Numerical Differentiation on Digital Data

Herman Jaramillo

November 25, 2016

1 Introduction

This article explains a set of techniques to compute differentiations on digital data. While differentiation touches almost any scientific field, we are more interested in signal processing and wave equation processes.

A few examples of the applications in wave equation processes appear both in modeling and imaging of seismic data using either integration methods (such as for example Kirchhoff methods) or differentiation methods such as finite differences for one or two-wave propagation algorithms. The application on each case suggest different algorithms. For example for integration methods, the frequency/wavenumber domain is a better choice, while for finite difference methods the design of a small set of coefficients to convolve with the data seems optimal.

Since we are interested on differentiating digital data, we see the differentiation as a digital filter. We are interested on finding trade-offs between accuracy and computational cost. We evaluate errors due to discretization and aliasing by testing the algorithms in multiple frequency windows on signals with cusps where differentiation fails even for exact analytic methods.

We explore time/space versus frequency/wavenumber representations. We find differentiation filters by using digital signal theory. That is, by knowing the exact analog filter in the frequency/wavenumber domain and by using Fourier transformation to find the ideal filters in time/space domain. The ideal filters are digital filters with infinite length, so they are not good for practical implementations in a computer. We explore the using of windowing methods (mainly Hanning window) to obtain finite filter representations which are adequate. Additional techniques such as least square spectral matching, Lagrange polynomial interpolation and Taylor series are explored to obtain filter coefficients. The errors of all filters are evaluated in a range of low to high frequencies.

We start with a first derivative, then extend the analysis to high order derivatives (mainly second order) and fractional derivatives (mainly half derivative). The half derivative is important because it represents the rho filter for Kirchhoff migration/modeling of two-dimensional data, which has a 45 degree phase-shift correction together with a $\sqrt{\omega}$ amplitude correction. In the case of fractional derivatives, we do
not perform a full analysis as done in the case of first and second order derivatives. It seems that for fractional derivatives the frequency/wave number domains seem more appropriate where the implementation is straightforward and accurate for modeling and migration of seismic data using integration methods.

Finally we show that the analysis for one dimension is ready to apply to multi-dimensional differentiation and illustrate this with an example of a Laplacian operator in a three-dimensional real space.

We observe from this study that the last word on differentiation of digital data is not yet said and there is plenty of room for research on the topic.

Numerical Differentiation Algorithms

2 Basic Facts

We assume that the forward/inverse temporal Fourier transforms applied to a real function \( f(t) \) are given by

\[
F(\omega) = \int_{-\infty}^{\infty} f(t)e^{i\omega t} dt
\]

\[
f(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\omega)e^{-i\omega t} d\omega.
\]  

we say that the analog differentiator is a \(-i\omega\), which reverses the phase of the input for each frequency \( \omega \) by \( \pi/2 \) radians (observe that \(-i = \exp(-\pi/2))\. Figure 1 shows the ideal impulse response for a differentiator and its Fourier transform drawn in FFTLAB.
Figure 1: Ideal impulse response for differentiator and its Fourier transform.

This figure was hand–mouse drawn known that the frequency impulse response was $-i\omega$.

The ideal impulse response of a sampled data set is a flat box defined by

$$B(\omega_N) = \begin{cases} 1 & -\omega_N \leq \omega \leq \omega_N \\ 0 & \text{otherwise} \end{cases}$$

where $\omega_N = \pi/\Delta t$ is the circular Nyquist frequency.

To find the analytic time domain impulse response of the ideal differentiator, let us first compute inverse transform of the box.

$$b(t) = \frac{\Delta t}{2\pi} \int_{-\omega_N}^{\omega_N} e^{-i\omega t} d\omega = \frac{\Delta t}{2\pi} \frac{e^{-i\omega t}}{-it} \bigg|_{-\omega_N}^{\omega_N} = \frac{\Delta t}{\pi} \sin(\omega_N t)$$ (2.2)

Now, the Fourier transform of $\delta'(t)$ is given by $-i\omega$. So the inverse Fourier transform of $-i\omega B(\omega_N)$ is given by

$$D = \delta'(t) * \frac{\Delta t}{\pi t} \frac{\sin(\omega_N t)}{t} = \frac{\Delta t}{\pi} \int_{-\infty}^{\infty} \delta'(\tau) \frac{\sin(\omega_N (t-\tau))}{t-\tau} d\tau = -\frac{\Delta t}{\pi} \frac{d}{d\tau} \frac{\sin(\omega_N (t-\tau))}{t-\tau} \bigg|_{\tau=0}$$

That is

$$D = \frac{\omega_N \Delta t}{\pi} \left( \frac{\cos \omega_N t}{t} - \frac{\sin \omega_N t}{t^2} \right)$$
is the ideal analog differential operator in the frequency domain. Note that by using the L’Hôpital’s rule we verify that D=0, for t=0.

If we sample our time domain data with a sampling rate of Δt so that $t = n \Delta t$ and being $\omega_N = \pi/\Delta t$ then

$$D = \begin{cases} \frac{\cos(n\pi)}{n\Delta t} = \frac{(-1)^n}{n\Delta t} & \text{if } n \neq 0 \\ 0 & \text{if } n = 0. \end{cases}$$ \hspace{1cm} (2.3)

So, taking a derivative of a function is equivalent to the convolution with this operator. Here $n$ runs from $-N$ to $N$ for a $2N+1$ points filter. The ideal filter can not be achieved because we have limited memory on computers. We are forced to truncate the filter. How to truncate this operator is part of the purpose of this appendix.

Sometimes I use the name rho filter to refer to the $-(i\omega)^p$ for some $p$ number (typically $p = 1/2$, 1 for 2D and 3D Green’s functions respectively).

Figure (2) illustrates the time domain 3D derivative filter for a few points. Note that this filter is not centered around zero (it is not a zero phase filter). This is because we can not handle in C, negative indexes. Therefore we create it as causal and then we do a phase shift (by multiplying it with $\exp(-i n \omega \Delta t/2)$) to centered it around zero. Here $n$ is the number of coefficients of the filter.

Figure 2: A set of filter coefficients for the first order derivative filter on the time domain.

A few issues with this filter are

• The filter is not causal. However the finite difference literature does not have a problem with this. The filters applied in practice have only a few coefficients
and any finite filter could be made causal by design. This also should apply to
the integration filter, however different from the integration filter this filter has
the coefficients going to zero as $1/t$ in the time domain, while the ideal bilateral
integration does not have any decay in its time coefficients

- The filter amplifies high frequencies linearly. This is bad news since high frequency
  noise exists in all field data.

3 Analog to Digital Conversion.

3.1 The relation between analog and digital

The knowledge of a wealth of analog filters in analytical form could be of great use to
design digital filters that will produce some desired results. A few highlights which we
would like to preserve when converting filters from analog to digital (or vice-versa) are:

- Bandwidth (frequency breadth)
- Amplitude and Phase Spectra.
- Phase behavior. In particular the minimum phase property. This is:
  - Stability.
  - Causality.

In the filter design literature there are several techniques to relate analog to digital
filters. Bose, [2] describe the following techniques to convert between analog and digital
filters:

- Use the bilinear transformation
- Impulse Invariant Method
- Numerical Integration Method
- Matched Z-transforms.

Bose (in his Table 3.6) presents advantages and disadvantages of each of these meth-
ods in terms of the highlights itemized above and in addition to difficulties on imple-
mentation and aliasing contamination. These methods are suited for Infinite Impulse
Response (IIR) filters, which are efficient for implementation due to short recursive
formulas. Integration filters are naturally recursive since they are accumulators that
carry the history of previous data. Differentiators, (and this is the topic of appendix[1])
on the other hand, are not recursive and need a different design.

We see that the bilinear transformation is well suited for low pass filters. Since
integration is basically a low pass filter, we will choose the bilinear transformation as
3.1 The relation between analog and digital

our tool to convert from analog to digital. According to some authors (see for example, Oppenheim et. al., [6]), the bilinear transformation is motivated by the trapezoidal rule of integration.

The transformation from analog to digital is done by mapping the Laplace parameter \( s = i\omega \), on the continuum to the \( Z \)-transform parameter \( z = e^{s\Delta t} \) on the discrete, with sampling rate \( \Delta t \).

The digital representation of filters is through the \( Z \)-transform. Polynomial representations are easy to implement, but many filters come in fractions of polynomials. Filters written in terms of fractions of polynomials are evaluated recursively. These are the type of filters used for integration (accumulators). The simplest rational function of polynomials is a bilinear representation (bilinear here means that the numerator and denominator are linear functions of the \( z \) argument. The bilinear function is not even linear).

The idea is to find a rational approximation for the parameter \( s \) as a function of \( z \). Let us start by finding first a rational function of \( z \) as a function of \( s \) (Claerbout, [3]).

\[
\begin{align*}
    z &= e^{s\Delta t} \\
    &= \frac{e^{s\Delta t/2}}{e^{-s\Delta t/2}} \\
    &\approx \frac{1 + s\Delta t/2}{1 - s\Delta t/2}.
\end{align*}
\]

Note that this approximation we can see the assumption of small values of \( s \). That is \(|s\Delta t| \ll \pi\). This is why the bilinear transformation is fitted for low pass filters. We will show in Appendix 1 that the bilinear transformation does not work well with differentiation filters, which are high pass filters.

