Sampling

In order to calculate the Fourier integrals swiftly by applying FFT techniques we have to prepare a given data-set. We are not making any assumptions on the abscissa. The abscissa may be linearly equidistant, may be logarithmically distributed or of any distribution. It is only assumed that the data-set resembles a function, i.e. each abscissa value is unique and it is ordered. General distributions we term warped. Either the abscissa is linear or warped. We have a linear abscissa if all points are equidistant up to a numerical threshold, which is set to be at 1ppm. A warped abscissa needs to be re-sampled onto a linear grid.

Before the actual sampling takes place the original data are investigated and some switches are set according to user specifications and according to the type of subsequent processing. The routine just doing that is XG_Procs_Utils.Process_FourierParams(), which precedes all sampling for Fourier transformations.

The first check is therefore on linear distribution, because in this case some calculations can be accelerated. The second step is to determine the number of sampling points. We know the range (xRange) to be sampled, either by the starting- and end-points of the data-set or by the bounds of the Process Window. If the sampled data-set needs to be grounded then xRange.min = 0. More on grounding see the chapter on Fourier transformation. We also calculate the smallest distance, Delta.min, of all points within xRange. Then the number of points is

\[ N := \text{floor}\left(\frac{\text{xRange.max} - \text{xRange.min}}{\text{Delta.min}} + 1.0\right) \]  (1)

If we want to make use of the FFT-algorithm then we may have to increase to the next power of 2, which may yield over-sampling:

\[ N := \text{FFT}_\text{NextPow2}(N) \]  (2)

Excessive numbers of N needs to be limited by preference-values:

\[
\text{Limit(gc Linear Nmin, gc Linear Nmax, N);} \quad \text{(for linear abscissa)} \\
\text{Limit(gc Warped Nmin, gc Warped Nmax, N);} \quad \text{(for warped abscissa)}
\]  (3)

Sampling types

The third step yields the sampling type, which depends on whether the data-set has a linear abscissa or a warped one. The sampling type is determined in the second half of XG_Procs_Utils.Process_FourierParams().

Linear abscissa

On a linear grid we want to know weather we can use the data-set directly (\textit{pint Direct}), weather we need more points (over-sampling, \textit{pint Interpolate}) or weather we need less points than the original data-set (under-sampling, \textit{pint Convolve}).
Why not to use the data-set as is when it is linear anyway? There are some reasons

1) Rounding to the base of 2.
2) Limiting excessive number of sampling points.
3) Application of the Process Window
4) Grounding (Zero padding to zero)

The sampling rate, \( dx \), is calculated from the range and the number of sampling points. The latter are given by formulae (1 ... 3):

\[
dx := \frac{\text{abs}(x\text{Range.max} - x\text{Range.min})}{N - 1.0}; \tag{4}\]

In a next step we have to sort out the type of sampling. If \( dx = \text{Delta.min} \) then we can potentially use the data-set directly. If \( dx < \text{Delta.min} \) then the new sampling rate is smaller then the delta of the original data-set, hence we have to insert points in-between and use interpolation from one point to the next in the original data-set. If \( dx > \text{Delta.min} \) then the new data-set becomes smaller than the original one. In this case there are multiple original points to select from. However, it is better to use a mean-value forming algorithm, rather than to pick one and skip others. The mean-value forming is equivalent to band-limiting the spectrum (more on this later).

\[
\begin{align*}
\text{if } \text{CompZahl}(dx, \text{Delta.min}, 6) \text{ then begin} \\
x_1 := \text{CD.Get_xAxisValue}(i1); \\
x_2 := \text{CD.Get_xAxisValue}(i2); \\
\text{if } (\text{CompZahl}(x_1, \text{xRange.min}, 6) \text{ and} \\
\text{CompZahl}(x_2, \text{xRange.max}, 6)) \\
\text{then} \\
\text{IntType} := \text{pint_Direct} \\
\text{else} \\
\text{IntType} := \text{pint_Interpolate}; \\
\text{end else if } dx < \text{Delta.min} \text{ then } \{ \text{Oversampling} \} \\
\text{IntType} := \text{pint_Interpolate} \\
\text{else if } dx > \text{Delta.min} \text{ then } \\
\text{IntType} := \text{pint_Convolve} \{ \text{Undersampling} \} \tag{5}\end{align*}
\]

