Least-squares estimators on periodically sampled noisy signals

1. Introduction

In this report, we will describe the situation of attempting to extract useful information from sampled data. We will describe how least-squares polynomials are a useful tool, but potentially expensive. We will then describe how these formulas can be simplified if the data is sampled periodically. We will then look at extracting data using least-squares linear and then quadratic polynomials to find various estimators, including taking into account the concept of *jitter*. We will then see the most efficient mechanism to extract these estimators if not only one or two of these estimators are required, but perhaps three, four or more, in which case, we can return to using algebra and calculus to derive other estimators based on the calculations of a subset of the estimators. We will conclude the descriptive component of this report with a brief description of when to use least-squares linear polynomials for calculating these estimators, and when to use least-squares quadratic polynomials. This will be followed by a description of the associated C++ library that calculates these estimators and displays the associated coefficients.

2. Background

Suppose we have data we have read from a sensor:

$$\ldots,(t_{n-1},y_{n-1}),(t_n,y_n).$$

When data read from a sensor is noisy, it is in general a poor approach to using interpolating polynomials as the noise amplifies the error of the interpolating polynomials. Instead, it is often beneficial to use least-squares linear or quadratic polynomials through the last N points to extract information from the data. Thus, we want to find those polynomials $at + b$ or $at^2 + bt + c$ such that

$$\sum_{k=0}^{N-1}\left(at_{n-k}+b-y_{n-k}\right)^2 \text{ or } \sum_{k=0}^{N}\left(at_{n-k}^2+bt_{n-k}+c-y_{n-k}\right)^2,$$

respectively, is minimized. This can be done through the application of linear algebra, although, finding the coefficients of these least-squares polynomials is normally a computationally expensive task. For example, suppose we are finding the least-squares best-fitting linear or quadratic polynomial through N points. This would require us to solve what is called the *normal equations* $A^{\mathrm{T}}A\mathbf{c}=A^{\mathrm{T}}\mathbf{y}$. In this case, we must perform the following calculations as described in Table 1 indicating the calculations that must be made, and the total number of floating-point operations (FLOPs).

Table 1. Required operations for finding best-fitting polynomials.

Linear polynomial Operations	FLOPS	Quadratic polynomial Operations	FLOPS
$\sum_{k=0}^{N-1} t_{n-k}$, $\sum_{k=0}^{N-1} t_{n-k}^2$	$3N-2$	$\sum_{k=0}^{N-1} t_{n-k}$, $\sum_{k=0}^{N-1} t_{n-k}^2$, $\sum_{k=0}^{N-1} t_{n-k}^3$, $\sum_{k=0}^{N-1} t_{n-k}^4$	$7N-4$
$\sum_{k=0}^{N-1} y_{n-k}$, $\sum_{k=0}^{N-1} t_{n-k} y_{n-k}$	$3N-2$	$\sum_{k=0}^{N-1} y_{n-k}$, $\sum_{k=0}^{N-1} t_{n-k} y_{n-k}$, $\sum_{k=0}^{N-1} t_{n-k}^2 y_{n-k}$	$5N-2$
Gaussian elimination	9		27
Total	$6N+5$		$12N+21$

Note that these are under the assumption the programmer is competent and minimizes the number of floating-point operations.

Once the coefficients of the least-squares polynomial are found, then appropriate calculations can be made. For example, we can find the best estimator for the current value y_n using

$$at_n + b \text{ or } at_n^2 + bt_n + c,$$

respectively, and we can find the best estimator for the next value y_{n+1} using

$$at_{n+1} + b \text{ or } at_{n+1}^2 + bt_{n+1} + c,$$

respectively. Similarly, the best estimator of the velocity at the current time is either

$$a \text{ or } 2at_n + b,$$

respectively, and if you are using least-squares quadratic polynomials, the best estimator of the acceleration is

$$2a.$$

3. A more efficient approach: periodic sampling

This reduction in the number of FLOPs is beneficial in real-time applications with real constraints on power, memory and speed, and thus unnecessary computations are expensive. Fortunately, if the samples are taken periodically; that is, if $t_k - t_{k-1} = \Delta t$ for all $k > 0$ for some fixed period Δt, then many of the formulas, be it finding the best estimator of current value or the best predictor of the next value, or the best estimator of the current change per period or the actual rate of change, all these can be found via a simple linear combination of the y values; that is, we have reduced the runtime from $6n$ or $12n$ down to $2n - 1$ FLOPs. For example, to find the best approximation of the current value at time t_n using the current reading y_n and five previous readings $y_{n-1}, \ldots, y_{n-5}$ is

$$\begin{aligned}\hat{y}(t_n) \approx\ & 0.5238095238095238\,y_n + 0.3809523809523809\,y_{n-1} + 0.2380952380952381\,y_{n-2} \\ & + 0.0952380952380952\,3y_{n-3} - 0.04761904761904762\,y_{n-4} - 0.1904761904761905\,y_{n-5}\end{aligned}$$

This is independent of the period Δt between readings. As a proof of concept, the source file `least_sqrs_est.h` contains a number of functions that use either a least-squares linear polynomial or least-squares quadratic polynomial with m points y_n, y_{n-1}, y_{n-2}, …, y_{n-m+1} to return:

1. the best estimator of y_n,
2. the best estimator of y_{n+1},
3. the best estimator of the change per Δt (slope) at time t_n,
4. the best estimator of the integral of y from t_{n-1} to t_n, and
5. the best estimator of the change per Δt per Δt (concavity) at time t_n but for the least-squares quadratic polynomial, only.

Additionally, there are functions to:

1. use a least-squares linear polynomial to estimate how many time steps Δt from t_n that the signal will zero,
2. use a least-squares quadratic polynomial to estimate how many time steps Δt from t_n that the signal will zero, and
3. use a least-squares quadratic polynomial to estimate how many time steps Δt from t_n that the signal will reach a local extrema.

Finally, all systems experience *jitter*, in the sense that readings that are meant to be taken periodically are not actually periodically. If this difference is negligible, then this is not a problem, but suppose that the reading y_n took place at time $t_n + \varepsilon\Delta t$ where $-1 < \varepsilon < 1$, although, it is better if $-0.1 < \varepsilon < 0.1$. All of these coefficients were found using the symbolic computation language Maple, and an example of the algorithms used are shown in Appendix A. We will look at all of these, starting with least-squares linear polynomials.

3.1 Least-squares linear polynomials

To visualize least-squares linear polynomials, Figure 1 shows the least-squares linear polynomial passing through eight points, and that polynomial evaluated at t_n is the best approximation of y_n.

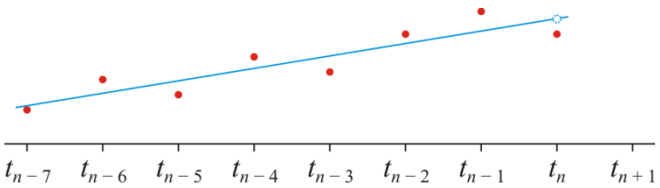

Figure 1. A least-squares linear polynomial passing through eight points and the best approximation of the point y_n.

The best linear estimator of the actual value at time t_n of this signal can be found by calculating

$$\hat{y}(t_n) = \tfrac{15}{36} y_n + \tfrac{12}{36} y_{n-1} + \tfrac{9}{36} y_{n-2} + \tfrac{6}{36} y_{n-3} + \tfrac{3}{36} y_{n-4} - \tfrac{3}{36} y_{n-6} - \tfrac{6}{36} y_{n-7}\,.$$

You will note that it does not depend on y_{n-5}. The reader is invited to observe that this in fact true through a few trials with different values at t_n. Also, the reader will note that the sum of the coefficients is one, as adding a constant to each reading should add that same constant to the estimator.