We find, from equation 3.4 that

\[
z(1 - s\Delta t/2) \approx 1 + s\Delta t/2.
\]

That is,

\[
1 - z \approx -\frac{s\Delta t}{2}(1 + z)
\]

and

\[
s \approx -\frac{2}{\Delta t} \frac{1 - z}{1 + z}.
\]

\[\text{In fact, I will go the opposite way. I derive the bilinear approximation and then show that it represents the difference equation for the trapezoidal rule of numerical integration.}\]
This approximation of $s$ as the ratio of two linear $z$ transforms is called the bi-linear transformation. Let us assume $z = a + ib$, with $a, b \in \mathbb{R}$. Then,

\[
s \approx -\frac{2 \,(1 - a) - ib}{\Delta t \,(1 + a) + ib}
= -\frac{2 \,[(1 - a) - ib][1 + a - ib]}{\Delta t \,(1 + a)^2 + b^2}
= -\frac{2}{\Delta t \,[(1 + a)^2 + b^2]} \left[(1 - a^2 - b^2 - ib(1 + a + 1 - a))\right]
= -\frac{2}{\Delta t \,[(1 + a)^2 + b^2]} \left[(1 - a^2 - b^2 - 2ib)\right]
= -\frac{2(1 - a^2 - b^2)}{\Delta t \,[(1 + a)^2 + b^2]} + i \frac{4b}{\Delta t \,[(1 + a)^2 + b^2]}
\]

The amplitude is approximated by

\[
|s| \approx \frac{2 \sqrt{(1 - a^2 - b^2)^2 + 4b^2}}{\Delta t[(1 + a)^2 + b^2]} = \frac{2\sqrt{1 - 2a^2 + 2b^2 + 2a^2b^2 + a^4 + b^4}}{\Delta t[(1 + a)^2 + b^2]}
\]

and the phase is approximated by

\[
\phi(s) \approx \arctan \left(\frac{2b}{a^2 + b^2 - 1}\right). \quad (3.8)
\]

If $z = \exp(i\omega \Delta t)$, then $a = \cos \omega \Delta t$ and $b = \sin \omega \Delta t$, and $1 - a^2 - b^2 = 0$, then

\[
s \approx \frac{4b}{\Delta t[(1 + a)^2 + b^2]}
\]

and with

\[
(1 + a)^2 + b^2 = 2 \cos \omega \Delta t, \quad b = \sin \omega \Delta t
\]

then

\[
s \approx \frac{2}{\Delta t} i \tan(\omega \Delta t/2). \quad (3.9)
\]

from which its amplitude and phase responses are given by

\[
|Y(f)| = \frac{2}{\Delta t} |\tan(\pi f \Delta t)| \quad \phi(f) = \text{sgn} \omega \pi/2.
\]

A geometrical way to understand the amplitude and phase behavior is by thinking of $z$ values as two-dimensional vectors. Figure 3 shows a point $z$ and the vectors $z + 1$ (red) and $z - 1$ (green). The numerator is represented by $z - 1$ as the green line, the denominator $z + 1$ is the red line. The amplitude of the filter is dominated by
3.1 The relation between analog and digital

Figure 3: This figure aids in the geometrical illustration of the phase and amplitude spectra from associating $z$ to a vector. The zero is represented by a small circle and the pole by a “×” sign. Here we observe the ratio $(z - 1)/(z + 1)$. The amplitude is the ratio of the norms of the green over the red vector. The phase is the difference between the green ($\phi_z$) and the red ($\phi_p$) phases.
3.1 The relation between analog and digital

the pole. That is, as \( z \) gets close to the pole \((-1, 0)\) the amplitude grows without bounds. Of course for \( z \) close to the zero, the amplitude should be low, but we are not much interested on the amplitude spectrum at this moment. We will come to this later with an analytic approach. The phase is the difference between the phase of these two vectors. At any point \( z \) in the upper half plane, the phase is the difference

\[ \phi = \phi_z - \phi_p, \]

and at any point \( z \) in the lower half plane, the phase is the difference

\[ \phi = -\phi_z + \phi_p. \]

Since the upper half plane is associated with \( \omega > 0 \) and the lower half plane with \( \omega < 0 \) we can write compactly

\[ \phi = \text{sgn} \omega (\phi_z - \phi_p). \]

Along the real line inside the unit circle and coming from above (that is for \(-1 \leq \text{Re}(z) \leq 1, \text{Im}(z) > 0\)), \( \phi_z = \pi \) and \( \phi_p = 0 \), so \( \phi = -\pi \). Now if \( \text{Im}(z) < 0 \) then \( \phi = -\pi \).

This means that the segment \(-1 \leq z \leq 1\) is a branch cut. On the real line but on the right of the zero \( z = (1, 0) \), both angles \( \phi_z \) and \( \phi_p = 0 \), so \( \phi = 0 \). On the real line, to the left of the pole \( z = (-1, 0) \) both angles are \( \pi \) so also \( \phi = 0 \).

Let us now study the phase along the unit circumference. From elementary geometry, the angle at \( z \) is always \( \pi/2 \) and

\[ \phi_z = \phi_p + \pi/2, \]

so

\[ \phi_z - \phi_p = \pi/2. \]

That is, along the unit circumference the phase is \( \text{sgn} \omega \pi/2 \). Figure 4 illustrates this situation.

Digital filters can be easily explained based on their zero–pole plots. The zero–pole plot for the digital integration filter has a zero at -1 and a pole at 1, as shown in Figure 5.
3.1 The relation between analog and digital

Figure 4: Phase of bilinear transformation \((z - 1)/(z + 1)\) with \(z = \exp(i\omega \Delta t)\). Here \(1 = (1, 0)\). Clearly along the upper semi–circumference \(\phi = \phi_z - \phi_p = \pi/2\).

Figure 5: Low frequency enhance integrator filter. Here \(z = e^{i\omega \Delta t}\). The zero is represented by a small circle and the pole by a “×” sign.
3.1 The relation between analog and digital

Here we use phase plots to illustrate the filters responses. Phase plots come in many flavors. Wegert and Semmler\cite{8} show the power and art of phase plots to illustrate complicated complex functions. Poles, zeros, brunch cuts, essential singularities, etc. All can be easily identified with the help of phase plots. To understand the coloring system, I show in figures 6 and 7 the phase plots for the functions $z$ (left) and $1/z$ (right). In all cases the C function $\text{atan2}$ was used. We picked three color plots: red, green and blue and color bars displaying the phases between $-\pi$ and $\pi$.

Figure 6: Phase plot for the function $f(z) = z$. Angle increases in the counter clockwise direction.

Figure 7: Phase plot for the function $f(z) = 1/z$. Angle decreases in the counter clockwise direction.

Close to zero, the identity map $f(z) = z$ has all colors between green (low) to blue (high positive) to red as we wind around zero in a counter clockwise direction. Close to zero the inverse function $f(z) = 1/z$ has a pole and all the colors between green to red (high negative) to blue. Which is what we expect since for $z = A \exp i \phi$

$$\arg(z) = \phi \quad \arg(1/z) = -\phi.$$  

A lower resolution plot where we can see the color divisions shows the mapping of constant phase lines which helps even more to understand the phase flow. Figures 8 and 9 show the phase plots for $f(z) = z$ and $f(z) = 1/z$, this time in a lower resolution scale that let us see the phase transitions, which in these two functions are radians but as we will obverse later (see for example Figure ??) could be more paths.

\cite{2}http://www.arxiv.org/pdf/1007.2295
3.1 The relation between analog and digital

Figure 8: Phase plot for the function $f(z) = z$. Angle increases in the counter clockwise direction. Lines of constant phase are seen as radial lines.

Note that the unit circumference is included since it will be the reference contour for the Z transform. Figure 10 shows the phase function 3.8.

Figure 9: Phase plot for the function $f(z) = z$. Angle decreases in the counter clockwise direction. Lines of constant phase are seen as radial lines.

Figure 10: Phase plot of the bilinear transformation.
Phase plots are good because they highlight important features of their corresponding complex mappings. Referring to the plot in Figure 10 we observe:

- The zero at \( z = 1 \) and the pole at \( z = -1 \). Note that the sequence of phases as we rotate counter-clockwise around the zero is reversed as compared with the same oriented rotation around the pole. There is not a clear definition of the phase at a zero and a pole. All phases take place around them. What is the phase of a zero vector? Had we chosen the electrical engineers convention for \( z^{-1} \) instead of \( z \), those zeroes and poles inside the unit circle would be mapped outside to their reciprocals, and those outside to the inside.

- The line between the the pole \((-1, 0)\) and the zero \((1, 0)\) is a brunch cut of the bilinear function.

- From equation 3.8 we observe that the phase is even (symmetric) with respect to the real axis \((a)\) and odd with respect to its imaginary axis \((b)\) as shown in the picture.

- The phase is zero in the real axis. At a small \( b > 0 \), the phase is close to \( \pi \) inside the unit circle (where \( a^2 + b^2 - 1 < 0 \)), and for small \( b < 0 \), the phase is close to \(-\pi\) inside the unit circle. Outside of the unit circle the values of the phase are close to 0 above (for \( b > 0 \)) and zero below (for \( b < 0 \)). This is the case of the regular \( \arctan(y/x) \) evaluation.

- At the unit circle \((z = \exp(i\omega\Delta t))\), the phase is constant \( \pi/2 \) for \( b > 0 \) and \(-\pi/2\) for \( b < 0 \). This means that the real component of \( s \) is zero and that the imaginary axis in the \( s \) plane is mapped into the unit circle in the \( z \) plane.

- If we take a point on the phase plot for which the phase is \( 0 < \phi(s) < \pi/2 \) that point is outside of the unit circle, and since, in the \( s \) plane a point with phase \( 0 < \phi(s) < \pi/2 \) is located in the right of \( s \) plane, we conclude that the outside of the unit circle on the \( z \) plane maps to the right side of the \( s \) plane. Likewise, by using a similar argument, the inside of the unit circle maps into the left side of the real plane. This is very important since it indicates that the bilinear transformation preserves the minimum phase filter condition (all zeros and poles outside of the unit circle in the \( z \) plane and on the right of the \( s \) plane).

