Maximum Entropy Reconstruction of Velocity Map imaging Data Program Manual

Bernhard Dick

Institut für Physikalische und Theoretische Chemie, Universität Regensburg, 93053 Regensburg, Germany. E-mail: bernhard.dick@chemie.uni-regensburg.de

April 11, 2022

Abstract

Three programs are provided that perform the equivalent to an Abel inversion on data obtained from an ion imaging experiment, or angularly resolved photoelectron spectroscopy. In contrast to other methods used for Abel inversion these programs never apply an inversion or smoothing to the data. Instead, they iteratively find the map which is the most likely cause for the observed data using the likelihood criterion to obtain a best fit. From the large space of solutions that are compatible with the data within the likelihood criterion, the entropy criterion selects the solution that minimizes the information content in this map. Thus all other solutions contain information for which there is no evidence in the data.

Three implementations are available: Maximum Entropy Velocity Image Reconstruction (`MEVIR`) obtains a two-dimensional slice through the velocity distribution and can be compared directly to Abel inversion. Maximum Entropy Velocity Legendre Reconstruction (`MEVELER`) finds one-dimensional distribution functions $Q_l(v)$ in an expansion of the velocity distribution in Legendre polynomials $\mathscr{P}_l(\cos\theta)$ for the angular dependence. Both `MEVIR` and `MEVELER` use the correct likelihood criterion for data sampled from a Poissonian distribution. Maximum Entropy Legendre Expansion Image Reconstruction (`MELEXIR`) solves for the same functions as `MEVELER`, but uses a Legendre projection of the data as input. Hence the Gaussian likelihood criterion is used. `MELEXIR` is much faster than `MEVELER`, has much smaller memory requirements, and solves for all Legendre orders $L \leq 6$, in particular also odd orders which are not available with `MEVIR` or `MEVELER`.

Contents

1 What is in the program package?

Unpacking the zipped archive file creates the following directories:

bin contains four programs Mevir.exe Meveler.exe, Melexir.exe, and PrepareVMI.exe. The first three perform the corresponding analysis of a velocity map. PrepareVMI.exe takes a raw velocity map image and transforms it into the input format required by the various maximum entropy programs. I.e., for `MEVIR` and `Meveler` a quadrant matrix is formed, for `MELEXIR` the Legendre projection is done. All these four programs access MaxEntAbel.dll which must be in the search path when the programs are called.

binS contains the same programs, but statically linked to the library libMaxEntAbel.lib.

dll contains the MaxEntAbel.dll and the MaxEntAbel.lib which is required if one wants to use this DLL in ones own programs.

lib contains the library libMaxEntAbel.lib. It contains all internal functions of the maximum entropy routines and can be used to incorporate these in ones own programs for static linking.

source FORTRAN90 source files for the driver programs. Compilation and linking with the libraries will produce the executables in bin and binS. I hope they are useful as guidelines if one wants to incorporate the maximum entropy routines in ones own programs.

manual contains this manual.

test contains test input and output files.

Note that all programs are 64-bit code. The code of `MEVIR` and `MEVELER` has been completely rewritten in FORTRAN90. All programs use a common Fortran module that stores the options. In order to avoid ambiguities, the keys for a few options have been changed from the original `MEVIR` and `MEVELER` options. These are marked where the available options are discussed in detail.

2 Quick start

All programs have several options that can be set to fine tune performance. However, for many cases the default options will do. The analysis of a velocity map data set is done in two steps. In the first step, the raw data matrix is centered and the information required by the various programs is extracted and written to a new file. As an example, take the raw data file **test1.matrix** in the test-directory. The call

```
c:\path_to_program> PrepareVMI -P1 test1.matrix quadrant1.dat
```

will determine the number of rows and columns in the file **test1.matrix**, find the center, and fold the four quadrants into a single data set which is written to the file **quadrant1.dat**. The option **-P1** enables output of the protocol to the screen. The default is no output, with **-P2** the protocol is written to the file **PrepareVMI.log**.

The file can be analyzed either by `MEVIR` or by `MEVELER`. The call for `MEVIR` is

```
c:\path_to_program> Mevir -P1 quadrant1.dat
```

`MEVIR` will determine the number of rows and columns in the file, make a rough estimate of the map with the matrix inversion method, and then start the maximum entropy routine. The results are written to several files: The map (i.e. the slice of the velocity distribution), the scaled map (i.e. the velocity distribution integrated over the rotation angle around the cylinder axis: This is normalized to the number of events in the data), the best fit to the data and the residuals. These are all matrices of the same size as the quadrant data set. In addition, the map is projected onto the Legendre polynomials with L = 0,2,4,6, resulting in the distribution functions $Q_l(v)$. The option **-P1** again directs the protocol output to the screen, while **-P2** directs it to the file **Mevir.log**. The default is **-P0**, corresponding to no output.

The call to `MEVELER` is analogous to `MEVIR`:

```
c:\path_to_program> Meveler -P1 quadrant1.dat
```

`MEVELER` first determines the number of rows and columns in the file, makes a rough estimate of the map with the matrix inversion method, and then projects this onto the Legendre polynomials. The default is that the first two components, $Q_0(v)$ and $Q_2(v)$, are used as the starting map. Higher even Legendre components can be requested with the option **-Ln** with $n = 4$ or $n = 6$, however, performance is not good, and the use of `MELEXIR` is recommended instead. After finishing the initial guess, `MEVELER` starts the maximum entropy routine. The results are written to several files: The primary result are the $Q_l(v)$. The best fit to the data and the residuals are also saved, as is a reconstruction of the 2D-map (i.e. the slice of the velocity distribution). These three results are matrices of the same size as the quadrant data set. The option **-P1** again directs the protocol output to the screen, while **-P2** directs it to the file **Meveler.log**. The default is **-P0**, corresponding to no output.

The `MELEXIR` program does not use the quadrant file as input, but the projection of the raw data onto the Legendre polynomials in the polar coordinates of the 2D-image. This is also produced by the PrepareVMI.exe program. The Legendre projection is invoked by the option **-LPn**, where $n \leq 6$ is the highest Legendre order requested. The call

```
c:\path_to_program> PrepareVMI -P1 -LP2 test1.matrix LegPro1.dat
```

hence creates the file **LegPro1.dat** with the 3 columns corresponding to the Legendre components with $n = 0-2$, each accompanied with a column containing the corresponding standard deviations. This file is then read by `MELEXIR` by a call like

```
c:\path_to_program> Melexir -P1 LegPro1.dat
```

This uses the default **-L2** for the Legendre components requested, i.e. only the even orders $L = 0, 2$ are calculated. In this particular case the "sinus/cosinus" representation is used (see the paper [2]), which is more efficient than the direct use of the $Q_l(v)$.

The file test1.matrix in the test directory was made with model II described in the paper [2], with an average of 1 count per pixel.The resulting velocity distributions $Q_0(v)$ and $Q_2(v)$ from these three calculations are in the files MXdis.dat (from `MEVIR`), MEXdis.dat (from `MEVELER`), and LMEMdis.dat (from `MELEXIR`). They should look rather similar. This set of calculations is automatically performed by the "runTest1.bat" Windows batch file in the test directory.

