo-mfc_phicalc_map.inp` file distributed with X-PLOR 3.851, and assuming that you want to calculate a σ_A-weighted $2mF_o - DF_c$ map.

The first step is to add the following bold lines in the X-PLOR script :

```
...........
...........
   do (fcalc=$k2*fcalc) (all)            { apply scaling to all reflections }
   declare name=diff   domain=reciprocal type=complex end
```

declare name=testamp domain=reciprocal type=complex end

```
   if ($sigmaa_flag=true) then
                          {* Compute sigmaa weights. *}
      declare name=eobs   domain=reciprocal type=real end
.............
.............
      do (diff = combine($nn * fom * ampl(fobs) - $mm * dd * ampl(fcalc), phase(fcalc)) )
         ( acentric and sel=1 )
      do (diff = combine(fom * ampl(fobs), phase(fcalc)) ) ( centric and sel=1  )
```

do (testamp = combine($nn * ampl(fobs) - $mm * dd * ampl(fcalc) / fom, phase(fcalc)))
(acentric and sel=1)
do (testamp = combine(ampl(fobs), phase(fcalc))) (centric and sel=1)

```
      undeclare name=eobs     domain=reciprocal end
      undeclare name=ecalc    domain=reciprocal end
      undeclare name=sigmaa   domain=reciprocal end
      undeclare name=dd       domain=reciprocal end
   else
                                      {* Compute unweighted n fo-m fc difference. *}
      do (diff = combine($nn * ampl(fobs) - $mm * ampl(fcalc), phase(fcalc)) )  ( sel=1 )
   end if
```

do (SIGMA = $nn * SIGMA)(acentric and sel=1)
write reflection output=GraphEnt.hkl testamp SIGMA fom end

```
   declare name=map1   domain=real end
   do (map1=ft(diff))  ( sel=1 )
   remarks  ($nn fo- $mm fc, phicalc) map
   write map
..............
..............
```

Please note that the assignment of standard deviation for the quantity `$nn * ampl(fobs) - $mm * dd * ampl(fcalc)/fom` is wrong in this example. If you are fluent with error propagation and you have derived the correct expression, please do mail it to me as well. When you execute the script, and in addition to the map file, you should also get a reflection file with the name `GraphEnt.hkl` which would look like this :

```
 NREFlection=      7271
 ANOMalous=FALSe { equiv. to HERMitian=TRUE}
 DECLare NAME=TESTAMP      DOMAin=RECIprocal   TYPE=COMP END
 DECLare NAME=SIGMA        DOMAin=RECIprocal   TYPE=REAL END
 DECLare NAME=FOM          DOMAin=RECIprocal   TYPE=REAL END
 INDE     2    0    0 TESTAMP=    37.700   360.000 SIGMA=     2.050
                   FOM=     0.878
 INDE     4    0    0 TESTAMP=    43.500     0.000 SIGMA=     1.710
                   FOM=     0.099
.........................
```

You can now convert this to .mtz with f2mtz :

```
f2mtz hklin GraphEnt.hkl hklout GraphEnt.mtz << eof
TITLE           2fo-fc coefficients from X-plor
CELL            54.476   42.565   51.722   90.000  104.684   90.000
SYMMETRY        5
LABOUT          H K L  FP PHIB SIGFP FOM
CTYPOUT         H H H  F  P    Q     W
SKIP            5
FORMAT          '(6X,3F5.0,9X,2F10.3,7X,1F10.3/23X,1F10.3)'
END
eof
```

Before running *GraphEnt* with this file, you need one additional step in order to put the columns in the order that *GraphEnt* expects to find them. You can do this with CAD, or mtzutils, or interactively with sftools :

```
Origin ~/trm6/17
Origin ~/trm6/17 d *.mtz
-rw-r--r--   1 glykos   user       205508 Mar  3 17:12 GraphEnt.mtz
Origin ~/trm6/17
Origin ~/trm6/17 sftools


OPTIONS ARE:

    ABSENT      MODE      CALC  CHECKHKL  COMPLETE    CORREL
    DELETE    EXPAND       FFT    FOURPT    HLCONV       I2F
      LIST       MAP    MAP2SF     MAPIN  MAPLIMIT    MAPOUT
   MAPSTAT     MERGE   OPTION1   PHASHFT      PLOT     PURGE
      READ    REDUCE   REINDEX     RFREE    SELECT       SET
      SORT      STOP    WINDOW     WRITE

>> give your option (or hit <return> to list options)
read GraphEnt.mtz

selected: READ


 User:   glykos               Logical Name: GraphEnt.mtz
 Status: READONLY   Filename: GraphEnt.mtz

Reading file : GraphEnt.mtz
With format  : MTZ

  !!! WARNING, sort order improper !!!

  Sort order will be set to 1 2 3
  Use option SORT [h k l] later if needed
The following columns will be read:

TYPE  LABEL
===========

  F   FP
  P   PHIB
  Q   SIGFP
  W   FOM
  now sorting the reflections
  now merging the reflections

  7271 reflections read from file
     0 reflections appended to existing data
  7271 reflections newly created
  7271 reflections now stored in memory


>> give your option (or hit <return> to list options)
write ready.mtz column 1 3 2 4

selected: WRITE
```

```
Writing file : ready.mtz
With format  : MTZ

Columns used :
 1 3 2 4
The following columns will be written :

TYPE  LABEL
===========

  F   FP
  Q   SIGFP
  P   PHIB
  W   FOM

  (Q)QOPEN allocated #  1
 User:   glykos               Logical Name: ready.mtz
 Status: UNKNOWN    Filename: ready.mtz

>> give your option (or hit <return> to list options)
exit

selected: EXIT



Normal end program sftools

Origin ~/trm6/17
Origin ~/trm6/17
Origin ~/trm6/17 d *.mtz
-rw-r--r--   1 glykos   user       205508 Mar  3 17:12 GraphEnt.mtz
-rw-r--r--   1 glykos   user       205508 Mar  3 17:13 ready.mtz
```

6.9 Words of FFTW's wisdom

FFTW (the library responsible for all of *GraphEnt*'s FFTs) has a mechanism for saving to disk information about how best to perform the FFT for a given array. Because chances are that if you run *GraphEnt* once, you will probably run it again with the same grid, *GraphEnt* will save a file in your `HOME` directory containing this information. The name of the file is `.FFTW_wisdom` and I would suggest that you do not delete it after the end of each *GraphEnt* run. There is just one thing that you shouldn't do : *do not copy the* `.FFTW_wisdom` *file between different computers (even of the same company).* For more information why is that so, consult the FFTW manual.

