Problem 3 overview: Savitzky-Golay filters – Local least-squares fit… convolution-style !

Pertinent reading for Problem 3:

The wiki page for Savitzky-Golay (it's actually the best source for this topic !)

https://en.wikipedia.org/wiki/Savitzky%E2%80%93Golay_filter

1) Savitzky-Golay = Local least-squars fit, with a sliding window !

The concept here is really simple ! Suppose I have a nonlinear, noisy data stream $y(t)$ depicted in Figure 1, where we discretize it at a given sampling frequency f_{sample} (the time spacings *Δt has to be equally-spaced* !!). The goal is to use multiple, local least-squares fit to "smooth" our dataset.

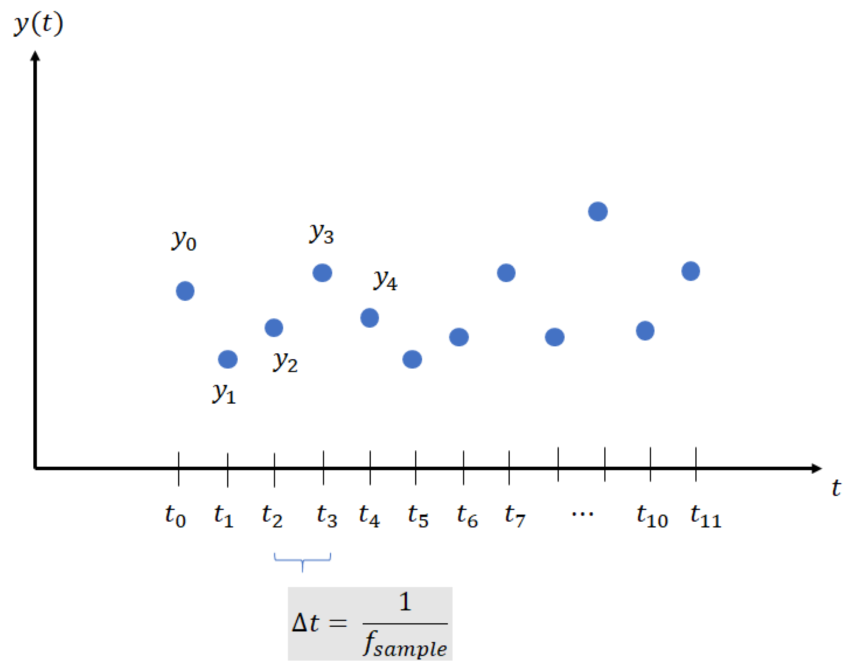


Figure 1: Our data $y(t)$ has been discretized at f_{sample}, and the first 5 data points are labeled as y_0 thru y_4

Step 1: Pick a window length and a local least-squares model

Let's pick a local window of length = 5. Usually, you want this to be an *odd* number....

$$m = 5 \qquad (using\ the\ Wiki\ page's\ notation)$$

And also, it is customary to pick either a local quadratic or cubic polynomial for our least-squares fit. Let's do a cubic for this example (and that's what the Wiki page has also):

Model curve: $$y_{local}(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$

Note: I've chosen the least-squares coefficients as a's such that they'll match the notations in the wiki page !

Step 2: (For internal points):- Perform local least-squares fit

Now, we'll window our left-most 5 data points and perform a least-squares fit. Focusing on Figure 2, the tasks are:

a) Pick 5 internal data points
b) Relabel the horizontal axis in "local" coordinates z, where the centerpoint reference is set at $z = 0$
c) Perform a local least-squares cubic fit for those 5 points
d) Extrapolate the fitted y –value at the centerpoint $z = 0$. We will call this value y_{2new}
e) Save the new value y_{2new} in a "results" vector $y_{smoothed}(t)$ at the corresponding center timepoint