**Warped abscissa**

A warped abscissa needs to be re-sampled to a linear abscissa. In a first try we want to use Delta.min as sampling rate, \( dx \), which is the smallest step size found (formula 1). For example if the data-set has a logarithmically distributed abscissa then Delta.min is the abscissa-difference of the first two data-points. However, with Delta.min we may exceed the allowed number of points (as specified in the preferences). In this case we have to limit \( N \) (formula 3). The sampling rate is given by:

\[
dx := \frac{\text{abs}(x\text{Range.max} - x\text{Range.min})}{N - 1.0}; \tag{6}\]

Similar to the linear case we need to determine the sampling type, only that we know that \( dx \not< \text{Delta.min} \) in the warped case.

\[
\begin{align*}
\text{if } dx < \text{Delta.min} \text{ then } & \{ \text{Oversampling} \} \\
\text{IntType} := \text{pint_Interpolate} \\
\text{else} \\
\text{IntType} := \text{pint_Convolve} \{ \text{Undersampling} \} \tag{7}\end{align*}
\]
Interpolation or over-sampling

Interpolation means that our new data-set has more points than the original one, and the new sampling rate, \( dx \) is smaller than \( \Delta \text{.min} \).

![Figure 1: Linear interpolation](image1)

![Figure 1: Linear interpolation](image2)

In Vacs we use a simple linear interpolation from point to point as pictured in figure 1. Regardless of the abscissa distribution of the whole data-set an intermediate point is always taken from a straight line between the next available neighbors on a linear grid.

The straight-line interpolation corresponds to sampling into bins of triangular shape as pictured on the right hand side of figure 1. Because the Fourier-spectrum of a triangle is broader than the equivalent Fourier spectrum of a rectangular bin the associated total band-width of the sampled data is assumed to be wider here than in usual audio signal-processing. The advantage is a swift processing and no leakage effects at start and end-points of the data-stream.

![Figure 2. Over-sampling. Left linear-, right logarithmic distribution](image3)

The strict interpolation on a linear grid can, of course yield arte-facts when viewed on different grids, such as, for example the logarithmic grid, as figure 2 demonstrates. In this example the original curve (red) has 5 points. The interpolated curve (blue) has 10 points. Viewed on a linear grid the inserted points are shown on straight lines connecting the five original points. Viewed on a logarithmic axis, however, the new points appear to be lower at the upper range. What happens? The error is caused by the graphic mapping of the red curve because the graphic engine connects any point by a straight line regardless of the current axis-grid (because of speed reasons). However, with the blue curve the interpolation is fixed to the linear grid and is therefore mapped more accurately to the logarithmic view.
**Convolving or under-sampling**

If the sampling rate of the new data-set is larger than the one of the original data-set then we do under-sampling. In the under-sampling situation we try to represent a data-set with fewer points. When does under-sampling happen? There are rare situations, were the processing imposes under-sampling automatically. This situation can happen if the calculated number of points exceeds the limits specified in the preferences. For example, if the original data-set points are distributed logarithmically then the sampling rate, dx, is given by the distance of the first two points. The number of points is calculated by function (1). If, further, the abscissa range is relative large then this could yield an excessive number of points. The limiting procedure, in turn, enforces a certain sampling rate, dx, as can be seen in the function (4). Then it can happen that, with this new sampling rate, parts of the warped data-set becomes under-sampled (usually at the lower end of the logarithmic axis). Note, that the automatic under-sampling routine is different to the explicit linear-sampling and smoothing processing. The latter makes use of the Continuous Phase Technique. More on this later.

