Cremat Charge Sensitive Preamplfier/Shaper Overview & Calibration

July 05, 2024

Luc Barrett
me@lucbarrett.info
Kumar Lab 028

Contents

1 Charge Sensitive Preamplifier

1.1 Introduction and Purpose

A Charge Sensitive Preamplifier (CSP) is a device used for detecting electrical pulses from detectors. A CSP can take in a pulse of current, and in its most basic form, the result is proportional to the integral of charge passing through

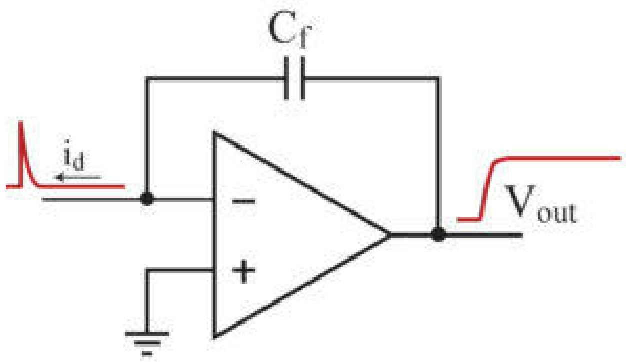


Figure 1: Simplified circuit diagram

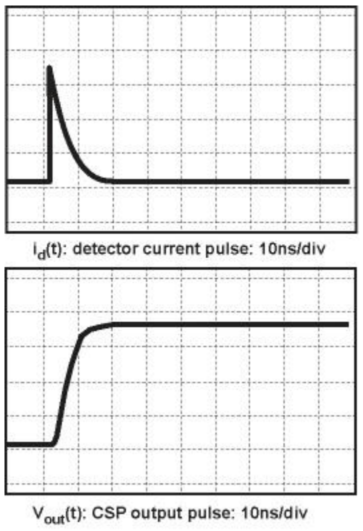


Figure 2: Detector pulse and resulting signal from CSP

This is generally useful as some types of detectors, for example a PMT, may release a small burst of electrons (comparable to a delta function spike in current), which is much harder to measure than a voltage difference.

This circuit arrangement is generally problematic, as multiple pulses will keep causing an increase to an unbounded size. This can be resolved by adding a bleed resistor

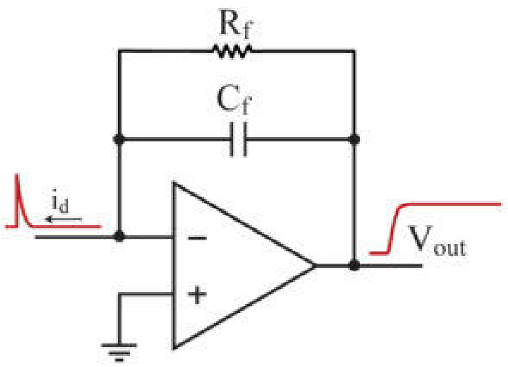


Figure 3: More typical circuit digram including the bleed resistor R_f

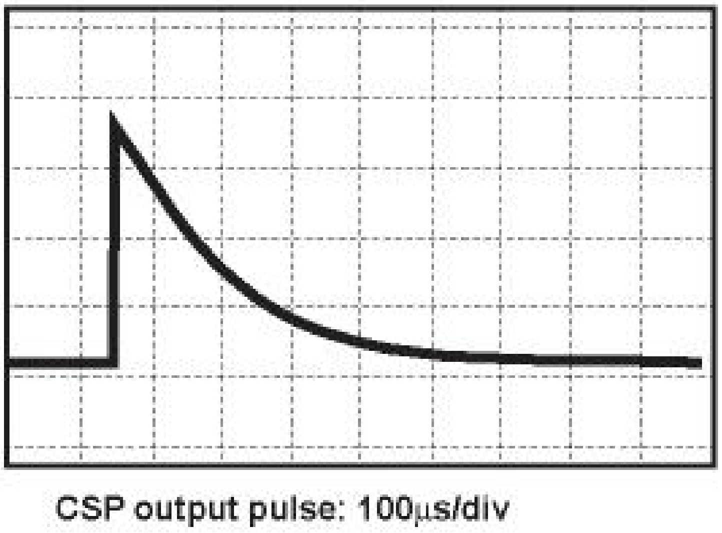


Figure 4: Resulting pulse from introducing the bleed resistor

This pulse will decay with a time constant of $\tau = C_f R_f$. It is not dependant on any other properties of the pulse, though a slow rise time will affect the overall shape.

