Jay N. Damask

Signal Processing for Quantitative Finance

Lecture Notes for Time-Series Analysis
Baruch College
City University of New York
MTH 9893, Winter 2015

January 19, 2015

Preface

This book originates from two sources. The first and more immediate is the class that I teach in the Masters Program in Financial Engineering at Baruch College, City University of New York. When I was asked to prepare a new course I used it as an opportunity to frame time-series analysis from an engineer's perspective. Over the several years that I have taught this class, and with input from my students, many of whom have asked insightful questions where I had gaps, I have been able to merge the sensibilities of practical financial analysis and tools from signal processing into a rigorous yet pedagogical exposition. The result is a tight coupling between online and offline analysis, an appreciation of the need to know the temporal scale of the problem, and the development of domain-specific criteria with which to judge filter performance for financial applications.

The second source is my industry experience and education. Over the years I have built filters in electronic servo systems; in waveguide and free-space optics; and more recently in software systems used in finance. Many years ago I invented a then-new class of filter, one that operates on the group-delay properties of optical polarization. In algorithmic trading there are many opportunities to *manage detail*, as I call it, that require a thorough understanding of signals analysis. In sum, filtering problems simply come up again and again.

I intend this book to complement econometrics. Econometrics' strongest contribution, in my opinion, is to the study of heteroscedasticity, a singularly financial concept. Moreover, like some aspects of signals analysis, the objective of finding a white-noise source, variously known as innovation sequences and market invariants, is correct in that predictable structure is identified and separated from truly random sources. Where signals analysis contributes is in the connection of canonical ARIMA models from econometrics to the temporal scale embedded in these models once coefficients are chosen; to a critical view of ARIMA solutions and the problems of overfitting; and to an understanding of the spectral properties of a filter alongside the temporal. In contrast to much of econometrics, the signals perspective typically starts

with a filter, not a finite-difference equation, and looks to apply that filter in suitable ways. Together, signals analysis and the study of heteroscedasticity form a powerful combination.

Lastly, I have used the majority of the material covered in this book in production systems to trade in the capital markets. Therefore, while this monograph covers a great deal of theory, in the end this is a practitioner's guide.

Unlike other works of mine, this monograph is not intended as a literature survey. Rather, I base the principal material on the texts from my education at the Massachusetts Institute of Technology and from my professional experience. Key texts are from courses 6.003, *Signals and Systems*, Prof. W. Siebert [15]; 6.341, *Discrete-time Signal Processing*, Prof. A. Oppenheim [6]; 6.302, *Feedback Systems*, Prof. J. Roberge [11]; and 18.04, *Complex Variables with Applications*, Prof. L. Trefethen [12]. To these professors, and to the Institute, I am deeply grateful. From professional experience, additional topics on filters derive from Brown and Hwang [1]; Press, Teukolsky, Vetterling and Flannery [9]; and Schlichtharle [13]. Topics in stochastic processes, which are central to Chapter **??**, derive from Davenport [2], Karlin and Taylor [3], Oppenheim and Verghese [7], and Shreve [14]. A comprehensive review of econometrics is provided by Tsay [16].

Since my earlier works, Wikipedia [17] has become a resource that warrants recognition. From Wikipedia itself,

> *As with any source, especially one of unknown authorship, you should be wary and independently verify the accuracy of Wikipedia information if possible. For many purposes, but particularly in academia, Wikipedia may not be an acceptable source...*

While my work is not originally sourced from Wikipedia, the pages on convolution [18], Fourier transforms [19], Laplace transforms [20] and z-transforms [22] are excellent. It is appropriate that I acknowledge the wealth of readily available information on these pages.

In addition to my principal sources, I cite in place the papers and lecture notes that I have found useful while researching and writing this monograph.

Acknowledgements...

Contents

Chapter 1
The Investment Horizon and Managing Detail

To manage detail is to create consistency between the rate of investment actions one wishes to take and the rate of arrival of investment-related information, especially when such information arrives at a higher frequency. The managing of detail can be done *online* when real-time investment decisions have to be made, or *offline* for calibration or historical analysis purposes. In either case, high-quality smoothing has an integral role in the pursuit of sound estimators for the current market state and for investment forecasts.

In this chapter I make the case for managing detail by smoothing techniques in the context of investment decisions. To make the case I appeal to the reader's experience in finance rather than mathematical background. Yet the story of how to smooth with mathematical rigor and how to connect off-line with on-line smoothing requires substantial digressions into Fourier analysis, complex analysis and polynomial analysis. And still, the goal is always the same: how is it that we can manage detail to support our investment decisions.

1.1 Temporal Scale

Temporal scale is a principal concept of this book. Throughout engineering and the sciences, scale, whether real or effective or equivalent, is fundamental to theory, experiment and understanding. There are many scales: length, time, mass and force are obvious examples. *Scale* provides a basis for comparison to determine what features may be relevant and what are not.

A motivating scale for this work is the interval of time between an investment decision and the target investment horizon, whether measured in continuous- or discrete-time, wall clock or event clock. Figure 1.1 marks these two points in time. The horizon interval h is a starting point for any investment decision, it is an exogenous variable to any analysis.

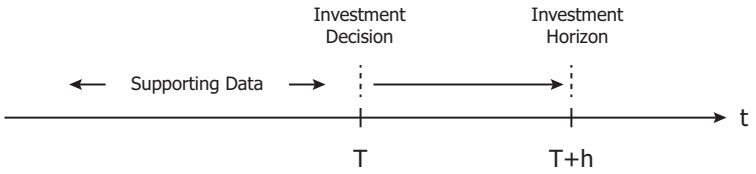

Fig. 1.1 The interval of time between an investment decision and the associated investment horizon provides a natural scale against which supporting data can be analyzed.

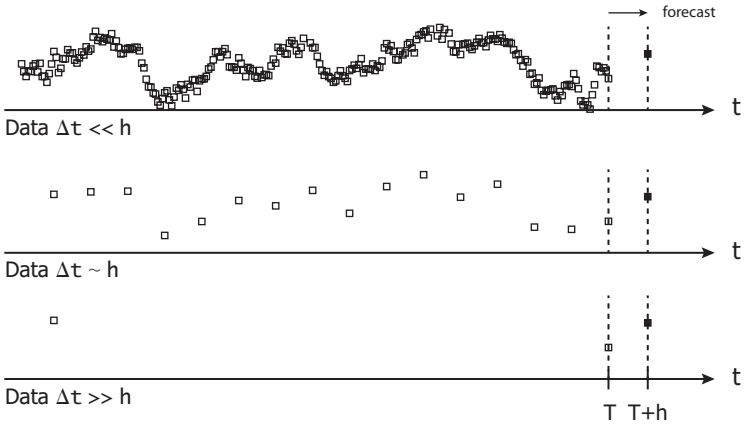

Fig. 1.2 Data intervals with respect to the investment horizon time interval h – top to bottom: a high level of detail; detail on the scale of the investment horizon; and far too little detail to make a sound decision.

Data that is used to support a decision must be commensurate with horizon h. As illustrated in Fig. 1.2, a characteristic data interval much smaller than the horizon, $\Delta t \ll h$, is useful in that there are many updates during an open investment, but yet this data has too much detail that left unmanaged will likely result in actions on the original investment that are closer to Δt than h in scale.

The middle figure illustrates an update interval that is commensurate with the horizon, $\Delta t \sim h$. Here one can train a supervised learning system to forecast order-h ahead of the current time. A drawback is that if, underlying this data interval, a faster data set is equally informative then one is left waiting longer than others for the next event.

