The calibrated parameter plugin

Ron Fox

The calibrated parameter plugin

by Ron Fox

Revision History

Revision 1.0 February 8, 2008 Revised by: RF
Original Release

Table of Contents

List of Figures

Chapter 1. Introduction

This document describes a plugin for SpecTcl that provides the capability of fitting calibrations to raw parameters and then using those fits to create calibrated parameters.

The remaining chapters of this document:

1. Describe how to obtain and install this plugin.
2. Describe how to load the plugin and how to use it.
3. Provide reference material describing the SpecTcl commands added by the plugin.
4. Describe a sample graphical user interface to the plugin that is installed along with the plugin itself.

1.1. What does the calibrated parameter plugin provide?

In many applications, a simple linear transform relates raw ADC values to some more meaningful value. For ADC's this value is often the energy left in a detector by a detected particle or photon. For TDC's this value is usually a time relative to some other 'happening' in the event.

The Calibrated parameter plugin provides a means to compute and apply these linear transforms to any raw parameter value to produce a value with 'real world' units, or a *calibrated parameter*. The plugin supports creating the calibration function as well as providing, and registering an event processor to apply calibration functions to raw parameter to produce calibrated parameters.

The plugin adds two commands to SpecTcl. **calibrationfit** accumulates takes a set of raw parameter/actual parameter pairs, and produces a linear least squares fit that can be used as the calibration function for calbirated parameters. The **calibparam** command associates a raw parameter, with a fit to direct the calibration event processor to produce a new calibrated parameter.

Chapter 2. Obtaining and installing the plugin

The plugin source code can be downloaded from http://www.sourceforge.net/projects/nsclspectcl It is a tarball nmed `calibratedparams-`*version*`.tar.gz` in the `plugins` package.

Download the source tarball. In the previous paragraph: *version* is the version of the plugin in the form: `major.minor-editlevel` e.g. `1.0-001`. Where the `-editlevel` is optional. Suppose for the sake of this example, that the version is 1.0-001. Download `calibratedparams-1.0-001.tar.gz` to some work directory.

Unwrap the tarball e.g.:

```
tar xzf calibratedparams-1.0-001.tar.gz
```

In this example, the directory `calibratedparams-1.0-001` is created.

cd, to the desired directory, configure and build the software using the usual **configure** and **make install**.

The **configure** script configures the software build system and creates the final makefiles. It understands quite a few options. The ones important and unique to the plugin are:

`--prefix=`*`install-path`*

Determines where the plugin will be installed. Note that the actual plugin software is installed in `install-path/TclLibs/calibration`.

By default, SpecTcl is also assumed to be installed in *`install-path`*. Note that The TclLibs directory and one level of subdirectory are automatically part of the SpecTcl package load path.

`--with-tcl-header-dir=`*`tc.h-path`*

The **configure** script searches for the Tcl header file `tcl.h` in several popular directories. If it is not able to find it, you can use this switch to give it the the directory that contains it.

`--with-spectcl-home=`*`spectclhome`*

While I encourage you to install the plugin in the SpecTcl installation directory, It is not necessary that you do so. If the value of `--prefix` is not the same as the value of the `SpecTclHome` directory when you run SpecTcl, use this switch and provide the correct *`spectclhome`*

`--with-gsl-home=`*`path`*

The software uses the Gnu Scientific Library (gsl) to perform its fits. On Debian linux, this is the package `libgsl0-dev`. If your installation of the gsl does not make the header and libraries findable, you should use this switch and set *`path`* to the top level directory of the GSL installation. The header is then assumed to be in *`path`*`/include` while the library is assumed to be installed in *`path`*`/lib`

Once you have configured the software, install it using **make install**. The example below shows how to install version 1.0-001 of the plugin at the NSCL for SpecTcl 3.2.

Example 2-1. Installing the plugin for SpecTcl 3.2 at the NSCL

```
tar xzf calibratedparams-1.0-001.tar.gz
cd calibratedparams-1.0-001
./configure --prefix=/usr/opt/spectcl/3.2
...
make install
...
```

Chapter 3. Loading and using the plugin

The calibrated parameter plugin is a Tcl loadable package which is a compiled extension.

If you install the plugin in SpecTcl's installation directory, you will not need to do anything to prepare to load the plugin. If you installed the plugin anywhere else, you will need to add the directory the plugin is installed in to the SpecTcl package load path.