7 The real thing : keyworded input.

The core of *GraphEnt* understands nothing of AUTO-labelled files, or even worse, .mtz files. What is really happening, is that when a .mtz file is specified on input, a function is called which prepares an AUTO-labeled ASCII file (with the name `MAXENT_FROM_MTZ.in` which is left behind after the program is finished). But, again, the core of *GraphEnt* can not interprete the AUTO flag. So, another function is called which translates the AUTO-labeled file to a keyworded format that *GraphEnt* can understand[14], like this one :

```
CELL              94.14900       24.17000       64.31901       90.00000      130.36700       90.00000
SPACEGROUP   1
VERBOSE
GRACYCLES    20
GRATWOWINDOWS
MAP_FORMAT   CCP4
DIFF_PATT
PERMUTATION  3 1 2
GRID           128    128      1
REFLECTIONS
   -22     0     6          50.32293        2.43270       67.66310        2.33278
   -22     0     7         148.78082        3.24875      139.47423        2.21939
   -22     0     8         189.90724        3.95280      210.31883        2.26664
   ..............................................................................
    14     0     3         250.23837        4.61909      307.68475        4.29308
    16     0     0         760.94031       15.55367      687.91992        8.57631
    16     0     1         350.09189        6.37660      328.42670        3.76660
```

No matter what the name of your .mtz or AUTO-labelled file is, the input file for the *GraphEnt* calculation is called `MAXENT_AUTO.IN` and is left behind after the calculation is finished. Needless to say that you can prepare or edit such a file and give it to *GraphEnt* by typing `GraphEnt <my_file.in>` (but, again, you can not specify resolution limits or projections if you run the program this way).

Although you will probably stick to using one or the other wrapper, there are situations where editing and using the keyworded input file is neccessary. Examples include the following : increasing the grid size (because you are getting splitted peaks for example), changing the axes permutation, excluding reflections from the calculation, selecting which map section to plot, and whether or not to use a grayscale plus contour representation, reducing the amount of output from the program, etc.

Comments can be incorporated in the file (but not after the `REFLection` keyword) by starting the comment line with a `!`, `#` or an asterisk (`*`). Please also note that the keywords are recognised using only the first four characters. A detailed description of the various keywords follows :

> **NOTE :** The default (expected) reflection structure for *GraphEnt* is to read $h, k, l, |F|, \sigma(|F|), \phi$ for each reflection record. This default is modified by the keywords `PATT`, `DIFF` and `FOM`. If none of these keywords is present, then you are expected to define six (and only six) columns for the reflection records containing $h, k, l, |F|, \sigma(|F|)$ and ϕ, even if what you are calculating is not a phased synthesis (to make it clear, if you are calculating a Patterson function, you would have to have a last column containing zeros).

7.1 CELL AND SYMMETRY RELATED KEYWORDS

7.1.1 CELL *a b c* α β γ

This keyword defines the unit cell dimensions. Six floating point numbers are expected.

7.1.2 GRID *nfast nmedium nslow*

This keyword sets the number of (integer) divisions along the whole unit cell edges for the fast, medium and slow axes. Which axis is fast, medium and slow is determined by the `PERMutation` keyword. To keep the FFT as fast as possible, *GraphEnt* will only cooperate if the grid sizes are any of the following :

```
  1,    2,    4,    6,    8,   10,
 12,   14,   16,   20,   24,   28,
 30,   32,   40,   42,   48,   56,
```

[14] In other words, .mtz files are going through two translation stages, and AUTO-labelled files through one.

```
 60,     64,     70,     80,     84,     96,
112,    120,    128,    140,    160,    168,
192,    210,    224,    240,    256,    280,
320,    336,    384,    420,    448,    480,
512,    560,    640,    672,    768,    840,
896,    960,   1024
```

If you need more than 1024 grid points on any axis, then you are better off finding a program encoding for a more efficient maximum entropy algorithm (because with *GraphEnt* you will probably never finish the calculation).

7.1.3 **PERMutation** *fastID mediumID slowID*

This keyword defines the axes permutations, ie which axis is the fastest changing, the medium, and the sectioning. The various axes are identified with the following convention : *x* is 1, *y* is 2, *z* is 3. For example, to define a valid permutation for a monoclinic space group with *b* unique, you give `PERM 3 1 2`. In this case the first argument of `GRID` should be the number of divisions of the *c* axis, followed by the number of divisions along *a* and, finally, *b*. **NOTE WELL :** Changing the axes permutation without careful thinking is a standard way to inverse the chirality of your molecule. *GraphEnt* will NOT stop you from changing enantiomorph.

7.1.4 **F000** *f*

This keyword sets the value for the F_{000} term. Although *GraphEnt* will chose a value by default, more often than not this will not be ideal. The importance of this term for the calculation is discussed analytically in section 8.1 and will not be repeated here. I will just mention that a completely wrong value of F_{000} is the most common source of problems with the program.

7.1.5 **SPACegroup** *n*

Where *n* is the spacegroup number of your crystals. Since *GraphEnt* is doing the calculation in *P*1, what you give here is absolutely irrelevant with respect to the actual calculation performed. It is only used to have a correctly formed map header in the case of the CCP4-related map files.

7.2 GRAPHICS-RELATED KEYWORDS

7.2.1 **GRACycles** *n*

The maxent plot will be updated every *n* iterations. Only applicable if your executable was compiled with graphics support.

7.2.2 **GRAGrayscale**

The density will ploted using a combination of grayscale representation and contouring. Nice if you are looking at protein maps at low resolution. Confusing for ‘peaky’ maps. Only applicable if your executable was compiled with graphics support.

7.2.3 **GRATwowindows**

This tells PGPLOT to open a new window in which to plot the maxent map, while keeping the window containing the conventional map (thus allowing you to compare the maps as you go along the calculation). I can see no reason for not having two (or more) windows. Only applicable if your executable was compiled with graphics support.

7.2.4 GRAWait

If this keyword is specified, *GraphEnt* will prompt you to press ENTER every time that the maxent map must be updated. I bet you will be bored pressing ENTER quite soon. Only applicable if your executable was compiled with graphics support.

7.2.5 GRASection *n*

This keyword allows you to specify which section *GraphEnt* should plot during the calculation. The value defined is the fractional coordinate (along the sectioning axis, see keyword `PERM`) of the required section. The default is the zero level section. Only applicable if your executable was compiled with graphics support.

7.2.6 GRANsections *n*

This keyword allows you to plot not just one section, but a stack of successive sections, where *n* is the number of sections to stack. Note that the program does not plot successive sections one on top of the other : instead, for every grid point it will find the maximum density on any of the defined sections, and will plot the resulting (maximum density) map. Only applicable if your executable was compiled with graphics support.

7.2.7 GRAFirst f

This keyword specifies the density level for plotting the first (dotted) contour line. This contour will be plotted at $\overline{\rho} + f\sigma(\rho)$, where $\overline{\rho}$ is the mean density of the section that is being plotted, and $\sigma(\rho)$ the corresponding rms deviation (again, not of the whole map but only of given section). To the best of my knowledge, *GraphEnt* will allow you to give f a negative value, thus allowing you to start contouring from below the mean density level (although I bet that you could crash either *GraphEnt* or PGPLOT by giving random values to this parameter). Default is 0.0 (ie, first contour at the mean density).

7.2.8 GRALevel f

This keyword defines the interval (in terms of number of rms deviations) for plotting the contours in the two graphics windows. Following the first contour (defined by the `GRAFirst` keyword), contours will be plotted every f times the rms deviation of the given map section. Default is 0.5 (ie, plot every 0.5 rmsd).