1.2 Calibration

To know the gain of the CSP, we must inject it with a current pulse with a known charge. To do so, a capacitor can be placed in series with a function generator producing a square wave. Then, when there is a change in voltage, the amount of charge injected is $Q = C_f \Delta V$ (discussed more in the next section). In the evaluation boards, there is one of these capacitors with $C_f = 1$ pF already in the test input.

It is not usually necessary to calibrate the preamp on it's own, as it will generally be used in series with a shaper. It is also a more challenging task to measure the ampltiude on it's own. A shaper (as will be seen in Section 2) provides a gaussian output, making an amplitude and determining an uncertainty easy with a curve fit.

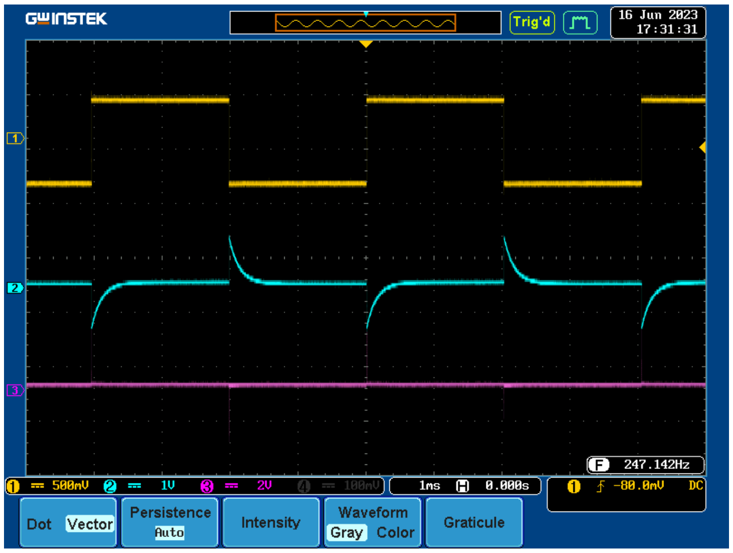


Figure 5: View from oscilloscope of function generator signal (yellow), preamp (blue), and shaper (pink). The pulse width of the shaper is so small it is difficult to see in this image; Refer to Figure 7

2 Shaper

2.1 Introduction and Purpose

The CR-200 is a Gaussian shaping amplifier module, and is used to read out the "tail pulse" signals such as from charge sensitive preamplifiers, PMTs, and other similar detection circuits. Gaussian shaping amplifiers are also known as 'pulse amplifiers', 'linear amplifiers', or 'spectroscopy amplifiers' in the general literature. They accept a step-like input pulse and produce an output pulse shaped like a Gaussian function (bell curve). The purpose of these amplifiers is not only to transform the shape of the event pulse from a tail pulse to a bell curve, but also to filter much of the noise from the signal of interest. Use of shaping amplifiers will reduce the fall time of the pulse signals, reducing the incidence of pulse 'pile up', and improve the signal-to-noise of the detection system.