Next, Figure 2 shows that least-squares polynomial evaluated at t_{n+1}, the best approximation of the signal one period into the future, which can be found using the formula

$$\hat{y}\left(t_{n+1}\right)=\tfrac{14}{28}y_n+\tfrac{11}{28}y_{n-1}+\tfrac{8}{28}y_{n-2}+\tfrac{5}{28}y_{n-3}+\tfrac{2}{28}y_{n-4}-\tfrac{1}{28}y_{n-5}-\tfrac{4}{28}y_{n-6}-\tfrac{7}{28}y_{n-7},$$

where the reader will note that the some of the coefficients is one, where again, as adding a constant to each reading should add that same constant to the same estimator.

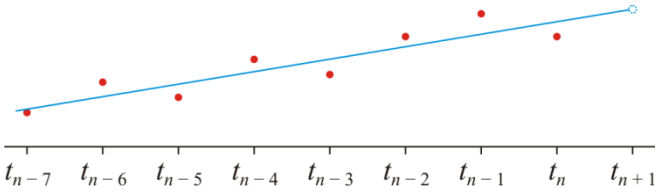

Figure 2. A least-squares linear polynomial passing through eight points and the best current approximation of the point y_{n+1}.

The approximation of the change per period is simply the slope of the best-fitting line, given by the formula

$$\hat{y}^{(1)}\left(t_n\right)\Delta t=\tfrac{7}{84}y_n+\tfrac{5}{84}y_{n-1}+\tfrac{3}{84}y_{n-2}+\tfrac{1}{84}y_{n-3}-\tfrac{1}{84}y_{n-4}-\tfrac{3}{84}y_{n-5}-\tfrac{5}{84}y_{n-6}-\tfrac{7}{84}y_{n-7},$$

and to find the actual rate of change per unit time (speed), one need calculate $\hat{y}^{(1)}\left(t_n\right)=\dfrac{\frac{7}{84}y_n+\cdots-\frac{7}{84}y_{n-7}}{\Delta t}$. The astute reader will note that $\hat{y}\left(t_{n+1}\right)=\hat{y}\left(t_n\right)+\hat{y}^{(1)}\left(t_n\right)\Delta t$; that is, the estimator of the next point is found by adding the estimation of the current point plus the estimation of the change per period. As expected, the sum of the coefficients of the estimator $\hat{y}^{(1)}\left(t_n\right)\Delta t$ is zero, for adding a constant to each value should not change such an estimate.

The estimator of the average value of the signal over the last time period would be given by

$$\hat{\bar{y}}_{[t_{n-1},t_n]}=\tfrac{189}{504}y_n+\tfrac{153}{504}y_{n-1}+\tfrac{117}{504}y_{n-2}+\tfrac{81}{504}y_{n-3}+\tfrac{45}{504}y_{n-4}+\tfrac{9}{504}y_{n-5}-\tfrac{27}{504}y_{n-6}-\tfrac{63}{504}y_{n-7},$$

and again, the astute reader will note that this is the approximation of the average value over the last period is found by subtracting half of the estimation of the change per period from the estimation of the current point. You will note that the sum of the coefficients is one, for adding a constant to each reading should increase the estimator by that same value.

This average value can then be used to estimate the integral, as shown in Figure 3, by calculating $\int_{t_{n-1}}^{t_n}y(t)\,dt=\hat{\bar{y}}_{[t_{n-1},t_n]}\Delta t=\left(\tfrac{189}{504}y_n+\tfrac{153}{504}y_{n-1}+\tfrac{117}{504}y_{n-2}+\tfrac{81}{504}y_{n-3}+\tfrac{45}{504}y_{n-4}+\tfrac{9}{504}y_{n-5}-\tfrac{27}{504}y_{n-6}-\tfrac{63}{504}y_{n-7}\right)\Delta t$.

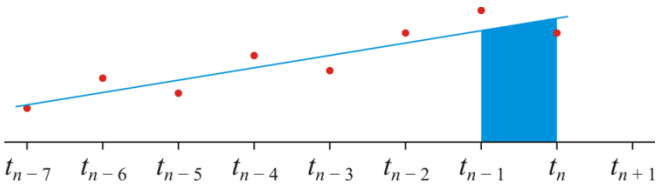

Figure 3. A least-squares linear polynomial passing through eight points and the area under that line from t_{n-1} to t_n.

3.2 Least-squares quadratic polynomials

The same technique is used for approximating y_n and y_{n+1} when using a least-squares quadratic polynomial passing through the eight points, as shown in Figures 4 and 5.

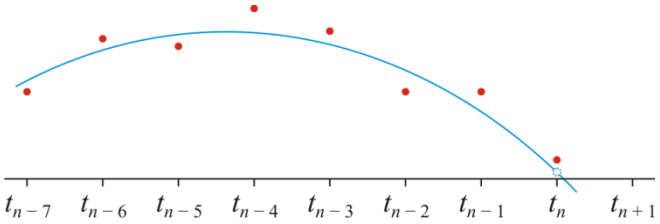

Figure 4. A least-squares quadratic polynomial passing through eight points and the best approximation of the point y_n.

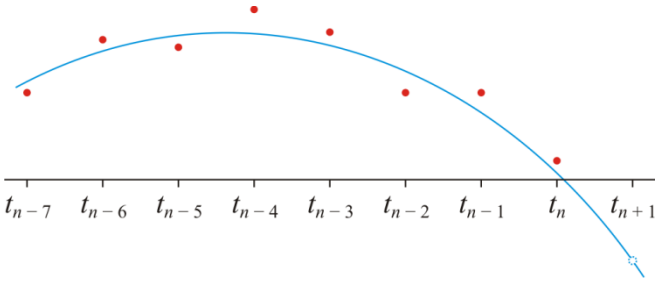

Figure 5. A least-squares quadratic polynomial passing through eight points and the best current approximation of the point y_{n+1}.

These formulas are

$$\hat{y}(t_n) = \tfrac{85}{120} y_n + \tfrac{45}{120} y_{n-1} + \tfrac{15}{120} y_{n-2} - \tfrac{5}{120} y_{n-3} - \tfrac{15}{120} y_{n-4} - \tfrac{15}{120} y_{n-5} - \tfrac{5}{120} y_{n-6} + \tfrac{15}{120} y_{n-7},$$

and

$$\hat{y}(t_{n+1}) = \tfrac{63}{56} y_n + \tfrac{27}{56} y_{n-1} + \tfrac{1}{56} y_{n-2} - \tfrac{15}{56} y_{n-3} - \tfrac{21}{56} y_{n-4} - \tfrac{17}{56} y_{n-5} - \tfrac{3}{56} y_{n-6} + \tfrac{21}{56} y_{n-7},$$

respectively. The reader will also note that, as before, the sum of the coefficients is one.

In order to estimate the instantaneous change per period at time t_n, we must find the derivative of the least-squares quadratic polynomial and then evaluate that at the point t_n, as shown in Figure 6.