7.2.9 GRAMaxContours *n*

This keyword defines the maximum number of contour lines that can be drawn in the graphics windows. The intension is to reduce the workload on the X server when a Patterson function is plotted and the rms deviation of the current map section is so small that several hundred (or even thousand) contour lines must be drawn for the origin peak.

7.2.10 VT125

This keyword switches-on PGPLOT's support for ReGIS graphics, making it possible to see *GraphEnt*'s graphics from a whole series of ReGIS-capable VT terminals (eg VT125, VT240, VT241, VT330, VT330+, VT340, ...). To make things go faster (especially if you are calculating a Patterson function), consider adding a `GRAMaxContours` keyword, and increasing the value of `GRACycles`. You may also want to define the `GRAWait` keyword. Enjoy (and don't let anyone touch your VTs).

7.2.11 ONEDimensional *u v* u_0 v_0

This keyword allows you do make 1D x-y plots (instead of 2D contour plots) of the distribution of density along a specified line of the *current section* (ie plotting the distribution of density along an arbitrary direction in 3D is not

supported. The line must belong to the current map section). The first two parameters define the direction of the axis $[uv]$ whose density is to be plotted, where *u* is measured along the fast-changing axis of the current section, and *v* along the medium axis (see keyword `PERM`, section 7.1.3). The two additional parameters define the starting point (origin) for the 1D graph in the two-dimensional section (usually 0.0). u_0 and v_0 should be given in fractional coordinates (in the crystallographic frame) along the fast- and medium-changing axes. When 1D data are given on input (in the form of *h*00, 0*k*0 or 00*l* data), *GraphEnt* will automatically make the decisions for you.

When 1D data are plotted, the program will draw a horizontal dotted line at the mean density, and a series of tick marks at $n \times \sigma(\rho)$ where $\sigma(\rho)$ is the rms deviation of the 1D data. Please also note that the program will take values (from the underlying 2D section) at regularly spaced intervals using a simple 4-point linear interpolation.

7.3 REFLECTION SELECTION AND MODIFICATION.

7.3.1 REJEct

When this keyword is present, *GraphEnt* will attempt to open a file named `REJECT.HKL` **(case in important)** from the current directory, which contains in its first three columns the indeces of reflections that should be excluded from the calculation. The following is a valid `REJECT.HKL` file :

```
  0     0     6          -2.99385       -30.69588
  2     0     4          -2.64107       -28.07780
-12     0     8
  0     0    11          Everything after the first three numbers is ignored
 -4     0     8
```

7.3.2 EXCLude_diff *f*

When calculating a difference Patterson function, reflections for which the magnitude of the observed difference is larger than f will be excluded from the calculation.

7.3.3 EXFOm *f*

Reflections with a figure-of-merit less than f will be excluded from the calculation (treated as if unobserved). This is yet another of the unsuccessful *ad hockeries* tried for improving FOM-weighted phased syntheses.

7.3.4 SQRT_sigmas *f*

For testing purposes only : the standard deviations are set to $\sigma(F) = f\sqrt{F}$.

7.3.5 AVERage_sigma *f*

For testing purposes only : the standard deviations are set to $\sigma(F) = f$.

7.3.6 KFOM

For testing purposes only : the input FOMs will be multiplied by f.

7.3.7 MAXFom *f*

For testing purposes only : all input FOMs greater than f, are set to f.

7.3.8 MINFom *f*

For testing purposes only : all input FOMs less than f, are set to f.

7.3.9 LIMIt f

This keyword instructs *GraphEnt* to exclude all reflections with $F/\sigma(F) < f$ from the calculation. The exclusion takes place after all processing of the data is finished (ie, taking differences, squaring, etc). It would appear that the presence of this keyword defeats the purpose of *GraphEnt.* This is correct with one exception : When calculating a 3D isomorphous difference Patterson function, and because the phases of $\mathbf{F}_H$ and $\mathbf{F}_P$ are not correlated, a small value for $(F_{PH} - F_P)$ (for acentric reflections) will only contain valid information about F_H for only half of the cases. Excluding small differences from this type of calculation is not as harmful as it would be in other cases. My experience is that giving a `LIMIt 1.0` gives almost identical difference Patterson functions, but in a fraction of the time that the proper (no exclusions) run would take. In some cases adding this keyword makes the difference between achieving convergence and wasting CPU time.

7.3.10 SCALe f

When this keyword is present, the data (amplitude and its standard deviation) will be multiplied by the given constant f. The multiplication takes place after all processing of the data is finished (ie, taking differences, squaring, etc). Downscaling the data may be useful when the (extreme) sharpness of the MaxEnt map suggests that the data may be way off the absolute scale (on the high side, ie they must be downscaled). I think that rescaling the data at this stage is totaly unjustified. The only good excuse that I can think of, is in the case of an isomorphous difference Patterson function calculation : if the derivative is non-isomorphous[15], then there are good chances that with increasing resolution, the mean fractional isomorphous difference will increase (instead of decreasing). This could fool maxent into believing that there are good-strong data even at high resolution. This, of course is correct, the only problem being that these "strong" high resolution data is noise from our point of view[16].

7.4 CALCULUS AND LIMITS-RELATED KEYWORDS.

7.4.1 TARGet f

This keyword defines the value of the χ^2 at which the calculation will stop. Normally this is equal to the number of observations. If you want to stop earlier (a more uniform map), or later (a more peaky map), you can achieve this either by changing the `TARGet`, or changing the standard deviations of your measurements. I should warn you, however, that both procedures are probably wrong (and dangerous), unless you have reasons to believe that the standard deviations of your measurements are incorrect.

7.4.2 PHASeless f

When this keyword is present and the calculation performed is a FOM-weighted synthesis, then all reflections with a figure of merit less than f will enter the calculation with only amplitude restraints, but no phase restraints. Use at your own risk. Note that the iteration is started using the phase angles given to the program (*ie.* it is not a random-phase seeded calculation), and that the FOMs given are not ignored but are used to adjust how fast the corresponding amplitudes will approach convergence.

7.4.3 SWITch f

This is yet another *ad hockery* closely related to the `PHASeless` keyword described above. The idea in this case, is to start the calculation of a FOM-weighted synthesis in the usual way, but when the *R*-factor reaches the value f, to switch to a `PHASeless` calculation for all reflections in the data set. Use at your own risk.

[15] Assuming that an isomorphous derivative ever existed ...

[16] These large differences arise from the non-isomorphism and not from the heavy atom structure.

7.4.4 LAMBda *f*

This sets the initial value of the Lagrange multiplier for the calculation to f. If this keyword is NOT given, *GraphEnt* will determine a suitable starting value for it by performing a limited number of iterations at different starting λ values. Better leave it to the automatic mode.

7.4.5 CONStant_lambda

The normal way to perform the calculation is to gradually increase the value of the Lagrange multiplier λ till the iteration starts diverging. Then the program switches to a constant λ mode, during which the value of the multiplier can only be decreased (again when the iteration diverges). If you define the keyword `CONST`, then *GraphEnt* will start directly from the constant λ mode.