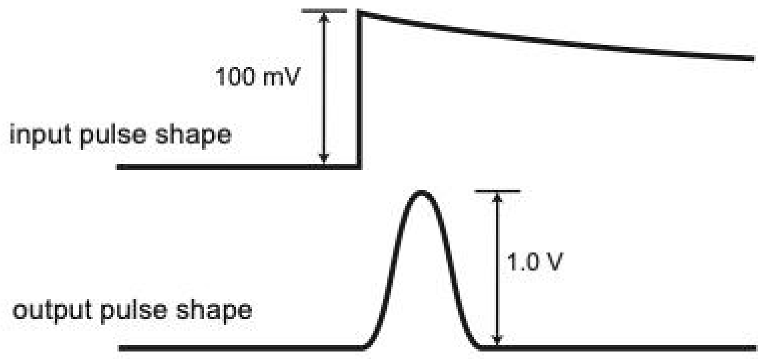


Figure 6: Response of shaper unit

The shaping time is defined as the time-equivalent of the "standard deviation" of the Gaussian output pulse. A simpler measurement to make in the laboratory is the full width of the pulse at half of it's maximum value (FWHM). This value is greater than the shaping time by a factor of 2.4. For example, a Gaussian shaping amplifier with a shaping time of 1.0 μs would have a FWHM of 2.4 μs.

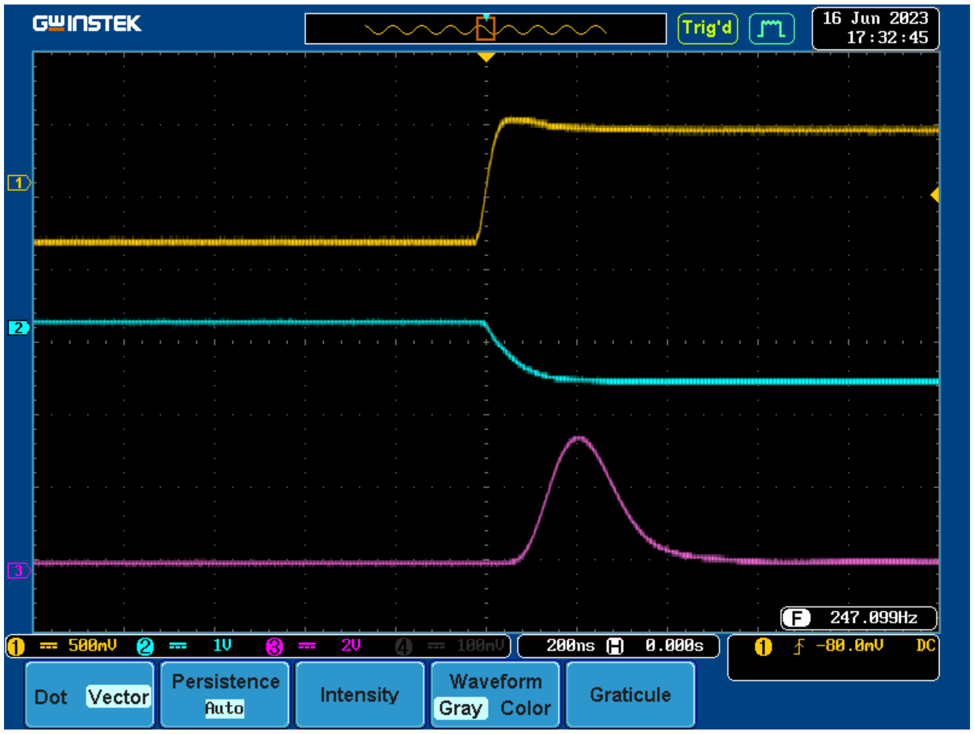


Figure 7: View from oscilliscope, with same color coorrespondance as before. The view is zoomed in to make the signal of the shaper clear

2.2 Calibration

The shaper has it's own gain that can be adjusted with controls on the front of the device. There is a fine gain along with two switches that can be toggled. Per the specification document, each switch adds a gain of a factor of 10. Any time the fine gain is changed, the stack will have to be recalibrated as there is no fixed position. Theyre generally calibrated for three gain switch configurations (x1, x10, x100) so these can be switched on and off as needed,