The "runTest2.bat" Windows batch file analyzes the file test2.matrix, which was made according to model III of the `MELEXIR` paper [2], also with an average of 1 count per pixel. This data set contains contributions from even and odd order Legendre components up to $L = 4$. Since the centering algorithm does not function well with data sets containing odd order components, the center of the matrix is explicitly given in the call to PrepareVMI:

```
c:\....> PrepareVMI -P1 -LP4 -IX513 -IZ513 test2.matrix LegPro2.dat
```

This creates a Legendre projection with $L = 0..4$. This is then analyzed by `MELEXIR`:

```
c:\....> Melexir -P1 -L43 LegPro2.dat
```

Here, the option **-L43** selects the Legendre options for which the reconstruction is solved: The first integer after the **-L** is the largest even order, the second integer the largest odd order. I.e., **-L43** solves for all even and odd Legendre orders up to $n = 4$.

The results of the two batch runs (run on a i5 notebook computer) can be found in the subdirectories ResultTest1 and ResultTest2.

3 The algorithms

For a detailed description the reader is referred to the original papers [1, 2]. Here only a short summary is given and illustrated by flow diagrams. The program finds the map F which yields the best simulation of the data D. This approach avoids any numerical modification or transformation applied to the data but uses instead only the numerically stable forward Abel transform. This is equivalent to regarding all elements of F as variational parameters aiming at maximizing the probability of the map given the data, $\Pr(F|D)$. The latter can be written by the Bayesian equation

$$\Pr(F|D) = \frac{\Pr(D|F)\Pr(F)}{\Pr(D)} \tag{1}$$

The a-priori probability of the data, $\Pr(D)$, is a constant for a given data set and model. Since probabilities must be positive numbers, the two factors in the numerator in equ. 1 can be written as

$$\Pr(D|F) = \frac{1}{Z_L}\exp(-L) \tag{2}$$

$$\Pr(F) = \exp(\alpha S) \tag{3}$$

where Z_L is a normalization constant. Maximizing the likelihood $\Pr(D|F)$ is equivalent to minimizing the likelihood estimator L. The resulting map makes the actual data the most likely event of a measurement. The likelihood estimator L measures the agreement between the simulated data A predicted by the map F and the actual data D. For the model of Gaussian noise this is given by

$$L_G = \frac{1}{2}\sum_{J=1}^{N_D}\left(\frac{A_J - D_J}{\sigma_J}\right)^2 = \frac{1}{2}\chi^2 \tag{4}$$

where N_D is the number of data values. For Gaussian noise maximizing the likelihood is equivalent to minimizing the sum of square deviations between the measured data and the simulated data predicted by the model. When the data are particle counts, the Poissonian likelihood estimator L_P must be used,

$$L_P = \sum_J (A_J - D_J \ln A_J + \ln(D_J!)) \tag{5}$$

Since the number of variable parameters (i.e. elements of F) is similar to the number of data values to be fitted, many maps F are compatible with the data in the sense that the corresponding estimator L is below a given threshold L_0. Maximizing the a-priori probability of the map $\Pr(F)$ selects the most likely map from this manyfold. This is equivalent to maximizing the entropy function S which, for a strictly positive map, can be given by

$$S = -\sum_{J=1}^{N_F} F_J \ln \frac{F_J}{eB_J} \tag{6}$$

where B_J is a default. We follow an approach proposed by Skilling an Gull in 1985 [3]. First L is minimized below a certain threshold value L_0. Subsequently, S is maximized with the constrained $L = L_0$. The first step selects from all possible maps those that are compatible with the data within the desired accuracy. The second step selects from these maps the one with the smallest information content. In other words, every other map has information for which there is no evidence in the data. For this reason the method is known as Maximum Entropy method.

3.1 MEVIR: maximum entropy velocity image reconstruction

The strategy used in the `MEVIR` method is schematically depicted in figure 1. The aim is finding the map F which is a slice cut along the z-axis through the center of the velocity distribution. We take this map as a matrix with the same dimensions $N \times M$ as the data matrix $\mathbf{D}$. Application of the Abel transform $\mathcal{R}$ to the map yields the simulated image $\mathbf{A}$, again as a $N \times M$ matrix. The map is iteratively varied until the simulated image agrees within a given tolerance with the data - measured by the likelihood estimator L. Application of the Abel transform is accomplished by a matrix multiplication.

$$\mathbf{A} = \mathbf{F}\mathbf{R} \tag{7}$$

Obviously, this transforms every row of $\mathbf{F}$ into the corresponding row of $\mathbf{A}$, i.e., no correlation between adjacent rows in $\mathbf{F}$ is assumed. The entropy measure defined in equ. 6 is apparently independent of the ordering of the values in the map and hence also ignorant of any correlation that might exist between adjacent elements of $\mathbf{F}$. In order to account for such a correlation, which obviously must exists in the map, Gull and Skilling have proposed to use a so called hidden map $\mathbf{H}$ behind the visible map $\mathbf{F}$ [4]. These two maps are related by a linear transformation $\mathcal{T}$ which we define as a convolution process which may be repeated several times, creating a sequence of hidden layers between the hidden map and the visible map.

$$F_{ij}^{(0)} = H_{ij} \tag{8}$$

$$F_{ij}^{(K)} = \left(1 - \frac{9\gamma}{8}\right) F_{ij}^{(K-1)} + \frac{\gamma}{8} \sum_{k=i-1}^{i+1} \sum_{l=j-1}^{j+1} F_{kl}^{(K-1)} \tag{9}$$

Each iteration replaces the value of a particular pixel with a weighted average of this pixel and its 8 immediate neighbors. A convenient value for γ is 1/2.

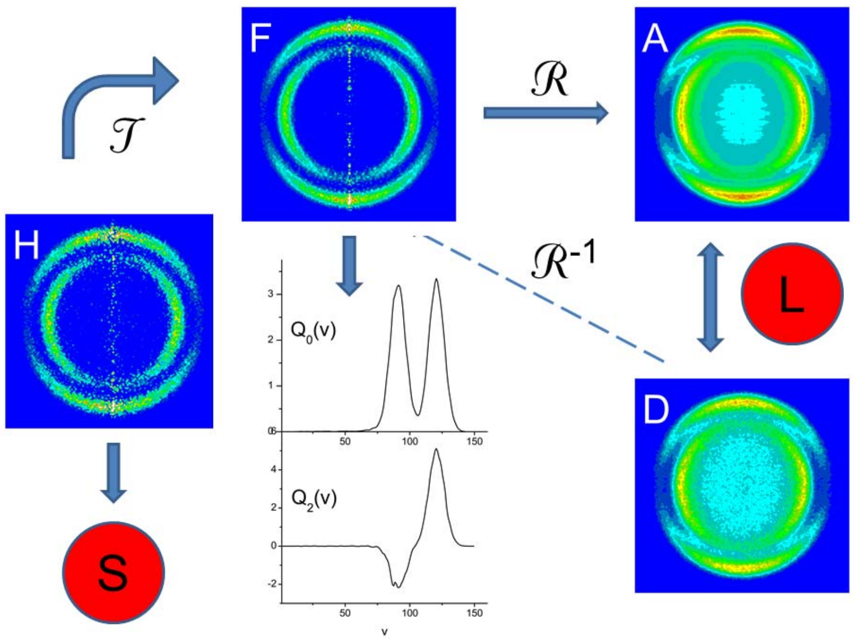

Figure 1: Schematic of MEVIR (maximum entropy velocity image reconstruction) strategy: The Abel transform $\mathcal{R}$ calculates a simulated image A from the map F. The likelihood estimator L compares this simulation with the data D. Projection of the map F onto Legendre polynomials yields the angular velocity distributions $Q_l(v)$. The map F is defined by application of the linear transformation $\mathcal{T}$ on a hidden map H, and the entropy S measures the information contained in H. This introduces correlation between adjacent elements of F. The hidden map H is varied until L is minimized below a given threshold while S is maximized. The inverse Abel transform $\mathcal{R}^{-1}$ indicated by the dashed arrow is never used.