7.4.6 REMOve_origin_peak

This keyword is intended for calculation of origin-removed Patterson functions. This is achieved by subtracting the average F^2 from all contributing observations. DO NOT USE THIS KEYWORD : the correct way to do the calculation is to subtract the local (in thin resolution shells) average of the observations (and not the global average). If you want an origin-removed Patterson function and you have CCP4 installed, use the program `ecalc` to calculate the coefficients.

7.4.7 CHILimit *f*

All reflections that contribute to the final value of χ^2 by more than f times the rmsd contribution of all reflections, will be written to the file `CHIcontributions.dat`. Outstandingly large contributions to χ^2 may indicate a problem with the measurement, or a significant underestimation of its standard deviation. The default is 10.

7.5 MISCELLANEOUS KEYWORDS.

7.5.1 VERBose

If this keyword is given, *GraphEnt* will be writing out quite a lot of info. From release 0.3 onwards, this is *not* the default.

7.5.2 TIME *n*

n is the maximum number of minutes that *GraphEnt* may use for the calculation. When the time limit is reached, *GraphEnt* will write out the current map and die peacefully. Please note that the time is given in minutes and is absolute, ie it is the actual time passed since the calculation started, and not the CPU time consumed by *GraphEnt.* When this keyword is not present, *GraphEnt* will just keep on going (hopefully not for ever).

7.5.3 PSOUt

When this keyword is present (and if the program was compiled with graphics support), two postscript files will be produced containing a plot of the contents of the two graphics windows (conventional and *GraphEnt* map). The section that will be plotted is the same as the one selected with the `GRASection` keyword.

7.5.4 TRANsforms

USEFUL ONLY FOR CREATING COVER PICTURES : When this keyword is specified, *GraphEnt* will produce two additional ASCII files, with the names `MAXENT_TRANS.dat` and `START_TRANS.dat`. These files contain information about the coordinates (in reciprocal Å) and amplitudes of the reflections lying on the first section of

the reciprocal lattice, both before and after the calculation. The first section of the reciprocal lattice before the calculation is simply a section of the input data (without phase and FOM information). The same section after the calculation corresponds to a section from the modulus of the Fourier transform of the *GraphEnt* map. An example of using these files for preparing a plot, was the cover image of the previous version of this document.

7.5.5 SHOW

If this keyword is given, *GraphEnt* will be writing out (when in `VERBose` mode) an additional column containing the value of the entropy of the current map.

7.6 MODE SELECTION AND OUTPUT FORMATS.

7.6.1 MAP_format *ASCII* | *CCP4* | *NA4*

This keyword determines the map format that *GraphEnt* will use. These are discussed on page 22.

7.6.2 PATTerson

When this keyword is present, *GraphEnt* expects to read 5 (and only 5) columns for each reflection record, which will be interpreted as $h, k, l, |F|, \sigma(|F|)$ for a Patterson function calculation. The data will be squared, the standard deviations corrected accordingly and the phases set to zero before the calculation is started.

7.6.3 DIFF_patterson

When this keyword is present, *GraphEnt* expects to read 7 (and only 7) columns for each reflection record, which will be interpreted as $h, k, l, |F_1|, \sigma(|F_1|), |F_2|, \sigma(|F_2|)$ for a difference Patterson calculation. The difference and its standard deviation will be calculated, the resulting data will be squared and the phases set to zero before the calculation is started.

7.6.4 FOM

When this keyword is present, *GraphEnt* expects to read 7 (and only 7) columns for each reflection record, which will be interpreted as $h, k, l, |F|, \sigma(|F|), \phi, FOM$ for a $m|F|\exp(i\phi)$ synthesis.

7.6.5 REFLections

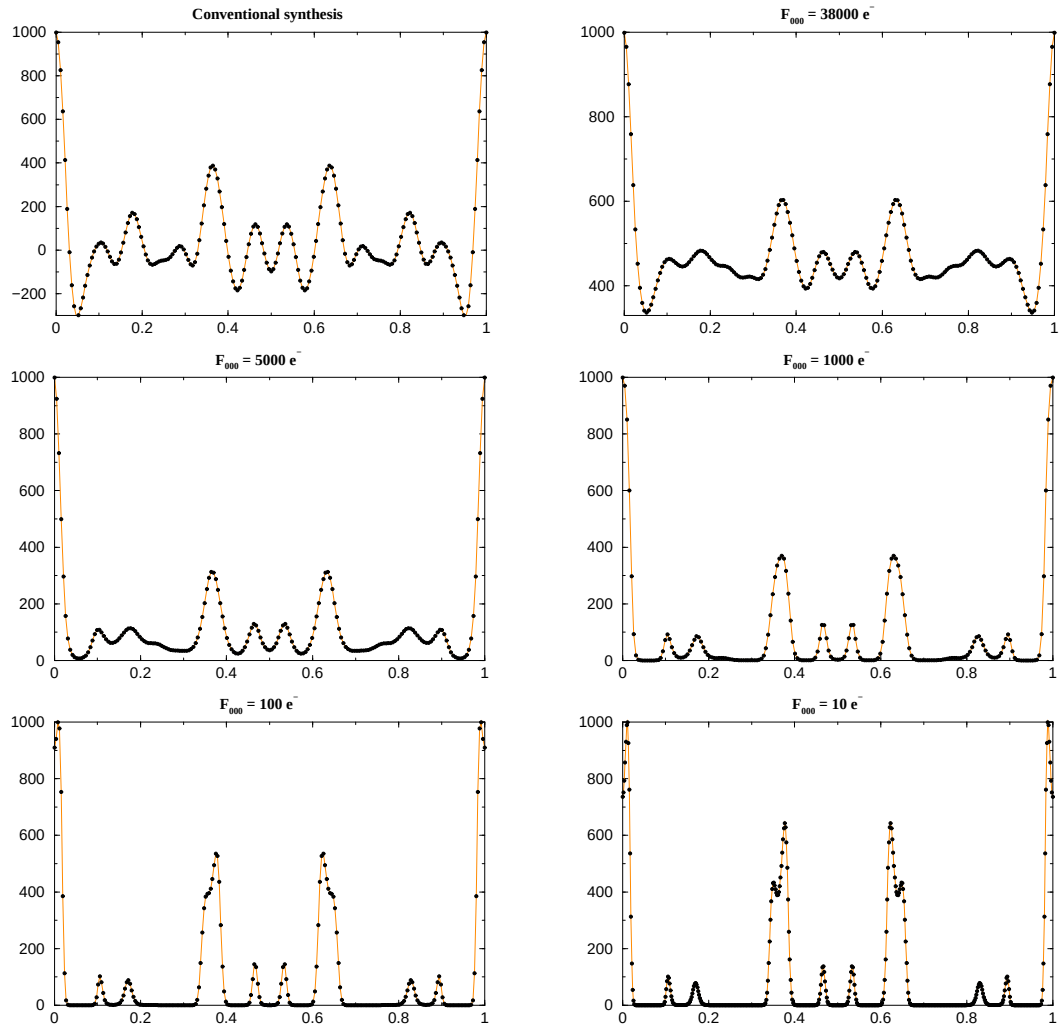
This must be the last keyword before the reflection records begin.