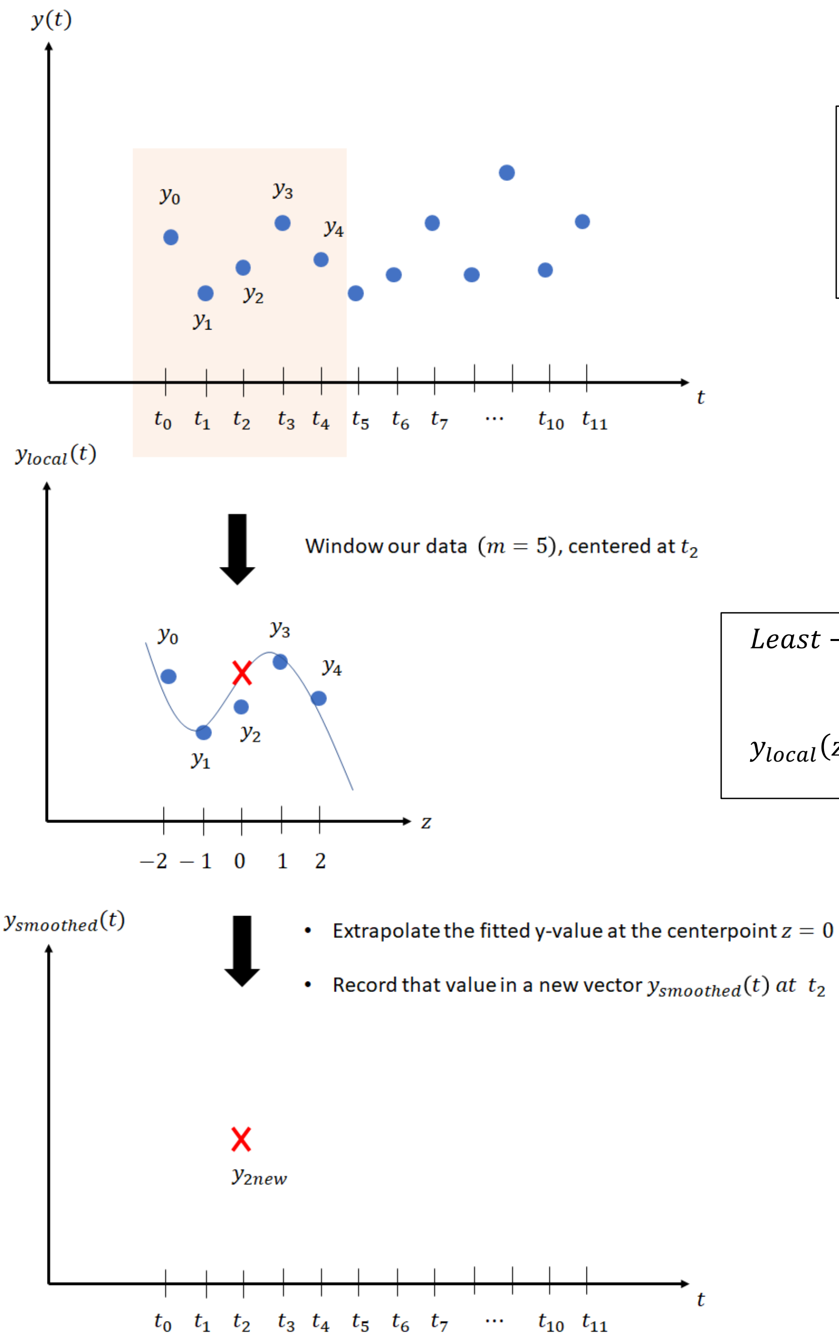


Figure 2: Local least-squares fit, followed by a centerpoint value extrapolation for 5 "internal" time nodes, with the centerpoint at t_2

$Least - squares\ cubic\ fit \quad (local\ z - axis)$

$$y_{local}(z) = a_0 + a_1 z + a_2 z^2 + a_3 z^3$$

Step 3: (For internal points):- Slide your window, and repeat the least-squares fit

Now, all you have to do is to slide the window 1 tick to the right.... and then, repeat the process for the next 5 points !