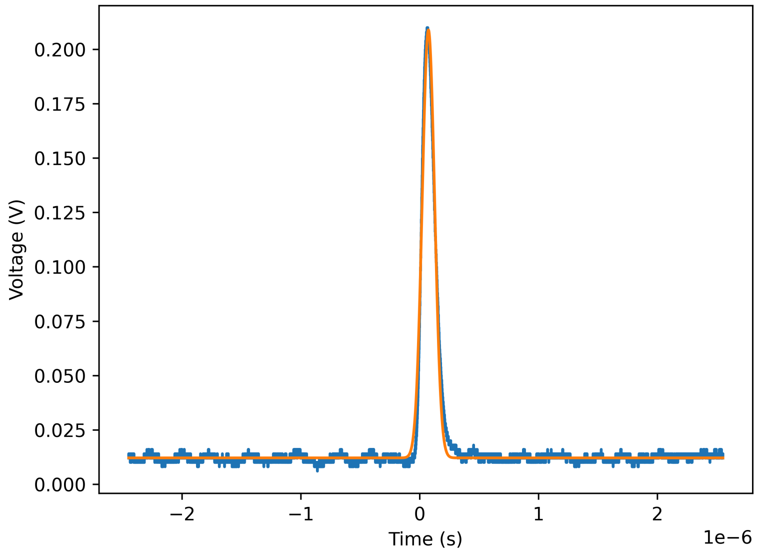


Figure 8: Typical fit to a shaper signal. $A = 0.197 \pm .0001$ V, $\sigma = 52$ ns

$$f(x) = Ae^{-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}} + c$$

Since the resulting output signal of the shaper is approximately gaussian, a calibration with uncertainty can be performed by injecting an input charge (calculated with the C_f from the preamp and a square wave as described) and doing a gaussian fit to the output signal and extracting the amplitude A. This is generally an easier way to get good data than calibrating both individually, so each particular "stack" (particular preamp and shaper unit) should be calibrated together.

2.3 Jumper Wire

In the absense of a CR-200X module, as in our case, a jumper wire had to be installed that was previously missing. This was done according to the documentation of the unit. If a 200X module is added in the future this will need to be removed.

3 Calibration Code

Calibrating a preamp/shaper stack can be done quickly and accurately using the PicoScope and it's python API. Found on the github (https://github.com/lab57/lab28-calibrate-cremat-stack), there is a jupyter notebook that performs the calibration. Here I'll describe how this works. The instructions for doing it are mostly given in the notebook itself, as it's intended to be self guided.

3.1 PicoScope API

There are several functions written in `./picoScopeControl.py` which aid in interacting with the API for this use. It uses the picosdk.ps3000a module (with corresponds to the unit we have in the lab). This code is commented but generally should not be edited.

Ensure the picoSDK is installed before running any part of the notebook (instructions within the notebook).

3.2 Wiring

It's important to connect the components together correctly to perform the calibration. The steps are outlined here:

- Connect the output of the signal generator on the picoscope to the CSP test-in
- Connect the output of the CSP to the input of the shaper
- Connect the output of the shaper to an input channel of the picoscope (default is channel C)
- **Connect a function generator to the ext. in of the picoscope**
  - This is important. The code assumes things are triggered on an external trigger
  - This is done because there can be large amounts of noise when doing low-amplitude pulses (for example when measuring high gain without saturating) and doing a regular trigger makes this difficult
  - As of writing there is one on the side of the optical table next to the cryostat. It can be wired into the picoscope using the BNC cables that terminate there (I use cable 5). This makes it easy to wire without moving stuff around (if it's no longer there, sorry! D:)
  - I set it to 1V, 500hz square wave and that worked well for me. Most importantly it should be a square wave.

3.3 Metadata

After opening the device, the next cell is to configure the metadata. This will be used for naming the plot file and in the plot itself. For example,

```
metadata = {
  "switchconfig" : "00",
  "stacknum"     :   2,
}
```

Replace switch config with the configuration of gain switches you're running it for (usually 00, 01, 10), and the stacknum with the ID number of the stack youre testing (ex preamp and shaper 1 go together, and preamp and shaper 2 likewise).

