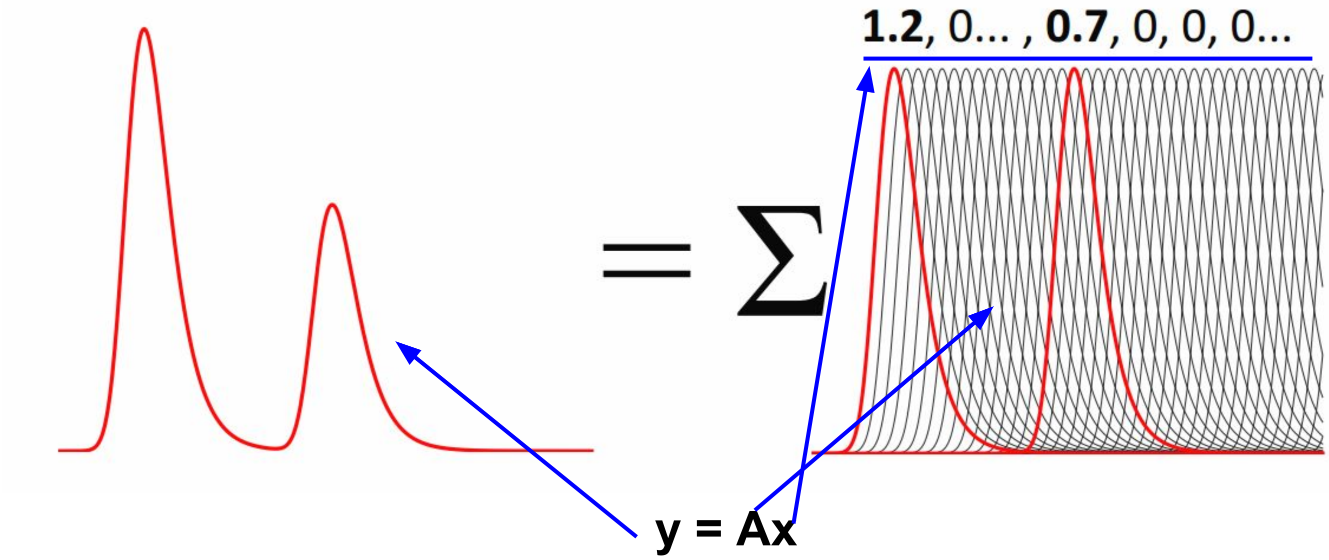
Wavedeform Unfolding



y - the waveform.
A - the matrix of basis functions created with SPE templates shifted in time.
x - the result of unfolding.

John Kelley, PINGU Hardware Call, June 5, 2014

Non-negative Least Squares problem

Given matrix A size of MxN and vector y size of N, vector x size of M is the solution to

$$\min_x \|\, Ax-y \,\|^2_2 >= 0$$

The complementary problem can be created* :

$$w = A^T y - A^T Ax\text{: } w<=0,\ x>=0,\ x^T w=0^{**}$$

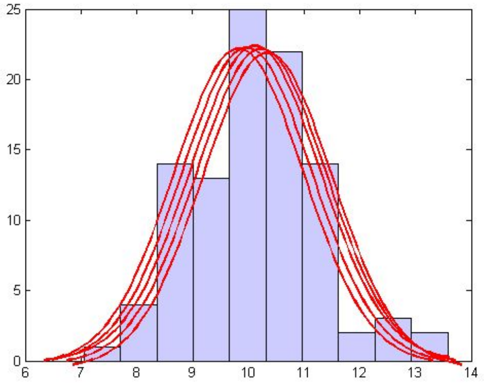
$$x^T w = x^T(A^T y - A^T Ax) = x^T A^T y - x^T A^T Ax = y^T y - y^T y = 0$$

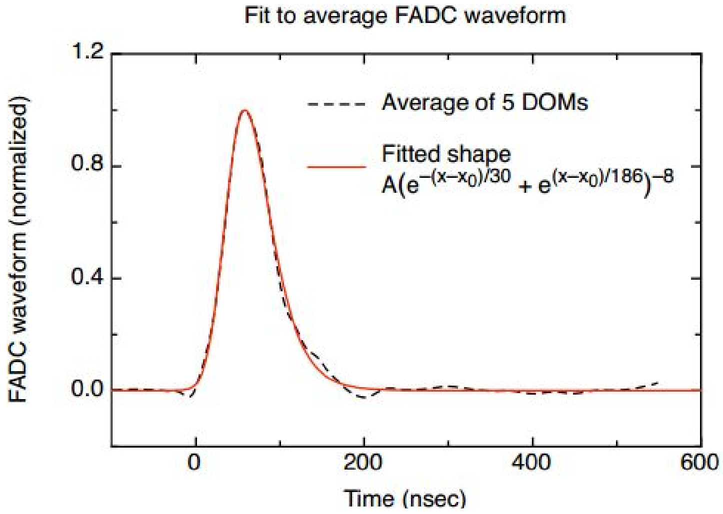
* A FAST NON-NEGATIVITY-CONSTRAINED LEAST SQUARES ALGORITHM, RASMUS BRO AND SIJMEN DE JONG, JOURNAL OF CHEMOMETRICS, VOL. 11, 393–401 (1997)
** J. van Santen, N.Whitehorn. IceCube internal note on Photosplines icecube_201011001_v2.