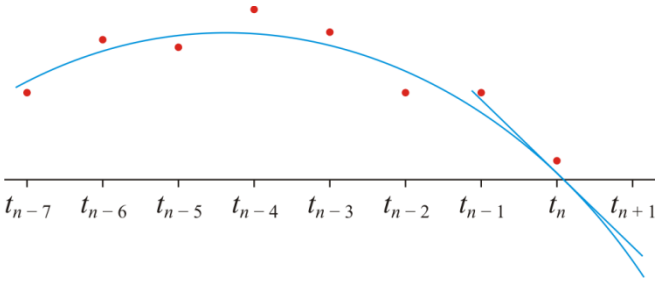

Figure 6. A least-squares quadratic polynomial passing through eight points and the best approximation of the derivative at t_{n+1}.

The formula for this is

$$\hat{y}^{(1)}(t_n)\Delta t = \tfrac{945}{2520} y_n + \tfrac{255}{2520} y_{n-1} - \tfrac{225}{2520} y_{n-2} - \tfrac{495}{2520} y_{n-3} - \tfrac{555}{2520} y_{n-4} - \tfrac{405}{2520} y_{n-5} - \tfrac{45}{2520} y_{n-6} + \tfrac{525}{2520} y_{n-7},$$

and an estimator of the speed is therefore $\hat{y}^{(1)}(t_n) = \dfrac{\frac{945}{2520} y_n + \cdots + \frac{525}{2520} y_{n-7}}{\Delta t}$. In order to find an approximation of the instantaneous change per period per period, all we need do is calculate the second derivative of the least-squares quadratic polynomial, a result that is a constant, and is

$$\hat{y}^{(2)}(t_n)\Delta t^2 = \tfrac{21}{252} y_n + \tfrac{3}{252} y_{n-1} - \tfrac{9}{252} y_{n-2} - \tfrac{15}{252} y_{n-3} - \tfrac{15}{252} y_{n-4} - \tfrac{9}{252} y_{n-5} + \tfrac{3}{252} y_{n-6} + \tfrac{21}{252} y_{n-7},$$

so an approximation of the acceleration or concavity is $\hat{y}^{(2)}(t_n) = \dfrac{\frac{21}{252} y_n + \cdots + \frac{21}{252} y_{n-7}}{\Delta t^2}$. Again, the astute reader will note that $\hat{y}(t_{n+1}) = \hat{y}(t_n) + \hat{y}^{(1)}(t_n)\Delta t + \dfrac{1}{2}\hat{y}^{(2)}(t_n)\Delta t^2$; that is, the estimator of the next point is the estimator of the current point plus the change per period plus one-half the change per period per period.

Finally, the estimator of the average value of the signal over the last time period would be given by

$$\hat{\bar{y}}_{[t_{n-1}t_n]} = \tfrac{8085}{15120} y_n + \tfrac{4935}{15120} y_{n-1} + \tfrac{2475}{15120} y_{n-2} + \tfrac{705}{15120} y_{n-3} - \tfrac{375}{15120} y_{n-4} - \tfrac{765}{15120} y_{n-5} - \tfrac{465}{15120} y_{n-6} + \tfrac{525}{15120} y_{n-7}$$

and , as shown in Figure 7 and therefore an estimator of the integral of the signal over the last time period is $\hat{\bar{y}}_{[t_{n-1}t_n]}\Delta t$.

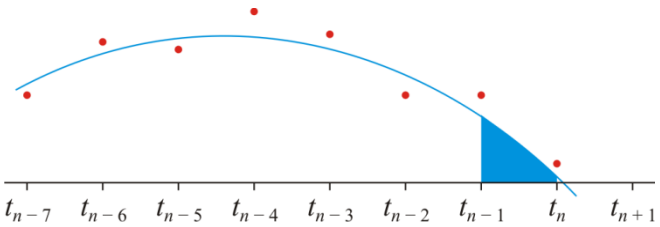

Figure 7. A least-squares quadratic polynomial passing through eight points and the area under that curve from t_{n-1} to t_n.

3.3 Estimating roots and extrema

To approximate when the signal will be zero, if we are finding a least-squares linear polynomial, we need only find the root of that line. Thus, given the estimators above for least-squares linear polynomials for $\hat{y}(t_n)$ and $\hat{y}^{(1)}(t_n)$, then the polynomial passing through $(t_n, \hat{y}(t_n))$ with this slope is $\hat{y}(t_n)+\hat{y}^{(1)}(t_n)(t-t_n)$ and, and therefore the root will be $-\dfrac{\hat{y}(t_n)}{\hat{y}^{(1)}(t_n)}$ relative to t_n. Thus, the actual estimated time that the signal will be zero is $t_n - \dfrac{\hat{y}(t_n)}{\hat{y}^{(1)}(t_n)}$ and an estimator for the number of periods from the current time that the function will be zero is $-\dfrac{\hat{y}(t_n)}{\hat{y}^{(1)}(t_n)\Delta t}$.

Similarly, given the least-squares quadratic polynomial estimators for $\hat{y}(t_n)$, $\hat{y}^{(1)}(t_n)$ and $\hat{y}^{(2)}(t_n)$, it follows that the polynomial is

$$\hat{y}(t_n)+\hat{y}^{(1)}(t_n)(t-t_n)+\frac{1}{2}\hat{y}^{(2)}(t_n)(t-t_n)^2,$$

and therefore the roots relative to t_n are

$$\frac{-\hat{y}^{(1)}(t_n)\pm\sqrt{\hat{y}^{(1)}(t_n)^2-2\hat{y}(t_n)\hat{y}^{(2)}(t_n)}}{\hat{y}^{(2)}(t_n)}$$

periods relative to t_n, and thus, the estimated times the signal will be zero are

$$t_n-\frac{\hat{y}^{(1)}(t_n)}{\hat{y}^{(2)}(t_n)}\pm\frac{\sqrt{\hat{y}^{(1)}(t_n)^2-2\hat{y}^{(2)}(t_n)\hat{y}(t_n)}}{\hat{y}^{(2)}(t_n)},$$

and therefore, we may conclude that the number of periods from the current moment that the zero of the signal will occur is

$$\frac{-\hat{y}^{(1)}(t_n)\pm\sqrt{\hat{y}^{(1)}(t_n)^2-2\hat{y}(t_n)\hat{y}^{(2)}(t_n)}}{\hat{y}^{(2)}(t_n)\Delta t}=\frac{-\hat{y}^{(1)}(t_n)\Delta t\pm\sqrt{\left(\hat{y}^{(1)}(t_n)\Delta t\right)^2-2\hat{y}(t_n)\hat{y}^{(2)}(t_n)\Delta t^2}}{\hat{y}^{(2)}(t_n)\Delta t^2},$$

where you will note that $\hat{y}^{(1)}(t_n)\Delta t$ and $\hat{y}^{(2)}(t_n)\Delta t^2$ are the estimators found above.

Similarly, the extreme point of the least-squares quadratic polynomial is the point $-\dfrac{\hat{y}^{(1)}(t_n)}{\hat{y}^{(2)}(t_n)}$ relative to t_n, and therefore the actual estimated time that the signal will reach an extreme point will be at $t_n-\dfrac{\hat{y}^{(1)}(t_n)}{\hat{y}^{(2)}(t_n)}$

and so the maximum will be reached $-\frac{\hat{y}^{(1)}(t_n)}{\hat{y}^{(2)}(t_n)\Delta t} = -\frac{\hat{y}^{(1)}(t_n)\Delta t}{\hat{y}^{(2)}(t_n)\Delta t^2}$ periods from the present, where again we note that $\hat{y}^{(1)}(t_n)\Delta t$ and $\hat{y}^{(2)}(t_n)\Delta t^2$ are the estimators found above.