8 Of FOOOs, SCALes and TARGets

8.1 F_{000}-related things

For the purposes of calculating a conventional Fourier synthesis, both the presence and the value of F_{000} can safely be ignored. The reason is, of course, that for the conventional syntheses, F_{000} is simply a constant term that is added to the electron density distribution. Changing its value will only change the mean electron density and nothing more. Given that most macromolecular crystallographers prefer to contour their maps with first contour at the mean electron density plus something×rmsd, it has become a macromolecular norm to actually prefer setting the F_{000} to zero, so that the first contour of the maps is always at something×rmsd.

F_{000}=0 is bound to fail with the maxent maps. Let me illustrate this with an example. The following graphs show the distribution of density along a line containing the origin peak of a Patterson function projection[17], both for the conventional synthesis and a number of *GraphEnt* maps calculated with different values for the F_{000} term (all scaled to 999.0). I'm probably taking the fun out of it, but I think it is worth mentioning that this is a Harker line for a single-site platinum derivative : The signal is the major non-origin peak. The other peaks do not arise from the heavy atom structure.



Taking the trends apparent from these graphs to their extreme, you could argue that as the value of F_{000} tends to 0.0 e^-, the peaks in the map will tend towards δ functions. This line of reasoning immediately warns you that by "adjusting" the value of F_{000}, you can make your map look as sharp as you please although your data (meaning the data that you have indeed measured) are the same. The point is of course that F_{000} is NOT an adjustable quantity :

[17] This is the line $v = 0.5$ from the example `Patt_projection.in` included with the distribution of *GraphEnt*.

the sharpness of these maps is not required by the data that you measured, but by the value that you arbitrarily decided to assign to the F_{000}. What *GraphEnt* will give you is (or, better still, I hope it is) what is required by the data (including the assignment of F_{000}). If you tell the program that $F_{000} = 10.0\ e^-$, then *GraphEnt* will give you peaks as sharp as needed for the sum of electron density on the unit cell to be $10.0\ e^-$. The result will be that noise will also appear as sharp peaks, and you are bound to mis-interprete your map[18].

The one and only consistent way of doing the calculation is to give F_{000} its correct value. This sounds very nice, but in real life things are not so straightforward : what should the F_{000} value be for an isomorphous difference Patterson calculation using acentric terms (in which case even knowing from before-hand the number of substitution sites doesn't help because $F_{PH} - F_P \neq F_H$) ? what should the F_{000} value be for a $(2mF_o - DF_c)\exp(i\phi_c)$ difference map phased from an incomplete poly-alanine model ? should the F_{000} include the number of electrons due to bulk solvent although I only have data from 8Å (and some strong data are missing because they were overloaded) ? etc. For these reasons, and in order to keep the procedure of running *GraphEnt* automatic (at least for the first time), I have resorted to the following unjustified and arbitrary assumptions about your F_{000}s :

- **Phased syntheses :** Assuming a that your crystals contain 50% 2M ammonium sulphate and 50% protein, their mean electron density is expected to be around $0.40\ e^-/\text{Å}^3$. The assignment then is $F_{000} = 0.40 V_{cell}\ e^-$, where V_{cell} is the volume of your unit cell in Å^3. I sincerely hope that for the majority of macromolecular problems this is an overstimate of the true value (which is no harm. The maps will not be as sharp as they ought to, but it will not be possible to mis-interprete them). If on the other hand, you are calculating a Fourier synthesis for the heavy atom structure (in which case the assumed F_{000} is much too high), you are better off stoping the calculation after the `MAXENT_AUTO.IN` file has been produced, edit it and add a reasonable definition for `F000`.

- **Patterson syntheses :** In this case, and because I expect most Patterson calculations to involve macromolecular isomorphous differences, I have resorted to $F_{000} = [2\max(F)]^2$, where $\max(F)$ is the largest amplitude observed. This is a rather dubious choice which will almost certainly fail if you are calculating, for example, a native Patterson function.

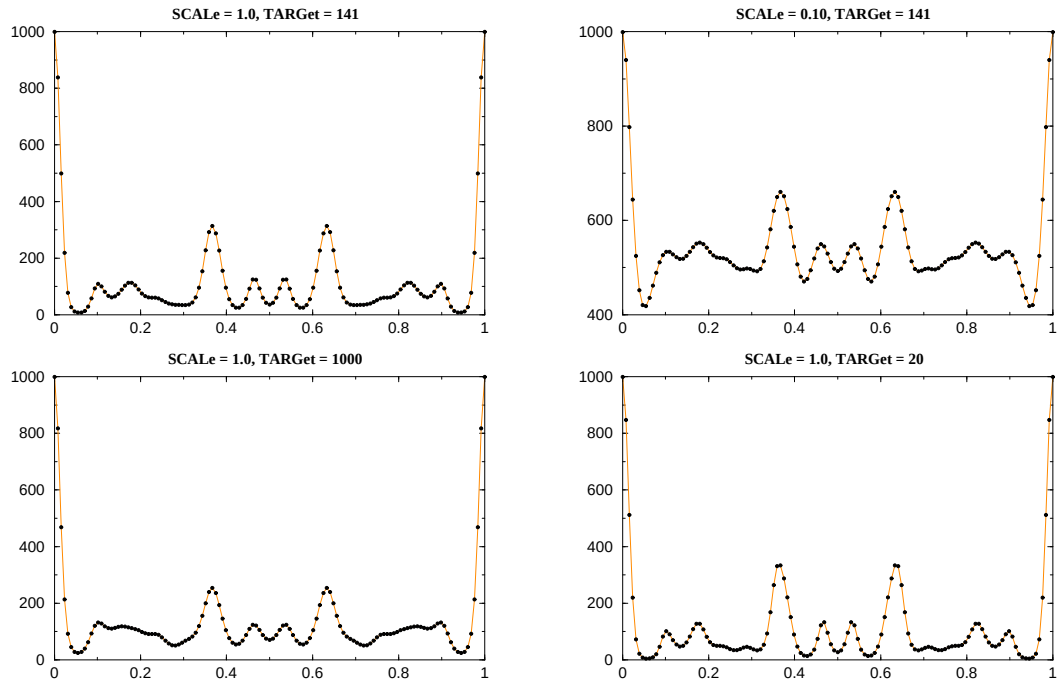
> **A pragmatist's view :** If your *GraphEnt* maps look unjustifiably sharp, increase F_{000}. If they look smooth, decrease F_{000} till the point where you can still "interprete" the features that you see.

