GetPulses - the main function of domwave that is responsible for waveform unfolding.
It calls NNLS function.

input: waveform wf, pulse template
output: pulse series

constants: **template of spe, PERIOD_NS, SPE per bin.**

number of bins (nbins) = length of waveform;

create arrays of right and left time bin edges: ledges, redges (size nbins+1);
fill them with times, time difference is **PERIOD_NS**;

create data vector size of nbins from waveform values in that bin multiplied by weights; (weight come from waveform)

create an array of spe start times evenly distributed over the waveform range;
for that find:

- number of spes = **SPE per bin** * nbins;
- the first spe is one **PERION_NS** early;
- the last is nbins-1 **PEROID_NS** ahead if the first;
- the spacind is time (first-last)/(number of spes)

create waveform template:
template spacing = **PERIOD_NS/ SPE per bin/** 2;
amount of template bins = width of template/ template spacing;

now fill the template vector of size equal to amount of template bins, with template values using spePulseShape function; (this function just returns the value of template bin if you give it time (from the start of template))

now begin creating basis matrix A.

first we estimate max amount of non zero elements (nzmax) in A:

Cycle through all bins: (bins are described by redges array.)
find the position (number) of first spe affecting this bin (template width before it).

when count all the followling spes after the first up to the last, which is template min in the future from the bin.

add all those spes to possible number on non zero matix elements (nzmax).
go to next bin.

now we fill basis matrix A size of nzmax:

Cycle through all bins:

find first spe affecting the bin (like before);

if weighted charge of that bin is 0 continue; -/ could be moved first ?/

go from that first spe to the last spe affecting the bin and set:
row number = bin number;
column number = spe number;
data = template bin value at the point of time of current spe (relative to the first)* weighted
charge of that bin;

actual number of non-zero bins++;

(the time for template is determined as difference of right edge of bin and current spe start time minus template minimum divided by pulse template bin spacing);??

go to the next bin.

call nnls with A and waveform data vector:

---

nnls: w = A'*A*x-A'*y, w>=0, x>=0, x*w'=0

input: A,y, tolerance, min_it, max_it, npos, norm
output: x

(min_it, max_it - minimum and max of iterations allowed by user, norm is flag what tell us if the normal equations are preformulated (i am not completely sure what it means))

code:
npos - amount of active coefficients. (those are constrained to zero)
if their number is not specified, npos = number of columns in A. (those are > 0)

Z - vector of active constraints {0,1,2 , ... ,npos}
P - vector of passive constraints {npos+1, ... ,number of columns in A} (default elements of P > npos)

x = 0, w = 0. (note: in the beginning all of the constraints are active => x=0)

for (n = 0, n <= max_it, n++) - (in this loop w is optimized. it finds most negative element of w and removes it from active set.)

wtemp = y; |
wtemp = A*x-wtemp = A*x-y; | basically w = A' *(A*x-y);
w= A'*wtemp; |
(if normal equations are preformulated then w=y-A*x)

if there are no active coefficients - break, we are done.

find wmax = maximum of w. (t is index of maximum)
if wmax <=0 break, we are done.

check if w is within tolerance:
if wmax < tolerance and and n>min_it check:
if P is empty or -wmin < tolerance break, we are done.

move index (t) of wmax from Z (active) to P (passive) set

while (1) (- loop for solving LS problem in subspace of passive set with trial solution x)

create a submatrix Ap which contains only the coefficients what are in passive set P;

p = Ap/y; - solves LS problem for Ax=y in P subset by QR decomposition.

(if normal equations are preformulated then create yp - a submatrix of y containing only the passive set P coefficients and p = Ap/yp)

Cycle through the passive set P. If elements of p with those indexes is <=0 stop.

if the are all positive x=p and break into (beginning of) outer loop.

Cycle through the passive set and find elements [j] of p that are <=0;

alpha = x[j]/(x[j]-p[j]);

find minimum alpha (among all j) called min_alpha;

find new x by cycling through the passive set P and setting x=x+min_alpha*(p-x);

Cycle through the passive set and find elements of x that are <=0;

set them to zero;

move their coefficients into active set Z;

if min_alpha=0 break; - equilibrium;

if min_alpha = 0 break; - equilibrium;

return x;

---

record the result of nnls into pulse series;

end.