You can do this by using **lappend** to append that directory to `auto_path`. For example:

Example 3-1. Adding the plugin directory to the package load path

```
lappend auto_path [file join ~ calibratedparameters]
```

Once the plugin can be found it can be loaded via the Tcl **package require** command:

Example 3-2. Loading the plugin

```
package require Calibrations
```

3.1. Using the plugin

The sequence for using the plugin, once it is loaded is

1. Create a calibration fit that will provide a calibration function.
2. Add points to the calibration fit. Each point consists of a raw parameter value and the corresponding desired calibrated value. Often fit points are gotten by identifying features in spectra of the raw parameter (e.g. known energy peaks in gamma spectra).
3. Ask the calibration fit to perform itself. Performing a calibration fit does a linear least squares fit to the fit points computing the slope and offset of the best fit line throug the points you provided.
4. Create a calibrated parameter by applying your calibration fit to a raw parameter to produce a new parameter. Once this calibrated parameter has been created it looks to SpecTcl like any other parameter; Spectra can be created, pseudo parameters can be scripted using it and so on (creating a calibrated sum parameter from several calibrated parameters e.g.).

The Example below shows this process for calibrating a parameter named `adc1.00` to produce a parameter named `adc1.e.00`. Don't take the actual calibration points too seriously.

Example 3-3. Calibrating a parameter

```
calibrationfit linear   testfit
calibrationfit -add     testfit  {0.0 0.0}  {1.0 2.0} {100.0 202.0}
calibrationfit -perform testfit

calibparam -create adc1.e.00 1234 adc1.00 testfit KeV
```

The first three lines of the example above create a fit named `testfit` add three points to the fit that describe the relationship between channel values and energies. The relationship is a linear with intercept very nearly 0.0 and slope very nearly 2.0. The **calibparam** command creates the new calibrated *`adc1.e.00`* parameter as paramteer number `1234` computing it by applying the fit `testfit` to the raw parameter *`adc1.00`* for each event. The `KeV` indicates that the units for this new parameter will be KeV.

The fit is a snapshot of the fit. If you later delete the fit, the calibrated parameter will continue to function on the original fit data. If you later re-create the fit with a different set of values, the parameter will continue to compute on the old fit. To change the fit you would need to delete and recreate the calibrated parameter. This is a compromise that is mandated by performance considerations (it would be expensive to have to locate the correct fit for each event, rather than storing a copy of the fit itself).

Chapter 4. Calibrated parameter plugin command reference

calibrationfit

Name

`calbrationfit` — Manipulate fits used to compute calibrated parameters

Synopsis

calibrationfit **`?-create?`** ***`type name`***

calibrationfit **`-list`** ***`?pattern?`***

calibrationfit **`-delete`** ***`name`***

calibrationfit **`-perform`** ***`name`***

calibrationfit **`-add`** ***`name point1 ?point2...?`***

calibrationfit **`-evaluate`** ***`name xValue`***

DESCRIPTION

The **calibrationfit** command manipulates the set of fits that are used to compute calibrated parameters. The fitting software is extensible to any sort of calibration fit that can be computed. At present, however only `linear` fits are supported.

The general form of the **calibrationfit** command is the command, followed by an option that describes the action the command will take, followed by the parameters that action needs.

The options, their actions and the parameters they need are documented in the OPTIONS section below.

Fits have state associated with them. The state determines what actions you are allowed to perform. The states are: `accepting` and `performed`.

OPTIONS

?-create? ***type name***

Creates new fit. The option `-create` is optional. if not provided, this is the default action. The fit subsystem is extensible to support any type of fit. Fit types have names. The *`type`* parameter is the fit type. At present this can only be `linear`. The *`name`* parameter is the name to assign to this fit. The fit name must be unique amongst all fits and will be used to refer to this fit in future **calibrationfit** and **calibparam** commands.

The initial state of a fit is `accepting` indicating that it is capable of accepting fit data points.

-list ***?pattern?***

Lists the known fits. If *`pattern`* is supplied, it is a pattern with *glob* wild cards that restricts the listing to those fits whose names match the pattern. glob pattern wild-cards are essentially those supported by filesystem matching wild cards in the Linux command shells.

The command returns a (possibly empty) list of fit descriptions. Each fit description is itself a sublist containing in order the *`fit-name`*, *`fit-type`*, *`fit-state`*, A list of the points used to compute the fit; each point is itself a two element raw/real coordinate pair, if the fit state is `accepting`, the fit description contains a final empty list element. If the state is `performed`, the final list element contains a list that is a set of name, value pairs that define the computed fit parameters and the Chisquare. The name for the linear fit are: `slope`, `offset`, and `chisquare`.

`-delete` *`name`*

Deletes the fit named *`name`*. Note that calibrated parameters are created with a copy of the fit. To stop a calibrated parameter from computing, you will need to delete it.

