Fitting PACS ramps with analytical models. Part I

Martin Groenewegen (ICC KUL)

DOCUMENT CHANGE RECORD

Version	Date	Changes	Remarks
Draft 0	16-February-2004	–	New document.
Draft 1	18-February-2004	small throughout	Comments by PR. Uploaded to document tree.

Reference Documents

RD1 – Alternative concept for data reduction/compression, PACS-ME-TN-040, Draft 1, May 2003

1. Introduction

This technical note describes the first attempt to fit Spectroscopic PACS data with an analytical model within the framework of the PCSS/IA, as presented during ICC meeting #18 (29-30 January 2004).

Using an analytical model to describe PACS ramps could be useful for OBSW data compression or for the on-ground data reduction. Also it could be useful in numerical simulations of ramps, since the ramps are generally not straight lines.

This document accompanies the two Python scripts PyRamp1Model.py and PlotPyRampModel.py that, respectively, implement the mathematical model described below, and plots the output generated.

2. Model

The (only) model available to describe the shape of the PACS ramps is derived by Albrecht Poglitsch in a unnumbered document (a verbal description and some plots can be found in RD1 however). It is a combination of an analytical description of the CRE circuit plus an ad-hoc description of the “bump”.

The functional form derived:

$$V(y) = A\left(\sqrt{\frac{F}{F_0R_1}/\left(\left(\frac{F}{F_0R_1}/V_\mathrm{b}^2+\frac{F}{F_0R_0}\right)\exp\left((2\,\frac{F}{F_0R_1}\,y)/(C+C_\mathrm{f}(1+A))\right)-\frac{F}{F_0R_0}\right)}-V_\mathrm{b}\right)$$

$$+a+b\,(1-\exp(-t/\tau))$$

with t the time and $y=t/256$ (256 Hz read-out frequency). A is de open-loop cascode gain, C_f the feed-back capacitance, C the parasitic capacitance, F/F_0 represents a flux scaling factor, V_b the bias voltage, and R_1 and R_0 are resistors to model the equivalent circuit describing the detectors. a is a constant, and b,τ are a ad-hoc description of the “bump”.

In the first implementation of this *physical* model the following *mathematical* representation was chosen, with 8 parameters:

$$f(x,p) = p_0\left(\sqrt{p_1/\left((p_1/p_2^2+p_3)\exp\left((2\;p_1\;y)/p_4\right)-p_3\right)}-p_2\right)$$

$$+p_5+p_6(1-\exp(-x/p_7))$$

with $y = x/256$. In other words:

$$p_0 = A$$
$$p_1 = F/(F_0R_1)$$
$$p_2 = V_\mathrm{b}$$
$$p_3 = F/(F_0R_0)$$
$$p_4 = C + C_\mathrm{f}(1+A)$$

$$p_5 = a$$

$$p_6 = b$$

$$p_7 = \tau$$

Some expected values as mentioned in Poglitsch's document are A = 300, R_1 = 54 GΩ, R_0/R_1 = 0.0015 (as measured), C = 5 pF, $F/F_0 = 1$..

2. IA/Python

The necessary files are located at the Leuven CVS server at / develop/ pcss/ herschel/ pacs/ scripts/ obswscripts

The related files are PyRamp1Model.py, input.list, Ramp1Model.java and PlotPyRampModel.py.

Ramp1Model.java is the actual implementation of the mathematical model as an extension of the NonLinear java class. Compile it with "javac Ramp1Model.java"

input.list (the name is arbitrary) contains the pathnames to the file(s) you want to analyse. This needs to be MPE data, as there is only this reader available in the PCSS for the moment[1].

PyRamp1Model.py does the actual fitting. There are loops over the file, the module, the channel and the ramps (loop-index i, j, k, l, respectively). In JIDE one can run the script as is, or execute it line-by-line and set the appropriate i, j, k, l-index manually (and skip the **FOR**-loop) to select a particular ramp. Similarly, selected ramps (and fits) are plotted to the screen, and this is selected by two **IF**-statements.

A note on performance. When running the script on larger datasets comment out all print statements to standard output, and limit the number of plots to the screen ! This optimisation trick was realised the day before the ICC meeting and so the results presented there were based on analysis of one module and one detector per file, i.e. 256 ramps of 64 files[2].