getPulses:

constants: **SPES_PER_BIN** = 5 - number of basis functions to unfold per bin,

PERIOD_NS = 5.0 ns - sampling period (waveform bin)





template: **c*((exp(-(x-x0)/b1)+exp((x-x0)/b2))^p)**

c, x0, b1, b2, width, min - template parameters

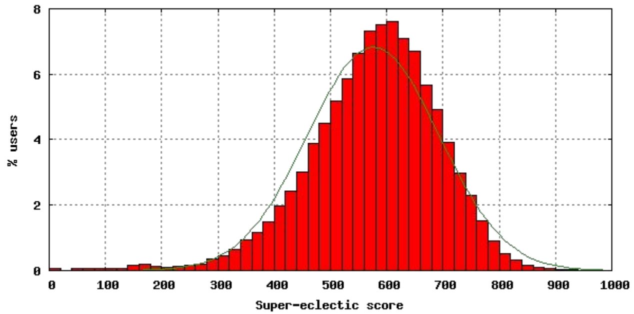
function **spePulseShape** which returns template value at given time **t**.

Prepare the A (basis) matrix (bins x number of spes)

i index - bin number - number of bins in waveform (**nbins**)

j index - spe number

data - template value at the time



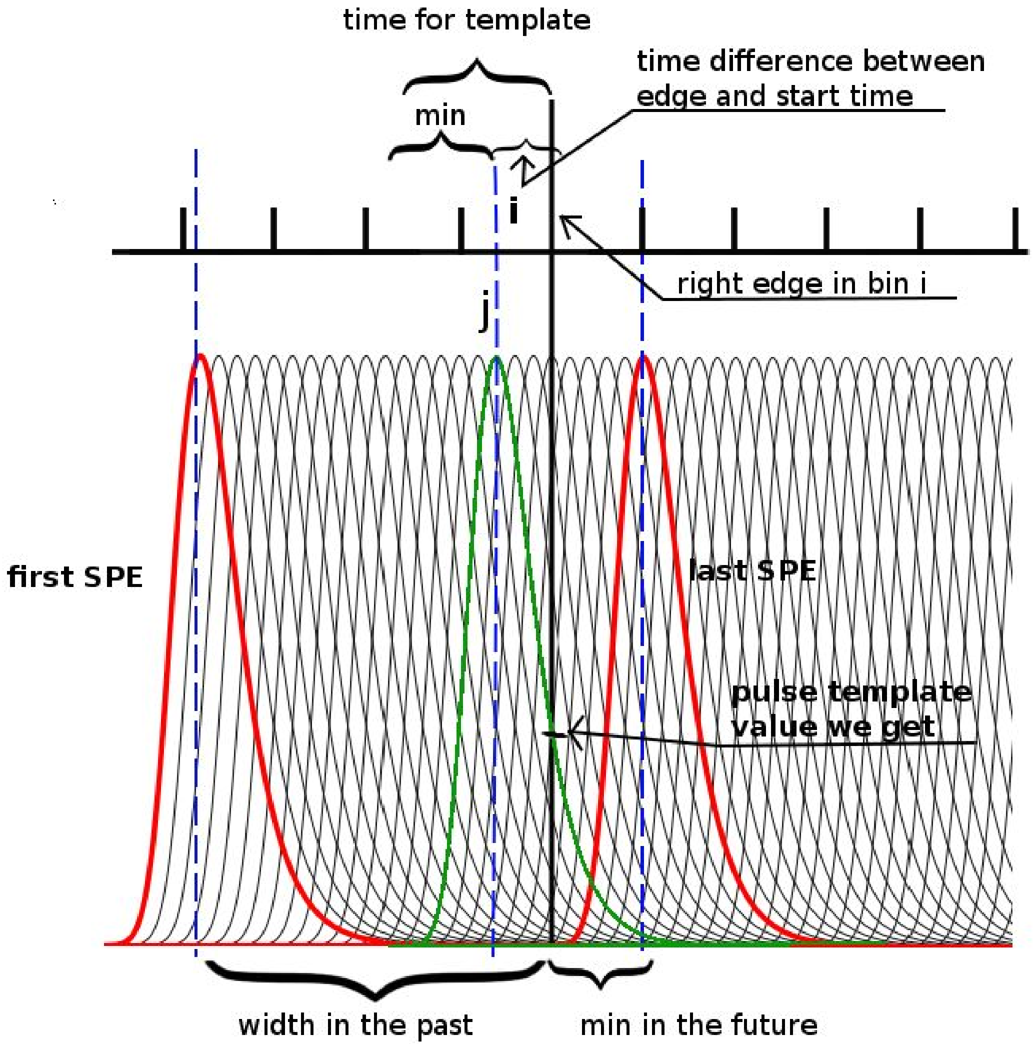
Create 1D arrays:

Name (**array name**)	size (bins)	values (of each bin)
bin edges (**redges**)	**nbins** (number of bins)	spacing: **PERIOD_NS *** edge number (0,5,10..) ns
data (y)	**nbins**	waveform data * weight
pulse **start times**	**SPE_PER_BIN * nbins** (number of spes)	spacing: evenly distributed along the waveform length (-5,-4,-3,..0,1,2,...) ns
pulse template	size of template (55 ns) / (**PERIOD_NS** / **SPE_PER_BIN** / 2 - 0.5 ns) = 110	pulse template values at each time from **min** to last **pulse template** bin. (values at -5, -4.5, -4, .., 0, 1, ..) ns

Prepare the A (basis) matrix (bins x number of spes)

- Estimate max number of matrix elements (3 cycles). Needed to initialize cholmod matrix.
- Initialize the matrix A
- Fill it with basis:
- Cycle through all the **bin**s:
  - if weight if the **bin** = 0 continue; - position
  - find first SPE affecting the **bin** (its start time is no more than one template width in the past)
  - find the last SPE affecting the **bin** (its start time is no more than one template min in the future)
  - for each SPE between the first and the last:
    - find which exact bin of **pulse template** contributes to this **bin** from that SPE. (find time difference between the **bin** and SPE and take value of corresponding bin from **pulse template**)
    - **i** index - **bin** number
    - **j** index - that SPE number
    - **data** - the found **pulse template** value at the time

matrix A is ready and can be given to NNLS along with **data** to find the x (y=Ax)


time for template
time difference between
edge and start time
min
i
right edge in bin i
j
first SPE
last SPE
pulse template
value we get
width in the past
min in the future

NNLS $w = A^T y - A^T A x$, $w <= 0$, $x >= 0$, $x^T w = 0$

<u>input:</u> **A, y, *tolerance, min_iter, max_iter, npos, norm***

<u>output:</u> **x**

max size = nbins*SPE_PER_BIN

P - passive set = {0} elements of **x** with those indexes will be > 0

Z - active set = {0,1,.. , number of columns} elements of **x** with those indexes = 0 (initially **x** = {0})

<u>It can be shown that for optimal solution x, w{Z} < 0, w{P} = 0</u>

Main loop: **(optimization of w. finds least negative element of w and moves it into passive set P, it goes on until we are done or *max_iter* is reached)**

calculate:

- **w = A'Ax- - A'y;** (cholmod own procedure)
- **wmax** = max{**w{Z}**}; (one cycle (comparison))
- **wmin** = max{**w{P}**}; (one cycle)

see if:

- **Z** is empty;
- **wmax** <= 0;
- **wmax** < ***tolerance*** and **-wmin** < ***tolerance***

if true → done;

NNLS $w = A^Ty - A^TAx$, $w<=0$, $x>=0$, $x^Tw=0$

Main loop: (continued)

if **wmax** > 0 $\Rightarrow$ move index of wmax into passive set **P**.

Second loop: (•every cycle goes through whole passive set P)

- create a submatrix **Ap** of **A** (**yp** of **y**), what contains only columns with passive set indexes;
- Solve LS problem **p = Ap/yp** by QR decomposition (SuiteSparseQR);
- if p completely positive, set **x = p;** $\Rightarrow$ **Main loop;**
- if there are negative elements of **p** for each of them find **q = x/(x-p);**
- find **alpha**= min{**q**};
- for passive set **P** indexes find new **x = x + alpha*(p-x);**
- if there are any elements of **x** < 0 set them to zero and move their indexes to active set **Z**;
- if **alpha** =0 (equilibrium) $\Rightarrow$ done;

return **x**;

SuiteSparseQR_C_backslash_default :

A = QR, where Q is orthogonal and R is upper triangular.

$$Ax=y,\ A=QR;$$
$$A^TAx=A^Ty;$$
$$R^TQ^TQRx=R^TQ^Ty;$$
$$R^TRx=R^TQ^Ty; \quad (Q^TQ=I)$$
$$Rx=Q^Ty$$