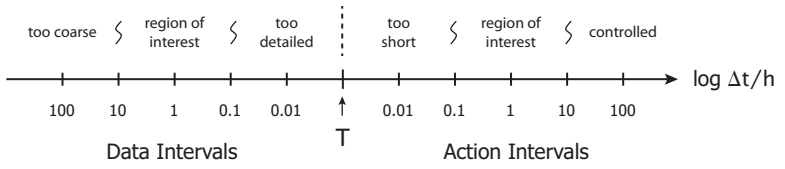

Fig. 1.3 Log-scale of intervals before and after the investment decision. The region of interest is centered at the target investment horizon and includes a suitable surrounding region. This region of interest for the trade duration is reflected over to the relevant data interval.

The bottom figure illustrates the $\Delta t \gg h$ case. From my perspective this case speaks to finding an alternative data set so that at least $\Delta t \sim h$. There are always exceptions of course: an event-based trade such as a economic release typically has a short horizon compared to the data interval. This is a valid counter example but one that I will not focus on.

As an obvious example of data interval and investment horizon, consider your 401(k)[1]. Does streaming data from the equity exchanges really effect your investment decision? When is the last time you even looked at your 401(k)? On the opposite extreme, when trying to win a tick in the order book, does daily open/high/low/close data have anything to do with model calibration?

Figure 1.3 bisects the time axis at the moment of investment decision. Human perception of time is logarithmic, so accordingly I've marked interval sizes for data on the left and actions on the right, both normalized to the agreed upon investment horizon h. The target action will be made at the horizon, which in this figure is labeled 1, or $\Delta t = h$. One cannot be inflexible to events as they happen, so a realized action may come sooner or later than the target. Still, I have labeled a *region of interest* around the target horizon to indicate the range within which actions are consistent with the investment thesis. Actions taken on a scale $\Delta t \ll h$ are too quick and will not lead to the returns anticipated around h. Actions taken on a scale $\Delta t \gg h$ are too long; funding that would otherwise be available for the next investment is tied up.

The region of interest for action intervals is reflected in a region of interest for data intervals. A band within an order of magnitude or so of an update interval h provides an investment-consistent data set. Data intervals that are too long are too coarse. Intervals that are too short have too much detail, but they have the benefit of frequent updates.

Since it is unlikely to find a data set whose increments coincide with h, how can we transform a data set that has too much detail to a consistent one? Figure 1.4 illustrates a classic sample-and-hold approach. At every interval h the underlying series is sampled, that sample value is taken as the prevailing value over the interval. Sample and hold solves the interval consistency problem, but in a suboptimal manner. During a hold interval no update is

[1] A 401(k) is a tax-advantaged investment vehicle for retirement in the United States.

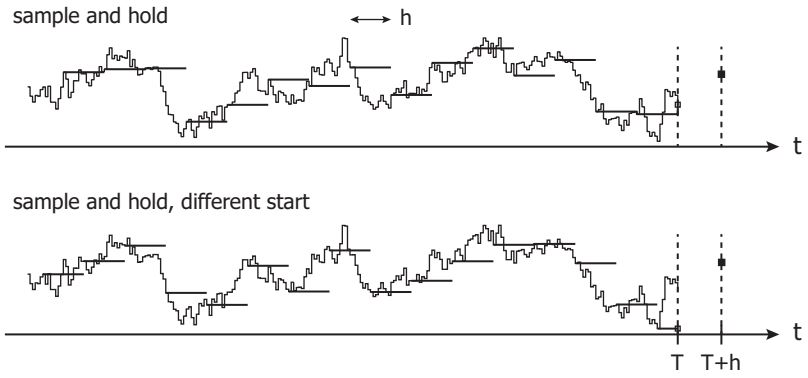

Fig. 1.4 Sample-and-hold is one method to manage detail. Here the data is sampled at period h. The sample-and-hold series are not stable to shifts in starting location, as illustrated in the two figures.

available, regardless of the motion of the underlying. Moreover, the sample series is critically dependent on the initial sample moment. The plots in the figure illustrate two significantly different series that arises from changing the sample location along the series. It is hard to have confidence in sample and hold.

However, confidence aside, sample and hold is popular. Candlestick charting adds high, low and close attributes to the sampled value which is the "open", and the hold interval is often in order-minutes regardless of the underlying activity. Candlesticks are one way to manage detail: summary features are calculated for each interval, the details are discarded.

1.2 Signal, Noise and Detail

I want to consider the terms *signal*, *noise* and *detail* in the context of today's public securities markets so that we can establish a clear vocabulary. Liquid public markets such as the U.S. equities markets operate at very high speeds. Two-sided quotes are continuously produced by market makers and trades occur when a buyer or seller is willing to pay the spread to execute immediately[2]. The U.S. markets operate as a distributed system, where dozens exchange centers run their own order books and trades can be executed on any of the venues, provided the price is not away from the National Best Bid and Offer (N.B.B.O). The distributed nature admits concurrent actions that only become known as concurrent after the information-transit time from one exchange to another.

[2] Here I ignore hidden liquidity effects and NBBO price protection.

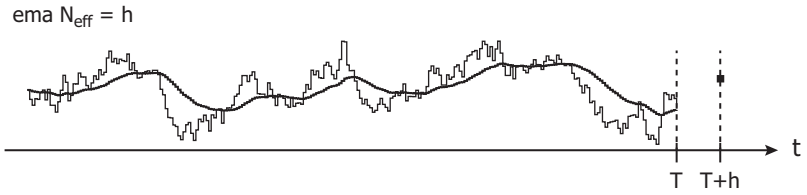

Fig. 1.5 Application of an *ema* to the series in Fig. 1.4 where $2\tau \equiv h$. In comparison to the sample-and-hold, the smooth series is stable to shifts in initial condition and updates for each new input.

Let's set aside execution errors that happen in the market, which are themselves real but do not constitute the median or typical behavior.

I claim that there is *virtually* no noise in the market. Noise exists within the race conditions of packets sent to and from the exchanges as well as orders fed to the matching engines. Wins and losses in these races are quickly corrected with counter actions. However, outside of this distributed system envelope there is price discovery. Price discovery is order-millisecond in time scale among the most liquid U.S. equity securities.

Once we observe prices on their natural price-discovery scale and above, there is no noise. Quote and trades are intended, market participants seek to adjust their positions for their own benefit. What we have is *detail*.

Moreover, at and above the natural price-discovery scale there is no signal. A signal refers to a communications link where there is a transmitter and receiver. While the data that is transmitted through the link is not known *a priori*, the transmission scheme is known. The receiver expects a transmission with a certain format, and conditioned that the format is observed, the receiver extracts a signal that has become noisy due to deleterious physical processes.

So all we have is detail. This book is about managing detail. As we smooth out input market series, such as prices, we will superimpose smooth curves on detailed curves, but we do not have a signal. What we will have is transformation of a detailed curve to a smoother curve that is consistent with a time scale that we supply.

1.3 Managing Detail

As an investor I certainly prefer to have as many updates as possible throughout the whole investment cycle, as long as those updates are informative for my investment and horizon. The electronic markets and attendant data sets such as analyst expectations and news feeds provide detail. Likewise I prefer

to set my investment horizons at a multiple of the median data interval. In this way sound decisions can be made.

In this monograph I focus on smoothing as a means to manage detail. Figure 1.5 illustrates the application of an exponential moving average (*ema*) to the input data set. The output is what I call a *smooth*, in contrast to the *rough* that is the input series. The smooth is updated on each new input point. However, the update is blended with a neighborhood of points in proximity, creating a *local average*. The consequence of this process is that the motion of the smooth has a scale consistency that I have imposed, one that is related to horizon h.