In addition to the properties of the bilinear transformation listed above, this method of filter design is immune to aliasing. However, we should be aware of the fact that the frequencies are mapped non-linearly. From equation 3.9 we find that

\[
s = i\omega_a \approx \frac{2}{\Delta t} i \tan(\omega \Delta t/2).
\] (3.10)
where \( \omega_a \) is the frequency corresponding to the analogue filter. We find the relation (and its inverse) between the analog and digital frequencies:

\[
\begin{align*}
\omega_a &= \frac{2}{\Delta t} \tan \left( \frac{\omega \Delta t}{2} \right) \\
\omega &= \frac{2}{\Delta t} \arctan \left( \frac{\omega_a \Delta t}{2} \right)
\end{align*}
\]

An important observation is that the variation of analog frequencies between \(-\infty < \omega_a < +\infty\) will map the discrete range of frequencies \(-\pi/2 < \omega < \pi/2\), with Nyquist extrema. This frequency compression (or stretching if looked in the other direction) is known as frequency warping.

The design of a digital filter with transfer function \( Y_d(\omega_d) \) from an analog filter with transfer function \( Y_a(\omega_a) \) is achieved with the substitution

\[
Y_d(\omega) = Y_a(s) \bigg|_{s=-\frac{2}{\Delta t} \frac{1+z}{1-z}}
\]

with \( z = \exp(i\omega \Delta t) \).

Returning to the integration theory, we could observe two basic types of numerical integrations. The analog integration and the digital integration. While physical filters are analog by nature, the word “analog” here means that the computer implementation is a direct discretization of the continuous Laplace operator \( 1/i\omega \) as shown in the next section. The Laplace transform signature \( s = i\omega \) in the continuous domain is mapped to the Z–transform signature \( z = \exp (i\omega \Delta t) \) in the discrete domain with the sampling rate \( \Delta t \). In the following sections we show the specifics of integration under each category.

We start by explaining why the analog to digital conversion through the bilinear transformation should not work as well as it does for integration filters. This is expected since the bilinear transformation has validity only for low frequencies.

Recall from equation 3.7 for small \( \omega \):

\[
-i\omega = -s \approx \frac{2}{\Delta t} \frac{1-z}{1+z} = -\frac{2}{\Delta t} i\tan(\omega \Delta t/2)
\]

(3.11)

The problem with the bilinear transformation is that it has a zero and a pole in the unit circle (see Figure 10). A stable approximation for the low frequency approximation \( \omega \ll 1 \), implies \( 1+z \approx 2 \). With this, the analog to digital transfer function conversion for the differentiation operator is

\[
Y_a(s) = -s \quad \Rightarrow \quad Y_d(z) = 2 \frac{1-z}{2\Delta t} = \frac{1-z}{\Delta t}
\]

The inverse Z transform of this operator is given by the time impulse response:

\[
y_d(t) = \frac{1}{\Delta t} [\delta(t) - \delta(t-\Delta t)]
\]
and convolution of a time series $f_i$ with this operator produces the backward differences

$$\frac{df}{dt}(t_i) \approx \frac{f_i - f_{i-1}}{\Delta t}$$

numerical scheme. The frequency response is given by

$$\frac{1 - z}{\Delta t} = \frac{1 - \cos \omega \Delta t - i \sin \omega \Delta t}{\Delta t}.$$  

That is, the real component is $(1/\Delta t)(1 - \cos \omega \Delta t)$ and the imaginary component is $-(1/\Delta t) \sin \omega \Delta t$. So the filter does not shift the phase by $-\pi/2$ for each of its frequency components.

Figure 11 shows the two point backward finite difference filter and its Fourier transform, from FFTLAB.

![Figure 11: Backward finite difference filter and its Fourier transform](image)

We observe that the imaginary component is a sinusoidal function which approaches the linear slope for small frequencies, but due to the non–zero real parts of its Fourier transform, the filter does not shift the data by the phase angle $-\pi/2$ as we would expect.

As we did for the integration filter, we can correct the phase by zeroing out the real part in the FFTLAB to find the filter and its response in Figure 12. The analog to digital transfer function conversion for this filter is

$$Y_a(s) = -s \quad \Rightarrow \quad Y_d(z) = \frac{z^{-1} - z}{2\Delta t} = \frac{e^{i\omega \Delta t} - e^{-i\omega \Delta t}}{2\Delta t} = \frac{1}{2\Delta t} i \sin \omega \Delta t$$
3.2 The “1/2” trick

Figure 12: Central finite difference filter and its Fourier transform

as shown in the figure. The time domain representation of the filter is given by

$$y_d(t) = \frac{\delta(t + \Delta t) - \delta(t - \Delta t)}{2\Delta t}.$$ 

and convolution of a time series \( f_i \) with this operator produces the central differences

$$\frac{df}{dt}(t_i) \approx \frac{f_{i+1} - f_{i-1}}{2\Delta t} \quad (3.12)$$

We see that to be able to acknowledge the \(-\pi/2\) phase shift, an antisymmetric filter in time domain should be designed.

3.2 The “1/2” trick

It is common to see finite differences referring to points that are midway between two
grid points. We see how by shifting the backward difference scheme by 1/2 we can get
to the central difference from the bilinear transform. If in the bilinear equation $3.11$ we
multiply numerator and denominator by $z^{-1/2}/2$ we find

$$-i\omega \approx \frac{1}{\Delta t} \frac{z^{-1/2} - z^{1/2}}{(z^{-1/2} + z^{1/2})/2}.$$ 

Now,

$$\frac{z^{-1/2} + z^{1/2}}{2} = \frac{e^{-i\omega \Delta t/2} + e^{i\omega \Delta t/2}}{2} = \cos \omega \Delta t/2 \approx 1,$$
and so we find the analog to digital transfer function conversion:

\[ Y_a(z) = -i\omega \quad \Rightarrow \quad Y_d(z) = \frac{z^{-1/2} - z^{1/2}}{\Delta t}, \]

which in time domain is

\[ y_d(t) = \frac{\delta(t + \Delta t/2) - \delta(t - \Delta t/2)}{\Delta t}. \]

If we resample our data so that the new sampling rate is half of the old sampling rate then this formula applies well. However what is commonly used is to replace in the previous formula \( \Delta t \) by \( 2\Delta t \) and obtain

\[ y_d(t) = \frac{\delta(t + \Delta t) - \delta(t - \Delta t)}{2\Delta t}, \]

which after convolved with a time series \( f_i \) will produce the central differences scheme 3.12.

### 3.3 The Windowing Method

A way to implement the differential filter is by using windows to taper the edges of the ideal filter. The filter can be designed directly in the frequency domain by tapering the function \(-i\omega\) so that it goes smoothly to zero at \( \omega = 0 \) and \( \omega = \omega_N = -\pi/\Delta t \). Another way is to use the time domain ideal (see Figure 2 and equation 2.3) and taper it with a window.

There is a large collection of tapering functions to smooth the edges: Barlett (a tent function), Parzen (piece wise cubic), Hann (known as Hanning), Hamming (these two are cosine tapering windows), Blackman, Lanczos, Kaiser, Gaussian, etc. A website about the topic can be found [here](http://www.cg.tuwien.ac.at/ theussl/DA). Figure (13) illustrates the Hanning, Hamming, Blackman, Welch and Gaussian windows. Note that the Gaussian window never dies to zero. This discontinuity will create ringiness on the dual domain. The same thing happens with the Hamming window. The parameters used here were \( \alpha = \gamma = 0.5 \) and \( \alpha_1 = 0.54 \). Both the Hamming and Hanning windows are cosine tapering functions with the same mathematical formulation but different \( \alpha \)'s as indicated above. For the examples shown here I will use Hanning windows.

Oppenheim et. al. [6] illustrate in their example 7.10 the use of the Kaiser window to taper a differentiator in the frequency domain. The implementation of the rho filter on the frequency domain amounts to a multiplication by \((-i\omega)^p\). In the case of 3D data, \( p = 1 \), for 2D or 2.5D data \( p = 0.5 \). Figure (14) illustrates the imaginary part of the 3D rho filter after and before tapering with the Hanning window. For reference the second order central difference (two points) is also shown. The time domain version of the ideal first order derivative, the Hanning window filtered, and the central difference is shown in Figure 15.

\[^3\text{http://www.cg.tuwien.ac.at/ theussl/DA}\]
3.3 The Windowing Method

Figure 13: Some common windows for tapering filters and data.

Figure 14: Amplitude spectrum of the ideal analytical and Hanning window tapered version of the derivative filter. The two point central difference ($\sin \omega \Delta t$) is shown also.
3.3 The Windowing Method

To test the derivative filter, we simulate signals with different frequency content. The signal is given by the sum of sine functions

\[ s_n(t) = \sum_{f=1}^{n} \frac{\cos \omega(t + t_0)}{\omega}, \]

with \( \omega = 2\pi(f + 1) \), and \( t_0 = 1.25s \). The reason for the choice of these functions are several:

- The sum of different functions let us control a spectral band for the simulated signals. For testing the accuracy of the different algorithms as a function of the spectral content of the signals.
- The factor \( \omega \) damps the high frequency so that they do not have high oscillation amplitudes.
- The function has a cusp at about 800 milliseconds (see Figure 16) for high frequencies. This will break any algorithm, but we want to see which algorithm is more robust around the cusp.
- It is easy to obtain an analytical derivative of \( s_n(t) \).