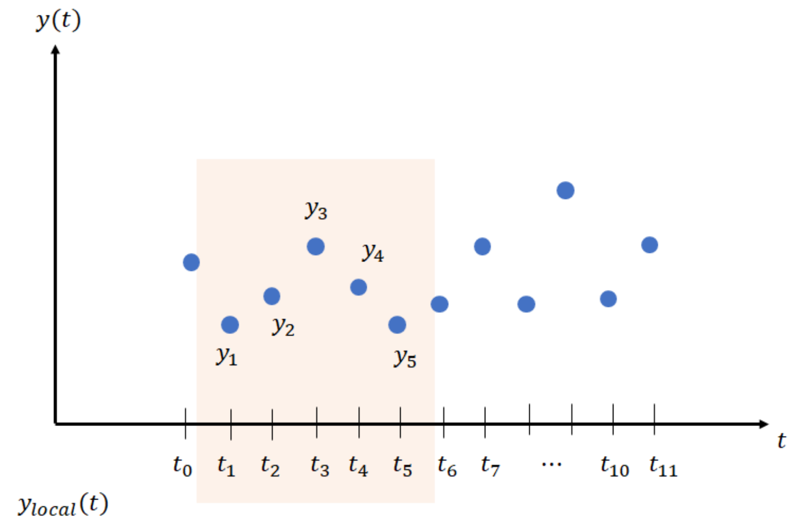
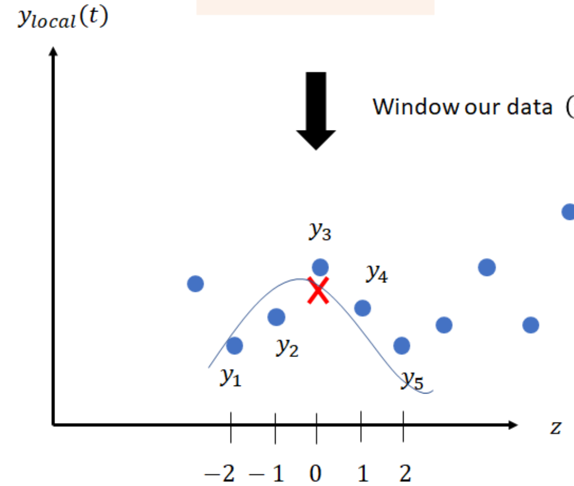


Figure 3: Do local least-squares fit again on the next 5-point frame

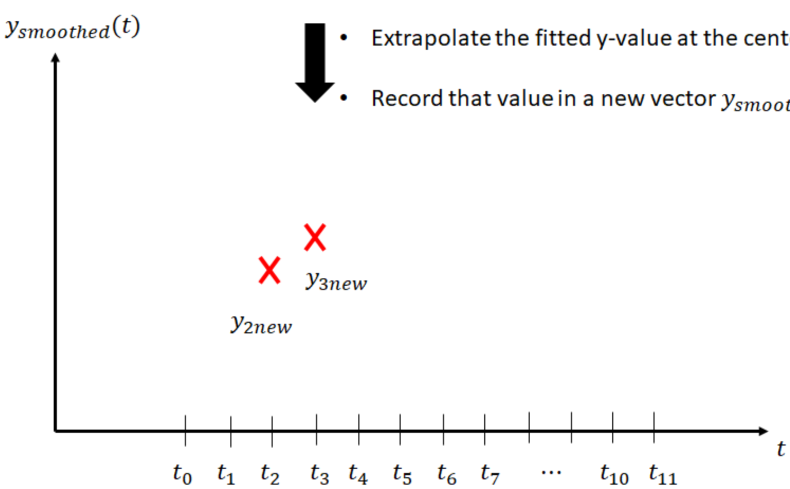
Window our data $(m = 5)$, centered at t_3



$$Least - squares\ cubic\ fit \quad (local\ z - axis)$$

$$y_{local}(z) = a_0 + a_1 z + a_2 z^2 + a_3 z^3$$

- Extrapolate the fitted y-value at the centerpoint $z = 0$
- Record that value in a new vector $y_{smoothed}(t)$ at t_3



Step 4: (For internal points):- Keep sliding until you reach the last 5-point set

Yup…. All you have to do is to keep doing it until you've reach the right-most 5 point set ! However, you might be wondering:

a) Our sliding window really starts at t_2 (midpoint of any given window) and we're ending it at t_9
b) What are we going to do at points $t_0,\ t_1\ ,\ t_{10}\ ,$ and t_{11} ?
c) Those are *boundary-condition points* for Savitzy-Golay filters, and we'll address those points next !