Comparing the rough to the smooth in the figure, following the motion of the smooth leads to investment decisions made on a scale consistent with the horizon. The smooth reacts slowly, by some measure, to abrupt changes. A jump, as they say, is only ever a jump after the fact. Unless a cause is identified, in the moment it is unclear whether the underlying process will revert or persist at a new level. This uncertainty is reflected in the hesitancy of the smooth to adjust any faster than it's natural response.

Managing detail, at its simplest, is the act of matching a well designed smoothing scheme to the behavior and characteristics of the underlying. Moreover, smoothing schemes can be built into models of market dynamics such as market or announcement impacts, and built into curve stripping, volatility models and risk analysis, to name a few. In any case these schemes need to be implemented for real-time, or *online*, use; and calibration, or *offline*, use. The duality of on- and off-line implementations are explored in detail in this monograph.

1.4 A Direct Smoothing Method

A smoothing process transforms an input time series to an output series using a smoothing window. As an abstraction, we want an algorithm Smooth which reads

$$y(t) = \textsc{Smooth}\,(w(t), x(t)) \tag{1.4.1}$$

where the input and output series are $x(t)$ and $y(t)$, respectively, and the smoothing window is $w(t)$. Let's focus on a point in time t_o. To smooth $x(t)$ about t_o is to take a weighted average of values $w_t\, x(t)$ in a neighborhood $\mathcal{N}_\tau$ of t_o, as in

$$y(t_o) = \text{Avg}\,(w_t\; x(t) \mid t \in \mathcal{N}_\tau\,(t_o))$$

This equation can be formally implemented with a weighting window $w(t)$, given that $w(t)$ has certain characteristics.

Figure 1.6 illustrates two windows $w(t)$: a *box* window and a Gaussian window. Both windows implement a neighborhood by concentrating the highest weight about the center. Weights well outside of the neighborhood are zero

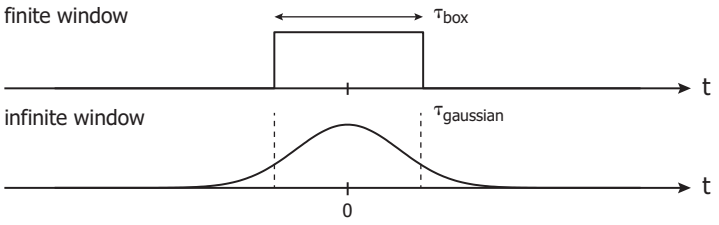

Fig. 1.6 Two comparable smoothing windows: a *box* and a Gaussian. The windows share a common scale, however the *box* is finite while the Gaussian is infinite.

or effectively so. The *box* window is finite while the Gaussian is infinite in extent, however the two windows have a similar width. In this respect it is only one's choice to use one or the other.

Given a choice of window, a direct implementation of a local average about point t_o is

$$y(t_o) = Z^{-1} \int_{\mathbb{R}} w(t - t_o)x(t)dt \tag{1.4.2}$$

where Z is the area under the window curve.

Figures 1.7-1.8 illustrate the development of series $y[n]$, a discrete-time analogue to the continuous case. In the first figure, we first identify the time point n_c about which a local average will be taken; I have a *cursor* point to this target point (thus the c subscript in n_c). The *box* window is aligned to the cursor by translating the window zero point to coincide with it. Once aligned, the input and window series are multiplied to form a product series. The product series is then summed and the value of the sum y is positioned under the cursor location n_c, that is we have point $y[n_c]$.

In the second figure, the cursor is moved to the right. The window is aligned to the cursor, the product series is formed, and the value of the product series sum is marked as the output value located at the current n_c. The dotted circle in the figure illustrates the input point (hollow) and output point (filled) based on this smoothing.

A complete smooth curve $y[n]$ repeats this local-averaging calculation for each point n in the series $x[n]$.

Figure 1.9 is the same as Fig. 1.8 except that the *box* window is replaced with the Gaussian window. Although the replacement window has a different shape, the smoothing process is the same. Since the Gaussian window has infinite extent (at least theoretically) all points along $x[n]$ contribute to $y[n_c]$, however the majority of the weight is centered about the peak of the window.

The free parameter in all of this is the window width. That width is ours to choose, and I choose it to be order-h in scale. This is the key point that connects this smoothing overview to the discussion of managing detail.

Causality: Investment decisions can only ever be made with currently available information. In this light, there is a major flaw with the *box* and Gaus-

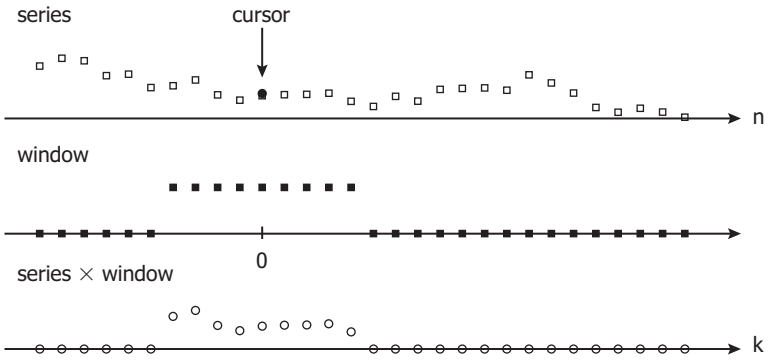

Fig. 1.7 A direct smoothing method. A cursor is used to identify the point to be evaluated along the series. A window is aligned to the cursor and the product between input series and aligned window is taken, see bottom. The sum of the product series is the value of the output, and this value is associated with the current cursor location (black dot).

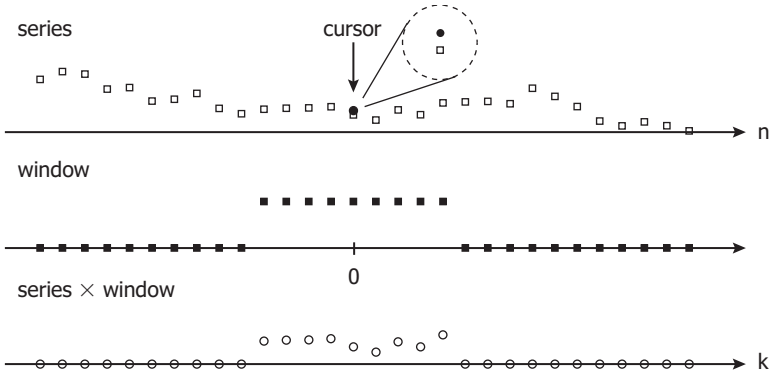

Fig. 1.8 The cursor has advanced. The smoothing method is applied to produce the next output point. The inset in the top figure shows the original point, square, and the output point, circle.

sian window shapes of the preceding: at each point t_o values $x(t)$ *after* t_o are admitted to the local average. The output value $y(t_o)$ consequently contains future information at time t_o. One way to enforce causality is to take a partial local average,

$$y(t_o) = Z^{-1} \int_{-\infty}^{t_o} w(t - t_o)x(t)dt$$

where the upper limit of the integral is fixed to t_o.

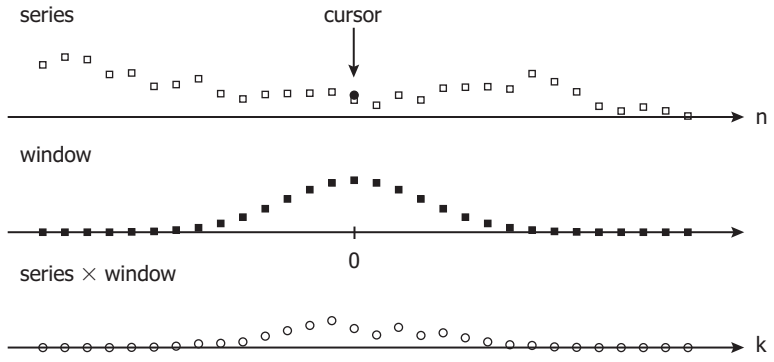

Fig. 1.9 At the same cursor location as shown in Fig. 1.8 but with a Gaussian window instead of the *box*. While the Gaussian window is technically infinite, points at and around the cursor position are weighted more heavily than others.