The optimization is performed on the hidden map. Starting with the default map $H_J = B_J$ all elements of $\mathbf{H}$ are varied until the likelihood estimator L falls below a given threshold and the entropy S is maximized. We use an algorithm which performs the search for the optimum in a three-dimensional subspace. Two coordinates of this space are the gradients of L and S with respect to H, the third is found by application of $\mathbf{R}$ and its transpose to a linear combination of the two gradient vectors. A short description can be found in [5]. We find the algorithm rather efficient, reaching convergence in ca. 20 - 50 iterations.

The visible map $\mathbf{F}$ is the primary result of this procedure. Although this map contains the full information that can be extracted from the experimental data, further reduction into a velocity distribution and anisotropies is frequently desired. For this purpose the velocity distribution in spherical coordinates is usually written in the form

$$P^S(v,\theta) = \frac{1}{2}p(v) \cdot \left(1 + \sum_{l>0} \beta^{(l)} \mathscr{P}_l(\cos\theta)\right) \tag{10}$$

When the photofragmentation is the result of a dipole transition and the sample was initially isotropic, the sum may be restricted to the term with $l = 2$. Equ. 10 is not convenient for our purpose, however, since the anisotropy parameters $\beta^{(l)}$ must be considered functions of the velocity and are ill defined when $p(v)$ becomes small. Therefore, we use the form

$$P^S(v,\theta) = \frac{1}{v^2} \sum_{l=0}^{L} Q_l(v) \mathscr{P}_l(\cos\theta) \tag{11}$$

Due to the orthogonality of the Legendre polynomials the functions $Q_l(v)$ can be obtained from the map by first transforming to spherical coordinates followed by projection

$$Q_l(v) = v^2(2l+1) \int_0^\pi P^S(v,\theta) \mathscr{P}_l(\cos\theta) \sin\theta \, d\theta \tag{12}$$

The factor v^2 in equ. 12 ensures that the area under each peak in $Q_0(v)$ is proportional to the total ion count for this velocity component. The anisotropy parameters may be obtained by the ratio

$$\beta^l(v) = \frac{Q_l(v)}{Q_0(v)} \tag{13}$$

which is, of course, only meaningful for velocities where $Q_0(v)$ has significant intensity. The final result of the `MEVIR` method is thus a set of distribution functions $Q_l(v)$, and these can be extracted to any desired degree l by the projection of equ. 12 since the map F contains the full information that is hidden in the data.

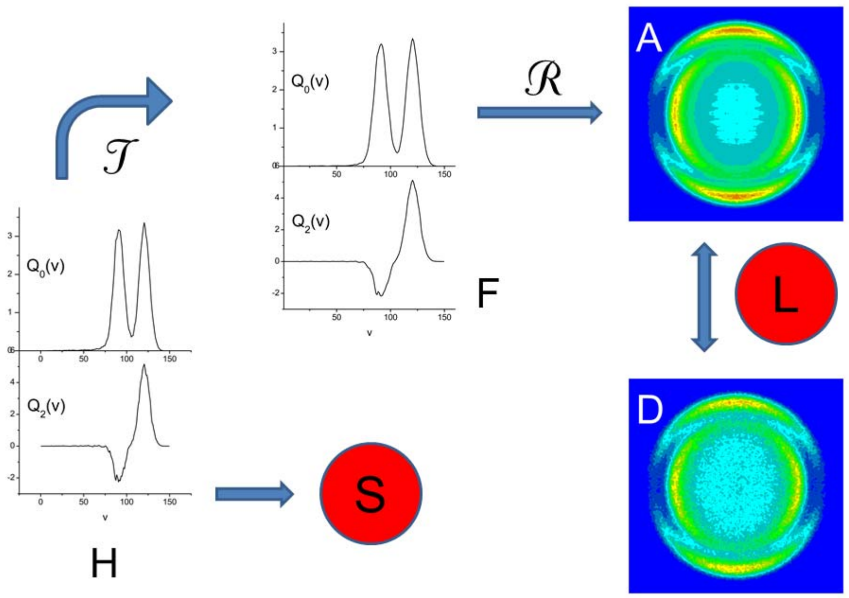

Figure 2: Schematic of MEVELER (maximum entropy velocity Legendre reconstruction) strategy: The simulated image is directly calculated from the expansion of the angular velocity distribution in Legendre polynomials, $Q_l(v)$. These distributions play the role of the map F. They are obtained by a linear transformation $\mathcal{T}$ from a corresponding set of hidden velocity distributions H which is subject to the entropy measure.

3.2 MEVELER: maximum entropy velocity Legendre reconstruction

The `MEVIR` algorithm makes no assumption on the form of the velocity distribution, except that it is rotationally symmetric around the z-axis. In most experimental situations it is, however, known that the velocity distribution should be well represented by equ. 11 with only a small number of Legendre polynomials involved. The numerical velocity information can then be represented by a $N \times L$ matrix $\mathbf{Q}$ with matrix elements $Q_{kl} = Q_l(v_k)$. The `MEVELER` algorithm, schematically shown in figure 2, uses this matrix as the visible map. It is related to the hidden map $\mathbf{H}$ by the iterative convolution

$$Q_{kl}^{(0)} = H_{kl} \tag{14}$$

$$Q_{kl}^{(K)} = \frac{\gamma}{2}\left(Q_{k-1,l}^{(K-1)} + Q_{k+1,l}^{(K-1)}\right) + (1-\gamma)Q_{kl}^{(K-1)} \tag{15}$$

In practice we found good performance of the algorithm with $\gamma = 1/2$. With $K = 0$ the hidden map and the visible map are identical. With data matrices that contain on