> **Please note :** The value of the F_{000} is only used for the calculation of the initial uniform map, but is *not used to constrain the sum of densities in the GraphEnt maps that follow.* In other words, do not expect the F_{000} calculated from the *GraphEnt* map to be identical with the value that you defined.
>
> Quoting from Gull & Daniell, (1978), "... Exact fitting also implies the existence of numerous separate constraints, resulting mathematically in an unwieldy proliferation of Lagrange multipliers and preventing calculation of the solution in all but the simplest cases". In the case of F_{000} things may not be that complex (I would think that one additional re-scaling step is all that is required), but given the difficulties with estimating F_{000} in the case of Patterson and difference Fourier synthesis, I thought I would better leave F_{000} unconstrained.

8.2 Connection with the `SCALe` and `TARGet` keywords

Continuing with the graphical approach, the following diagrams illustrate the effects of using the `SCALe` and `TARGet` keywords as if they were adjustable parameters (which they should not).

[18] You can actually see one of the artifacts of having too small a value of F_{000} in the last two graphs. If you look carefully, you will see that it is not only the major peak that is beginning to show line splitting, but also the origin peak. The splitting of the origin peak is only indirectly due to the F_{000} being too small : as the peaks in the *GraphEnt* map tend towards δ functions, the amplitudes of the transform of the *GraphEnt* map tend to a set of normalised E-values with $\overline{|E|^2} = 1$ for all resulution shells. Now, because you are sampling data that go to the infinity on a finite grid (ie, the grid of your map), the power of the transform that is outside the limits of your finite grid folds back into the limits of your transform (this is usually called "aliasing"). The most notable result is that some of the phases of the Patterson function coefficients will become *negative*, and the origin peak will start developing a hole in the middle.



Examination of these graphs shows that the effect of `SCALe` is rather similar to changing the `F000`. Actually, their effects should be identical, ie giving a `SCALe 2.0, F000 50000` should give identical results with `SCALE 1.0, F000 25000`. The reason for this behaviour is that *GraphEnt* will NOT apply the scale factor to the F_{000} term.

With the `TARGet` keyword things are different. The difference of the two maps above (in terms of their sharpness) has nothing to do with scaling or the F_{000} term. The argument in this case is that reducing the target χ^2 value is to a good approximation equivalent to dividing the standard deviations of your measurements by a constant $c > 1.0$. In that case, *GraphEnt* will fit your data closer, meaning that the high resolution data (which usually have the lowest $F/\sigma(F)$, will now be reproduced more accurately and will contribute more to your map. Having said that, if the standard deviations were correctly estimated in the first place, you will be fitting noise. Increasing the target χ^2 value has the opposite effect : *GraphEnt* will now fit your data less closely, and the *GraphEnt* map will be more uniform. See page 42 for an example of using the `TARGet` keyword in the case of macromolecular anomalous Patterson function calculations.

Take home message : You need data on an absolute scale, with correctly estimated standard deviations. If you have an estimate of a suitable —for your problem— value for the F_{000} term, use it (edit the `MAXENT_AUTO.IN`, add a line with the `F000` value, re-run with `GraphEnt MAXENT_AUTO.IN`).

9 Pathology of *GraphEnt* calculations, and frequent problems

9.1 Slow convergence, or no convergence

For most of the time, this is due to me being lazy (and mathematically unapt) and not coding a more efficient maxent algorithm. In the rest of the cases, it is either your data, or that the decision made by the program about the F_{000} value was completely wrong (see section 8.1). Now, if the problem is the value of F_{000} being too small, you should be seeing an unjustifiably sharp map. If that is the case, stop the program, edit the `MAXENT_AUTO.IN` file, add a line with the `F000` keyword (see 7.1.4), and re-run the program with `GraphEnt MAXENT_AUTO.IN`.

If you are calculating a *3D isomorphous difference Patterson function* you can improve the convergence properties (hopefully without loosing much of the signal) by giving `LIMIT 1.0` or `LIMIT 2.0` in the `MAXENT_AUTO.IN` file and re-running the program (see discussion in section 7.3.9).

If you are calculating a native Patterson function, see page 42.

You can get problems with slow convergence even when you calculate something as simple as a two-dimensional projection. To my experience, slow convergence indicates the presence of some signal (to take the extreme view, if your data are consistent with a uniform map, *GraphEnt* will stop immediately), but this is not necessarily the signal you expect, or wish to have[19]. In other cases, it indicates the presence of outliers in your data (which makes it difficult to find a solution that satisfies the constraints they impose). This last case is easily identified from the contributions to χ^2 table (and the normal probability plot in the case of difference Patterson functions).

9.2 Wrong symmetry elements in the map

This is probably a problem with the data expansion to $P1$: *GraphEnt* knows nothing about your space group and only checks for the presence of some axial reflections. If there are data missing from your $P1$ set, the expected symmetry elements will not be there.

9.3 When I plot the exported *GraphEnt* map, it looks different

Maximum entropy maps are always positive, which means that their mean is not zero (as happens with conventional syntheses in the absence of F_{000}). *GraphEnt* plots the sections with the first (dashed) contour at the mean, and then every 0.5 rmsd *of the given map section* (and not of the whole map).

9.4 The *GraphEnt* map looks worringly sharp (and noisy)

The most common reason for this is that *GraphEnt* chose a completely wrong value (on the low side, ie must be increased) for the F_{000} term. You can correct this by editing the `MAXENT_AUTO.IN` file, add a line with the `F000` keyword (see 7.1.4), and re-executing the program with `GraphEnt MAXENT_AUTO.IN` (see also discussion in 8.1).