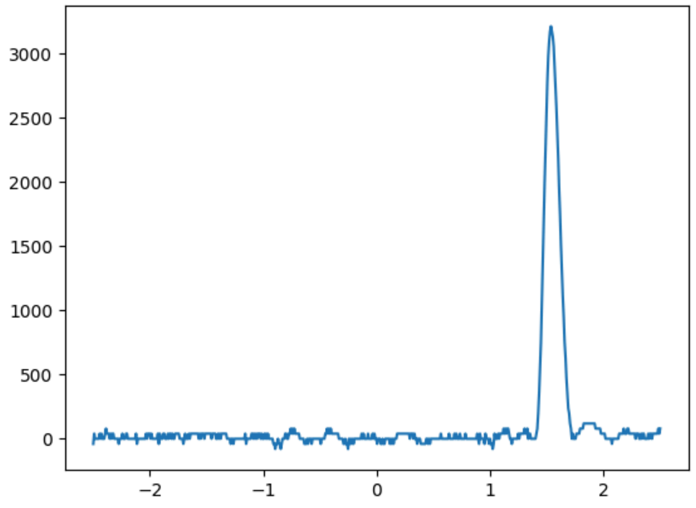
3.4 Test Measurement and Fit

```
t, data = takeDataVal(chandle, .5, rangeMap[1], trigger=0.5, window=5e-6)
pps, errs = fitGaussian(t, data)
plt.plot(t, data)
plt.plot(t, gauss(t, *pps))
plt.show()
```

This runs a short test to make sure the results look as expected. A quick explanation for each parameter in `takeDataVal`:

- chandle is the pointer to the device
- 0.5 is the amplitude of the square pulse to send, in volts.
- trigger sets the threshhold (should be 0)
- window sets the width of the window, should be 5e-6 generally to match the later calibration

The test output should look something like this



If it looks different, try slightly expanding your time window (in case theres a longer than expected delay) and check your wiring. You may find it easier to use the PicoScope software to debug wiring. (make sure you run the ‘close device’ cell at the end of the notebook before opening picoscope)

3.5 Run Measurements

This is the main part of the calibration. This will sweep across the voltages defined in `input_vs`, record the output of the shaper for that input, and store it in `datas`. Then it will calibrate the gain (fit a gaussian to it) using `getGains()` and put it in `gains`. It will show a simple scatter plot of the results so it can be verified it looks as expected.

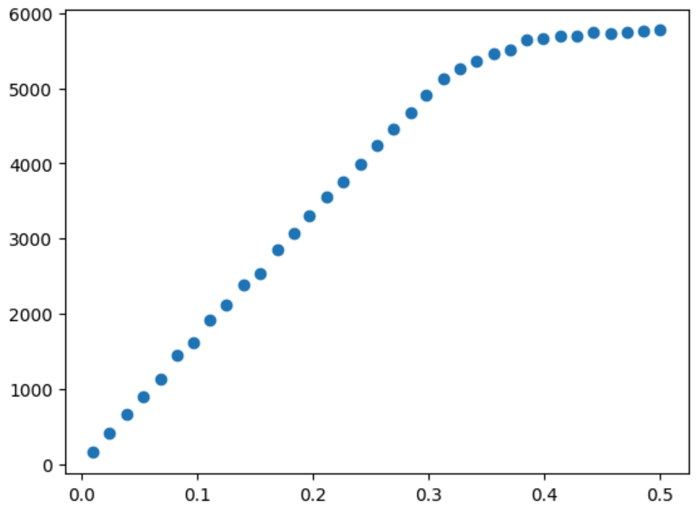


Figure 10: Plot of gains. Note it saturates when the output goes past 5V (5000mV)

3.6 Plots

Following this, the next cell plots these values and does a linear fit to extract the overall gain value.

```
indicies = np.where(np.asarray(gains) > 5000)[0]
saturation_index = indicies[0] if len(indicies) > 0 else len(input_vs)
```

An important first step is to find where the output saturates, or becomes nonlinear. This is the maximum output of the device. We don't want to include these in the fit, so we find the index where the output goes above 5V and only perform the fit up to that point.

Following that, the fit is performed and plotted along with the data. See below what might be expected.