Given the mechanics of the smoothing process, an alternative way to enforce causality is to select window shapes that are single sided. That is, the window itself is zero for $t > 0$ (where t is on the window axis). Figure 1.10 shows two examples of causal windows. The first is a *box* again, but where the *box* has been left shifted so that finite values end at the window origin. The second is an *ema* window, which in continuous time is a decaying exponential. In both cases window values for $t > 0$ are zero, and in both cases the window widths have some equivalence. With a causal window we can return to using Eq. (1.4.2) for our calculations.

And what happened to the Gaussian? To enforce causality we would have to cut the window such that it is a one-sided Gaussian with a vanishing value for $t > 0$. Or, we could delay the peak of the curve, as I did with the *box*, so that the body remains causal, and then truncate the window where the weights are low. Both of these are candidates.

However, the question of where our Gaussian window went speaks to a more fundamental point, one that will take several chapters to arrive at with rigor. In short, the windows I consider in this monograph have dual representations: the window shapes such as those already introduced, and finite-difference equation (FDE) counterparts. I prefer to study windows that have simple FDE expressions, and while an expression for a causal truncated Gaussian exists it is inelegant. For this reason I will part with Gaussian windows.

The *box*- and exponential-moving averages, on the other hand, have the simplest structure in difference-equation form among the set of possibilities, which is one reason the *box* and *ema* windows are omnipresent in financial time-series analysis.

Figure 1.11 revisits the direct smoothing method illustrated before, here with causal windows. The method is the same: a cursor points to the position along the original series that will be smoothed; a causal window is aligned so that the zero position sits under the cursor; and the sum of the product series

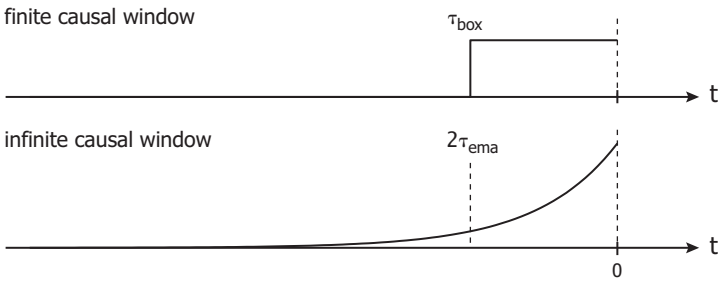

Fig. 1.10 Two common causal windows. The *box* is finite in length, the *ema* semi-infinite. The *box* has a clear length, τ_{box}. Infinite-length windows with finite area are characterized by their moments M_k. The full width of an *ema* is 2τ.

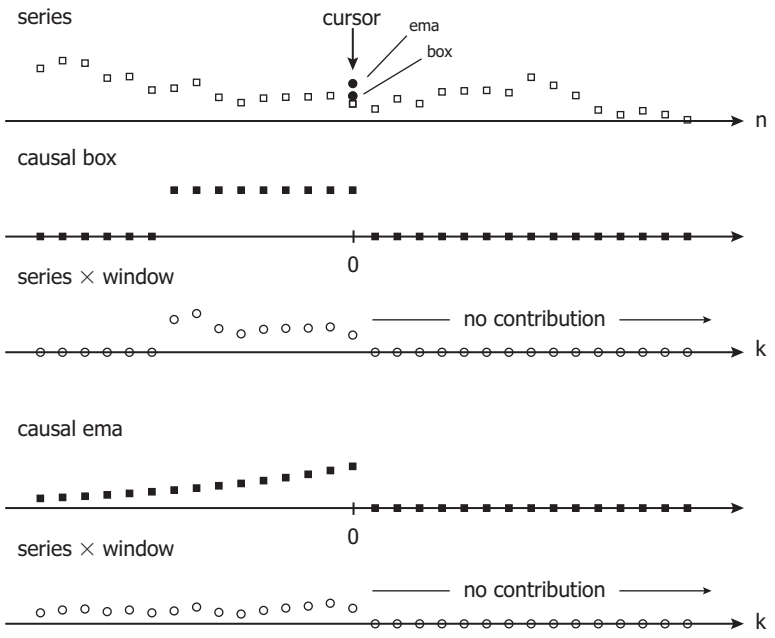

Fig. 1.11 Causal windows: points in the original series that are ahead of the cursor must not contribute to the smoothed value. Illustrated are a causal (shifted) *box* and an *ema*.

is the smoothed value $y[n_c]$. Now, however, since the windows are causal, no information $x[n > n_c]$ (analogous to $x(t > t_o)$) contributes to the smoothed value at position n_c.

As with the non-causal *box* and Gaussian windows, the causal *box* and *ema* windows can be stretched to have the same width. Electing one shape or the other changes details in the smoothed result but not the overall structure.

1.5 A Variety of Smoothing Windows

Smoothing windows considered in this monograph come in one of three forms: those with finite area, those with zero area, and those with infinite area.

Figure 1.12 shows three windows with finite area: a *box*, an *ema* and an integrated *macd*. The illustrated windows have similar widths, they have unit area, and their first moments are matched ($t = \tau$). The smooth curves created by these windows from the same input data set will share the same overall structure; differences in structural detail are a consequence of the shape differences between windows, but that is all.

The bottom window in Fig. 1.12 shows a delayed delta function located at the first-moment position of the others, $\delta(t - \tau)$. A delta function that is delayed imparts an *ideal delay*. The area of a delta function is one, so in fact the only difference between this window and the *box*, *ema* and integrated *macd* windows is the shape. Indeed the local averaging property of the delta function is simply to select a single point.

Wireframe is a method I use where I replace a smoothing window with a delta function at the location of the original window's first moment. Wireframe separates the concern of window *shape* from window *location*: the delta function preserves the location while deferring the smoothing to a later step.

Let's make a wireframe window for returns. The return on price px at time t between past times $t - \tau$ and $t - 3\tau$ is

$$r(t; t - \tau, t - 3\tau) = \log\left(\mathrm{px}(t - \tau)\right) - \log\left(\mathrm{px}(t - 3\tau)\right) \tag{1.5.1}$$

This equation can be represented as a window applied to the log of the price series $\mathrm{px}(t)$; the window equation is

$$w_r(t; \tau) \equiv \delta(t + \tau) - \delta(t + 3\tau)$$

$w_r(t; \tau)$ is illustrated in the top figure of Fig. 1.13. Clearly if we apply the smoothing technique from §1.4 to log prices we will recover Eq. (1.5.1).

The next step is to add smoothing to our returns. As it is, no local averaging is in effect from the input series $\mathrm{px}(t)$. The *box* differencer[3] in the

[3] I call a window a *differencer* when the window takes a balanced difference between an upper and lower arm.