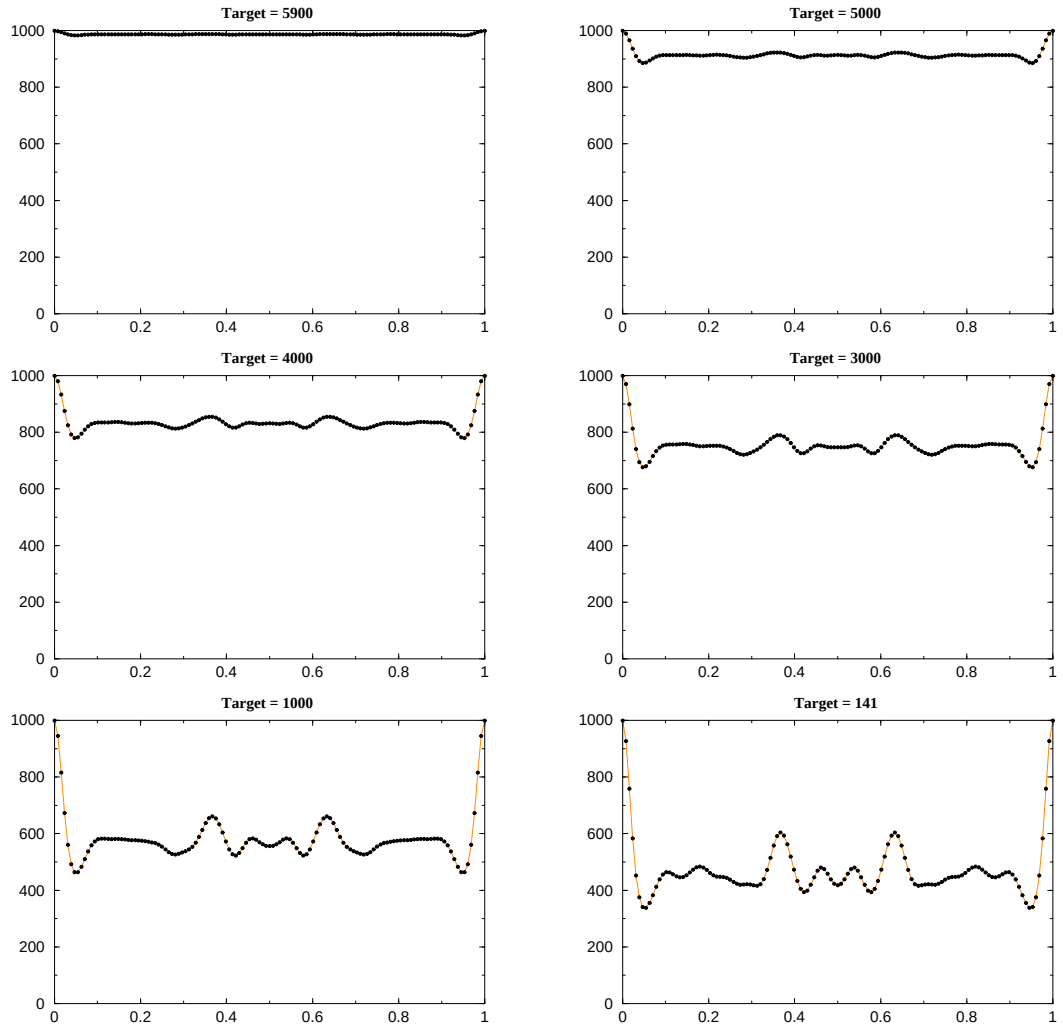
Other possible reasons are : (i) Seriously underestimated standard deviations (see keyword `TARGet`, page 34), (ii) Completely wrong scaling of data (on the high side, ie you must downscale them, see keyword `SCALe`, page 34), (iii) Doing a low resolution run with strong data throughout the resolution range used for the calculation, and, (iv) Calculating an isomorphous difference Patterson function for a markedly non-isomorphous derivative (see also page 34).

Case (iii) is one of *GraphEnt*'s deficiencies : the program should adjust the size of grid depending on the data quality, instead of using a fixed correspondence between the range of *hkl* indeces and the grid size that it will chose. There is a solution however : if your *GraphEnt* map shows contours that are not smooth (and maybe you also see peak-splitting), then, edit the `MAXENT_AUTO.IN` file in your current directory, increase the grid size given in the `GRID` keyword (see page 30), and re-run the program by giving `GraphEnt MAXENT_AUTO.IN`.

[19] For example, you may have measured very good quality native and derivative data, but if they are not isomorphous, *GraphEnt* will be fitting their (very accurately) measured differences, the only trouble being that the result will appear to be just noise to us.

9.5 The *GraphEnt* map changes considerably during the calculation

At the very beginning of the calculation (especially of a Patterson function), the *GraphEnt* map may show a great deal of peaks, which later on disappear giving a map with essentially only the origin peak. Then new features start to emerge slowly, which more often than not, remain till the end of the calculation. *This is an artifact of how the program contours the section that is plotted in the graphics window* : *GraphEnt* will always plot with the first (dashed) contour at the mean, and then every 0.5 rmsd *of the given map section* (and not of the whole map). At the very beginning of the calculation, the *GraphEnt* map is almost uniform, but because the program contours the plot from the mean and every 0.5 rmsd (however small this may be compared with the mean), the graphics window will show peaks (which in reality are just slight modulations of an otherwise uniform map). As the calculation progresses the major features start appearing (which in the case of Patterson functions is the origin peak), and then as the data are being fitted more closely the finer detail starts building up. I will illustrate these events with a series of intermediate *GraphEnt* maps produced during the calculation of a difference Patterson projection. To show clearly the significance of the mean density, these graphs only show the density along a Harker line ($v = 0.5$). The data used for this example have been discussed in section 8.1 :

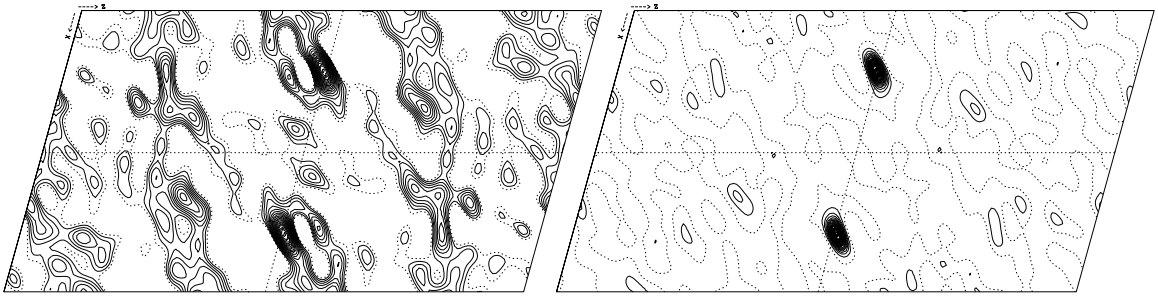


9.6 All my anomalous Pattersons are "consistent with a uniform map"

Why : Assume for a minute that all your reflections have the same $F/\sigma(F) = \mathcal{C}$. Then, because for the initial (uniform) map all Fourier coefficients are identically zero (with the exception of F_{000}), the statistic χ^2 is given by

$$\chi^2 = \sum_{\mathbf{h}} \frac{\mid F_{c,\mathbf{h}} - F_{o,\mathbf{h}} \mid^2}{\sigma(F_{\mathbf{h}})^2} = \sum_{\mathbf{h}} \frac{\mid F_{o,\mathbf{h}} \mid^2}{\sigma(F_{\mathbf{h}})^2} = \sum_{\mathbf{h}} \mathcal{C}^2 = N\mathcal{C}^2$$

where N is the number of observed reflections. Now, the target value for χ^2 during the calculation is $\chi^2 = N$. If $\mathcal{C} < 1.0$, ie $\overline{F/\sigma(F)} < 1.0$, the uniform map will satisfy the χ^2 constraint and *GraphEnt* will stop immediately. Just because *GraphEnt* stops, does not necessarily mean that there is no signal in the data : for Gaussian noise, $\chi^2 \approx N$ is the expected value of the distribution $\chi^2 \approx N \pm \sqrt{(2N)}$. This means that depending on the data, the target of the calculation could as well be significantly lower than the value aimed for by *GraphEnt.* I think it is worth emphasising this with an example. The following figure compares the conventional (left) and *GraphEnt* (right) map at the section $v = 1/2$ of the 20–3Å anomalous Patterson function for horse heart myoglobin crystals (dashed contour at the mean, and then every 0.5 rmsd of the whole map). The data were collected with CuK$_\alpha$ radiation and the anomalous signal comes from the iron atom of heme (this is one of the examples distributed with *GraphEnt*, file `Myoglobin_anom_Patt_no_outliers.in`).