3.4 Jitter

Suppose the most recent reading y_n was not taken at time t_n, but rather, it was taken at time $t_n + \varepsilon\Delta t$. In this case, this is still a valid reading, but it breaks the periodic sampling required for these formulas to work—we would be reduced to returning to solving the normal equations. Instead, as we are primarily concerned with the estimator of the signal at time t_n, our goal will be to find a value $\tilde{y}_n$ such that estimator of the signal at t_n using the points $\ldots,(t_{n-1},y_{n-1}),(t_n,\tilde{y}_n)$ is equal to the estimator of the signal at t_n when finding the least-squares polynomial through $\ldots,(t_{n-1},y_{n-1}),(t_n+\varepsilon\Delta t,y_n)$. This is shown in Figures 8 and 9.

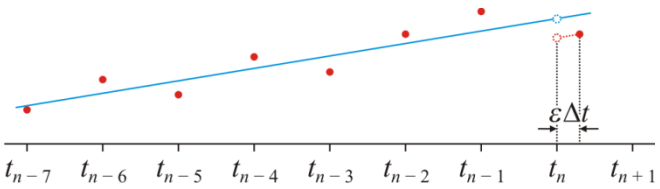

Figure 8. Given that the n^{th} reading was taken at time $t_n + \varepsilon\Delta t$, correcting for this error in the reading time, we approximate that point $\tilde{y}_n$ such that the estimator at time t_n for both data sets $(t_{n-7},y_{n-7}),\ldots,(t_{n-1},y_{n-1}),(t_n+\Delta\varepsilon,y_n)$ and $(t_{n-7},y_{n-7}),\ldots,(t_{n-1},y_{n-1}),(t_n,\tilde{y}_n)$ is equal.

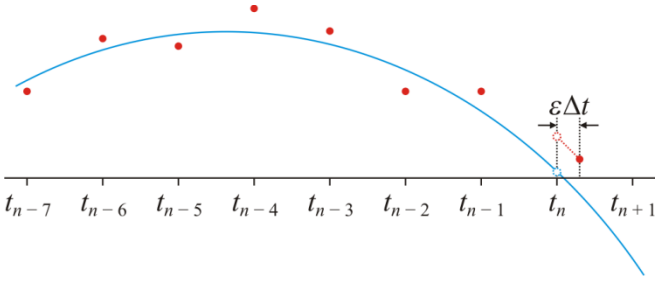

Figure 9. Given that the n^{th} reading was taken at time $t_n + \varepsilon\Delta t$, correcting for this error in the reading time, we find that point $\tilde{y}_n$ such that the estimator at time t_n for both data sets $(t_{n-7},y_{n-7}),\ldots,(t_{n-1},y_{n-1}),(t_n+\Delta\varepsilon,y_n)$ and $(t_{n-7},y_{n-7}),\ldots,(t_{n-1},y_{n-1}),(t_n,\tilde{y}_n)$ is equal.

Unfortunately, this is a rational polynomial expression in ε that would be prohibitive to calculate, and thus we will use only a linear approximation of this calculate. For eight points, the least-squares linear and quadratic polynomial approximations are

$$\tilde{y}_n = y_n + \varepsilon\left(\tfrac{126}{3780}y_n - \tfrac{477}{3780}y_{n-1} - \tfrac{324}{3780}y_{n-2} - \tfrac{171}{3780}y_{n-3} - \tfrac{18}{3780}y_{n-4} + \tfrac{135}{3780}y_{n-5} + \tfrac{288}{3780}y_{n-6} + \tfrac{441}{3780}y_{n-7}\right)$$

and

$$\tilde{y}_n = y_n + \varepsilon\left(-\tfrac{47250}{214200}y_n - \tfrac{64200}{214200}y_{n-1} + \tfrac{4950}{214200}y_{n-2} + \tfrac{46800}{214200}y_{n-3} + \tfrac{61350}{214200}y_{n-4} + \tfrac{48600}{214200}y_{n-5} + \tfrac{8550}{214200}y_{n-6} - \tfrac{58800}{214200}y_{n-7}\right),$$

respectively with the error being $O(\varepsilon^2)$, and thus, if ε is sufficiently small, this will be a reasonable approximation.

To give an example, consider the data

5.7344, 6.7585, 6.4353, 7.9155, 8.4797, 8.7731, 9.6916, 10.5857, 12.0946, 12.5924,

and suppose the last reading was read at time $t_n + 0.1\Delta t$. Thus, given the points $(t_{n-7}, 5.7344)$, …, $(t_{n-1}, 12.0946)$, $(t_n + 0.1\Delta t, 12.5924)$, if we were to find the least-squares linear and quadratic polynomials that best fit this data and evaluate that polynomial at t_n, we would get the estimators 12.3024378951912 and 12.6596226343653, respectively. If we were to ignore the jitter, and simply take the reading at time $t_n + 0.1\Delta t$ as the reading at t_n, then the estimators would be 12.3274 and 12.7280136363636, respectively, and thus introducing relative errors of 0.2% and 0.5%, respectively. We will now consider the two above approaches to correct for the jitter.

For the least-squares linear polynomial we get that the best approximation of the value y_n is 12.50866885714286, so if we find the best estimator of y_n through the data $(t_{n-7}, 5.7344)$, …, $(t_{n-1}, 12.0946)$, $(t_n, 12.50866885714286)$, we get the value 12.2984746970909, which is much closer to 12.3024378951912, reducing the relative error by a factor of ten in this example.

For the least-squares quadratic polynomial we get the best approximation of the value y_n is 12.48695035714286, so if we find the best estimator of y_n through the data $(t_{n-7}, 5.7344)$, …, $(t_{n-1}, 12.0946)$, $(t_n, 12.48695035714286)$, we get the value 12.6628265861818, which is much closer to 12.6596226343653, reducing the relative error by a factor of twenty in this example.

3.5 Summary of a more efficient approach

We have seen that if a sensor is periodically sampled, then information can be extracted from that data using much simpler algorithms than having to solve the normal equations associated with linear regression. It is not only possible to estimate the current or future values of the data, but also rates of change and concavity as well as integrals, and also estimators of when the signal will be zero or at a local minimum or maximum. Finally, we looked at the question of correcting jitter, the phenomenon that some physical systems may not have perfect periodic sampling and to try to correct this. We will now go on to see how we can estimate most of these estimators by only finding two or three estimates: the estimators of the current value, the current change per period and the current change per period per period. Finally, to correct for errors in when sensors are read, techniques can be used to correct for such jitter, as opposed to ignoring it altogether.

4. Multiple estimators required

Suppose you require more than just one estimator. In that case it is always possible to derive all other estimators from the estimators for the current value, change per period, if necessary, and acceleration per period.

4.1 Least-squares linear polynomials

Given the two estimators for $\hat{y}(t_n)$ and $\hat{y}^{(1)}(t_n)\Delta t$, the estimator of the current value and the estimator of the current rate-of-change per period, it follows that:

1. an estimator of the next value is $\hat{y}(t_n)+\hat{y}^{(1)}(t_n)\Delta t$,
2. an estimator of the average value of the signal over the previous interval is $\hat{y}(t_n)-\frac{1}{2}\hat{y}^{(1)}(t_n)\Delta t$ and thus an estimator of the integral over the previous period is $\left(\hat{y}(t_n)-\frac{1}{2}\hat{y}^{(1)}(t_n)\Delta t\right)\Delta t$, and
3. an estimator of when the signal is zero is $-\dfrac{\hat{y}(t_n)}{\hat{y}^{(1)}(t_n)\Delta t}$ as a multiple of periods from the current time and thus an estimator of the absolute time the signal will be zero is $t_n-\dfrac{\hat{y}(t_n)}{\hat{y}^{(1)}(t_n)\Delta t}\Delta t=t_n-\dfrac{\hat{y}(t_n)}{\hat{y}^{(1)}(t_n)}$.