Figure 15: Time domain impulse responses of the ideal, Hanning filtered, and two point central differences filters.
3.3 The Windowing Method

Figure 16: The signal $s_{400}(t)$ from equation 3.13

Direct evaluation shows that

$$s'_n(t) = - \sum_{f=1}^{n} \sin \omega (t + t_0)$$

Figure (17) shows a comparison of the window filtering method against the basic two point central difference algorithm.
First Derivatives for Window Filtering Design

Figure 17: Test of the window derivative filter for four different frequency bandwidths. For the 1 Hz and 1 to 10 Hz signals (top), the central differences (blue) has good accuracy. The window filter computation (green) does poor at the edges. For the 1 to 100 Hz there is image aliasing and it is hard to estimate the accuracy of the methods. For the 1 to 400 Hz, even tough there is aliasing it is easy to see that the analytical (red) and the window filter computation (green) follow close to each other.

We can observe from the top–left frame that the central difference (blue) does a good job at approximating the analytic derivative (red). The windowing filtered derivative does a good job inside the signal but a poor job at the edges. The poor job at the edges will be observed in all cases of different frequency bandwidth. For the bandwidth of 1 to 10 Hz, the results are similar. That is, the two point central difference operator does a good job in general and the windowing method does poor at the edges. However for high frequencies (here 100 Hz and 400 Hz) the situation is quite different. While the figure presents imaging aliasing, if properly zoomed it can be observed that for the
bandwidth from 1 to 100 Hz, the windowing method approximates close the analytical solution while the central difference drifts from the analytic solution. Finally, in the 400 Hz case the central difference has a large error as compared with the differentiation computed through the windowing method.

A better appreciation for the accuracy of the methods is done by computing the error (difference of the approximation versus the analytical computation) plots. Figure 18 shows the situation for the 4 different bandwidths considered.

Figure 18: Error: Difference of the analytical and the computed derivatives for four frequency bandwidths. For the 1 Hz signal (top–left), for the signals from 1 to 10 Hz, from 1 to 100 Hz and from 1 to 400 Hz, the error of the central differences increases with frequency bandwidth. In all cases the error computed from the windowing method is small except at the edges of the signal domain.

It is clear from this figure that the central difference error increases with frequency bandwidth. This is expected from the spectra in Figure 14 where we see that the
central frequency bandwidth departs soon (about 0.3 radians, that is about 10 percent of Nyquist, or 50 Hz) from its ideal spectrum. The windowing method, on the other hand, by design works well up to 75 of the Nyquist frequency (375 Hz) where it starts diverging from the true solution. It is interesting to observe that for low frequencies the central difference approximation works better than the window filtering approximation.

The windowing method is good when it has to be applied only a few times. For example for Kirchhoff seismic migration, the filter should be applied once for each data trace. There could be millions of traces but still this is not a big number as compared with the amount of times that the filter has to be used on a finite difference propagation code (at least once per grid point, per propagation time and per source location. A Reverse Time Migration (RTM) algorithm has two propagation wavefields, one from the source and another from the receivers. A Full Waveform Inversion (FWI) code can have many iterations of RTM–like processes.). In the case of large 3D volume with a large amount of source locations, the number of times the finite difference filters are applied is prohibitively expensive. In this case, we should design a short filter which at the same time is cost effective. For that reason the least squares spectral matching method provides a better design. This is the subject of the next section.

### 3.4 The Least Squares Spectral Matching

We want to find a fixed number of filter coefficients $c_j$ such that for the $n^{th}$ derivative of a function $f(t)$ could be approximated by the equation

$$f^n(t) \approx \sum_j c_j f(t - t_j).$$

Take the Fourier transform in both sides and find

$$(-i\omega)^n F(\omega) \approx \sum_j c_j e^{i\omega t_j} F(\omega).$$

We want to find coefficients $c_j$ that fit this for any arbitrary function $F$, then we can assume that $F(\omega) \neq 0$ and find the solution of the system

$$(-i\omega)^n \approx \sum_j c_j e^{i\omega t_j}. \quad (3.14)$$

for $c_j$. That is, we want

find $c_j$ such that yield $\min_\omega \|(-i\omega)^n - \sum_j c_j e^{i\omega t_j}\|$.

By choosing the $L_2$ norm, this is a least squares problem. Let us write it as a matrix vector multiplication. For this we can define

$$a_{ij} = e^{i\omega_i t_j}, \quad b_i = (-i\omega_i)^n$$
and write equation \[3.16\] as

\[Ac \approx b.\]

The least squares problem is

\[
\text{minimize } \|Ac - b\|_2
\]

One way to solve this problem is by using a weighted least squares with a weight matrix

\[
W = \begin{pmatrix}
w_1 & 0 & \cdots & 0 \\
0 & w_2 & \ddots & \vdots \\
\vdots & \ddots & \ddots & 0 \\
0 & \cdots & 0 & w_n
\end{pmatrix}
\]

(3.15)

where \(w_i > 0\) for each \(i\). The corresponding normal equation is

\[A^*WAc = A^*Wb,\]

(3.16)

where \(A^*\) is the adjoint matrix of \(A\). Given that the number of \(c_i\) coefficients is small compared with the number of frequencies chosen \(\omega_i\) the matrix \(A\) has many rows and few columns and is overconstrained. Then we expect the matrix \(A^*WA\) is non–singular, otherwise we need to pose the problem as

\[(A^*WA + \lambda I)c = A^*Wb,\]

where \(A^*\) is the adjoint matrix of \(A\). Given that the number of for a positive value \(\lambda\) (or \(\lambda = 0\)). The explicit form of equation \[3.16\] is written as

\[
\sum_j \sum_l e^{i\omega_l(t_j - t_i)}w_l c_j = \sum_j e^{-i\omega_l t_i}w_j(-i\omega_j)^n.
\]

This is a complex variable equation. We can equate its real an imaginary parts. For example, if we match the real component For \(n\) even

\[
\sum_l \sum_j \cos[\omega_j(t_j - t_i)]w_j c_l = \sum_j w_j \cos(\omega_j t_i) (-1)^{n/2} \omega_j^n,
\]

and if \(n\) is odd

\[
\sum_l \sum_j \cos[\omega_j(t_j - t_i)]w_j c_l = -\sum_j w_j \sin(\omega_j t_i) (-1)^{(n+1)/2} \omega_j^n.
\]

(3.17)

In the examples that follow we use a weighting function \(w_j = \omega_j^{-4}\). The idea of this weight is to attenuate the high frequencies which could increase undesired noise. The computer design of the filter coefficients could be done on the fly if the sampling rate is
non–uniform or stored in a look up table if the sampling rate is uniform. For example, Figure [19] shows the finite difference impulse response for a first order differentiator computed using the least squares technique described here for the computation of 9 coefficients on a uniform grid. Here we matched the real components, that is we used equation 3.17.

Figure 19: Impulse response to a first order differentiator using the least square spectral matching technique.

It is clear that the phase of this filter is $-\pi/2$ (since it is anti–symmetric). We compute the amplitude spectrum of this filter, with that of the central difference method and the windowing method explained in the previous section.

Figure [20] shows the amplitude spectrum corresponding to this the least squares method, the windowing method and the central difference operator.
3.4 The Least Squares Spectral Matching

Figure 20: Amplitude spectrum of the impulse response to a first order differentiator using the least squares spectral matching technique, the windowing method, and the central differences. For the least square spectral matching filter we used the 9 coefficients shown in Figure 19.

We see that the least squares method has a better performance that the central difference in terms of spectral bandwidth accuracy. This is expected since we are using 9 coefficients instead of 3. By the same token the windowing filter has a better spectral bandwidth by design. It was designed to start departing from its ideal behavior at 75 percent of Nyquist frequency. We also observe that the least squares method does not have a smooth landing on Nyquist (the central difference neither) but the windowing method does, because the Hanning window enforces that smoothness. We could have added constraints on the least squares filter to force zero derivatives of any order, but that is outside of the scope of this document.

Figure 21 shows the analytical derivative of the test function versus the derivative found by convolving the function $f$ with the least square filter shown in Figure 19, in addition to the central differences approximation.
3.4 The Least Squares Spectral Matching

Error on First Derivatives for Three Different Methods.

Figure 21: Error while computing the derivative using the Hanning Window filter, the least squares method and the central differences approach.

We observe that the Hanning window filter is the most accurate method, followed by the least squares spectral matching filter. The central difference works best for low frequencies but it is unusable for high frequencies. Close to the cusp, none of the filters works fine and this is expected.

Here is an interesting research project to further extend the research on numerical differentiation.

- Test the limits of frequency performance in the windowing and least squares spectral matching methods in terms of the number of coefficients being used.
- Add random noise and test the sensitivity of the filters to this noise.
- In the design of the least squares spectral matching method:
– Try different weighting functions.
– Try a $\lambda > 0$ to get provide more stability. Is it necessary?
– What if instead of finding the least squares coefficients for the real part of
the matching spectrum, we use the imaginary part or the amplitude?
– Measure the increase of the number of coefficients versus the extension of the
frequency range. That is, as we increase the number of computer coefficients,
the upper limit of matching frequencies should move toward Nyquist. How
fast?
– Impose maximum flatness at the ends. That is the function an many of its
derivatives are zero. Is it necessary this at $\omega = 0$?

4 Other Methods

4.1 Lagrange Interpolation

Let us assume a function $f(t)$ sampled at some discrete set of points $t_i$. The function
could be computed at any point in between by the Lagrange approximation formula for
polynomial interpolation. Let us assume the samples as follows

$$(t_{-j}, f(t_{-j})), \cdots (t_{-1}, f(t_{-1})), (t_0, f(t_0)), (t_1, f(t_1)), (t_k, f(t_k)).$$

The negative indices are chosen because we know that the derivative is not causal, so
they are convenient in our derivations, but alternative derivations could be done with
all positive indices.