A normal run of *GraphEnt* with the whole data set would immediately stop with the "uniformity" message. Even after rejecting all reflections with $F/\sigma(F) < 0.5$, *GraphEnt* would still refuse to co-operate (for 615 reflections with $F/\sigma(F) > 0.5$, the initial χ^2 —for the uniform map— was 502.8). The map shown above could only be produced after explicitly setting the `TARGet` χ^2-value to 100.0 (by editing the `MAXENT_AUTO.IN` file). As you see, it probably worth the effort[20].

Getting around it : Start *GraphEnt* the usual way. When the program stops with the "uniformity" message, edit the `MAXENT_AUTO.IN` and add a line with a new `TARGet` value (which should be less than the starting χ^2 value reported by the program if the `VERBose` flag is set on, see page 34). Depending on the circumstances, you could also add a line with `LIMIt 0.5` to exclude reflections with $F/\sigma(F) < 0.5$ (this should reduce the amount of computation required for convergence).

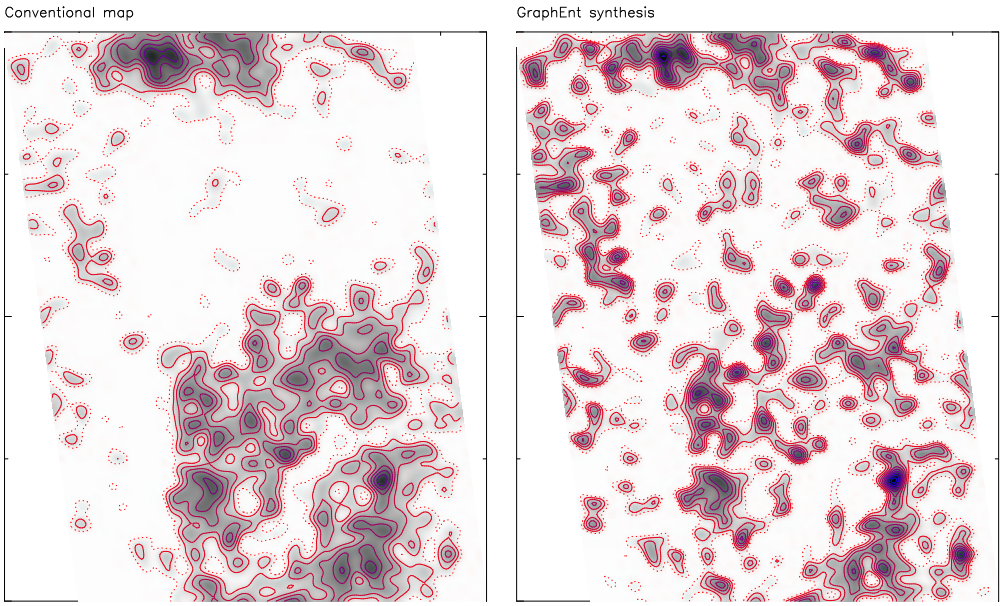
9.7 My native Patterson function calculations will take two years of CPU time to complete.

So are mine, I'm afraid. Actually, you may well find that the program will stop much earlier than the two years, with an error message "Failed to reach convergence ...". I do not think that there is a native-Patterson-specific bug in *GraphEnt.* Rather, it is probably that the data for a native Patterson are usually complete and of high quality. The higher the data quality, the longer *GraphEnt* will take to fit the constraints imposed by them (see also quotation on page 38).

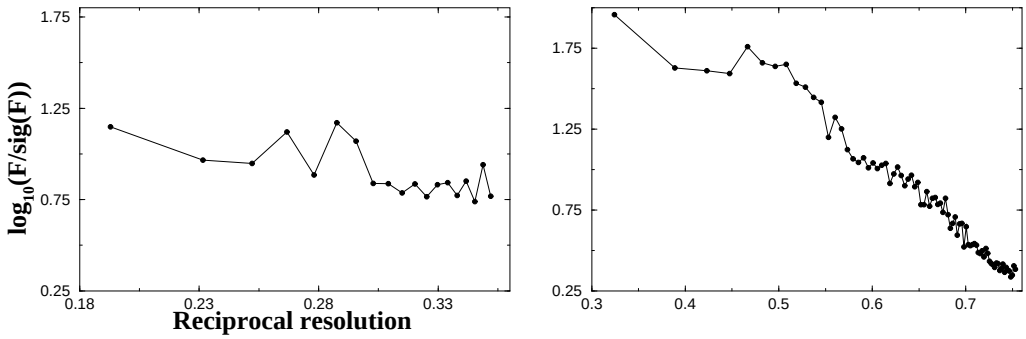
[20] I should add, however, that I am not convinced that changing the `TARGet` χ^2 is the correct way around. To continue with the example, even if we take the expected value of χ^2 to be $\chi^2 \approx N - 3\sqrt{(2N)} = 509$ (ie 3σ away from the mean), the uniform map is still consistent with the data. The fact that there seems to be some signal in the data when we reduce the `TARGet`, probably points the way to over-estimated standard deviations.

9.8 My molecule disappeared from the *GraphEnt* EM projection map.

Electron microscopy data quite often have a problem with the estimated standard deviations of the amplitudes. Let me illustrate this with an example. The figure below compares the conventional and *GraphEnt* maps for a 8Å potential density projection of a large complex.



It looks as if all low resolution information disappeared from the *GraphEnt* map, and this is more-or-less what has indeed have happened. The reason is shown in the next figure. The two graphs show on the same scale the distribution of $\log_{10}(F/\sigma(F))$ versus resolution for the EM data (left graph) and of a typical X-ray crystallographic data set (right graph).



Whereas the X-ray data have a dynamic range extending approximately over two orders of magnitude, the EM data show a flat distribution with the (strong) low resolution terms having a value of $F/\sigma(F)$ not much different from the data in the highest resolution shell. Because I have seen this behaviour with almost all EM data sets that I have come across, I suspect that the problem is with the data processing programs used by the EM community.

Note added in proof
(for CCP4–bound users only)

If you intend to use *GraphEnt* to calculate **FOM–weighted protein density maps,** then it is strongly suggested to use σA coefficients for the calculation, as follows :

– Use SIGMAA to obtain the required coefficients. If you use REFMAC you already have them in the .mtz file written by it.

– Start the program with 'graphent <my_file.mtz>'

– For a (2Fo–Fc) map, chose (in this order) the columns :
2FOFCWT, SIGFP, PH2FOFCWT where SIGFP is the standard deviation of your native structure factor amplitudes.

– For a (Fo–Fc) map, chose (in this order) the columns :
FOFCWT, SIGFP, PHFOFCWT

– If your .mtz file comes directly from a SIGMAA run, the coefficients will be : FWT, SIGFP, PHFWT and DELFWT, SIGFP, PHDELFWT.

Please note that depending on how high (or low) the figure of merit is, this procedure may underestimate the standard deviations of the coefficients. The result is that the maps may be somewhat sharper than they ought to.