Thus, the total number of calculations required to find all five estimators is $4n$ + 2 FLOPs.

4.2 Least-squares quadratic polynomials

Given the three estimators $\hat{y}(t_n)$, $\hat{y}^{(1)}(t_n)\Delta t$ and $\hat{y}^{(2)}(t_n)\Delta t^2$, the estimators of the current value, current change per period, and change per period per period, it follows that:

1. an estimator of the next value is $\hat{y}(t_n)+\hat{y}^{(1)}(t_n)\Delta t+\frac{1}{2}\hat{y}^{(2)}(t_n)\Delta t^2$,
2. an estimator of the average value over the previous period is $\hat{y}(t_n)-\frac{1}{2}\hat{y}^{(1)}(t_n)\Delta t+\frac{1}{6}\hat{y}^{(2)}(t_n)\Delta t^2$, and thus an estimator of the integral over the previous period is $\left(\hat{y}(t_n)-\frac{1}{2}\hat{y}^{(1)}(t_n)\Delta t+\frac{1}{6}\hat{y}^{(2)}(t_n)\Delta t^2\right)\Delta t$,
3. an estimator for the number of periods to the next extreme point is $-\dfrac{\hat{y}^{(1)}(t_n)\Delta t}{\hat{y}^{(2)}(t_n)\Delta t^2}=-\dfrac{\hat{y}^{(1)}(t_n)}{\hat{y}^{(2)}(t_n)\Delta t}$ from the current time, and thus an estimator of the absolute time to the next extreme point is $t_n-\dfrac{\hat{y}^{(1)}(t_n)}{\hat{y}^{(2)}(t_n)}$, and
4. an estimator for the zeros are found with $\dfrac{-\hat{y}^{(1)}(t_n)\Delta t\pm\sqrt{\hat{y}\left({}^{(1)}(t_n)\Delta t\right)^2-2\hat{y}(t_n)\hat{y}^{(2)}(t_n)\Delta t^2}}{\hat{y}^{(2)}(t_n)\Delta t^2}$.

Consequently, all the estimators can be found with $6n$ + 23 FLOPs.

Estimators required each period

If all estimators are required with each period, it is possible to calculate all of these estimators in O(1) time, requiring no more memory apart from less than a dozen local variables. In the above cases, we assumed the data was periodically sampled,

$$(t_{n-N+1}, y_{n-N+1}), (t_{n-N+2}, y_{n-N+2}), \ldots, (t_{n-2}, y_{n-2}), (t_{n-1}, y_{n-1}), (t_n, y_n),$$

however, the data is really independent of the period, so instead we will use the model

$$(1-N, y_{n-N+1}), (2-N, y_{n-N+2}), \ldots, (-2, y_{n-2}), (-1, y_{n-1}), (0, y_n).$$

In this case, finding the least-squares polynomial reduces to solving the linear equations

$$\begin{pmatrix} \sum_{k=1-N}^{0} k^2 & \sum_{k=1-N}^{0} k \\ \sum_{k=1-N}^{0} k & \sum_{k=1-N}^{0} 1 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} \sum_{k=1-N}^{0} k y_{n-k} \\ \sum_{k=1-N}^{0} y_{n-k} \end{pmatrix}$$

and

$$\begin{pmatrix} \sum_{k=1-N}^{0} k^4 & \sum_{k=1-N}^{0} k^3 & \sum_{k=1-N}^{0} k^2 \\ \sum_{k=1-N}^{0} k^3 & \sum_{k=1-N}^{0} k^2 & \sum_{k=1-N}^{0} k \\ \sum_{k=1-N}^{0} k^2 & \sum_{k=1-N}^{0} k & \sum_{k=1-N}^{0} 1 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} \sum_{k=1-N}^{0} k^2 y_{n-k} \\ \sum_{k=1-N}^{0} k y_{n-k} \\ \sum_{k=1-N}^{0} y_{n-k} \end{pmatrix}.$$

We can used the known formulas to evaluate the left-hand matrix, and if we represent the sums in the right-hand vector as $S_y = \sum_{k=1-N}^{0} y_{n-k}$, $SP_{xy} = \sum_{k=1-N}^{0} k y_{n-k}$ and $SP_{x^2y} = \sum_{k=1-N}^{0} k^2 y_{n-k}$ we get

$$\begin{pmatrix} \frac{N(N-1)(2N-1)}{6} & -\frac{N(N-1)}{2} \\ -\frac{N(N-1)}{2} & N \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} SP_{xy} \\ S_y \end{pmatrix}$$

and

$$\begin{pmatrix} \frac{N(N-1)(3N^2-3N-1)}{30} & -\frac{N^2(N-1)^2}{4} & \frac{N(N-1)(2N-1)}{6} \\ -\frac{N^2(N-1)^2}{4} & \frac{N(N-1)(2N-1)}{6} & -\frac{N(N-1)}{2} \\ \frac{N(N-1)(2N-1)}{6} & -\frac{N(N-1)}{2} & N \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} SP_{x^2y} \\ SP_{xy} \\ S_y \end{pmatrix}.$$

Thus, solving for the coefficients is a simple formula:

$$a = \frac{6\left((N-1)S_y + 2SP_{xy}\right)}{(N-1)N(N+1)}$$
$$b = \frac{2\left((2N-1)S_y + 3SP_{xy}\right)}{N(N+1)}$$

If N is known at compile time, both coefficients can be calculated with only 6 FLOPs. Similarly, the coeffiicients for the least-squares quadratic polynomial can be found with

$$a = \frac{30\left((N-1)(N-2)S_y + 6(N-1)SP_{xy} + 6SP_{x^2y}\right)}{(N-2)(N-1)N(N+1)(N+2)}$$
$$b = \frac{6\left(3(N-1)(N-2)(2N-1)S_y + 2(2N-1)(8N-11)SP_{xy} + 30(N-1)SP_{x^2y}\right)}{(N-2)(N-1)N(N+1)(N+2)}$$
$$c = \frac{3\left((3N^2-3N+2)S_y + 6(2N-1)SP_{xy} + 10SP_{x^2y}\right)}{N(N+1)(N+2)}$$

where if N is known at compile time, these can be calculated with only 15 FLOPs. With these coefficients, we can now compute the following:

Estimator	Linear	Quadratic
Estimate the current value	b	c
Estimate the next value	$b + a$	$a + b + c$
Estimate the rate of change per period	a	b
Estimate the rate of change per period per period	–	$2a$
Estimate the average over the last period	$-\frac{1}{2}a + b$	$\frac{1}{3}a - \frac{1}{2}b + c$
Estimate the average over the next period	$\frac{1}{2}a + b$	$\frac{1}{3}a + \frac{1}{2}b + c$
Estimate when the signal will be zero	$-b/a$	
Estimate when the signal will reach an extremum		$-\frac{1}{2}b/a$

Thus, given the coefficients of the least-squares polynomials, we can quickly find all estimators in O(1) time. The only issue is, however, that we must have S_y, SP_{xy} and for the quadratic case, SP_{x^2y}. Nominally, each of these requires O(N) time to calculate, however, we are fortunate, as we will see next.