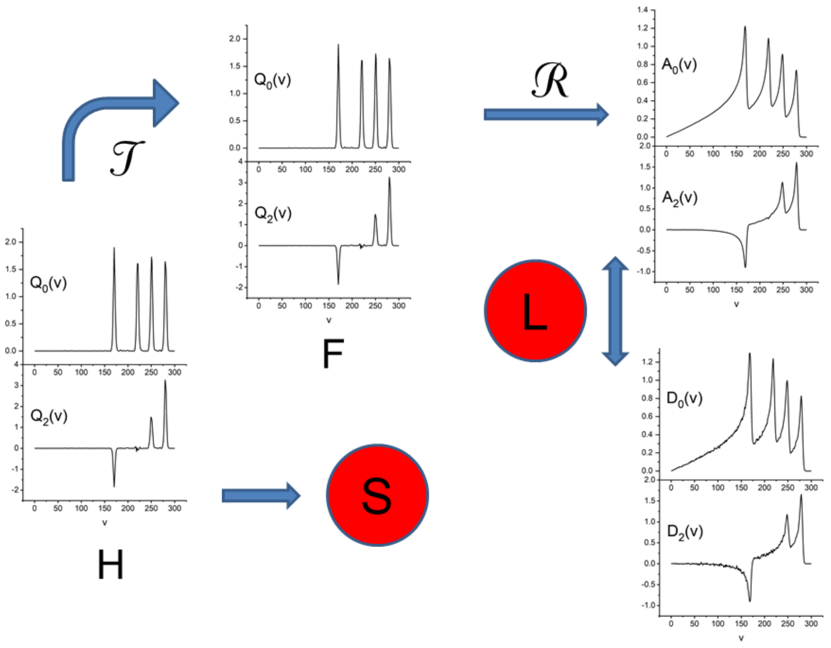

Figure 3: Schematic of MELEXIR (maximum entropy Legendre expansion image reconstruction) strategy: The simulated Legendre projection $A_l(v)$ is calculated by forward transform ($\mathcal{R}$) from the expansion of the angular velocity distribution in Legendre polynomials, $Q_l(v)$. These distributions play the role of the map F. They are obtained by a linear transformation $\mathcal{T}$ from a corresponding set of hidden velocity distributions H which is subject to the entropy measure. The agreement between the simulation $A_l(v)$ and the data $D_l(v)$ is measured by the Gaussian likelihood.

average one or more ions per pixel no hidden map is usually required.

For the `MEVELER` algorithm the transformation $\mathcal{R}$ is programmed so that it directly transforms $\mathbf{Q}$ into $\mathbf{A}$. The velocity map F used in the `MEVIR` algorithm is not involved in the `MEVELER` algorithm but may be calculated from the final result for $Q_l(v)$.

3.3 MELEXIR: Maximum Entropy Legendre Expansion Image Reconstruction.

This algorithm is described in detail in [2]. It uses the same Legendre expansion model for the velocity distribution as `MEVELER`. However, instead of the matrix quadrant a projection of the ion image onto Legendre polynomials in the polar coordinates of the ion image is used as the data. The `MELEXIR` algorithm is schematically shown in figure 3. The relation between the hidden map and the visible map is the same as for `MEVELER`. As for `MEVELER`, for $K = 0$ the hidden map and the visible map are identical. With data matrices that contain on average one or more ions per pixel no hidden map is usually required.

4 Preparing the data for MaxEnt analysis

4.1 Mapping detector to matrix

Before the description of the program the coordinates and other conventions used will be shortly described. The experiment yields the two-dimensional coordinates of the charged particles (electrons or ions) that hit the detector (see figure 4). We label these coordinates by x and z, where the latter corresponds to the polarization vector of the laser, i.e. the cylindrical symmetry axis. Usually the detection area is divided into pixels, and ions hitting the same pixel are counted together. This naturally results in a numerical representation of the data as a matrix, where each matrix entry corresponds to one pixel. In the matrix form, each entry is identified by two indices (I_Z, I_X), where I_Z indicates the row and I_X the column. Traditionally, $A(1,1)$ is the upper left corner of the matrix, and $A(N,1)$ is the Nth value in the first column going downwards (see figure 4). When

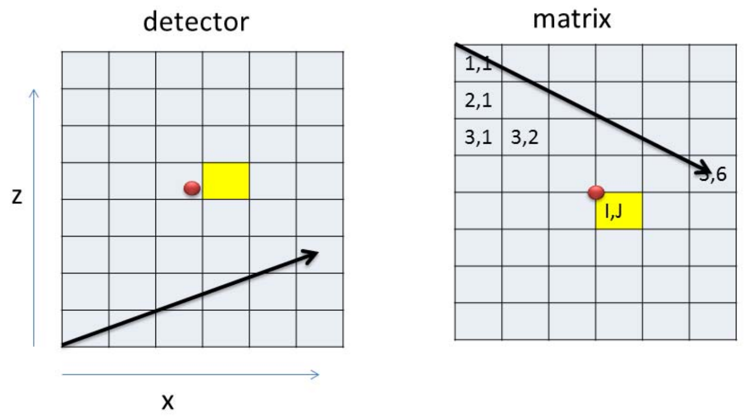

Figure 4: Pixel structure of the detector (left) and its representation as a matrix (right). The z-axis is the polarization direction of the photolysis laser, i.e. the axis of cylinder symmetry. Pixels along the x-axis correspond to a row of the matrix, pixels going up along the z-axis correspond to the matrix elements of a column going down. The red dot indicates the center of the image. The indices of the matrix element in yellow are used to define the center of the image.

images are obtained in the event counting mode, the primary result for each ion is a pair of real numbers (z, x), and it is not necessary to assign these to the actual pixel structure of the detector. Instead, binning into pixels could be done later by the software, e.g. by

$$
\begin{aligned}
I_X(x) &= \mathrm{INT}(x/d) + 1 \qquad (16)\\
I_Z(z) &= \mathrm{INT}(z/d) + 1 \qquad (17)
\end{aligned}
$$

where d is the size of the bin (i.e. virtual pixel). For calculations with the data, however, pixel indices have to be converted back into coordinates. This is done in all programs by

$$
\begin{aligned}
x = (I_X - 0.5) \qquad (18)\\
z = (I_Z - 0.5) \qquad (19)
\end{aligned}
$$

I.e., the coordinate associated with a given matrix element is the center of the corresponding pixel. So far we have set the origin of the coordinate system at the lower left corner of the detector. For the analysis of the data it is more convenient to define the center of the image as the coordinate origin. In the "real" world the coordinates of the center (z_0, x_0) are floating point numbers and can point to any position within any pixel (see the red dot in the left part of figure 4). For numerical operations with the matrix this is, however, inconvenient. Therefore, we assign the center of the matrix always to a corner of a pixel, i.e., the four pixels meeting at this corner are the lower left corners of the four quadrants of the image. We hence define as the center of the matrix the coordinates of the corner that is closest to the "real-world" coordinates (see red dot in the left part of figure 4). The " matrix-world" coordinates used in the programs for the center are the indices of the matrix element that is to the right and below the red dot, i.e. (I, J) in figure 4. This means: if the matrix is already a quadrant (as assumed by `MEVIR` and `MEVELER`), then the "center" is at $A(1, 1)$.

