# Plotting the performance diagram (R1)

The DLL includes five different functions, which provide you with the information required to plot the four families of curves which compose the diagram.

Each family is composed by up to fifteen curves, with the actual number varying with the fan and the type of the curve family. The functions shall be plotted point by point.

This is a guidance document giving a possible procedure to plot the fan performance diagram.

The first step is to define the number of points Npoints required to represent the curves smoothly enough. This can be as low as 20 for very small, low-resolution diagrams, or as high as 100 for large diagrams. The larger diagram shown by Ventil (the one called up from the Catalogue window) is plotted with 60 points along each curve.

The second step is to dimension two arrays, abscissas and ordinates:

*FlowVals(4 curve types, 15 curves, Npoints) As Double*

*Pressures(4 curve types, 15 curves, Npoints) As Double*

And an array of curve labels

*Labels(4 curve types, 15 curves) As Double*

Four extremes of the two scales should be defined as doubles and also initialised:

*MinFlow = 1e+9*

*MaxFlow = -1e+9*

*MinTot = 1e+9*

*MaxTot = -1e+9*

The minimum values must be larger than any possible value, and the maximum values smaller than any possible value.

Then the points can be calculated and plotted:

For each family of curves among CONSTANT SPEED, CONSTANT POWER and CONSTANT EFFICIENCY,

> Call the appropriate GET_GRAPH_RPM, GET_GRAPH_POWER or GET_GRAPH_ETA function
>
> This will return in the first position (OUT(0)), of the OUT array of double precision real values, an error code, while the second value (OUT(1)) is the number of curves to be plotted in the required family.
>
> For each curve with curve_nr from 1 to OUT(1)
>
>> read in the OUT vector the value of the constant level (be it speed, power or efficiency), in the element OUT(1+(curve_nr -1)*9+1), and assign it to Labels(curve type, curve_nr)
>>
>> read in OUT(1+ (curve_nr -1)*9+8) the value of the minimum flow to be plotted for this curve
>>
>> read in OUT(1+ (curve_nr -1)*9+9) the value of the maximum flow to be plotted for this curve
>>
>> read from OUT(1+ (curve_nr -1)*9+2) to OUT(1+ (curve_nr -1)*9+7) the value of the six coefficients of the fifth degree polynomial representing the curve:
>>
>> $y = \Delta p_{total}[Pa] = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + a_4 x^4 + a_5 x^5$
>>
>> For each flow step, from the minimum flow to the maximum flow of the curve, with step equal to the difference divided by the number of required steps Npoints –1,
>>
>>> calculate the flow and assign the flow value to FlowVals(curve_type, curve nr, point)
>>>
>>> If the calculated flow is smaller than MinFlow update this value
>>>
>>> If the calculated flow is larger than MaxFlow update this value
>>>
>>> calculate the value of the polynomial function and locate the total pressure value in Pressures(curve_type, curve_nr, point)
>>>
>>> If the calculated Total Pressure is smaller than MinTot update this value
>>>
>>> If the calculated Total Pressure is larger than MaxTot update this value
>>
>> End For
>
> End For

End For

To cover the remaining family of CONSTANT DB curves:

Call the GET_GRAPH_SOUND function to get a similar OUT vector.

The polynomial coefficients are meaningless.

The first position (OUT(0)) of the OUT vector contains an error code, the second value (OUT(1)) is the number of curves to be plotted in the required family.

For each curve with curve_nr from 1 to OUT(1)

> read in the OUT vector the value of the constant sound power level in the element OUT(1+(curve nr -1)*9+1), and locate it in Labels(curve type, curve nr)
>
> read in OUT(1+ (curve nr -1)*9+8) the value of the minimum flow to be plotted for this curve
>
> read in OUT(1+ (curve nr -1)*9+9) the value of the maximum flow to be plotted for this curve
>
> Call the PRESS_DB_CONST function, with Npoints as required points number, to get the vectors VX (flow values) and PY (total pressures). The value in VX(0) is the echo of the required dB, while the value in PY(0) is the actual number of points calculated by the function (could be lower than Npoints)
>
> Define the actual nr of points dBPoints(curve nr) = PY(0)
>
> For every point from 1 to dBPoints(curve nr)
>
> > Copy the flow value from VX(point) in FlowVals(curve type, curve nr, point)
> >
> > If the calculated flow is smaller than MinFlow update this value
> >
> > If the calculated flow is larger than MaxFlow update this value
> >
> > Copy the total pressure from PY(point) in Pressures (curve type, curve nr, point)
> >
> > If the calculated Total is smaller than MinTot update this value
> >
> > If the calculated Total is larger than MaxTot update this value
>
> End For

End For

Define your plot window

Draw the actual plot-enclosing rectangle, which may be smaller than the actual plot window to leave space for scale numbers.

Define two transformation functions, to convert physical co-ordinates (flows and pressures) into graphical co-ordinates x and y, leaving margins around the actual plot rectangle for the scale numbers, and applying logarithmic transformations if required. Remember that pressures increase from bottom to top, while often plot co-ordinates increase from top to bottom.

For each family of curves

> Define the pen colour and thickness
>
> For each curve in the family
>
> > If the curve label in Labels(curve_type, curve_nr) is zero, there are not any more curves in the family; skip to the next family
> >
> > Locate the pen on the first point (x(Flow, Total), y(Flow,Total))
> >
> > For each point from 2 to Npoint
> >
> > > If the curve is a SWL curve, and the point nr > dBPoints(curve_nr) then stop plotting and skip to the following SWL curve
> > >
> > > Draw a line to the point (x(Flow, Total), y(Flow,Total))
> >
> > End For
> >
> > At the end of each curve locate appropriately the cursor and write the curve label.
>
> End For

End For

Plot axes titles, ticks and labels