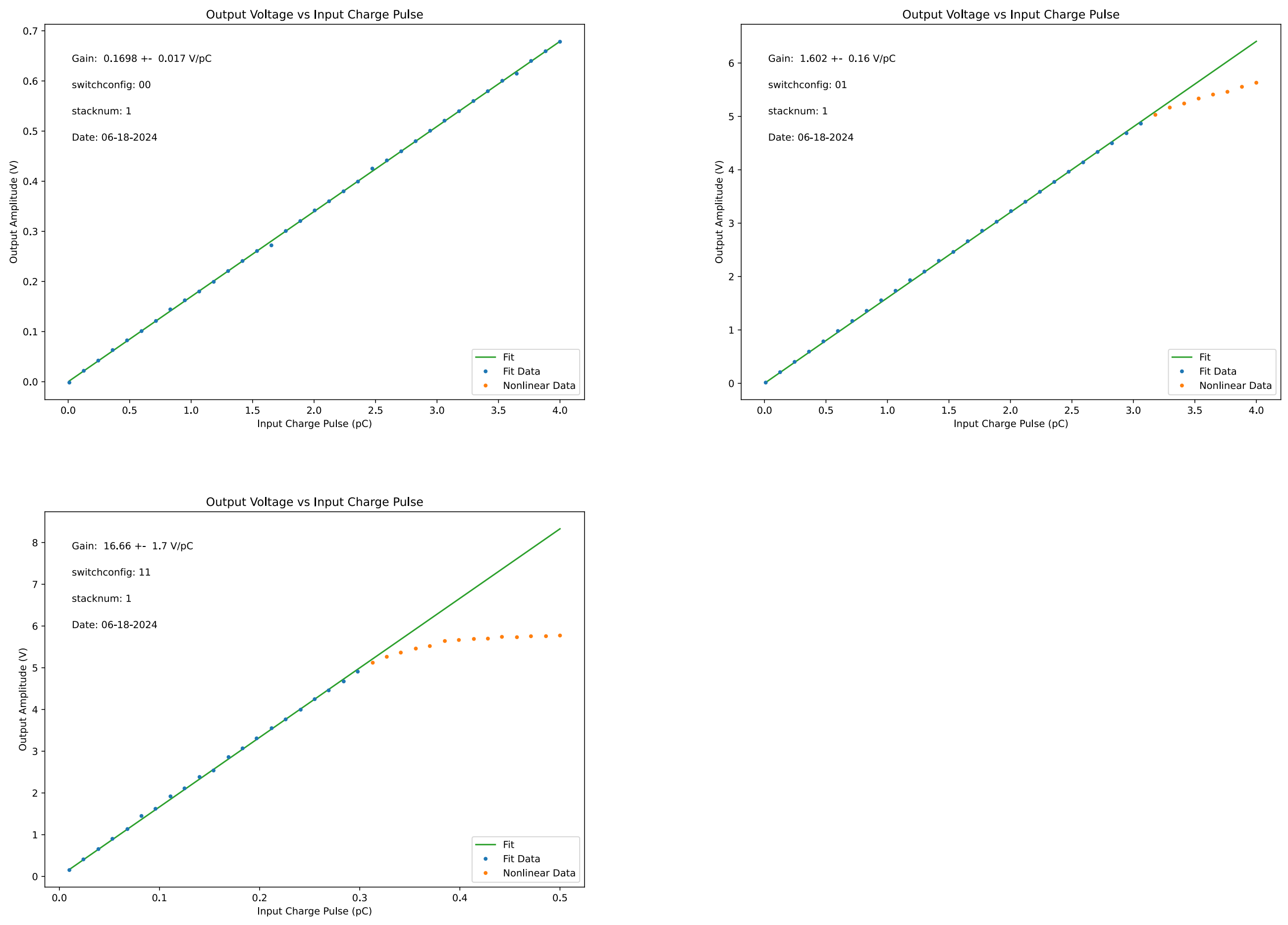


Figure 11: Example calibration data from 06/17/2024

Notice that for each enabled gain switch, the gain is approximately 10x higher as expected.

Our error is dominated by the error in the 1pF capacitor used to make the charge pulse. Its ±10%, and since this is used as a conversion factor, more data will not help this result. There are ways to reduce this by building a larger circuit, but this isn't straight forward and it's been decided is probably not worth it.

These three calibration graphs should be generated for each stack when needed. (If fine gain is never moved, and on some regular time interval to be decided)