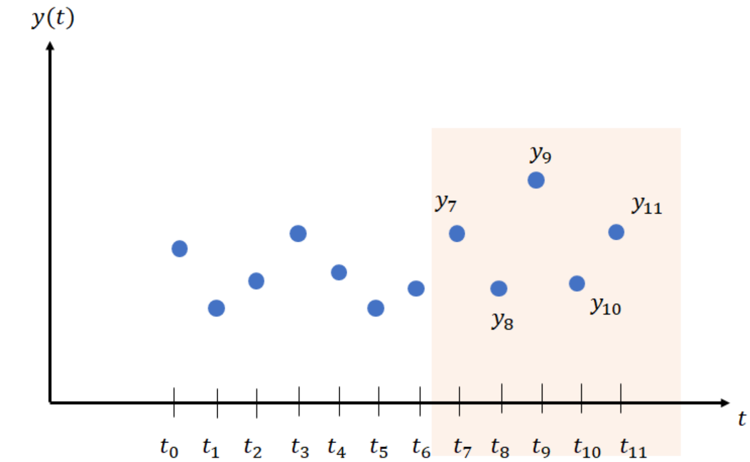
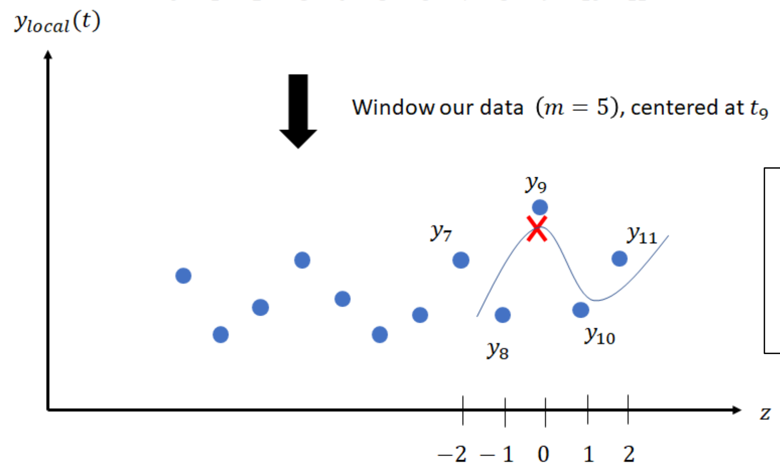
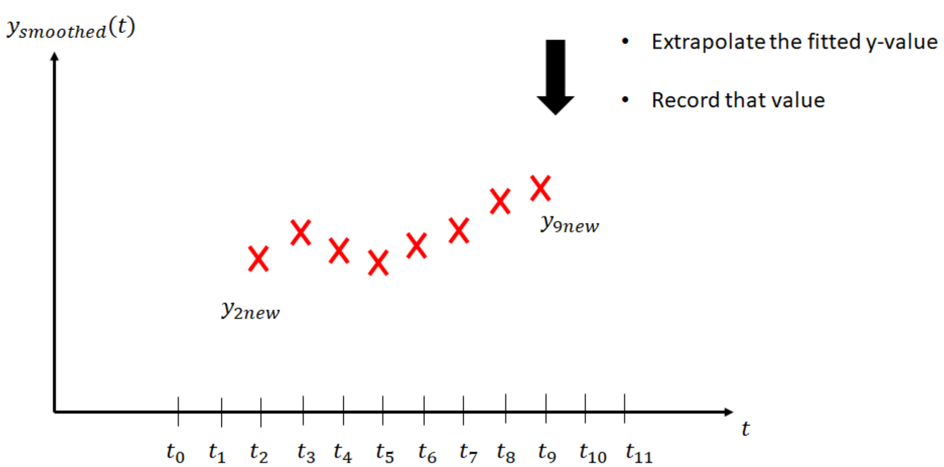


Figure 4: The very last "internal 5-point" window, centered at t_9



Least − squares cubic fit (local z − axis)

$$y_{local}(z) = a_0 + a_1 z + a_2 z^2 + a_3 z^3$$



Step 5: Left / right boundary conditions: Add mirror-image ghost nodes on either side !