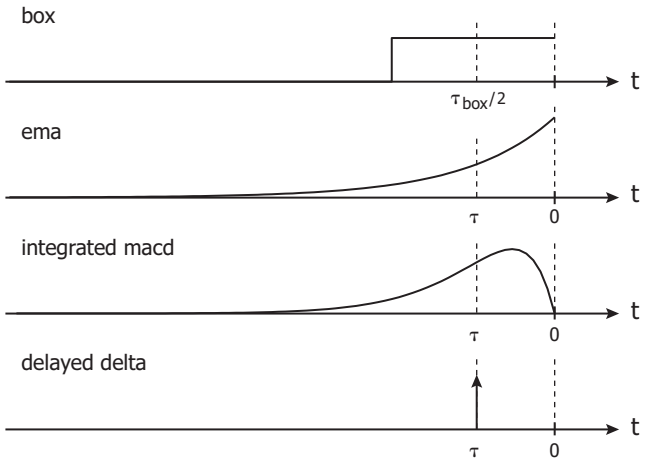

Fig. 1.12 The location of four unit-area windows. Location is defined as the first moment M_1. Here four different window shapes share the same location.

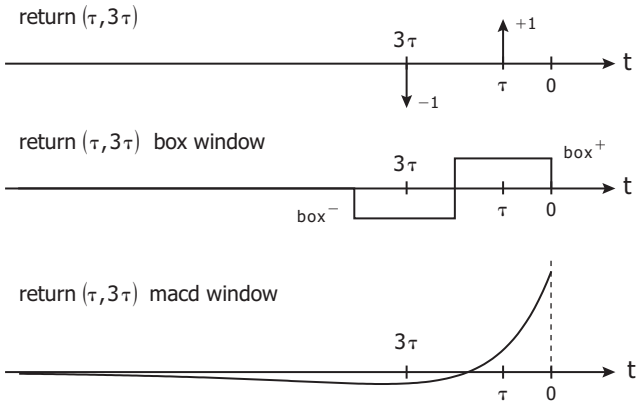

Fig. 1.13 Differencing windows have two location parameters, one for the positive arm and one for the negative arm, here τ and 3τ. In a financial context, the difference of two equal-area windows makes a window that represents a financial return to within a scale factor. Here, three differencers share the common locations yet have different shapes. The returns are the therefore same, the regularization results are different.

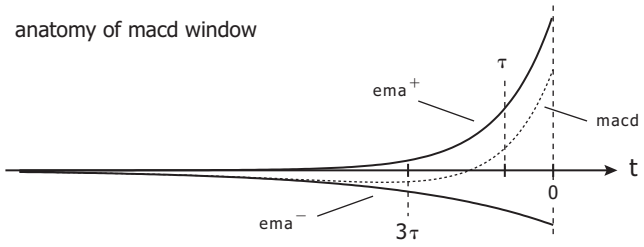

Fig. 1.14 An *macd* window is the result of the difference between two *ema* windows with different characteristic lengths $\tau_{\pm}$. Since the area of the two *ema* windows, when properly defined, is unity, the areas under the positive and negative sections of the *macd* window are equal, albeit less than one. The difference of the two *ema*s here must be scaled to what I call the *financial gauge* in order to properly represent a regularized return.

middle figure of Fig. 1.13 substitutes two equal-size, unit-area boxes for the delta functions. When used to smooth, each *box* averages (log) prices within the *box* bounds. The sum of the upper and lower local averages makes a smoothed return.

The arms of the *box* differencer don't overlap, which makes it simple, but we can also admit differencers where there is overlap. An *ema* differencer is a common example in finance, and in finance, this window shape is called an *macd* window.[4] The details of an *ema* differencer are illustrated in Fig. 1.14. Both arms are exponential decay curves with unit area. Each *ema* is scaled so that its first moment coincides with a target location, in this case τ and 3τ. The upper and lower arms overlap for all time yet the difference curve is non-vanishing as long as the *ema* scalings are different.

The *macd* window smooths a returns curve because each arm creates a local average of prices. Unlike the *box* differencer, the lower *ema* arm "starts" at $t = 0$; neither arm has a pure delay involved.

Taken together, the three differencing windows in Fig. 1.13 are similar: they take balanced differences of an input series; the upper and lower arms have unit area; and the difference locations are the same. The window shapes are the only distinguishing feature. Applying any of these windows to a log price series will generate a return series. One additional property of these three windows is that their total area is zero. This zero-area property makes differencer windows distinct from finite- and infinite-area windows.

Lastly there is the infinite-area window. The only one that we will consider is the unit step function $u(t)$, illustrated in Fig. 1.15. The unit step function is causal, has a fixed amplitude of one, and due to its semi-finite extent, has infinite area. Moreover, the unit step function has no defined location and

[4] The term *macd* is an industry idiom, it stands for *m*oving *a*verage *c*onvergence *d*ivergence, which actually refers to a trading strategy. I will adopt this idiom and refer to an *ema* differencer window as an *macd* window. Other differencer types will be explicitly stated.

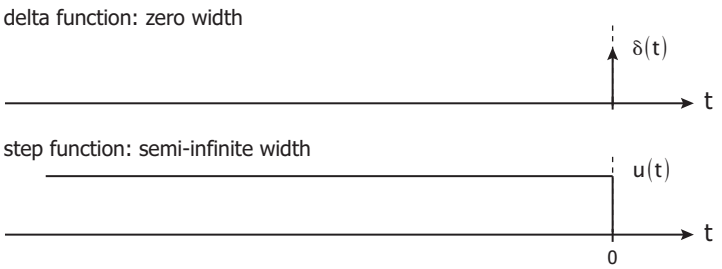

Fig. 1.15 Top, the continuous-time delta function $\delta(t)$, which has zero width. Bottom, the continuous-time step function $u(t)$, which has infinite width.

has infinite width. The unit step function will be our *integrator*, in contrast to the differencers just discussed. When a unit step function is used when "smoothing", a local average about the cursor location is not taken, rather the entire history up to the cursor contributes equally:

$$y(t_o) = \int_{-\infty}^{t_o} x(t)dt$$

Here I have dropped the normalization coefficient Z because one does not exist.

The windows I have introduced here are representative and in no way exhaustive. Yet with smoothing windows that are part of the financial vernacular we can classify them into groups of smoothers, smoothing differencers and integrators. With smoothers and differencers, we can wireframe our window and then replace the smoothing shape with another smoothing shape having the same scale. In this way, shape is its own degree of freedom. Certain shapes have superior properties when compared to others, but the overall structure of the output curves will possess strong similarities.

1.6 Smoothing Examples

Now let's apply the smoothing windows that were introduced in the last section. Figure 1.16 shows the result of applying the four finite-area smoothing functions of Fig. 1.12 to the price series first illustrated in Fig. 1.4. The top plot in Fig. 1.16 applies the delta function delayed by τ to the price series. The result is that the output is identical to the input but shifted by time τ. The delta function $\delta(t-\tau)$ has zero width and therefore imparts no local average; however, it imparts a shift in output location. The next three output curves are generated using a *box*, *ema* and integrated *macd* windows. As you can see, the resulting curves are overall the same: the curves are delayed by

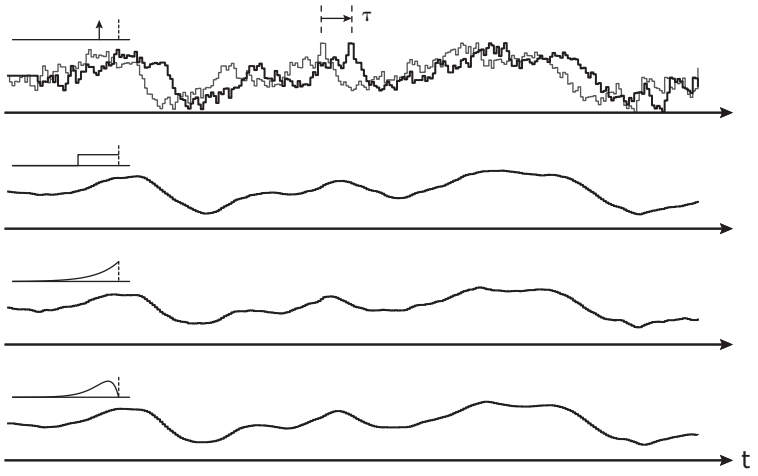

Fig. 1.16 Application of four windows with the same location to a price series: a delayed delta function, a *box*, an *ema*, and an integrated *macd*. The delta function reproduces the input series but imparts a delay. The remaining windows impart the same delay but apply local averages. The lower three curves are smooth and share the same general form. The details of the curves differ, which is due to the differing window shapes.