The Lagrange interpolation formula for a point $t \in [t_{-j}, t_k]$ is given by

$$f(t) = \sum_{i=-j}^{k} f(t_i)L_{N,i}(t) + \frac{1}{(N+1)!} \prod_{i=-j}^{k} (t - t_i) f^{N+1}[\xi(t)]. \quad (4.18)$$

where

$$L_{N,i}(t) = \prod_{i=-j \atop i \neq i}^{k} \frac{t - t_i}{t_i - t_l}$$

$N = j + k + 1$ and

This formula is exact at each node point $t_i$ since the error term (the second term) at
$t_i = t_l$ is zero. Otherwise the formula has an error that is proportional of the product
of the drifts between the output point and its nodes and the $N + 1$ derivative of $f$ at
some point $\xi(t) \in [t_{-j}, t_k]$. 
By differentiating \(4.18\) with respect to \(t\) we find
\[
f'(t) = \sum_{i=-j}^{k} f(t_i)L_{N,i}'(t) + \frac{1}{(N+1)!} \frac{d}{dt} \prod_{i=-j}^{K} (t - t_i) + \frac{1}{(N+1)!} f^{N+1}(\xi(t)) \frac{d}{dt} \prod_{i=-j}^{K} (t - t_i)
\]
(4.19)

If we pick a node \(t = t_i\), the second term becomes zero since at \(i = l\), \(t_l - t_i = 0\). The derivative in the last term expands on an \(N\) sum of factors but only the factor that does not have \(t - t_l\) is non–zero. That is the error is
\[
Error = \frac{1}{(N+1)!} f^{N+1}(\xi(t)) \prod_{i=-j}^{K} (t - t_i).
\]

Ignoring the error, we find
\[
f'(t_j) \approx \sum_{i=0}^{N} f(t_i) L_{N,i}'(t_j).
\]
(4.20)

So the filter coefficients are the derivatives of the Lagrange interpolation polynomials. That is
\[
L_{N,i}'(t_p) = \sum_{l=-j}^{k} \frac{1}{t_i - t_l} \prod_{q=-j}^{k} \frac{t - t_q}{t_i - t_q} \bigg|_{t=t_p}.
\]
(4.21)

This expression looks complicated because it is very compact. However we can evaluate it for special cases.

The first simplification is by imposing \(p = 0\). That is we choose a center scheme where our approximation stencil should match a central differences stencil. Then
\[
L_{N,i}'(t_0) = \sum_{l=-j}^{k} \frac{1}{t_i - t_l} \prod_{q=-j}^{k} \frac{t_0 - t_q}{t_i - t_q}
\]

This can be programmed to find a set of coefficients for non–uniformly sampled derivatives. However, we will simplify this equation further by assuming uniform sampling. For uniformly sampling \(t_i = t_0 + i\Delta t\), so equation \(4.21\) reduces to
\[
L_{N,i}'(t_0) = \frac{1}{\Delta t} \sum_{l=-j}^{k} \frac{1}{i - l} \prod_{q=-j}^{k} \frac{q}{q - i}.
\]
(4.22)
4.1 Lagrange Interpolation

and the coefficients can be programmed by using this equation. To further simplify we assume \( p = 0 \).

We show that, if \( j = k \) this equation is antisymmetric for \( i \). That is, if we change \( i \) for \(-i\) we should reverse the sign of the coefficient.

\[
L'_{N,-i}(t_0) = \frac{1}{\Delta t} \sum_{l=-k}^{k} \frac{1}{-i - l} \prod_{q=-k}^{k} \frac{q}{q + i}
\]

\[
= \frac{1}{\Delta t} \sum_{l=k}^{k} \frac{1}{-i + l} \prod_{q=k}^{k} \frac{-q}{q + i}
\]

\[
= \frac{1}{\Delta t} \sum_{l=k}^{k} \frac{1}{i - l} \prod_{q=k}^{k} \frac{q}{q - i}
\]

\[
= - \frac{1}{\Delta t} \sum_{l=k}^{k} \frac{1}{i - l} \prod_{q=k}^{k} \frac{q}{q - i}
\]

\[
= - \frac{1}{\Delta t} \sum_{l=k}^{k} \frac{1}{i - l} \prod_{q=k}^{k} \frac{q}{q - i}
\]

reversing indices

\[
= -L'_{N,i}(t_0).
\]

As a natural consequence

\[
L'_{N,0}(t_0) = 0
\]

Then we only have to compute \( k - 1 \) coefficients.

We illustrate the method with a few examples.

• Assume 3 points \( t_{-1} = t_0 - \Delta t, t_0 = t_0, t_1 = t_0 + \Delta t \).

\[
\Delta t c_i = L'_{2,i}(t_0) = \sum_{l=-1}^{1} \frac{1}{i - l} \prod_{q=-1}^{1} \frac{q}{q - i}
\]

So we find the coefficients \( c_{-1}, c_0, c_1 \).
For $i = -1$,

$$
\Delta t \ c_{-1} = \sum_{l>0}^{1} \frac{1}{-1-l} \prod_{q>0}^{2} \frac{q-1}{q} = \frac{-2 - 1}{2} = \frac{-1}{2}.
$$

For $i = 0$, from (4.23)

$$
\Delta t \ c_0 = 0.
$$

and for $i = 1$, from the antisymmetric property,

$$
\Delta t \ c_1 = -c_{-1} = \frac{1}{2}.
$$

So the coefficients for this scheme are

$$(c_{-1}, c_0, c_1) = \frac{1}{2\Delta t}(-1, 0, 1).$$

which correspond to the central difference scheme. Note that these coefficients are written backwards as compared with the coefficients being used this far. The reason is that the implementation for this filter is a zero lag cross-correlation instead of a convolution.

- A more interesting example is that of 5 coefficients. Let us work that example. We assume $t_{-2} = t_0 - 2\Delta t, t_{-1} = t_0 - \Delta t, t_1 = t_0 + \Delta t, t_2 = t_0 + 2\Delta t$ The coefficient equation (4.22) is given by

$$
L'_{4,j}(t_0) = \frac{1}{\Delta t} \sum_{l=\frac{-j}{i}}^{k} \frac{1}{i-l} \prod_{q=\frac{-j}{j}}^{k} \frac{q}{q-i}
$$

If $i = -2$, then

$$
\Delta t \ c_{-2} = \sum_{l=-2}^{2} \frac{1}{-2-l} \prod_{q<-2}^{2} \frac{q}{q+l}
$$

$$
= \left( \frac{1}{-1} \right) \left( \frac{1}{3} \right) \left( \frac{1}{2} \right) + \left( \frac{1}{-2} \right) \left( -1 \right) \left( 1 \right) \left( -1 \right) \left( 1 \right) \left( 1 \right) + \left( \frac{1}{-3} \right) \left( -1 \right) \left( -1 \right) \left( 1 \right) \left( 2 \right)
$$

$$
= \frac{-1}{6} + \frac{1}{12} + \frac{1}{6}
$$

$$
= \frac{1}{12}.
$$
If \( i = -1 \) then

\[
\Delta t c_{-1} = \sum_{l=-2 \atop l \neq -1}^{2} \frac{1}{-1-l} \prod_{q=-2 \atop q \neq -1 \atop q \neq l}^{2} \frac{q}{q+1} = \left( \frac{1}{-1} \right) \left( \frac{-2}{-1} \right) \left( \frac{1}{2} \right) \left( \frac{2}{3} \right) = -\frac{2}{3}
\]

Now, from antisymmetry \( c_0 = 0 \), \( c_1 = -c_{-1} = 2/3 \) and \( c_2 = -c_{-2} = -1/12 \) so the coefficients can be written as

\[
\frac{1}{12 \Delta t} (1, -8, 0, 8, -1).
\] (4.24)

Here is a C code that implements the computation of the coefficients \( c(i) \),

```c
/* generate coefficients */
for (j=k; j>0; j--)
{
    i = -j;
    sum=0.0;
    int l,p,r;
    for (p=0; p<n; p++)
    {
        l = p - k;
        float prod=1.0;
        if (l != i)
        {
            for (r=0; r<n; r++)
            {
                q = r-k;
                if ( (q != i) && (q != l) )
                    prod *= (float) q/(q-i);
            }
            prod *= 1.0/(i - l);
            sum += prod;
        }
    }
    printf("Coefficient c[%d]=%f \n",i,sum);
}
```

and the output of the program for 9 coefficients is

```
lena,haramil $ lagrange
Enter number of coefficients
9
Number of coefficients n=9, k=4
Coefficient c[-4]=0.003571
Coefficient c[-3]=-0.038095
Coefficient c[-2]=0.200000
Coefficient c[-1]=-0.800000
Coefficient c[0]=0.000000
Coefficient c[1]=0.800000
Coefficient c[2]=0.200000
Coefficient c[3]=-0.038095
Coefficient c[4]=0.003571
```

Note that we only need to compute 4 coefficients for the total of 9. The zero coefficient is always zero and the other four are the antisymmetric version of the first four coefficients.

Figure 22 shows a comparison of the normalized coefficients computed using the least squares spectral matching and the Lagrange polynomial interpolation method.
Figure 22: Comparison of the impulse responses from the least squares spectral matching versus the Lagrange polynomial interpolation, for the first derivative filter.

Figure 23 shows the approximations of the amplitude response for the least squares spectral matching, Lagrange interpolation, windowing, and central differences approach to differentiation.
4.1 Lagrange Interpolation

Figure 23: Comparison of the amplitude spectra from the least squares spectral matching, the Lagrange polynomial interpolation, the windowing method, and the central differences operator, for the first derivative filter.

We observe that the least squares spectral matching filter approaches the ideal amplitude response better than the Lagrange interpolator filter does for frequencies around half Nyquist. However as we show next the precision for low frequencies (on function 3.13) is higher for the Lagrange interpolator.
Error on First Derivatives for Four Different Methods.

Figure 24: Error of computing the derivative of function 3.13 by using the four different differentiation methods: Hanning window tapering, second order central differences, least squares spectral matching, and Lagrange polynomial interpolation.