To take care of the left and right boundary conditions, all you have to do is:

a) Take the left bounddary point t_0 , and mirror-image:

$$floor\left(\frac{m}{2}\right) = round\ down\left(\frac{5}{2}\right) = round\ down\ (2.5) = 2\ points$$

on the other side of the boundary. This will create points t_{-2} and t_{-1} , with the corresponding mirror-imaged y-values of y_2 and y_1, respectively.

b) Next, take the right boundary point t_{11} , and mirror-image:

$$floor\left(\frac{m}{2}\right) = round\ down\left(\frac{5}{2}\right) = round\ down\ (2.5) = 2\ points$$

on the other side of that boundary. This will create points t_{12} and t_{13} , with the corresponding mirror-imaged y-values of y_{10} and y_9, respectively.

c) If you take a close look at Figure 5, you will see that the mirrored ghost nodes on either side will now facilitate valid 5-point windows near the edges, where:

$Centerpoint$	$Valid\ 5-point\ window$	$Extrapolated\ y-value$
t_0	$[t_{-2}\ \ t_{-1}\ \ t_0\ \ t_1\ \ t_2]$	y_{0new}
t_1	$[t_{-1}\ \ t_0\ \ t_1\ \ t_2\ \ t_3]$	y_{1new}
t_{10}	$[t_8\ \ t_9\ \ t_{10}\ \ t_{11}\ \ t_{12}]$	y_{10new}
t_{11}	$[t_9\ \ t_{10}\ \ t_{11}\ \ t_{12}\ \ t_{12}]$	y_{11new}

★★★

And the resulting set $y_{smoothed}(t)$ is the output of our Savitzky-Golay filter !!!!! =) See Figure 5 on the next page