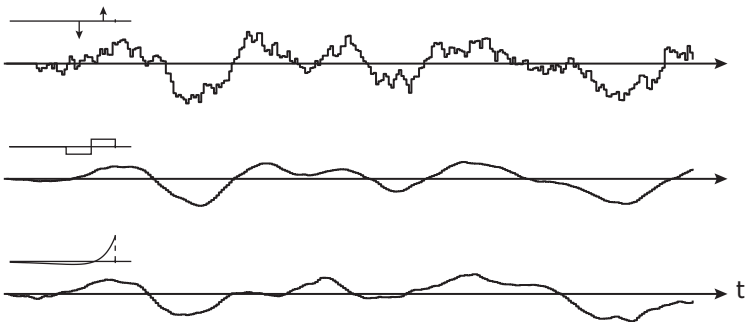

Fig. 1.17 Application of three differencer windows to a price series: a wireframe differencer, and *box*- and *ema*-differencers. The windows have the same upper- and lower-arm locations. The three curves share the same long-scale fluctuation. The wireframe differencer shows short-scale roughness; this is because the two delta functions have no width, no neighborhood is formed. The *box* and *ema* differencers are smoothed.

the same amount as for the delta function yet the roughness is significantly reduced. The remaining differences between these output curves are the result of the differences in window shape. The integrated *macd* output is in fact the smoothest of all the curves, the high-frequency ripple is suppressed more than the others. The reasons for this will become apparent when we consider window spectra.

Figure 1.17 shows the result of applying the three differencers of Fig. 1.13 to a price series. The wireframe differencer shows the structure that results from $(t-\tau, t-3\tau)$ differences along the original price series. Since there is no local averaging there is a lot of ripple on top of the slower moving underlying. The *box*- and *ema*-differencers significantly smooth out the ripple while preserving the long range structure of the wireframe. Still, there are non-trivial differences between the *box*- and *ema*-differencers smoothed curves, which is because the lower arm of the *box* differencer has an ideal delay that is not present in the *ema* differencer. In this sense the *box*- and *ema*-differencers, while sharing the same wireframe, have a fundamental difference.

1.7 Smoothing, Autocorrelation and Downsampling

The investment horizon, managing detail and temporal scale are the themes of this chapter. It is time to return to them so that we can complete the story of smoothing. Recall the sample-and-hold technique, Fig. 1.4. An input series is sampled at a rate consistent with one's investment horizon. This is a straightforward technique but inferior to smoothing. Smoothing, among other attributes, provides an update for each new event yet the smoothed curve fluctuates on an h-consistent scale.

A consequence of smoothing is the introduction of autocorrelation. Autocorrelation is the dependence of an update on the preceding values. The application of a smoothing window *imparts* such a dependence even if none existed before. However we know the scale of the dependence because we know the length scale of the windows we apply. All that is left, then, is to *downsample* the smoothed curve to recover an output series with nearly the same autocorrelation as the input. Figure 1.18 illustrates the process: we start with an input curve that is rough, we smooth it with a window whose scale is h-consistent, and then we downsample the smoothed series using a sampling interval that is related to the window scale. If the original data set had little autocorrelation, such as a returns series, the downsampled data set is then suitable for use in regression or statistical analyses.

So really we have two approaches to managing detail: sample-and-hold versus smooth-and-sample. The benefits of smooth-and-sample are curve stability against choice of initial condition, and curve integrity in that each surviving point is a local average of points leading up to that moment.

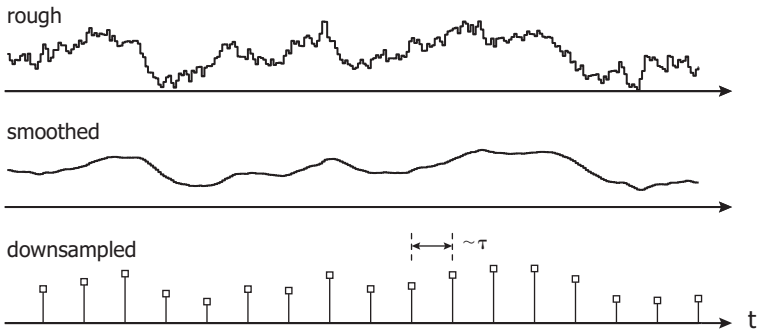

Fig. 1.18 Smoothing autocorrelates an input series. In order to recover the level of autocorrelation of the input, the output must be downsampled. The downsample period is on the order of the temporal scale of the window that was applied. A downsampled series is used for calibration.

The smooth-and-sample method is used for calibration. All calibration techniques rely on i.i.d. inputs so that the estimators are unbiased. For online use we don't downsample, we want each smoothed update as it arrives. The smoothing effect in fact provides some measure of interpolation between adjacent h-consistent intervals. There are technical reasons why smoothing is not interpolation, specifically, the market is not band limited and therefore there is no Nyquist rate that provides perfect reconstruction. Yet, in regular market conditions smoothing and interpolation are similar.

1.8 Filters: A Separation of Concerns

The application of a window to a data set, as in the local average of Eq. (1.4.2) on page 7, is a *transformation* of the data. It is a transformation and not a function because any subset of the input points, including the entire set, can be included in the evaluation of a single output point. The window functions previously illustrated capture, or describe, the transformation between input and output. The common and more general term is that a *filter* captures the input / output relationship, as illustrated in Fig. 1.19. When describing a filter we can talk about its window function, or as adopted in the next chapter, its impulse response, and we can also talk about its transfer function, its spectrum, and its pole-zero diagram. These terms all relate to ways of describing a filter, yet the purpose of a filter is always to transform an input into an output.

A filter that is not adaptive, which is to say that the filter does not change in time, is completely separable from its input. Alternative expressions to Eq. (1.4.2) are

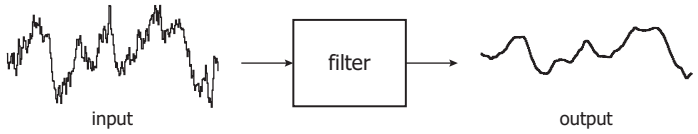

Fig. 1.19 A *filter* embodies a transformation. An input signal is transformed by a filter into an output signal. A wide variety of transformation types, or filter classes, exist.

$$\underbrace{y(t)}_{\text{output}} = \underbrace{h(t)}_{\substack{\text{impulse}\\ \text{response}}} * \underbrace{x(t)}_{\text{input}} \qquad \text{or} \qquad \underbrace{Y(\omega)}_{\text{output}} = \underbrace{H(\omega)}_{\substack{\text{transfer}\\ \text{function}}} \underbrace{X(\omega)}_{\text{input}}$$

On the left, the filter is represented by its impulse response $h(t)$, and $*$ is the convolution operator. This equation is cast in the time domain. On the right, the filter is represented by its transfer function $H(\omega)$. $H(\omega)$ and $X(\omega)$ are multiplied to produce output $Y(\omega)$. This equation is cast in the frequency domain. The importance of these expressions is that a filter and its input are independent.