`-perform` *`name`*

Peforms the fit. If there are sufficient points, the fit parameters are computed, and the state of the fit becomes `performed`. The command returns the list of fit parameter name/value pairs. For the linear fit type these are: `slope`, `offset`, `chisquare`.

`-add` *`name point ?...?`*

Adds calibration points to a fit that is in the `accepting` state. Any number of points can be added to the fit. At present, there is no way to edit the set of points, other than to delete and re-create the fit. Each point is a Tcl list of two elements, raw value and corresponding calibrated value.

`-evaluate` *`name raw-value`*

This action is only available for fits that are in the `performed` state. The *`raw-value`* is passed to the fitted function computed for *`name`*. The resulting calibrated value is returned as the command's value.

calibparam

Name

`calibparam` — Create, list and manipulate calibrated parameters.

Synopsis

calibparam `?-create?` *`name number raw fitname ?units?`*

calibparam `-list` *`?pattern?`*

calibparam `-delete` *`name`*

calibparam `-refresh` *`?pattern?`*

DESCRIPTION

The **calibparam** command creates and manipulates calibrated parameters. A calibrated parameter is a computed parameter that depends on a base, or raw parameter and a calibration function or calibrationfit. Calibrated parameters are implemented as event processors and run at compiled speeds.

The base parameter for a calibrated parameter can be any parameter known to SpecTcl at the time the calibrated parameter is created with the exception of scripted pseudo parameters, which are always last in SpecTcl's evaluation chain in this implementation of SpecTcl.

The first word of the **calibparam** command following the command keyword is an option switch that defines the action the command will be performing. These options and their actions are described in OPTIONS below.

OPTIONS

`-create` *`name number raw fitname ?units?`*

Creates a new calibrated parameter. The `-create` option is not required. Creation is the default action if no initial option is present. *`name`* is the name of the new calibrated parameter and *`number`* is the SpecTcl parameter number associated with that name. *`number`* must be unique as must *`name`*.

`raw` is the name of the raw parameter to which the fit named *`fitname`* will be applied to compute values of the *`name`* calibrated parameter. The optional *`units`* parameter allows you to associate

units of measure with the new parameter *name*. These units of measure will be displayed on axes of spectra that depend on this parameter.

`-list` *?pattern?*

Lists the calibrated parameters that are now active. If the optional *?pattern?* is present, only parameters whose names match the *pattern* will be listed. The *pattern* can contain all the wildcards recognized by the unix shells when expandind/finding filenames. If *pattern* is missing it defaults to `*` which lists all parameters.

The list is returned as the command result and therefore can be manipulated by other commands. The list is a well formed, possibly empty Tcl list. Each element of the list is a sublist that describes one parameter. The sublists have five elements, the name of the parameter, the number of the parameter, the name of the raw parameter, the name of the fit used to compute the parameter, and the units or an empty string if no untits were defined.

`-delete` *name*

Deletes the calibrated parameter *name*

`-refresh` *?pattern?*

Refreshes the calibrated parameters whose names match the *pattern*. *pattern* can contain any of the unix file matching wild-cards. If omitted, the *pattern* defaults to `*`, refreshing all parameters.

Refreshing a parameter updates any changes to the fit that computes the parameter.

Chapter 5. Sample GUI for the calibrated parameter plugin.

The GUI supplied with the plugin was originally funded for work at Duke University at TUNL. It is intended to calibrate parameters by having the user make correspondences between peaks in spectra and reference values for peaks that are expected to live in the spectra. This is a typical calibration method used in gamma ray spectroscopy, where a source with known energy lines can be used to extract a calibration.

This GUI may or may not be suitable to your application. If it is, use it. If not, don't.

This chapter will describe:

- How to setup the table of reference energies used by the GUI.
- How to incorporate the GUI into your SpecTcl application.
- How to use the GUI to calibrate your parameters.

5.1. Creating a table of reference energies.

The GUI requires a table of referene values. When you calibrate a parameter, you will be selecting the reference values that match the centroids of specific peaks in your spectrum. These correspondences provide the points for the calibration fit.

The calibration GUI reads, and merges a system-wide and a user-specific reference file. One or the other or both of these files must exist (or fitting is not possible). If `SpecTclHome` is a Tcl variable that points to your SpecTcl installation, `$SpecTclHome/etc/CalibrationPoints.dat` is the system-wide reference file. The user specific calibration file is `~/.SpecTcl/CalibrationPoints.dat`.

The contents of these files are just real parameter values, one per line. The units and scaling are entirely up to you. Comments and blank lines are not supported.