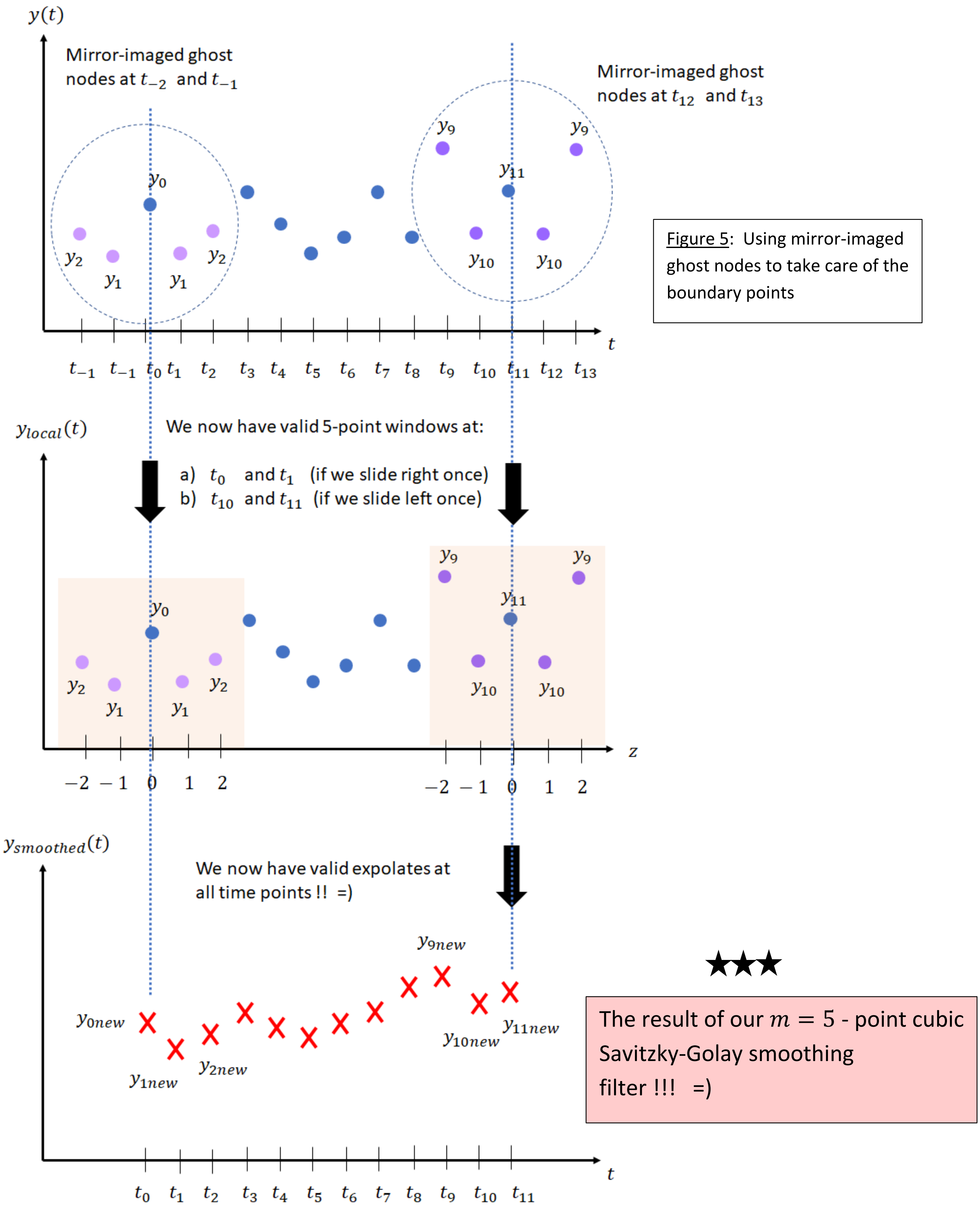


Figure 5: Using mirror-imaged ghost nodes to take care of the boundary points

Epilogue: The "Savitzky-Golay" convolution coefficients (in the wiki page discussions)

If you read the wiki page on Savitzky-Golay filters, you'll immediately see a lot of garbage math…….. and eventually, they list the so-called "convolution coefficients" the 5-pt cubic fit, 7-point cubic fit, etc. All the BS in those discussions has to do with the fact that:

a) If you discretize your original $y(t)$ data set so that they have *equal time points*

b) Then, when you perform your 5-point (or whatever # of points) local least-squares, your new, local z −axis coordinates will all be integers ! Moreover, they will *always be centered around z = 0*

You can then use factoid (b) to your advantage and "hard-code" your least-squares coefficients for your local model fit into matlab (Warning: It's usually *NOT WORTH IT* !!) . But just out of mathematical curiosity, let's see how it would work !

For instance, if you took Figures 2 in this PDF as reference, one can write out the least-squares equation for our 1st five-point window, centered around time t_2 , by making a small table:

$Index\ \ (z - axis)$	$Cubic\ model:\quad a_0 + a_1 z + a_2 z^2 + a_3 z^3 = y_{local}(z)$
$z = -2$	$a_0 + a_1(-2) + a_2(-2)^2 + a_3(-2)^3 = y_0$
$z = -1$	$a_0 + a_1(-1) + a_2(-1)^2 + a_3(-1)^3 = y_1$
$z = 0$	$a_0 + a_1(0) + a_2(0)^2 + a_3(0)^3 = y_2$
$z = 1$	$a_0 + a_1(1) + a_2(1)^2 + a_3(1)^3 = y_3$
$z = 1$	$a_0 + a_1(2) + a_2(2)^2 + a_3(2)^3 = y_4$