In the context of finance, the study of prices, or spreads or rates or volatilities, requires stochastic calculus. Stochastic calculus is a rich field and the reader can find any number of general and product-specific references.

This monograph, in contrast, is entirely dedicated to the study of filters: the shape of $h(t)$ or the spectrum of $H(\omega)$. In order to engage in this study, inputs are classified as either finite or infinite in length.

Finite-length inputs are the typical case, such as the daily closing prices of Google. The finite nature of the data set means that its energy is finite, so there are no convergence issues for the Fourier, Laplace or z transforms used in the following pages. Infinite-length inputs are used to model random series such as white noise. These series have infinite energy, so Fourier and related transforms do not converge. However, infinite-length series have finite power, so once the filtering system is recast to transform an input power density to an output power density, such as

$$\underbrace{R_Y(\tau)}_{\text{output}} = \underbrace{\kappa_h(\tau)}_{\text{S-ACF}} * \underbrace{R_X(\tau)}_{\text{input}} \qquad \text{or} \qquad \underbrace{S_Y(\omega)}_{\text{output}} = \underbrace{K_h(\omega)}_{\substack{\text{S-ACF}\\ \text{xform}}} \underbrace{S_X(\omega)}_{\text{input}}$$

we recover the entire framework of filter analysis and design. On the left, $R.(\tau)$ represents the autocorrelation function of an infinite series and $\kappa_h(\tau)$ is the system autocorrelation function (S-ACF), and on the right, $S.(\omega)$ and $K_h(\omega)$ represent the power spectral density and S-ACF Fourier transform. Once again, there is separation of concerns between a filter and its input.

Provided that the input is appropriately classified, the focus of this monograph is on the analysis and design of continuous- and discrete-time filters.

1.9 Filter Types and the Types of Filters In This Book

Filters come in many classes. Figure 1.20 illustrates one way to classify the set of all possible filters. At the top, a filter may be causal or non-causal. The distinction between these two filters was illustrated by Fig. 1.8 on page 8 and Fig. 1.11 on page 10. Only causal filters are admissible in a financial context.

Within the class of causal filters, a filter may vary in time or not. For instance, if the window functions of the previous sections changed shape in time, that is, from position to position when working out a local-average sequence, then they would be time varying. A time-invariant filter is one where the window shape is fixed for all time. A time varying filter may be adaptive, but adaptivity is beyond the scope of this work, and moreover, the language of adaptive filters builds on that of time-invariant filters. Therefore, with the exception of the *vwap* filter, page 111 and onward, I concentrate on time-invariant filters.

Lastly, there is the issue of update complexity. What is the computation cost for each update? The procedure to compute the local average for one point as illustrated in Fig. 1.11 on page 10 requires N_w multiplications and additions, where N_w is the window length. The complexity is $O(N_w)$. For a long window, the inner product of input samples and window weights can take a significant fraction of the time available to respond to the market[5].

[5] In particular, the US equities markets require short response times when executing low-latency trading strategies.

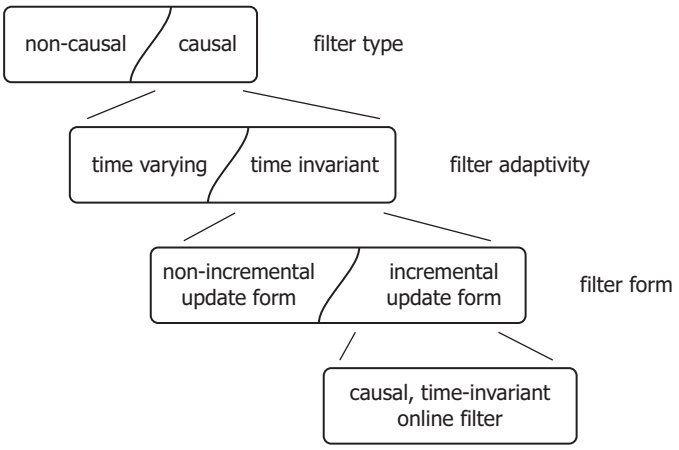

Fig. 1.20 One possible filter classification tree. The filters in this monograph are *causal*, *online*, and for the most part, *time invariant*. An online filter is expressible as a finite-difference equation and thus has an $O(1)$ update complexity. A time-invariant filter means that the window shape does not change over time.

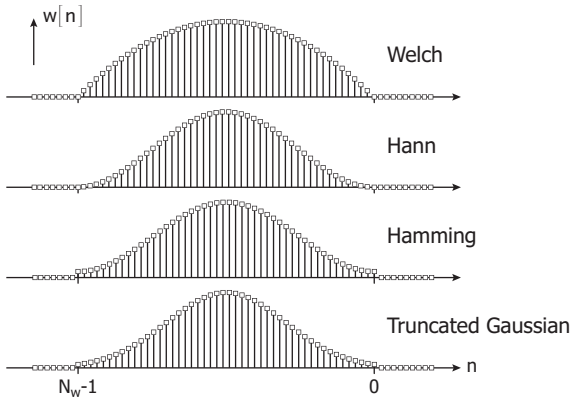

Fig. 1.21 Examples of filters that do not have an incremental-update form, they are only expressible as vectors of N_w weights. Each update at the input requires $O(N_w)$ input-window point pairs to be recomputed. In this book I seek filters with $O(1)$ update complexity.

Consequently, I seek $O(1)$ update complexity, a computational cost that is independent of the window length. The *box* and *ema* filters are traditional examples found in the financial markets. Their update equations are

$$y_{\text{ema}}[n] = p\, y_{\text{ema}}[n-1] + (1-p)x[n], \quad |p| < 1$$
$$y_{\text{box}}[n] = y_{\text{box}}[n-1] + N_w^{-1}\left(x[n] - x_{\text{box}}[n-N_w]\right)$$

These two finite-difference equations require trivial compute time to update, yet they can represent long window lengths. An essential aspect of this monograph is to introduce the reader to several filter classes, including polynomial-*ema* and Bessel filters, that also require only $O(1)$ in cost and yet significantly outperform the *ema* and *box* filters.

To contrast filters that offer $O(1)$ update complexity, several common window functions are illustrated in Fig. 1.21 [21]. These filters and others like them are nonzero only on a range $n \in [0, N_w - 1]$, which is to say, they are finite in extent. The *box* window and cascades of *box* windows are the only finite-extent windows that are expressible as finite-difference equations. The Welch window is quadratic within the range, the Hamming is trigonometric, and the Gaussian is a shifted, truncated form:

$$w_{\text{welch}}[n] \equiv 1 - \left(\frac{n - (N_w-1)/2}{(N_w-1)/2}\right)^2$$
$$w_{\text{hamming}}[n] \equiv \alpha - (1-\alpha)\cos\left(\frac{2\pi n}{N_w - 1}\right)$$

$$w_{\text{gauss}}[n] \equiv \exp\left(-\frac{1}{2}\left(\frac{n-(N_w-1)/2}{\sigma(N_w-1)/2}\right)^2\right)$$

None of these windows can be expressed as a finite-difference equation, the best update complexity is $O(N_w)$.

I tend to classify filters that have $O(1)$ update complexity as *online* filters because they are suitable for low-latency trading strategies. Filters with update complexities of $O(N_w)$, $O(N_w \log N_w)$ or worse, I classify as *offline*. However, graphical processing unit (GPU) technology can truly bring $O(N_w)$ complexity to online quality. That would admit finite-length filters such as Welch and Hamming to the pool of consideration. Finite-length filters have the advantages of guaranteed stability and the possibilities of linear phase and adaptiveness. Yet, such filters are no panacea, all aspects of good filter design need to be addressed and balanced.