Suppose we have the sum of the y-values from the last period up to y_{-1},

$$S_y^{(-1)} = y_{-N} + y_{1-N} + y_{2-N} + \cdots + y_{-3} + y_{-2} + y_{-1} \quad ;$$

and we now want to compute the sum of the y-values up to y_0,

$$S_y^{(0)} = \quad y_{1-N} + y_{2-N} + \cdots + y_{-3} + y_{-2} + y_{-1} + y_0 \,.$$

You will note that we can compute the next sum using the previous sum in O(1) time:

$$S_y^{(0)} = S_y^{-1} - y_{-N} + y_0 .$$

Thus, if we track the previous sum of the last N y-values, then calculating S_y for the current last N y-values can be quickly calculated in O(1) time.

More difficult, however, may be to see how we can calculate SP_{xy} given a previously calculated,

$$SP_{xy}^{(-1)} = -(N-1)y_{-N} - (N-2)y_{1-N} - (N-3)y_{2-N} + \cdots - 2y_{-3} - 1y_{-2} - 0y_{-1}$$

and we now want to compute the sum of the y-values up to y_0,

$$SP_{xy}^{(0)} = \qquad -(N-1)y_{1-N} - (N-2)y_{2-N} - \cdots - 3y_{-3} - 2y_{-2} - 1y_{-1} - 0y_0 .$$

If we subtract the first from the second, we see that

$$SP_{xy}^{(0)} - SP_{xy}^{(-1)} = (N-1)y_{-N} - y_{1-N} - y_{2-N} + \cdots - y_{-3} - y_{-2} - y_{-1} .$$

Recall, however, that we already have the sum of the y-values, so

$$\begin{aligned} SP_{xy}^{(0)} &= SP_{xy}^{(-1)} + Ny_{-N} - y_{-N} - y_{1-N} - y_{2-N} + \cdots - y_{-3} - y_{-2} - y_{-1} \\ &= SP_{xy}^{(-1)} + Ny_{-N} - S_y^{(-1)} \end{aligned}$$

Thus, we can calculate the current sum-of-products for the last N points as a linear combination of two previously known values, and thus can be calculated in O(1) time. Similarly,

$$\begin{aligned} SP_{x^2y}^{(-1)} &= (N-1)^2 y_{-N} + (N-2)^2 y_{1-N} + (N-3)^2 y_{2-N} + \cdots + 2^2 y_{-3} + 1^2 y_{-2} + 0^2 y_{-1} \\ SP_{x^2y}^{(0)} &= \qquad (N-1)^2 y_{1-N} + (N-2)^2 y_{2-N} + \cdots + 3^2 y_{-3} + 2^2 y_{-2} + 1^2 y_{-1} + 0y_0 \end{aligned}$$

Subtracting the second from the first, we get

$$\begin{aligned} SP_{x^2y}^{(0)} - SP_{x^2y}^{(-1)} &= -(N-1)^2 y_{-N} + (2N-3)y_{1-N} + (2N-5)y_{2-N} + \cdots + 5y_{-3} + 3y_{-2} + 1y_{-1} + 0y_0 \\ &= -N^2 y_{-N} + (2N-1)y_{-N} + (2N-3)y_{1-N} + (2N-5)y_{2-N} + \cdots + 5y_{-3} + 3y_{-2} + 1y_{-1} + 0y_0 \\ &= -N^2 y_{-N} + 2(N-1)y_{-N} + 2(N-2)y_{1-N} + 2(N-3)y_{2-N} + \cdots + 6y_{-3} + 4y_{-2} + 2y_{-1} \\ &\qquad + y_{-N} + y_{1-N} + y_{2-N} + \cdots + y_{-3} + y_{-2} + y_{-1} \end{aligned}$$

Thus, we note that $SP_{x^2y}^{(0)} = SP_{x^2y}^{(-1)} - N^2 y_{-N} - 2SP_{xy}^{(-1)} + S_y^{(-1)}$, and therefore all of the sums may be calculated in O(1) time based on previous values, even though each sum nominally requires O(N) calculations.

Thus, in conclusion, if these estimators are required with each period, then all estimators can be calculated in O(1) time, with a relatively small number of FLOPs.

Next, to deal with jitter, we want to find a y^* such that the constant coefficient solution b of

$$\begin{pmatrix} \frac{N(N-1)(2N-1)}{6}+\varepsilon^2 & -\frac{N(N-1)}{2}+\varepsilon \\ -\frac{N(N-1)}{2}+\varepsilon & N \end{pmatrix}\begin{pmatrix} a \\ b \end{pmatrix}=\begin{pmatrix} SP_{xy}+\varepsilon y \\ S_y \end{pmatrix}$$

has the same constant coefficient solution of

$$\begin{pmatrix} \frac{N(N-1)(2N-1)}{6} & -\frac{N(N-1)}{2} \\ -\frac{N(N-1)}{2} & N \end{pmatrix}\begin{pmatrix} a \\ b \end{pmatrix}=\begin{pmatrix} SP_{xy} \\ S_y-y+y^* \end{pmatrix};$$

that is, we want to find that y^* such that the estimator of the current value remains unchanged. In this case, it can be shown by solving both systems of linear equations and solving for y^*, we get that

$$y^*=\frac{\begin{array}{l} N^2(N-1)(2N-1)(N+1)y \\ \quad +\left(3N(N-1)\left(N^2+9N-4\right)y-12N(N-1)(2N-1)S_y-6N(7N-5)SP_{xy}\right)\varepsilon \\ \quad +\left(6(3N-1)(N-2)y-6(3N-1)(N-2)S_y+36(1-N)SP_{xy}\right)\varepsilon^2 \end{array}}{(2N-1)(N-1)\left((N+1)N^2+12\varepsilon(N+\varepsilon)\right)}.$$

While this appears complex, note that most of the coefficients can be calculated at compile time, so this requires in fact only 20 FLOPs—a small price to pay to deal reasonably with jitter. Note that in this case, we can use the exact formula, as opposed to the series approximation above, because we have S_y and SP_{xy}.

To demonstrate that this works, let us assume the most recent reading is taken 0.1 times the period after it should have been taken. The least-squares linear polynomial that passes through the data

(–9, 5.7344), (–8, 6.7585), (–7, 6.4353), (–6, 7.9155), (–5, 8.4797),
(–4, 8.7731), (–3, 9.6916), (–2, 10.5857), (–1, 12.0946), (0.1, 12.5924)

is 12.3024378951912 + 0.756427148149479*n*, which evaluated at $n = 0$ is 12.3024378951912. The above formula for the correction for jitter is $y^* = 12.52014127555338$, and the least-squares linear polynomial passing through the data

(–9, 5.7344), (–8, 6.7585), (–7, 6.4353), (–6, 7.9155), (–5, 8.4797),
(–4, 8.7731), (–3, 9.6916), (–2, 10.5857), (–1, 12.0946), (0, 12.52014127555338)

is 12.3024378951912 + 0.756351948363518*n*, which has the same constant coefficient and the relative difference in the slope is 0.01%.

If jitter was not corrected for (that is, the reading taken 0.1 times the period after the time it should have been taken), the least-squares linear polynomial that passes through

(–9, 5.7344), (–8, 6.7585), (–7, 6.4353), (–6, 7.9155), (–5, 8.4797),
(–4, 8.7731), (–3, 9.6916), (–2, 10.5857), (–1, 12.0946), (0, 12.5924)

is 12.3274 + 0.760293333333334*n*, where the relative difference in the estimator of the current point is close to 0.2% and the error in the slope is 0.5%. If we assume the jitter was 50% of a period, the relative difference in the estimator for the current point increases to 1% if the jitter is ignored.