Rewriting it out in matrix form, we'll get a rectangular $Ax = b$ equation:

$$\underbrace{\begin{bmatrix} 1 & -2 & 4 & -8 \\ 1 & -1 & 1 & -1 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 8 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}}_{b}$$

As you know, we can solve for the least-squares coefficients using our favorite equation:

$$A^T \; A \; \hat{x} \quad = \quad A^T \; b$$

If you plug in everything and evaluate the stuff on the left-hand side, you'll get:

$$\underbrace{\begin{bmatrix} 5 & 0 & 10 & 0 \\ 0 & 10 & 0 & 34 \\ 10 & 0 & 34 & 0 \\ 0 & 34 & 0 & 0 \end{bmatrix}}_{A^T \; A} \underbrace{\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}}_{\hat{x}} \underbrace{=}_{=} \underbrace{\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ -2 & -1 & 0 & 1 & 2 \\ 4 & 1 & 0 & 1 & 4 \\ -8 & -1 & 0 & 1 & 8 \end{bmatrix}}_{A^T} \underbrace{\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}}_{b}$$

And if you solve for $\hat{x}$ using LU factorizations, you'll get the expressions listed in the wiki page !

$$Local\ least-squares\ coeffs \quad (for\ \ m = 5\ \ point\ windows)$$

$$a_0 = \frac{1}{35}\left(-3y_0 + 12y_1 + 17y_2 + 12y_3 - 3y_4\right)$$

$$a_1 = \frac{1}{12}\left(y_0 - 8y_1 + 8y_3 - y_4\right)$$

$$a_2 = \frac{1}{14}\left(-2y_0 - y_1 - 2y_2 - y_3 + 2\,y_4\right)$$

$$a_3 = \frac{1}{12}\left(-\,y_0 + 2y_1 - 2y_3 + y_4\right)$$

Finally, you know that after we've determined our 5-point loca least-squares fit $y_{local}(z)$, our last job is to extrapolate the y-value at the window midpoint $z = 0$:

$Midpoint\ extrapolation$: $\quad y_{local}(0) \; = \; a_0 \; + \; \cancel{a_1\,(0)} \; + \; \cancel{a_2\,(0)^2} \; + \; \cancel{a_3\,(0)^3}$

$$y_{local}(0) = \frac{1}{35}\left(-3y_0 + 12y_1 + 17y_2 + 12y_3 - 3y_4\right)$$

This equation will be true for *every 5-point window* (you will just *change the indices of each "y"* as you slide your window)