4.2 Centering the data matrix, contracting of quadrants

The program PrepareVMI.exe allows several methods to identify the center of the data matrix:

a) In the simplest case, the indices (IZ,IX) of the "center pixel" as defined above are given. This is done with the options -IXnnn -IZmmm on the command line, where nnn and mmm are positive integer numbers.

b) The program can calculate the center of mass of the image, i.e.

$$
\begin{aligned}
I_X &= \mathrm{INT}(\langle x \rangle) + 1 \qquad (20)\\
\langle x \rangle &= \frac{\sum\limits_{j,k} k\ A(j,k)}{\sum\limits_{j,k} A(j,k)} \qquad (21)
\end{aligned}
$$

and the corresponding formula for I_Z. This is invoked by the command line option -MC=1.

c) The option -MC=2 invokes a centering algorithm. This is the default.

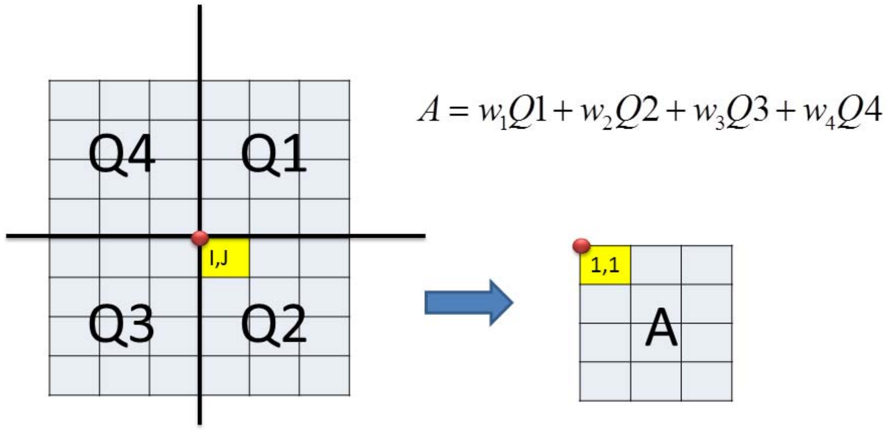

Figure 5: Contraction of the 4 quadrants of an image. The quadrants Q1 and Q4 are folded down, the quadrants Q3 and Q4 are folded to the right, and then added together with the weights w_j. Presently, the weights are either 0 or 1, and are set by the option -Wjklm, where j, k, l, m correspond to the quadrants in the order shown in the picture. I.e., -W1111 (the default) will add all 4 quadrants, W0001 will only use quadrant Q4.

The centering algorithm should add those 4 pixels together that have the same distance from the center and the same absolute value of the angle towards the z-axis. If the center is not chosen properly, those 4 pixels that should be added together will be mapped to four different pixels in the contracted image (i.e. the "quadrant"). In the present version of the program a function $G(x, z)$ is maximized which is the integral over the square of a distribution function $P(x, z; r)$

$$G(x, z) = \int_0^R |P(x, z; r|^2 \, dr \tag{22}$$

Where $P(x, z; r)$ is the probability of finding an ion at distance r from the point (x, z). This distribution function is calculated as a histogram with a bin-size equal to the pixel width. The rationale behind this choice is the assumption that a better centered image will lead to a radial distribution with a higher contrast. Since the area under $P(x, z; r)$ is normalized to the total number of ions in the image, sharper peaks in the distribution will lead to a larger integral of the square. Optimization is done with a simplex algorithm. This is not especially fast, but very robust, and does not require gradients of the function.

5 Input for program PrepareVMI

The program is invoked by the command line

```
C:\> PrepareVMI  filename1 filename2  -Option1 -Option2 ...
```

where filename1 is the name of the input file containing the raw data matrix. The input file must exist. filename2 is the name of the output file. If the output file exists and the "-OV" option is set, it is overwritten. Otherwise the filename "DefaultQ.dat" is used. The program interprets the first string that does not begin with a minus "-" sign as filename1, the second one as filename2. Otherwise the order of the command line arguments is not important.

All options begin with a minus "-" or a slash sign, followed by one or two letters (the key) which are followed by a number. An equal "=" sign between the key and the number is also accepted. Presently, the following options are available:

-IX=n: Column index of center point. with n a positive integer number

-IZ=n: Row index of center point, with n a positive integer number

-SC=f: Scale factor, with f a floating point number. -SC=0.001 multiplies all matrix entries with 0.001, the same is achieved by -SC=1.E-3

-TR: Use the transposed matrix if the data acquisition program stores the data this way (i.e. pixel parallel to the cylinder axis into rows of the matrix). Transposition must be done here, the transposition option is no longer available in the MaxEnt routines.

-ST=f: Step size for the initial simplex in optimization of center. Default is -ST=15.0, i.e., the initial simplex consists of the initial center point and two points shifted by 15 pixels (one stepsize) along the x- and z-axes.

-Wnnnn: The list of weights for the four quadrants. W1111 means add all four quadrants, W0110 adds only quadrants 2 and 3.

-MC=n: Chooses the method for centering. **(Note that this option was -M in the old F2QC.exe program. Now -M is reserved for the MaxEnt routines)**:

MC=0 uses the input values of -IX and -IZ

MC=1 uses the center of mass of the image

MC=2 uses simplex refinement starting with the center of mass. M2 is the default.

Any values given for -IX and -IZ are ignored with MC=1 and MC=2. When the values of either -IX or -IZ are not positive numbers (e.g. zero), MC=2 is used. When MC=0 is given but no values for -IX and -IZ, the center of the matrix is returned (i.e. half the largest index of columns and rows.

-SH:	The histogram is saved to a file named "histogram.dat". This can be done also with MC=0 or MC=1, i.e. when no refinement of the center is requested. Default is no file. **(Note that this option was -H in the old F2QC.exe program. Now -H is reserved for the MaxEnt routines)**
-RH=n:	Range for the histogram evaluation. E.g., -RH=350 uses all values of the original matrix within a circle of radius 350 pixel around the assumed center, and the histogram has 350 entries. Default is the largest circle that fits into the original matrix. **(Note that this option was -R in the old F2QC.exe program. Now -R is reserved for the MaxEnt routines)**
-LP=n:	This option switches to the Legendre projection mode, i.e. the output file contains the Legendre projected image up to L=n and their standard deviations.
-DV=f:	When the Legendre projection mode is chosen, the floating point number f is the stepsize in the velocity axis in pixel units. Default is -DV=1.0.
-P=n:	Print switch. -P=0 (default) generates no output, -P=1 directs the output to the screen, -P=2 sends it to the file PrepareVMI.log.

The only parameter that must be provided is the input data file name. Calling

```
C:\working directory\PrepareVMI datafile.dat
```

Is equivalent to

```
C:\working directory\PrepareVMI -MC2 -W1111 datafile.dat DefaultQ.dat
```

i.e., all 4 quadrants are added, the center is determined first by calculation of the center of mass and subsequently refined with the centering algorithm. The result is written to the file "DefaultQ.dat". The program determines whether all numbers are integer. If so, the number of digits needed for the largest number is determined, and the file is written in a format with this fixed field length, separated by blanks.