In Vacs, the automatic under-sampling is done the classical way by forming a mean-value at the point of sampling. The mean-value integration is done through low-pass filtering the associate spectrum with a rectangular filter. For derivation we start backwards and assume that we can write our new data-set as the superposition of sinc-functions:

$$h(x) = \sum_{n=-\infty}^{\infty} \left( h(n \cdot dx) \cdot \sin\left( \frac{x}{dx} \cdot \pi - n \cdot \pi \right) \right)$$  
(8)

This of course is the classic interpolation formula for data on a linear grid. Formula (8) is a convolution, which is equivalent to a rectangular low-pass filter with cut-off at \( f_c = 1/(2 \cdot dx) \).

What we have are points of the original set, which are arbitrarily distributed. Hence, we could write the original points, say \( g(x_0) \), in terms of formula (8):

$$g(x_0) = \sum_{n=-\infty}^{\infty} \left( h(n \cdot dx) \cdot \sin\left( \frac{x_0}{dx} \cdot \pi - n \cdot \pi \right) \right)$$  
(9)

What we are after are the \( h(n \cdot dx) \)-terms. In order to solve for these we make use of the orthogonal property of the sinc-function:

$$\int_{-\infty}^{\infty} \sin\left( \frac{x}{dx} \cdot \pi - m \cdot \pi \right) \cdot \sin\left( \frac{x}{dx} \cdot \pi - n \cdot \pi \right) dx = \delta_{m,n}$$  
(10)

If we multiply both sides of equation (9) by another sinc-function (of the same set) and integrate, finally we arrive at:

$$h(n \cdot dx) = \int_{-\infty}^{\infty} g(\xi) \cdot \sin\left( \frac{\xi}{dx} \cdot \pi - n \cdot \pi \right) d\xi$$  
(11)
Figure 3: Sinc function for integration

In figure 3 a sinc function is displayed with \( dx=0.1 \) and \( n=2 \). Hence, formula (11) integrates over all data-points but the weighting is such that the point of the original data-set close to \( n \cdot dx \) is amplified most. This works fine even on strongly distorted distributions, because in areas of high density the integration forms a mean value and at sparse areas the algorithm picks an interpolated value.

Of course we cannot integrate forever and, for speed reasons, we have to stop integrating at a reasonable distance from the centre of the sinc-function. For speed reasons we prepare a lookup table for the sinc function. The argument, \( x \) of the sinc-function can be expanded in terms of indices \( i \) and \( j \):

\[
x = x_i + x_j = i \cdot dx + j/N_{\text{int}} \cdot \text{Spread} \cdot dx
\]

Inserting this into \( \sin(\pi \cdot (x/dx-1)) \) yields

\[
\sin(\pi \cdot j/N_{\text{int}} \cdot \text{Spread})
\]

\( \text{Spread} \) controls the integration-width in terms of zero crossings of the sinc-function on either side. \( N_{\text{int}} \) is the number of integration points on either side. \( N_{\text{int}} \) should not be an integer number of \( \text{Spread} \) in order to avoid zeros, and there should be at least two points per cycle. For a real-valued data-set the convolution routine looks like this:

```plaintext
N_{\text{int}}:= 13;
\text{Spread}:= 5.0;
\text{Spread}_N_{\text{Int}}:= \text{Spread}/N_{\text{Int}};
for j:= -N_{\text{Int}} to +N_{\text{Int}} do
    W[j]:= \sin(\pi \cdot j/\text{Spread}_N_{\text{Int}});
for i:= 0 to \text{pred}(N) do begin
    y:= 0.0;
    \text{Norm}:= 0.0;
    for j:= -N_{\text{Int}} to +N_{\text{Int}} do begin  { Integrate region weighted with sinc function }
        x:= x_{\text{Range}.min} + (i + j*\text{Spread}_N_{\text{Int}})*dx;
        if (x >= x_{\text{Range}.min}) and (x <= x_{\text{Range}.max}) then begin
            yj:= CD.Get_Value(x).x;
            if abs(j) = N_{\text{Int}} then begin
                y:= y + 0.5*yj*W[j];
                \text{Norm}:= \text{Norm} + 0.5*W[j];
            end else begin
                y:= y + yj*W[j];
                \text{Norm}:= \text{Norm} + W[j];
            end;
        end;
    end;
    if \text{Norm} <> 0 then
        H[i]:= y/\text{Norm}
    else
        H[i]:= 0.0;
end;
```