The output of PyRamp1Model.py are 3 files, "output_of_ramp1model.dat", "failed_of_ramp1.dat" and "weird_of_ramp1.dat". The first file describes the input and output of the fitting in the majority of cases. The second file list the indices l, k, j, i of those ramps where the fitting failed (i.e. A java exception occurred), and the third file list the indices of those ramps with a very high (arbitrarily chosen) χ^2.

The file "output_of_ramp1model.dat" is read in by PlotPyRampModel.py. It automatically generates plots of all fit parameters against all input parameters, and all output parameters against all other output parameters.

2. Results and future work

The main aim of this first round of working with analytical models was "can we do it?" The answer is clearly yes; we can process sizeable amounts of data and analyse the output within the framework of PCSS/IA.

Examples of fits are given in Figures 1 and 2. Figure 2 shows the need for a "edge detection" algorithm. Figure 3 shows one result of the fitting where for the about 15 000 successfully fitted ramps the parameter p_2 is plotted versus the bias voltage. Although there is considerable scatter the two are strongly correlated, as they should be, according to the mathematical representation of the physical model.

Issues for the future:

[1]The contents as uploaded to the CVS server contains the 64 files analysed in the first attempt which are from the MPE April 2003 data that contain T*n256*.dat in the file name except T19bb50b0t025c3n256_3.dat, T18b20t05c01n256_6.dat, T18bb33b40d40t025c3n256_1.dat and T18bb33b30t025c1n256_1.dat, that had a deviating file length.

[2]Just to give an idea, printing 2 lines per fitted ramp, that is printing 30 000 lines to standard output, or not, makes a difference of a factor 20 in execution time !

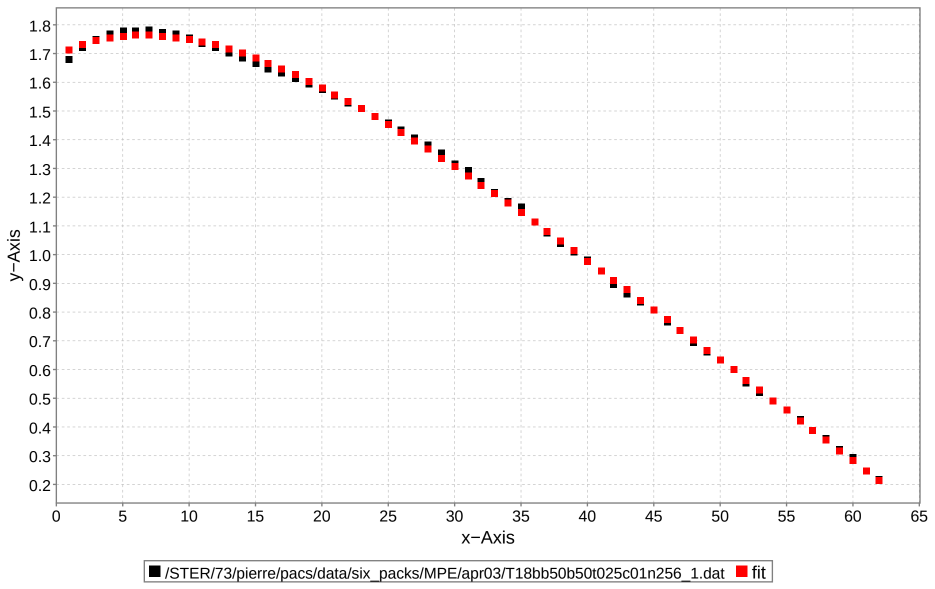

Figure 1: Data (black) and fit to the data (red). Volts versus NDR number (first and last NDR are always removed).