6 Input for MaxEnt programs

Both programs are called from the command line by

```
C:\> ProgramName [-CMDstring] datafile
```

Here, `datafile` is the name of the file containing the data. For Programname = `MEVIR` or `MEVELER`, the file contains the particle counts of the contracted quadrant of the ion image. The programs assume an ASCII file with every record corresponding to a row of the matrix. The number of columns in the matrix is automatically determined by analyzing the first record. The number of records give the number of rows of the matrix. The data can be integer numbers or floating point numbers, separated by either spaces or commas. The row index corresponds to the z-axis (i.e. the laser polarization axis), the column index to the x-axis. Both `MEVIR` and `MEVELER` assume that all valid data are either zero or positive, i.e., negative data entries are ignored in the analysis.

For ProgramName = MELEXIR, the input file must contain the Legendre projections and their standard deviations as produced with the PrepareVMI program. Make sure that the file contains the required number of Legendre components, i.e. when `MELEXIR` is called with -Lnm, PrepareVMI must be called with -LPk with k $\geq$ max(n,m).

All control of the programs can be done with (optional) command strings `CMDstring`. These must begin with a minus character or a slash, otherwise the string is interpreted as the data file name. The command string contains letters followed by optional numbers. The following options are recognized by all programs (the equal symbol can be omitted):

-I=n : maximum number of iterations. Default is -I=50.

-H=n : Number of hidden layers, default is no hidden layers (-H=0).

-T=n : Termination criterion. -T=0 is the classic method, i.e., the program iterates until the likelihood criterion reaches the target, which may never happen. -T=1 stops when the running average of the Gaussian likelihood criterion did not change by more than 1 % in three consecutive iterations. -T=2 stops when the Gaussian likelihood criterion did not change by more than 1 % in three consecutive iterations. -T=3 and -T=4 are the same as -T=1 and -T=2 but with the Poissonian likelihood criterion. Default is -T=2. `MELEXIR` only uses the Gaussian likelihood. (Note: even when the Poissonian likelihood estimator is used for the optimization, the Gaussian equivalent is also calculated. The latter shows less fluctuations and is hence useful for recognizing self-consistency.)

-P=n : Print level. -P0 means no output to screen or log file (default). -P1 puts output to the screen, -P2 to a log-file.

-R=n : Radius in pixel units for valid data area. Special cases: -R1 means all data (default for `MEVIR`), -R2 means all data in the circle with radius MIN(nrow,ncol), where nrow is the number of rows in the matrix, and ncol is the number of columns. -R2 is the default for `MEVELER`. The Legendre projection used in `PrepareVMI` also uses only data within a circle, and the default is the largest circle around the center that fits into the data matrix, i.e. it is equivalent to the -R2 option of `MEVELER`. In `MELEXIR` a smaller radius can be chosen with the -R=n option. However, to remain consistent with `MEVELER`, -R=2 uses all data from the largest circle (patch220411), but is actually the default.

-BA=f: sets the reference value (i.e. the "base") of the map to the floating value f. A negative value sets BA so that the integrated intensity of the image simulated with BA as the default map is f times the total ion count in the data. For `MEVIR` and `MEVELER` the default is -BA=1.0, `MELEXIR` uses the total ion count divided by the number of velocity bins as default.

-DV=f: sets the stepsize DELTAV for the final velocity distribution to the floating value f. Default is DV=1.0 Values less than DV=0.2 are probably not meaningful. DV is used in `MEVELER` as the stepsize of the functions $Q_l(v)$. In `MEVIR` it is used only for the Legendre projection at the end. `MELEXIR` assumes that DV was already set in PrepareVMI for the Legendre projection.

-S=n : Defines the starting map. -S=0 is a constant map (F = BA), -S=1 uses a distribution obtained by analysis of a crude Abel inversion of the data, -S=2 assumes that the initial map is in the array fmap when the subroutine is called. Default is -S=1.

-Lnm : sets the number of Legendre distributions in `MEVELER` and `MELEXIR`. The first digit n gives the largest even order, the second the largest odd order (`MELEXIR` only). Default is -L2.

-C : Start with a map from a previous calculation. The map is read from the file MXini.dat for `MEVIR`, MEXini.dat for `MEVELER`, and LMEMini.dat for `MELEXIR`. It is equivalent to the option -S=2. .

The following options are specific for a particular program:

-G : Gaussian likelihood estimator is used, with the standard deviation for each data value D given by $\sqrt{\text{MAX}(D,1)}$. Default is Poissonian likelihood. Applies to `MEVIR` and `MEVELER`.

-M=n : Method switch. in `MEVIR`, M=1 switches to an alternative algorithm to calculate the transform matrix. In `MELEXIR`, M=1 switches the sinus/cosinus representation off for -L=2. Only for test use.

-B=n :	Switch for background correction, `MEVELER` only. The background level is calculated as the average intensity in the area outside the largest circle in the data. This value is added to the simulated data in calculating the likelihood. B=0 is the default (no background correction), B=1 switches the correction on.

Calling `MEVIR` without any command string and only with the data file name will produce a standard run, starting with a crude Abel inversion by the matrix method (-S=1), allowing for 50 iterations (-I=50), using all the data in the matrix (-R=1) and Poissonian likelihood, no hidden layers (-H=0), and terminate when the Gaussian likelihood criterion becomes stationary (-T=2).

Calling `MEVELER` without any command string and only with the data file name will produce a standard run, starting with a crude Abel inversion by the matrix method (-S=1), followed by projection onto the Legendre polynomials with L=0 and L=2 (-L2). This is used as the starting map and the maximum entropy routine is called, allowing for 50 iterations (-I=50), using Poissonian likelihood, no hidden layers (-H=0), and terminate when the Gaussian likelihood criterion becomes stationary (-T=2).

Calling `MELEXIR` without any command string and only with the data file name will produce a standard run, starting with a crude Abel inversion by the DAVIS method (-S=1). Only the Legendre polynomials with L=0 and L=2 (-L2) are used. The maximum entropy routine is called with this starting map, allowing for 50 iterations (-I=50), using Gaussian likelihood, no hidden layers (-H=0), and terminate when the Gaussian likelihood criterion becomes stationary (-T=2).

7 Program Output

7.1 MEVIR

The results are written to files:

MXmap.dat	Final map F in the same format as the data matrix D.
MXsim.dat	Matrix with the best fit to the data.
MXres.dat	Matrix with the weighted residuals, i.e. the difference between data and simulation divided by the standard deviation. (Note: The equivalent to the standard deviation for Poissonian likelihood is the square root of the calculated function value.)
MXdis.dat	The velocity distributions $Q_l(v)$ corresponding to the Legendre polynomials with $l = 0, 2, 4, 6$ found by projecting the final map onto the Legendre polynomials.
Mevir.log	when printing to a log-file was requested (option P2).

7.2 MEVELER

This program produces the following files:

MEXmap.dat	Final map F in the same format as the data matrix D. **In contrast to the old `MEVELER` program, the output is now in scientific notation with 5 significant digits (e.g., 0.12345E+02) and separated by blanks.**
MEXsim.dat	Matrix with the best fit to the data.
MEXres.dat	Matrix with the weighted residuals, i.e. the difference between data and simulation divided by the standard deviation. (Note: The equivalent to the standard deviation for Poissonian likelihood is the square root of the calculated function value.)
MEXdis.dat	The velocity distributions $Q_l(v)$ corresponding the the Legendre polynomials with the requested l.
Meveler.log	when printing to a log-file was requested (option -P2).