Figure 24 shows the error with respect to the analytical function 3.13 made by the filters from least squares spectral matching, Lagrange interpolation, Hanning window and the central differences formula. In the order, central differences, least squares, Lagrange interpolator and Hanning window the accuracy is increasing in general (for medium to high frequencies and away from the edges). For the 400 Hz signal, the error is large for all the methods at the cusp zone. The Hanning window method has a relative small error away from the cusp zone.

Next we show how to design a first order differentiator using Taylor series to fit a desired order of accuracy.
4.2 The Taylor Series Method

4.2.1 Forward

For a function \( f(t) \) its Taylor series can be represented by the equation

\[
f(t + \Delta t) = f(t) + \Delta t f^{(1)}(t) + \frac{(\Delta t)^2}{2!} f^{(2)}(t) + \frac{(\Delta t)^3}{3!} f^{(3)}(t) + \cdots \quad (4.25)
\]

From here

\[
f^{(1)}(t) = \frac{f(t + \Delta t) - f(t)}{\Delta t} + O(\Delta t).
\]

This corresponds to the \((1, -1)\) filter and the order of accuracy is \(O(\Delta t)\).

4.2.2 Backward

\[
f(t - \Delta t) = f(t) - \Delta t f^{(1)}(t) + \cdots + (-1)^k \frac{(\Delta t)^k}{k!} f^{(k)}(t) + \cdots \quad (4.26)
\]

From here

\[
f^{(1)}(t) = \frac{f(t) - f(t - \Delta t)}{\Delta t} + O(\Delta t).
\]

This corresponds to the \((1, -1)\) filter and the order of accuracy is \(O(\Delta t)\). Note that both, forward and backward schemes have the same filter coefficients, but the backward is delayed by 1 sample. That is the forward is causal and the backward is anti–causal. That is, in the Z transform representation the forward operator is \(z - 1\), while the backward operator is \(1 - z^{-1}\).

4.2.3 Central

If we subtract the two Taylor series representations \(4.25\) and \(4.26\) and divide by \(2\Delta t\), we find, up to \(O(\Delta^{2m+1})\),

\[
\frac{f(t + \Delta t) - f(t - \Delta t)}{2\Delta t} \approx f^{(1)}(t) + \frac{(\Delta t)^2}{3!} f^{(3)} + \cdots + \frac{(\Delta t)^{2m}}{(2m + 1)!} f^{(2m+1)} \quad (4.27)
\]

from which

\[
f^{(1)}(t) = \frac{f(t + \Delta t) - f(t - \Delta t)}{2 \Delta t} + O(\Delta t)^2.
\]

This corresponds to the central difference scheme \(1/2, 0, -1/2\), which in Z transform notation is \(z^{-1}/2 - z/2\).
4.2 The Taylor Series Method

We see that the Taylor method provides an explicit accuracy indicator. We would think that the central differences is the best method of the three shown here. However it could happen that the central difference is unstable (\[7\]). Appendix ?? discusses the issues of numerical stability, accuracy, consistency, dispersion and diffusion in the context numerical solutions of partial differential equations.

How can we get to higher order accuracy using the Taylor series expansion? Let us replace $\Delta t$ by $2\Delta t \cdots k\Delta t$, in equation 4.27 and find up to $O(\Delta t)^{2m+1}$,

$$
\frac{f(t + 2\Delta t) - f(t - 2\Delta t)}{4\Delta t} \approx f^{(1)}(t) + \frac{(2\Delta t)^2}{3!} f^{(3)} + \cdots + \frac{(2\Delta t)^{2m}}{(2m + 1)!} f^{(2m+1)}
$$

\[4.28\]

$$
\frac{f(t + k\Delta t) - f(t - k\Delta t)}{2k\Delta t} \approx f^{(1)}(t) + \frac{(k\Delta t)^2}{3!} f^{(3)} + \cdots + \frac{(k\Delta t)^{2m}}{(2m + 1)!} f^{(2m+1)}.
$$

In this way we involve farther and farther points in the computation. Now we combine equations 4.27 and 4.28 for $k = 1, 2$. We multiply equation 4.27 by $2^2$ and from this, subtract equation 4.28 to find, up to $O(\Delta t)^4$,

$$
8 \frac{f(t + \Delta t) - f(t - \Delta t)}{4\Delta t} - \frac{f(t + 2\Delta t) - f(t - 2\Delta t)}{4\Delta t} \approx 4f^{(1)}(t) - f^{(1)}(t)
$$

and collecting terms

$$
f^{(1)}(t) = \frac{1}{12 \Delta t} (f(t - 2\Delta t) - 8f(t - \Delta t) + 8f(t + \Delta t) - f(t + 2\Delta t)) + O(\Delta t)^4,
$$

which has the normalized coefficients $(1/12, -8/12, 0, 8/12, -1/12)$. We observe that these coefficients match the coefficients (see 4.24) found using the Lagrange interpolation formula.

This is in general true since both Taylor approximations and Lagrange interpolation provide an exact computation of the derivative for a $k$ order polynomial.

In general we can set up the equation

$$
\sum_{i=-k}^{k} c_i f(t - i\Delta t) = \sum_{i=-k}^{k} c_i \sum_{j=0}^{\infty} \frac{(-i\Delta t)^j}{j!} f^{(j)}(t) = f^{(1)}(t) + O((\Delta t)^k). \tag{4.29}
$$

We already know some constraints. That is, $c_{-i} = -c_i$, $c_0 = 0$ (for the central difference approximations) so we only need to find $k$ coefficients. Let us reverse indices above and find

$$
\sum_{j=0}^{\infty} \frac{f^{(j)}(t)}{j!} \sum_{i=-k}^{k} c_i (-i\Delta t)^j = f^{(1)}(t) + O((\Delta t)^{2k}).
$$
4.2 The Taylor Series Method

We match coefficients based on the derivative order.

\[ j = 0 \Rightarrow c_0 + 2 \sum_{i=-1,i\neq 0}^k c_i = 0 \quad \text{balance of coefficients.} \]

That is \( c_0 = 0 \), which we already know from the anti–symmetry property of the coefficients.

\[ j = 1 \Rightarrow -\sum_{i=-k}^k c_i i \Delta t = 1. \quad \text{conservation of first momentum.} \]

Now, since \(-ic_{-i} = ic_i\) then

\[ j = 1 \Rightarrow 2 \sum_{i=1}^k c_i i \Delta t = -1. \quad \text{conservation of first momentum.} \]

Let us now assume \( j = 2p \), for \( p \geq 1 \), then

\[ j = 2p \Rightarrow \sum_{i=-k}^k c_i i^{2p} (\Delta t)^{2p} (2p)! = 0, \]

but since \( c_i \) are antisymmetric and \( i^{2p} \) is symmetric this sum is zero regardless any coefficients \( c_k \). That is, no new contribution arises for \( j \) even. Let us then assume \( j = 2p + 1 \) (odd). So, after grouping all coefficients for positive indices

\[ \sum_{i=1}^k c_i i^{2p+1} (\Delta t)^{2p+1} (2p+1)! = 0, \]

We can summarize this section with the following simultaneous equations

\[
\begin{align*}
c_0 &= 0 \\
\sum_{i=1}^k c_i i &= -\frac{1}{2\Delta t} \quad \text{First Momentum Conservation.} \\
p \geq 1 &\Rightarrow \sum_{i=1}^k c_i i^{2p+1} = 0 \quad \text{High order momentums vanish.}
\end{align*}
\]

for the coefficients \( c_0, \cdots c_k \) of a first order derivative filter which approximate a polynomial function perfectly up to the \( k \) power.

Let us apply this equation to two low order approximations
\[
\begin{align*}
\kappa &= 1. \\
\text{Only one equation is required} \\
c_1 &= -\frac{1}{2\Delta t} \\
c_{-1} &= -c_1
\end{align*}
\]
So obviously the first order central difference operator \([1/(2\Delta t)](-1,1)\) results.

\bullet \ \kappa = 2.

Here we need two equations
\[
\begin{align*}
c_1 + 2c_2 &= -\frac{1}{2\Delta t} \\
c_1 + 8c_2 &= 0.
\end{align*}
\]
Subtracting the top equation from the bottom equation we find
\[
6c_2 = \frac{1}{(2\Delta t)} \Rightarrow c_2 = \frac{1}{12\Delta t}
\]
So
\[
c_1 = -8c_2 = -\frac{8}{12\Delta t}.
\]
So the 5 coefficients are
\[
\frac{1}{\Delta t} \begin{pmatrix}
-1/12, & 2/3, & 0, & 2/3, & -1/12
\end{pmatrix}.
\]
This coincides with expression 4.24 derived from the Lagrange polynomial interpolation.

\section{Higher Order Derivatives}

\subsection{Basic Facts}

From Fourier analysis we know that a second derivative is computed in the frequency domain as a second power of the circular frequency. That is,
\[
\frac{d^2}{dt^2} \Rightarrow \mathcal{F}_\omega(-i\omega)(-i\omega) = -\omega^2.
\]
In the time domain the second derivative is the convolution of two first derivatives.
This is good because everything we know from first derivatives can be applied to by convolution to find second derivatives. In the \(Z\) transform domain convolution is seen
Second Order

as a product of polynomials. For example the forward difference \((1/\Delta t)(z - 1)\) for the first derivative can be use to find an \(n\)-th order derivative as follows

\[
\frac{d^n}{dt^n} \Rightarrow \left[ \frac{1}{\Delta t} (z - 1) \right]^n = \left( \frac{1}{\Delta t} \right)^n \sum_{i=0}^{n} (-1)^i \binom{n}{i} z^i.
\] (5.30)

This is an interesting equation. Note that for \(z = 1\) the left hand side of the equation is 0, and so the sum of all coefficients should be 0. This is an easy test to check for errors on stencils. The problem with repeated convolution (or exponentiation in the Z transform domain) is that it spreads quickly and this involve taking samples that reach far from the center. This reduces precision and increases computational cost. Another problem is that the representation 5.30 is skewed. It is all shifted to the right of the output sample. This problem can be fixed by using the central difference for the first derivative. That is

\[
\frac{d^n}{dt^n} \Rightarrow \left[ \frac{1}{\Delta t} \right]^n \sum_{i=0}^{n} (-1)^i \binom{n}{i} z^i,
\] (5.31)

which centers the coefficients around the output sample. Due to the spreading of the filter by exponentiation only in a few cases high order filters are designed from low order filter with recursive convolutions. We will treat the high order derivatives in a similar fashion as we did for the first derivative.