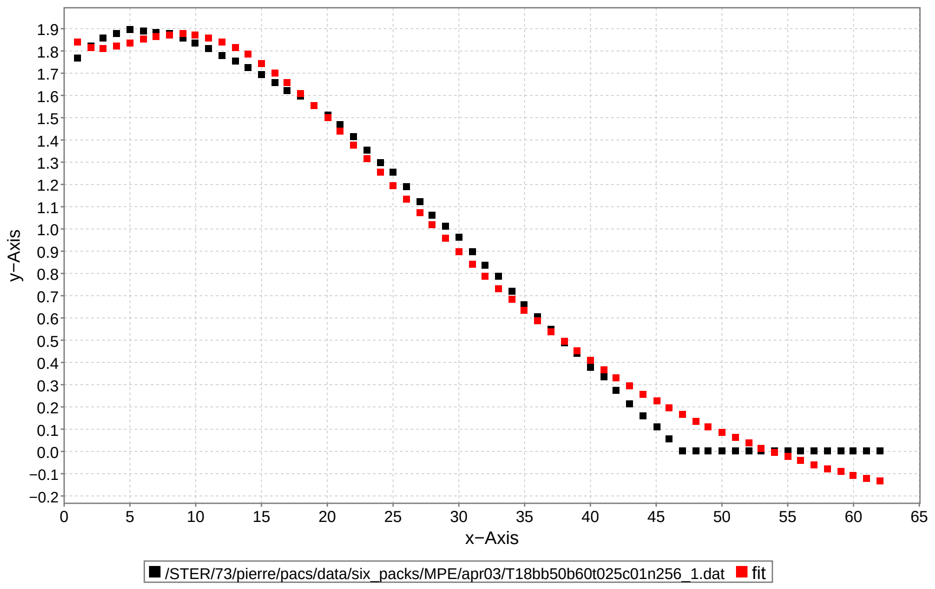

Figure 2: Data (black) and fit to the data (red). Saturation occurs at the end of the ramp which was not considered in the fitting.

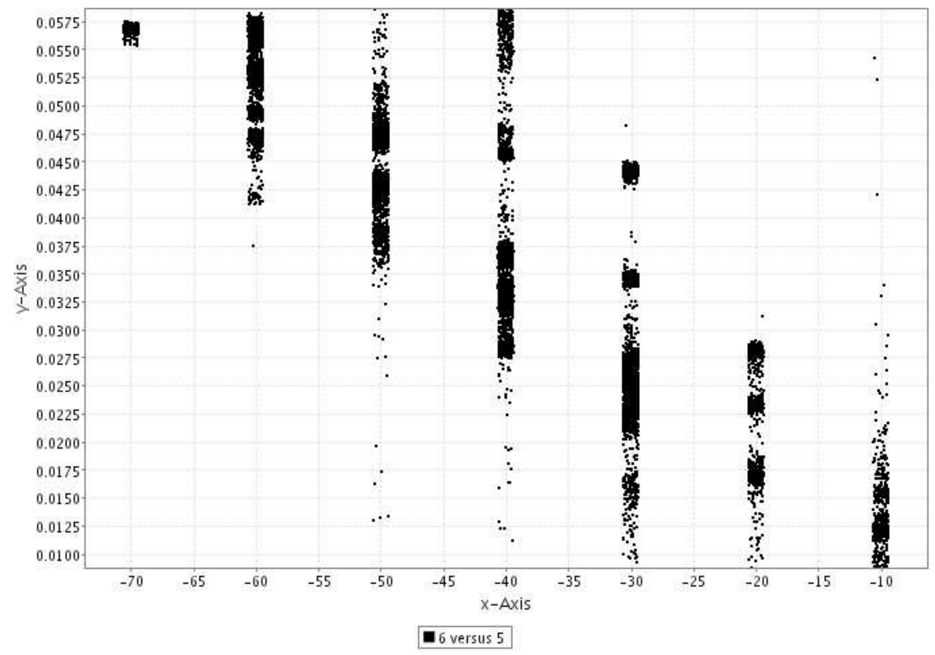

Figure 3: Parameter p_2 (in Volts) versus bias voltage (the MPE reader in the PCSS lists them as negative value, and in mV), which should be 1-to-1 correlated. The correlation is strong but not perfect.
($\pm$5% outliers are not plotted and some random noise has been added in x- and y-direction).

- Try out a different mathematical representation of the physical model.

  Take care that some physical parameters which should be positive at all times are mathematically described in such a way that the fitting does not allow negative values for this parameter.

- Parameters like the bias and the feed-back capacitance have so far not been *fixed* to the input values. This a-priori knowledge has been used to set the initial guess, but then left free (as evident from figure 3 for example).

- Now that the problem of performance with JIDE seems to have been solved (or at least using the workaround by not printing to standard output), analyse more (all) detectors and channels and more (all) files from the MPE april 2003 dataset.

- An “Edge detection” algorithm is needed to detect (numerical) saturation, and glitches.

- Check in detail why the fitting failed (wrong initial guess ?), or provides extreme χ^2 values.

- Related to the above items. Can we create “calibration tables” to optimise the initial guesses, and/or fix some parameters to decease the number of free parameters, and to make the fitting faster and more robust (which will be a critical issue when such a scheme would be implemented in the OBSW).

- Considering all ramps for a given file, what is the distribution of the derived parameters ? Is it Gaussian ? Define a Signal-to-noise, similar to the one use in the (sub)ramp fitting.