With the filter classification of Fig. 1.20, the remainder of this monograph is dedicated to the analysis and design of causal, time invariant, online filters.

1.10 Filter Selection Criteria

Returning to Section 1.6, *Smoothing Examples*, how is one to choose a filter? A principal thesis of this work is that practical criteria exist to choose one filter type over another for a given application. The qualities one looks for when selecting a filter are:

- Temporal shape of the window function,
- Temporal overlap of two or more stretched window functions,
- Spectral shape of the window function, with low attenuation of desired frequencies and high attenuation of undesired frequencies, and
- Low distortion across frequencies with low attenuation.

Figures 1.22-1.24 illustrate the first three criteria.

The three window functions shown in Fig. 1.22 highlight various temporal features. The *box* window is finite in length, constant in amplitude across the window, and has leading and trailing discontinuities. The sharp edges can cause abrupt changes at the filter output as samples enter and exit. Moreover, an outlying sample effects the output for the full time that it falls within the window. The Chebyshev window has no sharp edges but significantly rings in the tails. This ringing is not suitable for financial applications. The integrated *macd* window neither rings nor has sharp edges. Its initial value, $w(0)$, is zero (or nearly zero in discrete time) and the leading edge rises linearly. The varying amplitude across the window means that an outlier's effect is roughly limited to the region around the window peak.

The integrated *macd* window concentrates its weight about its first moment. This is a desirable quality. Polynomial *ema*s and Bessel filters have even

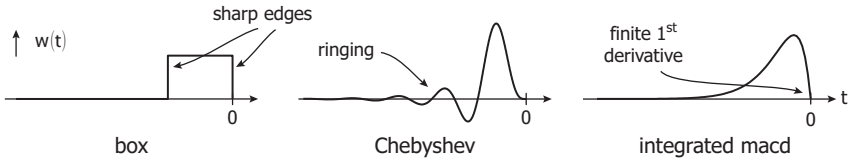

Fig. 1.22 Examples of various smoothing window characteristics. Sharp edges and significant ringing in the tail are two non-ideal characteristics of a window function. The integrated *macd* window is smooth and does not ring. Its leading edge rises linearly.

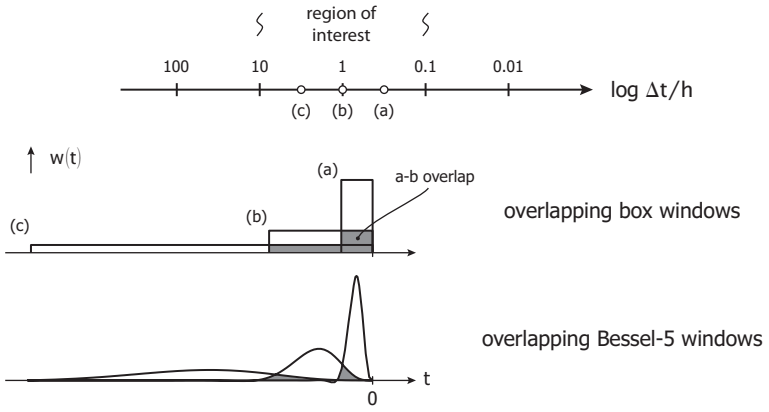

Fig. 1.23 Top: Sampling of the region of interest, two temporal scales per decade. Below: *box* and Bessel (5^{th} order) window functions stretched so that their first moments coincide with the sampled scales. The overlap between windows governs the correlation of the outputs of the filters (for a zero autocorrelated input). The overlap among *box* windows is higher than the corresponding Bessel windows.

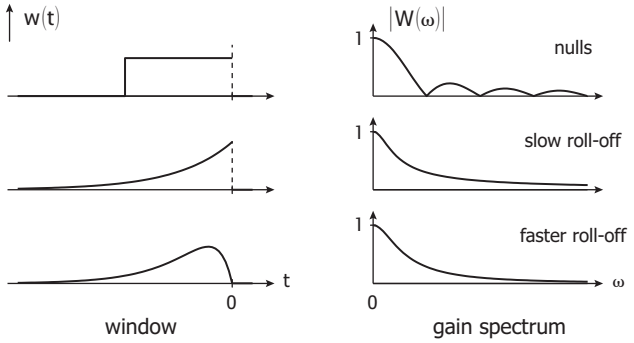

Fig. 1.24 Right column: the Fourier transform magnitude, or gain spectrum, of *box*, *ema* and integrated *macd* windows. The *box* gain spectrum has periodic nulls and a long tail. The *ema* spectrum has slow roll-off whereas the integrated *macd* spectrum exhibits a faster roll-off.

higher concentration of weight about their first moments. Such concentration improves locality and reduces skew.

Figure 1.23 highlights the significance of overlap of stretched window functions. In finance, rarely is a filter used in isolation. Instead, various temporal scales in the range of the investment horizon can play a role in prediction. The top of the figure focuses on the region of interest (after Fig. 1.3 on page 3), where the time interval between an investment decision and its target horizon is normalized to one. One might want to sample other intervals, in the figure I have indicated two samples per decade. Each sample point is associated with a stretched version of the filter window. The lower part of the figure shows *box* and Bessel windows stretched in a $(1/3, 1, 10/3)$ series. In particular, notice the overlap among the stretched windows.

As I show in Section **??** starting on page **??**, *Filter Bank Correlation*, for an input that is not autocorrelated, the correlation of multiple output signals (three outputs in this case) equals the overlap integral of the stretched window functions. Highly correlated outputs have collinearity tendencies which lead to poor statistical results. We want to choose window functions with low overlap across stretched versions so that the region of interest can be sampled effectively.

Windows that have high amplitude concentration about their first moments lead to low overlap integrals. In the extreme, a delta function is perfectly concentrated at its first moment since it has no width; however, it provides no smoothing. Clearly a balance of concerns must be achieved.

The spectral qualities of a filter are as important as the temporal. Spectral analysis of a window function yields two types of spectra: the gain spectrum and the group-delay spectrum. The gain spectrum governs the frequency-dependent attenuation and the group-delay spectrum governs the temporal distortion. A window and its spectra are linked by the Fourier transform, they are not independent. Accordingly, spectral analysis is only a complementary view on the behavior of a window function, but one with great value.

Figure 1.24 illustrates *box*, *ema* and integrated *macd* window functions and their associated gain spectra. The gain spectrum of the *box* filter has nulls at integral multiples of $1/T$, where the *box* width is T. One might question the financial significance of selectively zeroing out certain frequency components. The gain spectrum of the *ema* filter is shown below, this spectrum is smooth. However, the contrast between the low- and high-attenuation regions is, relatively speaking, poor. This is because the leading edge of the *ema* is abrupt, which in turn admits fast moving inputs. The gain spectrum of the integrated *macd* is also smooth, and has better capture of low frequency components at the input and higher suppression of high frequency components.

Low filter distortion is a subtle effect that is of importance only in certain circumstances. For instance, a price series that is passed through a smoothing differencer, such as those in Figure 1.17 on page 15, and then integrated back up to produce a smoothed (and delayed) price, can exhibit price-level artifacts caused by filter distortion. This is a case where low distortion has significance.

It turns out that for filters with infinite extent, low distortion leads to window functions that have high amplitude concentration about their first moments, which is already a sought after quality.

Finite-length filters can have zero distortion, the *box* filter is one example. However, absence of distortion for finite-length filters does not lead to window concentration about its first moment. The zero distortion feature of some finite windows is appealing; yet the temporal shape, window overlap, spectral quality and update complexity are attributes that have to be balanced.