5.2 Second Order

If in equation 5.31 we take \(n = 2\) then we find

\[
\frac{d^2}{dt^2} \Rightarrow \frac{z^{-2}}{(\Delta t)^2} (1 - 2z^2 + z^4) = \frac{1}{(\Delta t)^2} (z^{-2} - 2 + z^2),
\] (5.32)

which corresponds to the central difference stencil \((1, 0, -2, 0, 1)\). In this particular case we want to consider a grid step of \(\Delta t/2\), to bring the coefficients closer to the zero position. Then by mapping \(z \rightarrow z^{1/2}\) in equation 5.32 we find

\[
\frac{d}{dt} \Rightarrow \frac{1}{(\Delta t)^2} (z^{-1} - 2 + z^1)
\]

which corresponds to the central difference stencil \((1, -2, 1)\) which is more concentrated.

From equation 2.2 and by convolution we find that the second derivative operator in the time domain can be written as

\[
D^2 = \delta''(t) \star \frac{\Delta t \sin \omega_N t}{\pi t} = \Delta t \int_{-\infty}^{\infty} \delta''(\tau) \frac{\sin \omega_N (t - \tau)}{(t - \tau)} d\tau = \Delta t \frac{d^2}{dt^2} \frac{\sin \omega_N (t - \tau)}{t - \tau} \bigg|_{\tau=0}.
\]
Now, from
\[
\frac{d^2 \sin \omega x}{dx^2} = \frac{d}{dx} \frac{x^2 (\omega \cos \omega x - \omega^2 x \sin \omega x - \omega \cos \omega x) - 2x (x \omega \cos \omega x - \sin \omega x)}{x^4} = \frac{-\omega^2 x^2 \sin \omega x - 2 \omega x \cos \omega x + 2 \sin \omega x}{x^3}
\]
with \(\omega = \omega_N\), \(\omega_N = \pi/\Delta t\), and \(x = t - \tau\) we find that
\[
\frac{\pi}{\Delta t} D^2 = -\frac{\omega^2_N (t - \tau)^2 \sin \omega_N (t - \tau) + 2(t - \tau) \omega_N \cos \omega_N (t - \tau) - 2 \sin \omega_N (t - \tau)}{(t - \tau)^3}
\]
\[
= -\frac{\omega^2_N t^2 \sin \omega_N t}{t^3} + 2 t \omega_N \cos \omega_N t - 2 \sin \omega_N t.
\]
Let us find the behavior of this equation when \(t \to 0\). By calling \(y = \omega_N t\) and we can write
\[
\frac{\pi}{\Delta t} D^2 = \frac{-\omega^3_N y^2 \sin y + 2 y \cos y - 2 \sin y}{y^3}
\]
\[
= \frac{-\omega^3_N y^3 + 2 y - (2y)y^2/2 - 2y + 2y^3/6 + O(y^4)}{y^3}
\]
\[
\to \frac{\omega^3_N}{3}, \quad \text{as} \quad t \to 0.
\]
and so
\[
\lim_{t \to 0} D^2 = \frac{\Delta t}{3\pi} \left(\frac{\pi}{\Delta t}\right)^3 = \frac{\pi^2}{3\Delta t^2}.
\]
On the other hand, since \(t = n\Delta t\), \(\omega_N t = \pi n\) and \(\sin \omega_N = 0\), then by using equation 5.34 we find,
\[
D^2 = \begin{cases} 
\frac{-2 \cos n\pi}{n^2 \Delta t^2} = \frac{2(-1)^{n+1}}{n^2 \Delta t^2} & \text{if } n \neq 0 \\
\frac{\pi^2}{3\Delta t^2} & \text{if } n = 0.
\end{cases}
\]
Compare this equation with 2.3. Figure 25 shows the ideal impulse response for a second order differentiator.
5.2 Second Order

We observe that it decays faster than its corresponding first order differentiator in Figure 2. This is expected since its decay is as $1/t^2$, while that of the first order differentiator is $1/t$, where $t$ is the time from the center position.

5.2.1 The Windowing Method

As in the first order differentiator we consider a Hanning window that is good up to 75% of the Nyquist frequency and start decaying from there.

Figure 26 shows the amplitude spectra of the analogue (analytical) signal (red), the Hanning window tapered (green) and the central difference $(1, -2, 1)$ central difference impulse responses.
Figure 26: Amplitude spectra of the ideal signal (red) versus the Hanning tapered window (green), and three point central difference (blue) impulse responses.

To test the second derivative filter, we simulate signals with different frequency content. The signal is given by the sum of sine functions

\[ s_n(t) = \sum_{f=1}^{n} \frac{\cos(\omega(t + t_0))}{\omega^2}. \]  

(5.36)

The reasons for this test function are the same as those explained just after the equation 3.13. Note that here we scaled by $1/\omega^2$ to keep the function stable for second derivatives on high frequency components. Figure 27 shows the sum of the first 400 frequencies.
Figure 27: The signal $s_{100}(t)$ from equation 5.36.

Figure 28 shows the computations of the analytic, central difference and Hanning window filtered versions of the second derivative defined in equation 5.36.
5.2 Second Order

First Derivatives for Window Filtering Design

Figure 28: Test of the window derivative filter for four different frequency bandwidths. For the 1 Hz and 1 to 10 Hz signals (top), the central differences (blue) has good accuracy. The window filter computation (green) does poor at the edges. For the 1 to 100 Hz there is image aliasing and it is hard to estimate the accuracy of the methods. For the 1 to 400 Hz, even tough there is aliasing it is easy to see that the analytical (red) and the window filter computation (green) follow close to each other.

Figure 29 shows the error from the Hanning window (green) method as well as from the central differences approach (red).
Consistent with previous tests, the central differences works best for low frequencies. The Hanning window does not do good at the edges. For high frequencies the Hanning window filter does a good job.

### 5.2.2 Least Squares Spectral Matching

We developed the least square spectral matching technique in section 3.4. We choose, for the second order derivative $n = 2$ and find the coefficients that minimize the least square error. Figure 30 shows a plot of these coefficients.
Figure 30: Filter for a least square spectral matching of a second derivative.

Figure 31 shows the amplitude spectra of the analytical, Hanning window tapered, central difference, and spectral matching filters. The spectral matching filter does a good job up to about half the Nyquist frequency. The central difference operator does a good job only for low frequencies (up to about 20 percent of Nyquist).
Figure 31: Amplitude spectra of the ideal signal (red) versus the Hanning tapered window (green), three point central difference (blue), and least square spectral matching (purple) filters.

Figure 32 shows the errors made when using the central difference, Hanning window and least square approximations to the second derivative of our test function.
5.2 Second Order

Error on Second Derivatives for Three Different Methods.

Error on $s'_t$ of 1 Hz. Error on $s'_{10}(t)$ of 1 to 10 Hz.

Error on $s'_{100}(t)$ of 1 to 100 Hz Error on $s'_{400}(t)$ 1 to 400 Hz

Figure 32: Error of computing the derivative of function 5.36 by using the Hanning window (red) tapering, second order central differences (green), and the least square spectral matching method (blue).

We observe in Figure 32 that the spectral matching method does not do as well as the other methods except at high frequencies (400 Hz) where in general all methods do poorly since none are designed for this spectral window.

5.2.3 Lagrange Interpolation

We use the results developed in section 4.1. By differentiating 4.19 with respect to $t$ we find:
\[ f''(t) \approx \sum_{i=-j}^{k} f(t_i)L''_{N,i}(t) \]  

(5.37)

where now we dropped all terms with derivatives of order \( N + 1 \) and beyond. Taking one more derivative in equation 4.21 we find:

\[
L''_{N,i}(t_p) = \sum_{l=-j \atop l \neq i}^{k} \frac{1}{t_i - t_l} \sum_{n=-j \atop n \neq l \neq i}^{k} \frac{1}{t_i - t_n} \prod_{q=-j \atop q \neq l \neq n}^{k} \left. \frac{t - t_q}{t_1 - t_q} \right|_{t=t_p}.
\]

As we did before, and to simplify the problem, we assume uniform distribution of time samples with time interval \( \Delta t \). So

\[
L''_{N,i}(t_p) = \sum_{l=-j \atop l \neq i}^{k} \frac{1}{(i-l)\Delta t} \sum_{n=-j \atop n \neq l \neq i}^{k} \frac{1}{(i-n)\Delta t} \prod_{q=-j \atop q \neq l \neq n}^{k} \frac{p-q}{i-q}.
\]

To further simplify we assume \( p = 0 \), and samples are between the \(-k\) and \( k\) index position (that is \( j = k\)), to find

\[
L''_{N,i}(t_0) = \frac{1}{(\Delta t)^2} \sum_{l=-k \atop l \neq i}^{k} \frac{1}{(i-l)} \sum_{n=-k \atop n \neq l \neq i}^{k} \frac{1}{(i-n)} \prod_{q=-k \atop q \neq n \neq l \neq i}^{k} \frac{q}{q-i}.
\]  

(5.38)