The value y^* to correct for jitter in least squares quadratic polynomials is shown Appendix C. Again, this can be calculated exactly, as opposed to using a series approximation as above, because we have the three sums S_y, SP_{xy} and SP_{x^2y} already calculated.

Which to use: linear or quadratic?

If the sensor is read sufficiently often such that on that interval the reading is essentially linear, then using a least-squares linear polynomial is more than appropriate. For example, suppose we are taking an outdoor temperature reading outside once a minute. In this case, if we were using ten points to approximate the behavior of the temperature, then a linear polynomial would be sufficient. Similarly, in a self-driving vehicle, if a reading was being taken at a rate of 1 kHz, then even at 180 km/h, the vehicle would only move 0.5 m per 10 μs, giving access to 11 samples in that time. Essentially, there should not be the possibility for a significant acceleration or deceleration over the period in which the samples are taken. Alternatively, if samples are not being taken sufficiently often, it may be better to use least-squares quadratic polynomials. Samples should, however, still be taken sufficiently often to ensure that it is unlikely that a serious change in surge (third derivative) during the time during which the samples are taken.

Associated implementations

The library `least_sqrs_est.h` is a C++ implementation of these estimators, but it is not efficient, as the coefficients are calculated during the execution, and thus the number of FLOPs is by no means optimal. There is, however, output to the `std::clog` stream displays C++ code that would optimally calculate that specific estimator. Sample output to this stream is shown in Appendix B.

Summary

It is difficult to extract information from noisy data, and one of the most reliable techniques are least-squares polynomials. When the signal is sampled periodically, the evaluation of specific estimators simplifies to a linear combination of the samples, significantly reducing the number of floating-point operations.

Appendix A: Using Maple to find these coefficients

The tool used to find these expressions was Maple:

```
> for N from 3 to 10 do
    # Create the list [t, t - Delta[t], t - 2*Delta[t], ..., t - (N-1)*Delta[t]]
    t_values := [seq( t - k*Delta[t], k = 0..N - 1 )]:
    # Create the list [y[n], y[n - 1], y[n - 2], ..., y[n - (N-1)]]
    y_values := [seq(  y[n-k], k = 0..N - 1 )]:

    # Uncomment the first for a quadratic polynomial, and
    # the second for a linear polynomial
    p := CurveFitting:-LeastSquares( t_values, y_values, tau, curve = a*tau^2 + b*tau + c ):
    # p := CurveFitting:-LeastSquares( t_values, y_values, tau, curve = b*tau + c ):

    p := simplify( eval( p, tau = t ), size ) assuming real;
    # p := simplify( eval( p, tau = t + h ), size ) assuming real;
    # p := simplify( eval( diff( p, tau ), tau = t + h ), size ) assuming real;
    # p := simplify( eval( diff( p, tau, tau ), tau = t + h ), size ) assuming real;
    # p := simplify( int( p, tau = t - Delta[t]..t ), size ) assuming real;
    p := expand( p );
    coefficients := [seq( coeff( p, y[n - k] ), k = 0..N - 1 )];
    denominator := ilcm( seq( denom( k ), k = coefficients ) );
    numerators := expand( coefficients*denominator );
    q := CurveFitting:-PolynomialInterpolation( [seq( -k, k = 0..N - 1 )], numerators, n );
    print( q/denominator );
end do:
```

Appendix B

A formatted sample output of what appears on the `std::clog` stream with $N = 5$ and $N = 10$ points is shown here. Recall that the compiler will calculate any arithmetic operations on floating-point literals:

```
// Five points
// Linear polynomial approximating y[n]
   y_n = 9.0/15.0*y[n] + 6.0/15.0*y[n-1] + 3.0/15.0*y[n-2] - 3.0/15.0*y[n-4];

// Linear polynomial approximating y[n + 1]
   y_n_1 = 8.0/10.0*y[n] + 5.0/10.0*y[n-1] + 2.0/10.0*y[n-2] - 1.0/10.0*y[n-3] - 4.0/10.0*y[n-4];

// Linear polynomial approximating the change per period at y[n]
   dy_n = 4.0/20.0*y[n] + 2.0/20.0*y[n-1] - 2.0/20.0*y[n-3] - 4.0/20.0*y[n-4];

// Linear polynomial approximating the average value over the last period
   average_y_n = 60.0/120.0*y[n] + 42.0/120.0*y[n-1] + 24.0/120.0*y[n-2] + 6.0/120.0*y[n-3] - 12.0/120.0*y[n-4];

// Linear correction for jitter
   y[n] += epsilon*(-36.0/540.0*y[n] - 126.0/540.0*y[n-1] - 36.0/540.0*y[n-2] + 54.0/540.0*y[n-3] + 144.0/540.0*y[n-4]);

// Quadratic polynomial approximating y[n]
   y[n] = 31.0/35.0*y[n] + 9.0/35.0*y[n-1] - 3.0/35.0*y[n-2] - 5.0/35.0*y[n-3] + 3.0/35.0*y[n-4];

// Quadratic polynomial approximating y[n+1]
   y_n_1 = 18.0/10.0*y[n] - 8.0/10.0*y[n-2] - 6.0/10.0*y[n-3] + 6.0/10.0*y[n-4];

// Quadratic polynomial approximating the change per period at y[n]
   dy_n = 162.0/210.0*y[n] - 39.0/210.0*y[n-1] - 120.0/210.0*y[n-2] - 81.0/210.0*y[n-3] + 78.0/210.0*y[n-4];

// Quadratic polynomial approximating the concavity at y[n]
   ddy_n = 6.0/21.0*y[n] - 3.0/21.0*y[n-1] - 6.0/21.0*y[n-2] - 3.0/21.0*y[n-3] + 6.0/21.0*y[n-4];

// Quadratic polynomial approximating the average value over the last period
   average_y_n = 690.0/1260.0*y[n] + 411.0/1260.0*y[n-1] + 192.0/1260.0*y[n-2] + 33.0/1260.0*y[n-3] - 66.0/1260.0*y[n-4];

// Quadratic correction for jitter
   y[n] += epsilon*(
       -4374.0/6510.0*y[n] - 249.0/6510.0*y[n-1] + 4206.0/6510.0*y[n-2] + 3321.0/6510.0*y[n-3] - 2904.0/6510.0*y[n-4]
   );
```

```
// Ten points
// Linear polynomial approximating y[n]
   y_n = 19.0/55.0*y[n]   + 16.0/55.0*y[n-1] + 13.0/55.0*y[n-2] + 10.0/55.0*y[n-3] + 7.0/55.0*y[n-4]
       +  4.0/55.0*y[n-5] +  1.0/55.0*y[n-6] -  2.0/55.0*y[n-7] -  5.0/55.0*y[n-8] - 8.0/55.0*y[n-9];

// Linear polynomial approximating y[n + 1]
   y_n_1 = 18.0/45.0*y[n]   + 15.0/45.0*y[n-1] + 12.0/45.0*y[n-2] + 9.0/45.0*y[n-3] + 6.0/45.0*y[n-4]
         +  3.0/45.0*y[n-5]                    -  3.0/45.0*y[n-7] - 6.0/45.0*y[n-8] - 9.0/45.0*y[n-9];

// Linear polynomial approximating the change per period at y[n]
   dy_n = 9.0/165.0*y[n]   + 7.0/165.0*y[n-1] + 5.0/165.0*y[n-2] + 3.0/165.0*y[n-3] + 1.0/165.0*y[n-4]
        - 1.0/165.0*y[n-5] - 3.0/165.0*y[n-6] - 5.0/165.0*y[n-7] - 7.0/165.0*y[n-8] - 9.0/165.0*y[n-9];

// Linear polynomial approximating the average value over the last period
   average_y_n = 315.0/990.0*y[n]   + 267.0/990.0*y[n-1] + 219.0/990.0*y[n-2] + 171.0/990.0*y[n-3] + 123.0/990.0*y[n-4]
               +  75.0/990.0*y[n-5] +  27.0/990.0*y[n-6] -  21.0/990.0*y[n-7] -  69.0/990.0*y[n-8] - 117.0/990.0*y[n-9];

// Linear correction for jitter
   y[n] += epsilon*(
       459.0/9405.0*y[n]   - 831.0/9405.0*y[n-1] - 636.0/9405.0*y[n-2] - 441.0/9405.0*y[n-3] - 246.0/9405.0*y[n-4]
     -  51.0/9405.0*y[n-5] + 144.0/9405.0*y[n-6] + 339.0/9405.0*y[n-7] + 534.0/9405.0*y[n-8] + 729.0/9405.0*y[n-9]
   );

// Quadratic polynomial approximating y[n]
   y[n] = 136.0/220.0*y[n]   + 84.0/220.0*y[n-1] + 42.0/220.0*y[n-2] + 10.0/220.0*y[n-3] - 12.0/220.0*y[n-4]
        - 24.0/220.0*y[n-5] - 26.0/220.0*y[n-6] - 18.0/220.0*y[n-7]                     + 28.0/220.0*y[n-9];

// Quadratic polynomial approximating y[n+1]
   y_n_1 = 108.0/120.0*y[n]   + 60.0/120.0*y[n-1] + 22.0/120.0*y[n-2] - 6.0/120.0*y[n-3] - 24.0/120.0*y[n-4]
         - 32.0/120.0*y[n-5] - 30.0/120.0*y[n-6] - 18.0/120.0*y[n-7] + 4.0/120.0*y[n-8] + 36.0/120.0*y[n-9];

// Quadratic polynomial approximating the change per period at y[n]
   dy_n = 2052.0/7920.0*y[n]   + 876.0/7920.0*y[n-1] -  30.0/7920.0*y[n-2] - 666.0/7920.0*y[n-3] - 1032.0/7920.0*y[n-4]
        - 1128.0/7920.0*y[n-5] - 954.0/7920.0*y[n-6] - 510.0/7920.0*y[n-7] + 204.0/7920.0*y[n-8] + 1188.0/7920.0*y[n-9];

// Quadratic polynomial approximating the concavity at y[n]
   ddy_n = 36.0/792.0*y[n]   + 12.0/792.0*y[n-1] - 6.0/792.0*y[n-2] - 18.0/792.0*y[n-3] - 24.0/792.0*y[n-4]
         - 24.0/792.0*y[n-5] - 18.0/792.0*y[n-6] - 6.0/792.0*y[n-7] + 12.0/792.0*y[n-8] + 36.0/792.0*y[n-9];

// Quadratic polynomial approximating the average value over the last period
   averagey_y_n = 23580.0/47520.0*y[n]   + 15636.0/47520.0*y[n-1] + 9102.0/47520.0*y[n-2] + 3978.0/47520.0*y[n-3]
                +  264.0/47520.0*y[n-4] -  2040.0/47520.0*y[n-5] - 2934.0/47520.0*y[n-6] - 2418.0/47520.0*y[n-7]
                -  492.0/47520.0*y[n-8] +  2844.0/47520.0*y[n-9];

// Quadratic correction for jitter
   y[n] += epsilon*(
       -106704.0/1077120.0*y[n]   - 291504.0/1077120.0*y[n-1] -  82104.0/1077120.0*y[n-2] +  70056.0/1077120.0*y[n-3]
      + 164976.0/1077120.0*y[n-4] + 202656.0/1077120.0*y[n-5] + 183096.0/1077120.0*y[n-6] + 106296.0/1077120.0*y[n-7]
      -  27744.0/1077120.0*y[n-8] - 219024.0/1077120.0*y[n-9]
   );
```