(14)
Interference problems

The sub-sampling routine performs quiet well as long as the integration performs locally on the data-set, this is particularly true for complex-valued data, such as a spectrum. The main problem here is that in extreme cases the result can be zero because the integration sums negative and positive parts of the data-set. The problem arises on strong sub-sampling and, especially if the values are embedding a strong time-delay. Whenever you want to explicitly sub-sample a complex data set it is better to use Lin/Log Smoothing Processing because these routines make use of the Continuous Phase Technique, which circumvent these particular problems caused by interference.

An example may help to understand what is going on. Let us start with a data-set of a second order low-pass filter, which is going to be down-sampled as shown in figure 4. The original function is pictured on the left hand side as real and imaginary parts. In the middle the curves of the sub-sampled and original curves are compared as amplitudes. The agreement can be seen as fairly good for the used low density of points representing the original complex function. Further added is the associated time impulse response function of the original data-set. The blue marker indicates the right slope of the rect-function, which is the Fourier transformation of the sinc-function used for integration.

Figure 5 now displays the problem arising from integrating broadly heavily rippled real and imaginary parts due to an induced all-pass delay of 1s. Because the integration yields cancelling of positive and negative parts in the real and imaginary traces the amplitude of the sub-sampled data-set drops (centre, red). The same problem viewed in the time domain reveals, that because of the delay, the impulse responds is shifted to the right and will be truncated by the rectangular window. Of course, the more delay is present, the further right the impulse is shifted until the whole impulse is forced to zero yielding also zero amplitude.
**Sampling ranges**

In Vacs there are four sampling ranges, the limits are given by

1) the natural limits of the data-set

2) grounding, i.e. from zero to the maximum of the data-set

3) the limits of the process window

4) grounding, i.e. form zero to the upper limit of the process window.

**Sampling range given by data-set**

In the first case the sampling range is given by the bounds of the data set

![Figure 6: Left: Sampling a linear, right a warped distribution](image)

**Sampling range given by data-set but grounded**

For certain Fourier transformations the sampling range should start at zero. In this case the data-set is sampled from $x_{\text{min}}$ to $x_{\text{max}}$ but zero padded between 0 and $x_{\text{min}}$.

![Figure 7: Sampling and zero padding (grounding) of linear grid.](image)
As figures 7 and 8 demonstrate zero valued points are inserted between 0 and x_min. The sampling rate is \(dx\) and deduced from the data-set, however, interpolation can be necessary if zero is not an integral part of the data-set abscissa.

**Process Window**

If a Process Window is applied to a signal then Vacs samples the original data-set within new boundaries. The mechanics are similar to the sampling of the total data-set. The first point of the new data set is at the Process Window start, the last point at the Process Window end as shown in figure 9.

The number of samples, \(N\) is basically equal the number within the Process Window plus the neighboring two if cut by the Process Window, as given by formula (1). \(xRange\) is given by the Process Window. \(Delta.min\) is the smallest difference of abscissa points within the Process Window plus the neighboring two if cut by the Process Window. Depending on the process \(N\) may be modified according to function (2) and (3). Finally the new sampling rate, \(dx1\) is derived, according to function (4). Again, depending on the distribution of original points, either interpolation or convolution is applied.
### References

   ISBN: 0-521-37516-9
   ISBN 0.07-066454-4