7.3 MELEXIR

This program produces the following files:

LMEMdis.dat	The velocity distributions $Q_l(v)$ corresponding the Legendre polynomials with the requested l.
LMEMsim.dat	The best fit to the data, i.e. to the Legendre projections of the ion image.
LMEMres.dat	Residuals of the fit, i.e. DATA - FIT (no weighting here!)
LMEMuse.dat	The data that were actually used (the input file may contain more Legendre components and a longer velocity range).
Linverse.dat	The velocity distributions $Q_l(v)$ obtained by the direct inversion (i.e. the DAVIS type method).
Melexir.log	when printing to a log-file was requested (option P=2).

The main difference is that `MXmap.dat` is the primary result in `MEVIR`, from which `MXdis.dat` is derived, whereas in `MEVELER` and `MELEXIR` the primary result is `MEXdis.dat` and `LMEMdis.dat`, respectively, which is then used to calculate `MEXmap.dat` in `MEVELER`. The different prefixes `MX`, `MEX` and `LMEM` have been chosen to protect the files from overwriting if the methods are used on the same data set.

8 The library MaxEntAbel.dll

This Windows dynamic link library (DLL) provides the tools that allow the users to implement the routines of `MEVIR`, `MEVELER`, or `MELEXIR` into their own programs. It also contains the routines for PrepareVMI. All parameter settings are encapsulated into a FORTRAN module that is local in the DLL.

Together with MaxEntAbel.dll comes a library MaxEntAbel.lib which is used by the linker to import the exported symbols of the DLL. This library must be linked with the main program that should access the DLL. It should be possible to call the routines in the DLL also from other programs like LabView.

I also provide a static library with the name libMaxEntAbel.lib which can be statically linked to provide the same functions as those in the DLL, i.e. replacing MaxEntAbel.lib in the linking process. The stand-alone programs in the binS directory were compiled in this way. Note that in this case the compiler also needs access to the LaPack and BLAS libraries (linear algebra packages and basic linear algebra subroutines). These come usually with the compiler.

The DLL exports the following subroutines that can be called by the user.

8.1 SetOptions

This subroutine is called by

```
CALL SetOptions(OptString)
```

where Optstring is a string containing the options, separated by blanks. Each option begins with either a - (minus) or a / (slash) symbol, followed by a two-letter or one-letter code, and optionally a number. For example, the option -R300 sets the radius for the data to be analyzed to 300 pixels. The program recognizes the option also if an = (equal) symbol is inserted between the option code and the number. I.e., -R=300 or /R300 or /R=300 are all equivalent.

After returning from the subroutine SetOptions, the string OptString contains all strings that were not recognized as options. E.g., when the command line of the original `MEVELER` program is processed, OptString will contain the filename of the input file.

8.2 Image2Data

This subroutine is called by

```
CALL Image2Data(fimage,ldf,nrow,ncol,dat,ldd)
```

where

```
!-- subroutine arguments
  DOUBLE PRECISION, INTENT(IN OUT)         :: fimage(ldf,*), dat(ldd)
  INTEGER, INTENT(IN OUT)                  :: nrow, ncol
  INTEGER, INTENT(IN)                      :: ldf, ldd
!
!-------------------------------------------------------------------------------!
! on entry:                                                                     !
! fimage(ldf,*) = contains the matrix of raw data                               !
! ldf           = leading dimension of array fimage in calling program          !
! nrow, ncol    = logical dimension of the data in array fimage                 !
! dat(ndd)      = array for the results                                         !
! ndd           = length of array dat in calling program, must be large enough !
!                 to hold the data of one quadrant or the Legendre projection  !
!                                                                               !
! on exit, when -LP option is not set:                                          !
! nrow, ncol    = the new logical dimension of the  extracted quadrant. ndd    !
!                 must be large enough to hold these data, ndd >= nrow*ncol    !
!                                                                               !
! on exit, when -LPn option is set:                                             !
! nrow          = number of bins along the velocity axis (nv); This depends on !
!                 the position of the center, the -R option, and the -DV option!
! ncol          = number of Legendre components (i.e., nl = lmax+1).           !
!                 ndd must be larger than nrow*ncol*2 = nv*nl*2                !
!-------------------------------------------------------------------------------!
```

The array fimage has leading dimension ldf. On entry, nrow and ncol are the number of rows and columns in fimage that contain the data (double precision numbers). On exit, nrow and ncol are the number of rows and columns in the combined quadrant. This is stored on array dat with size nrow*ncol. The user must provide sufficient memory, i.e. ndd >= nrow*ncol.

When the option -LP is set, the program generates the Legendre projection instead of the quadrant. The result is again in the array dat, with all columns directly following each other. I.e., the size ldd must be at least 2*nl*nv, where nl is the number of Legendre components, and nv is the number of bins along the velocity axis. These values are returned on the variables nrow = nv and ncol = nl. Note that for each Legendre projected value also the corresponding standard deviation is stored.

If one calls image2data a second time from the same main program, the previously stored options remain valid. If, after doing a Legendre projection, the second call should make a quadrant extraction, the Legendre projection option must be switched off again, by supplying a negative number for the parameter, e.g. "-LP=-1".

Details of the use may be seen in the source code **PrepareVMI.f90**, which is the driver routine for **PrepareVMI.exe**.

8.3 MevirDLL

The subroutine is defined by:

```
  subroutine MevirDLL (dat,nrow,ncol,fmap,work,nwork)
!-- subroutine arguments:
  double precision, intent(IN OUT) :: dat(nrow,ncol), fmap(nrow,ncol)
  double precision, intent(IN OUT) :: work(nwork)
  integer         , intent(IN)     :: nrow,ncol,nwork
!
!------------------------------------------------------------------------------!
!  on entry:                                                                   !
!  dat(nrow,ncol) is the quadrant data                                         !
!  fmap(nrow,ncol) is the map. With "-S2" the initial map is already in fmap, !
!                  otherwise the initial map is set by the routine.            !
!  work(nwork) is a work array, where nwork >= 10*nrow*ncol                    !
!                                                                              !
!  on exit:                                                                    !
!  fmap          : contains the final (hidden) map.                            !
!  dat           : the true map in classical representation (i.e. slice       !
!                  through the velocity distribution).                         !
!                                                                              !
!  further results are located at the following memory positions:              !
!  work, block #1: residuals of the fit                                        !
!  work, block #2: best fit                                                    !
!  work, block #3: true map, internal scaling (i.e. sum = ncounts)             !
!  work, block #4: the distribution functions Q_l(v), l=0,2,4,6                !
!                                                                              !
!  the matrix results all have dimension (nrow,ncol), the distribution         !
!  functions have dimension Q(10*nrow,4)                                       !
!                                                                              !
!------------------------------------------------------------------------------!
```

Details of the use may be seen in the source code **Mevir.f90**, which is the driver routine for **Mevir.exe**.