Appendix C

Correcting for jitter in least squares quadratic polynomials. Note that most if N is known at compile time, most coefficients can be calculated explicitly, reducing the run time to 44 FLOPs at run time.

```
y = (
  (
    (
      (
        (
          ((((300.0*N - 2760.0)*N + 2940.0)*N - 1920.0)*N + 720.0)*y
            + ((((-300.0*N + 2760.0)*N - 2940.0)*N + 1920.0)*N - 720.0)*S_y
            + (((-1800.0*N + 8640.0)*N - 6840.0)*N + 2160.0)*SP_xy
            - 1800.0*(N - 1.0)*(N - 2.0)*SP_xxy
        )*epsilon + (
          720.0*N*(N - 1.0)*(((N - 6.0)*N + 3.0)*N - 2.0)*y
            - 720.0*N*(N - 1.0)*(((N - 6.0)*N + 3.0)*N - 2.0)*S_y
            - 120.0*N*(2.0*N - 1.0)*((17.0*N - 57.0)*N + 34.0)*SP_xy
            - 360.0*N*((11.0*N - 27.0)*N + 22.0)*SP_xxy
        )
      )*epsilon + (
        (((((((10.0*N + 622.0)*N - 3182.0)*N + 4030.0)*N - 1604.0)*N - 116.0)*N + 528.0)*N - 144.0)*y
          + ((((((-612.0*N + 3132.0)*N - 4080.0)*N + 1644.0)*N + 156.0)*N - 528.0)*N + 144.0)*S_y
          - 36.0*(N - 1.0)*((((89.0*N - 183.0)*N + 52.0)*N + 42.0)*N - 12.0)*SP_xy
          + ((((-3000.0*N + 6000.0)*N - 3840.0)*N - 1320.0)*N + 720.0)*SP_xxy
      )
    )*epsilon + (
      6.0*N*(N - 1.0)*(N - 2.0)*(2.0*N - 1.0)*(N + 1.0)*(((N + 21.0)*N - 16.0)*N + 12.0)*y
        - 36.0*N*(N - 1.0)*(N - 2.0)*(2.0*N - 1.0)*(N + 1.0)*(3.0*(N - 1.0)*N + 2.0)*S_y
        - 24.0*N*(2.0*N - 1.0)*(N + 1.0)*(((21.0*N - 60.0)*N + 56.0)*N - 20.0)*SP_xy
        - 180.0*N*(N - 1.0)*(N + 1.0)*((5.0*N - 8.0)*N + 4.0)*SP_xxy
    )
  )*epsilon + (N*N*(N - 1.0)*(N - 2.0)*(2.0 + N)*(3.0*(N - 1.0)*N + 2.0)*(N + 1.0)*(N + 1.0)*y)
) / (
  (
    (
      (
        180.0*(N - 2.0)*(N - 1.0)
           *((3.0*N - 3.0)*N + 2.0)*epsilon
        + 360.0*N*(N - 2.0)*(N - 1.0)*((3.0*N - 3.0)*N + 2.0)
      )*epsilon + 36.0*(N - 2.0)*((7.0*N + 1.0)*N - 1.0)
         *(N - 1.0)*((3.0*N - 3.0)*N + 2.0)
    )*epsilon + 36.0*N*(N - 1.0)*(N - 2.0)*(2.0*N - 1.0)
       *(N + 1.0)*((3.0*N - 3.0)*N + 2.0)
  )*epsilon + N*N*(N - 1.0)*(N - 2.0)*(2.0 + N)
     *((3.0*N - 3.0)*N + 2.0)*(N + 1.0)*(N + 1.0)
);
```