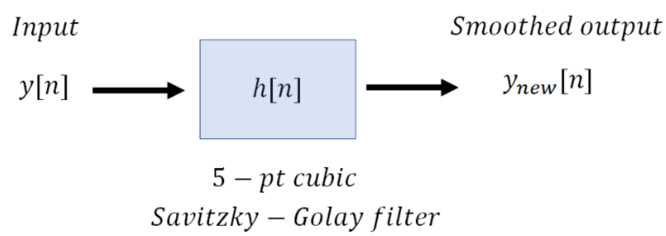
★★★

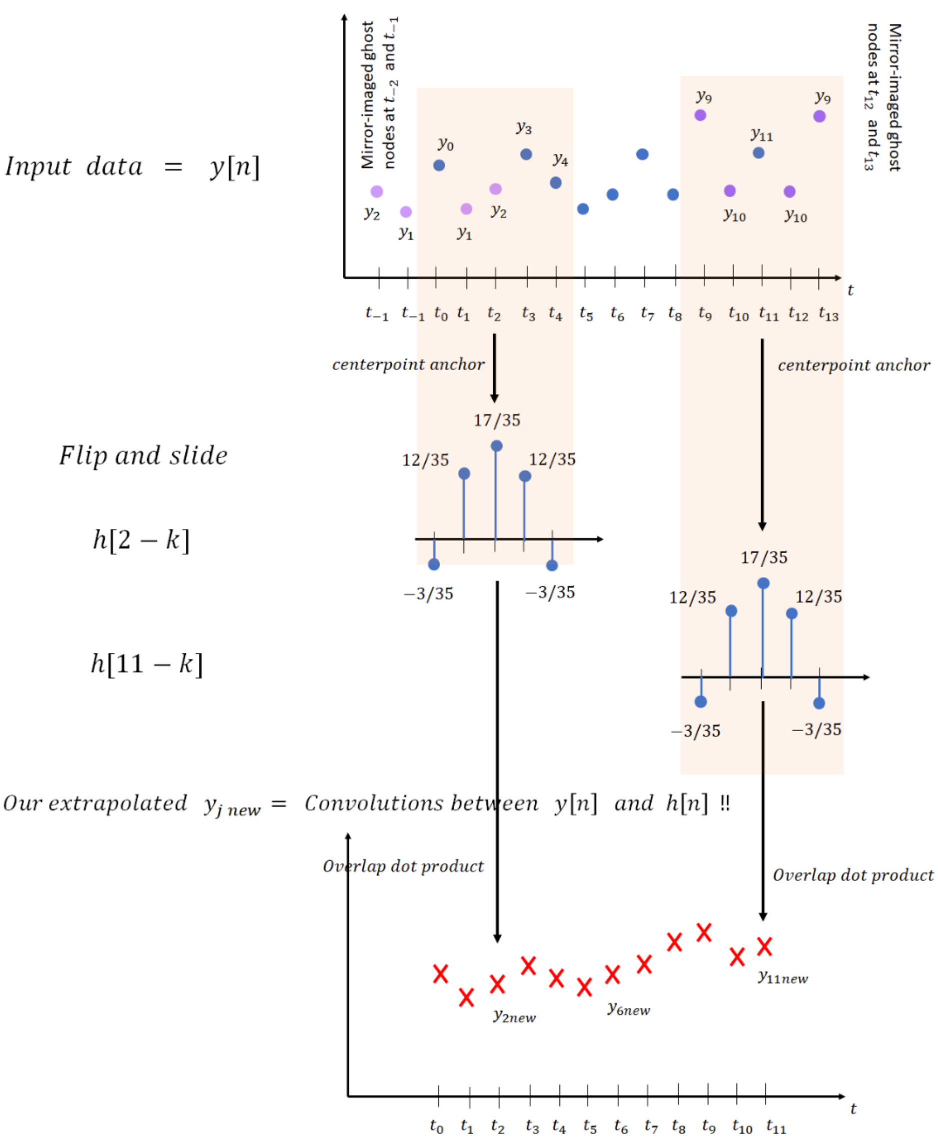
Therefore, if you have pre-processed your input $y(t)$, where you've added the required $floor(m/2)$ amount of "mirrored-imaged ghost nodes" at the boundaries, you can use this hard-coded formula to calculate *every* extrapolated $y_{j\ new}$ values (at every time point t_j) using discrete convolution, where we would:

a) Set the impulse response $h[n]$ to be equal to the numerical sequence for $y_{local}(0)$

b) And use the flip + slide method between $y[n]$ and $h[n]$ to calculate each new $y_{j\ new}$ (note that our $h[n]$ filter will be *non-causal* in this situation !)

Basically, all the messy math you saw in the wiki page can be summarized in Figure 6 below !!!! =)





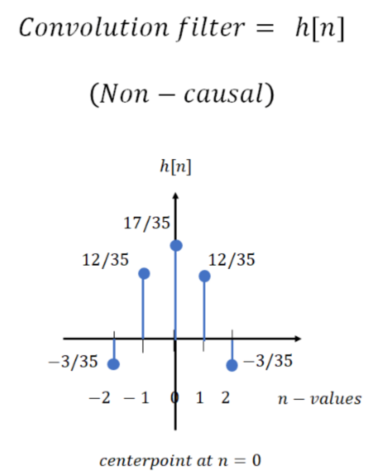


Figure 6: If you knew the exact expression for the local polynomial intercept term a_0, you will be able to use convolutions to perform Savitzky-Golay filtering !

** Note: Getting the expression for a_0 is, most of the time, not worth it… =(

It is much easier to just write loops + calculate the local least-squares fits from scratch !