5.2. Incorporating the GUI into your application specific SpecTcl

The calibration GUI is a complex of packages that can be loaded via the **pacakge require** command.

If you installed the plugin in SpecTcl's installation directory, you don't need to do anything to prepare to use this package. If you installed it elsewhere, you need to add that directory to SpecTcl's `auto_path` e.g.

```
lappend auto_path [file join ~ calibrationplugin]
```

Load the calibration GUI and attach its components to buttons by modifying the following example:

Example 5-1. Loading the Calibration GUI and attaching it to buttons

```
package require Calibrations
package require CalibrationGUI
package require CalibIOGUI

frame .calibration
button .calibration.create -text {Create Calibration...} -command CalibrationGUI::Calibratio
button .calibration.save   -text {Save Calibrated Parameters...} \
                           -command CalibIOGUI::WriteConfiguration
button .calibration.read   -text {Read Calibrated Parameters...} \
                            -command CalibIOGUI::ReadConfiguration

grid .calibration.create .calibration.save .calibration.read -sticky w

# Below here do what's needed to make the calibration frame visible in some
# parent.
...
```

The code in this example places three buttons in a frame. These buttons are laid out left to right as follows:

Create Calibration...

Create a new calibrated parameter.

Save Calibrated Parameters...

Prompts for a file into which will be written the Tcl commands needed to reconstruct the fits and calibrated parameters currently defined.

Read CalibratedParameters...

Prompts for a file saved with the Save Calibrated Parameters... button and reads it in. Really this just **source**es the selected file.

5.3. Calibrating parameters with the GUI.

The parameter calibration GUI operates by asking you to define correspondences between peak positions in a spectrum on a raw parameter, and values in the reference file. To define peak positions, you will need to give SpecTcl information about where the peaks are. You do this by creating *Cut* gates on a raw parameter spectrum in Xamine.

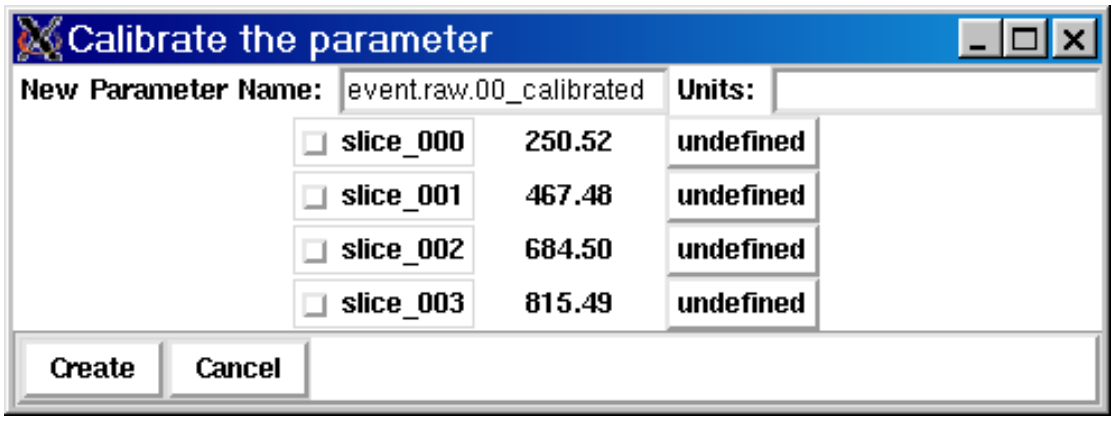
When you click the Create Calibration... button, you are first presented with a list of the SpecTcl parameters:

Figure 5-1. Raw parameter selection box

Select the raw parameter to be calibrated either by double clicking its name in the list box or selecting it and then clicking the Fit... button.

The next step in creating a calibrated parameter presents you with the following dialog:

Figure 5-2. Calibration dialog



Fill in the top part of this dialog with the name of the new parameter you will be creating, and optionally the units (e.g. Gamma1 and KeV).

Using the middle part of the dialog, select peaks by clicking the checkbox next to the name of the gate that has been integtated. The position of the peak in the raw parameter space is shown in the middle column. The right most column is a set of drop down menus that allow you to choose the reference value that corresponds to that peak.

If you don’t want to use a peak, simply leave the gate in which the peak position is unchecked.

Click the Create to create the calibrated parameter. If the calibrated parameter already exists you will be prompted to decide if you want to replace it.

Once the calibrated parameter is created, the plugin will compute it for each event. The parameter can be treated as any other SpecTcl parameter. Spectra can be made on it, gates set on it and so on.