8.4 MevelerDLL

This subroutine is defined by

```
  subroutine MevelerDLL (daten,nrow,ncol,fmap,nv,llm,work,nwork,qini)
!-- subroutine arguments:
  double precision, intent(INOUT) :: daten(nrow,ncol), fmap(nv,llm)
  double precision, intent(INOUT) :: work(nwork)
  integer         , intent(IN)    :: nrow,ncol,nv,llm,nwork
  logical         , intent(IN)    :: qini
!------------------------------------------------------------------------------!
!                                                                              !
! on entry:                                                                    !
!                                                                              !
! daten     :   array of one quadrant of the image                             !
! nrow, ncol:   dimension of the daten array (i.e. the array with the data of !
!               one quadrant of the ion counts. Note that this must also be   !
!               the physical dimension of the daten array, i.e., the second   !
!               column _MUST_ begin immediately following the first column.   !
!               The MAXENT subroutines think that this is a vector!           !
! fmap      :   initial guess for the map                                      !
! nv, llm   :   physical and logical (!) dimension of fmap, the MAXENT routine!
!               thinks this is a vector.                                       !
! work      :   work array                                                     !
! nwork     :   size of workarray, must be at least 5*nrow*ncol + 5*nv*llm    !
! qini      :   if .true., size of the coefficient array BSAVE will be cal-   !
!               culated and the array will be allocated.                      !
!                                                                              !
! on exit:                                                                     !
! fmap                  = contains the true map (the hidden map is discarded  !
! work(1:ndat)          = the residuals of the fit                             !
! work(ndat+1:2*ndat)   = the best fit to the data                             !
! work(2*ndat+1:3*ndat) = the simulated 2D-map                                 !
! work(ip8)   = rmsd    : Gaussian RMSD between data and fit                   !
! work(ip8+1) = ent     : normalized entropy                                   !
! work(ip8+2) = sum1    : integrated intensity of the reconstructed image     !
! work(ip8+3) = av      : average intensity                                    !
!                                                                              !
! where ip8 = 3*ndat + 1                                                       !
!------------------------------------------------------------------------------!
```

On entry, dat(nrow, ncol) contains the combined quadrant. The user must provide storage for the arrays basis and work. On exit, fmap contains the Legendre-distribution functions $Q_l(v)$. The first three blocks of the array work contain the residuals of the fit (i.e. work(1:ndat)), the fit to the data (i.e. work(ndat+1:2*ndat)), and the fitted slice through the velocity distribution (i.e. work(2*ndat+1:3*ndat), the 2D-representation of the map).

On the first call, qini = .true. initiates a dry run of the transform which determines the memory requirement for the transform. This can be very large. E.g., for a quadrant with 500 columns and rows and a map with 500 velocity steps (1 pixel wide) and 2 Legendre polynomials, ca. 66 MWORD of double precision (i.e. 530 MByte) need to be allocated. If a second call to MevelerDLL is made with the same values for nrow, ncol, nv, and llm, qini=.false. skips this step. Details of the use may be seen in the source code **Meveler.f90**, which is the driver routine for **Meveler.exe**.

8.5 MelexirDLL

The interface to this subroutine is:

```
  subroutine MelexirDLL (dat,sigma,fmap,base,datinv,nr,nt)
  integer, intent(IN)      :: nr, nt
  double precision, target :: dat(nt),fmap(nt),base(nt),sigma(nt),datinv(nt)
!------------------------------------------------------------------------------!
!  on entry:
!  nr   = number of bins along the r-axis and the v-axis.
!  nt   = total number of data values (and fmap-values), nt =  nr * ltotal,
!         where ltotal is the total number of Legendre components
!  dat  = contains the data, i.e. the Legendre projected ion image. First the
!         even orders, then the odd orders. The data must immediately follow
!         since the maximum entropy algorithm treats dat as a vector.
!  sigma= the standard deviations of the values in dat
!  fmap = the initial map (with option -S2). With -S0 fmap will be set to a
!         constant, with -S1 (default) the initial fmap is obtained by direct
!         inversion (i.e. the DAVIS algorithm).
!  base = the default value of the map for the entropy calculation. I.e.,
!         fmap = base maximizes the entropy.
!
!  on exit:
!  datinv = the map obtained by the DAVIS inverse
!  fmap   = the hidden map
!  base   = the best fit to the data
!  sigma  = the true map
!  dat    = the residuals of the fit
!
!------------------------------------------------------------------------------!
```

Details of the use may be seen in the source code **Melexir.f90**, which is the driver routine for **Melexir.exe**.

8.6 CheckOption

This subroutine offers a possibility to get individual parameters encapsulated in the DLL. The call is

```
CALL CheckOption(key,iopt,fopt,qopt)
```

where key is a character variable with the key for the requested option, whereas iopt, fopt, and qopt are the integer, double precision, and logical values associated with this option. In this way the calling program can know the actual settings of all options, including the default settings. For examples, see the code in **Mevir.f90**, **Meveler.f90**, or **Melexir.f90**.

8.7 SetDefaultOptions

This subroutine sets all parameters encapsulated in the DLL to their default values. The call is

```
CALL SetDefaultOptions ()
```

This function is useful if the main program calls any of the DLL routines more than once. All parameter settings remain in the DLL until a new value is set. Hence, for a sequence of calls that use the same settings, only one call to SetOptions is required at the beginning of the sequence. Each parameter value can be reassigned by a new call to SetOptions with the corresponding string. E.g.,

```
OptString = '-IX=0 -IZ=0'
CALL SetOptions (OptString)
```

resets the predefined center of the matrix to zero, hence the next call to Image2Data will optimize the center position.

The options -TR, -SH, -G, and -C have no value associated, i.e. their presence in the option string sets a flag. E.g., the option -G switches from Poissonian likelihood (default) to Gaussian likelihood. This applies to MEVIR and MEVELER only. If one wants to call MEVIR twice from the same program, first with Gaussian and then with Poissonian likelihood, one needs a way to reset the flag. This can be done with a call to SetDefaultOptions, thereby setting all parameters to default values, or by sending the string -G0 via SetOptions. Likewise, the other flags are reset by -TR0, -SH0, and -C0. (More precisely, any number following the key will reset this flag.)

9 citation

If you like my programs and use them in your scientific work, I would be grateful if you could cite the two papers [1, 2] in your publications. Thanks!

References

[1] B. Dick, *Phys. Chem. Chem. Phys.*, **16** (2014) 470.

[2] B. Dick, *Phys. Chem. Chem. Phys.*, **21** (2019) 19499.

[3] J. Skilling and S. Gull, in *Maximum-Entropy and Bayesian Methods in Inverse Problems*, C.R. Smith and W.T. Jr. Grandy, eds., Reidel, Dordrecht, 1985;

[4] S. Gull and J. Skilling Quantified maximum entropy, memsys5 users' manual Technical report, Maximum Entropy Data Consultants Ltd. South Hill 42 Southgate Street Bury St. Edmunds Suffolk, IP33 2AZ, U.K., 1999.

[5] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical recipes in Fortran 77: the art of scientific computing, chapter 18.7*, Cambridge University Press, 1988.