We now show that this equation is symmetric for \( i \). That is, if we change \( i \) for \(-i\), equation 5.38 does not change. Let us see

\[
L''_{N,-i}(t_p) = \frac{1}{(\Delta t)^2} \sum_{l=-k \atop l \neq -i}^{k} \frac{1}{(-i-l)} \sum_{n=-k \atop n \neq l \neq -i}^{k} \frac{1}{(-i-n)} \prod_{q=-k \atop q \neq n \neq l \neq -i}^{k} \frac{q}{q+i}.
\]

reverse signs

\[
= \frac{1}{(\Delta t)^2} \sum_{l=k \atop l \neq -i}^{k} \frac{1}{(-i+l)} \sum_{n=k \atop n \neq l \neq -i}^{k} \frac{1}{(-i+n)} \prod_{q=k \atop q \neq n \neq l \neq -i}^{k} \frac{q}{q-i}.
\]

reverse signs

\[
= \frac{1}{(\Delta t)^2} (-1)^2 \sum_{l=-k \atop l \neq -i}^{k} \frac{1}{(-i+l)} \sum_{n=-k \atop n \neq l \neq -i}^{k} \frac{1}{(-i+n)} \prod_{q=-k \atop q \neq n \neq l \neq -i}^{k} \frac{-q}{-q+i}.
\]

\[
= L''_{N,i}(t_p)
\]
This is important, because instead of having to find \(2k + 1\) coefficients we need to find only \(k + 1\) (this include the zero position coefficient and the others symmetrically distributed around this zero position.) We derive the second order central difference operator.

Assume 3 points \(t_{-1} = t_0 - \Delta t, \ t_0 = t_0, \ t_1 = t_0 + \Delta t\).

\[
(\Delta t)^2 c_i = L''_{2,3}(t_0) = \sum_{l=-1}^{1} \frac{1}{(i - l)} \sum_{n=-1}^{1} \frac{1}{(i - n)} \prod_{q=-1}^{1} \frac{q}{q - i}.
\]

Let us find the coefficients \(c_{-1}, \ c_0\) and \(c_1\).

- For \(i = 0\) we add two terms
  
  (i) \(l = -1\)

  \[
  \frac{1}{(-1)} \sum_{n=-1}^{1} \frac{1}{(-n)} = -1
  \]

  (ii) \(l = 1\)

  \[
  -1 \sum_{n=-1}^{1} \frac{1}{(-n)} = -1.
  \]

  Then

  \[c_0(\Delta t)^2 = -2.\]

- For \(i = 1\) we need to add two terms
  
  (i) \(l = -1\)

  \[
  \frac{1}{2} \sum_{n=-1}^{1} \frac{1}{(1 - n)} \prod_{q=-1}^{1} \frac{q}{q - 1} \frac{q}{q - 1} \frac{1}{q - 1} \frac{1}{q - 1} = \frac{1}{2} \prod_{q=-1}^{1} \frac{q}{q - 1} = \frac{1}{2}.
  \]

  1 by vacuity
(ii) \( l = 0 \)

\[
\sum_{n=-1}^{1} \frac{1}{(1-n)} \prod_{q=-1}^{1} \frac{q}{q+1} = \frac{1}{2} \prod_{q=-1}^{1} \frac{q}{q+1} = \frac{1}{2} \quad \text{by vacuity}
\]

Then

\[ c_1(\Delta t)^2 = 1. \]

Now, since the coefficients are symmetric,

\[ c_1(\Delta t)^2 = c_{-1}(\Delta t)^2 = 1. \]

So the three coefficient operator is \([1/(\Delta t)^2](1, -2, 1)\) which is the second order central difference operator.

Let us now find the three coefficients \(c_0\) and \(c_1, c_2\), for an \(N = 5\) points filter.

- For \(i=0\) We have that \(q/(q + i) = 1\), so \(\prod_{q=-1}^{1} \frac{q}{q+i} = 1\). We add 4 terms:

  (i) \( l=-2 \)

\[
\frac{1}{2} \sum_{n=-2}^{2} \frac{1}{(-n)} = \frac{1}{2} \left( 1 - 1 - \frac{1}{2} \right) = -\frac{1}{4}
\]

(ii) \( l=-1 \)

\[
\sum_{n=-2}^{2} \frac{1}{(-n)} = \left( \frac{1}{2} - 1 - \frac{1}{2} \right) = -1
\]

(iii) \( l=1 \)

\[- \sum_{n=-2}^{2} \frac{1}{(-n)} = - \left( \frac{1}{2} + 1 - \frac{1}{2} \right) = -1
\]

(iv) \( l=2 \)

\[- \frac{1}{2} \sum_{n=-2}^{2} \frac{1}{(-n)} = - \frac{1}{2} \left( \frac{1}{2} + 1 - 1 \right) = -\frac{1}{4}
\]
5.2 Second Order

So

\[(\Delta t)^2c_0 = -\frac{5}{2}\]

A similar analysis reveals that

\[(\Delta t)^2c_1 = (\Delta t)^2c_{-1} = \frac{4}{3} \quad \text{and} \quad (\Delta t)^2c_2 = (\Delta t)^2c_{-2} = \frac{1}{12}\]

so the 5 point second differentiator is given by

\[\frac{1}{(\Delta t)^2}(-1/12, 4/3, -5/2, 4/3, -1/12).\]  \hfill (5.39)

Here is a C code that implements the computation of the coefficients \(c(i)\),

```c
#include <stdio.h>
#include <math.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
    /* miscellaneous */
    int n; /* number of coefficients, hard coded to 5 */
    int k;
    int i,j,l,q; /* indices */
    float sumout, sumin, scalout, scalin; /* accumulators */
    // float *filt; /* filter coefficients */

    /* usage */
    if(argc < 1 )
    {
        printf("Program that generates Lagrange coefficients for derivatives \n");
        exit(0);
    }

    printf("Enter number of coefficients \n");
    scanf("%d", &n);
    k = n/2;

    printf("Number of coefficients n=%d, k=%d\n", n, k);

    /* generate coefficients */
    for(i=0; i<=k; i++) /* i-th coefficient */
    {
        sumout=0;
        scalout=1.0;
        for(l=-k; l <= k; l++) /* sum over l */
        {
            scalin=1.0;
            if(i != l)
            {
                scalout =1.0/(i-l);
                sumin=0;
                for(n=-k; n <=k; n++) /* sum over n*/
                {
                    if((n != l) && (n != i))
                    {
                        scalin = 1.0/(i-n);
                    }
                }
            }
        }
    }

    return(0);
}
```

for(q=-k; q<=k; q++) /* product over q */
{
    if( (q != n) && (q != l) && (q != i))
        scalin *= ((float) q)/( (float) (q-i));
    sumin += scalin;
}
sumout *= scalout*sumin;
}
printf("Coefficient c[%d]=%f \
", i, sumout);
}

and the output of the program for 5 coefficients is

hjaramil $ lagrange2
Enter number of coefficients
5
Number of coefficients n=5, k=2
Coefficient c[0]=-2.500000
Coefficient c[1]=1.333333
Coefficient c[2]=-0.083333
Coefficient c[3]=0.025397
Coefficient c[4]=-0.001786

We verify that these coefficients correspond to the filter in equation 5.39.

The computation for 9 coefficients produces

baggins3,hjaramil $ lagrange2
Enter number of coefficients
9
Number of coefficients n=9, k=4
Coefficient c[0]=-2.847222
Coefficient c[1]=1.600000
Coefficient c[2]=-0.200000
Coefficient c[3]=0.025397
Coefficient c[4]=-0.001786

Figure 33 shows the set of spikes corresponding to the spectral matching and Lagrange interpolation for the second order derivative.
Figure 33: Spikes for the least square spectral matching and Lagrange interpolator for the second derivative operator.

Figure 34 shows the spectra of all methods explained here for the second order differentiator.
5.2 Second Order

Figure 34: Amplitude spectra of the ideal signal (red) versus the Hanning tapered window (green), three point central difference (blue), least square spectral matching (purple) and the Lagrange interpolation (light blue), filters.

We see that the Lagrange interpolation filter has a lower degree of matching for the high frequencies. In any case all, except by the Hanning window method, amplify the high frequencies. The least square design should include a condition to smoothly decay to 0 at Nyquist.

Figure 35 shows the errors produced by the second order differentiators studied in this section.
Error on Second Derivatives for Four Different Methods.

We find that except for the central the second order central difference filter (which behaves good for low frequencies) the filters have a good approximation up to high frequencies, except in a neighborhood of the cusp (around 0.75). In all cases, except for the edges, the Hanning window method is more accurate, however these methods use many points (1261 instead the 9 used on the other filters) in its evaluation. A good compromise requires an analysis of the range of target frequencies, required accuracy and cost of the algorithm. For example, for Kirchhoff migration of seismic data the windowing method is ideal since it let us preserve high frequencies and it is not expensive compared to the cost of the ray tracing and wavefield integration. For an RTM code 3 to 9 point filters could be used depending on the frequency band required and the
5.2 Second Order

running cost of the program.

5.2.4 The Taylor Series Method:

The Taylor series method illustrates another way to find the Lagrange interpolation coefficients. As shown in section 4.2 we start with the Taylor series expansions

\[ f(t + \Delta t) = f(t) + \Delta tf^{(1)}(t) + \frac{(\Delta t)^2}{2!}f^{(2)}(t) + \frac{(\Delta t)^3}{3!}f^{(3)}(t) + \ldots \]

and

\[ f(t - \Delta t) = f(t) - \Delta tf^{(1)}(t) + \ldots + (-1)^k \frac{(\Delta t)^k}{k!}f^{(k)}(t) + \ldots \]

from which is easy to observe that

\[ f(t + \Delta t) + f(t - \Delta t) = 2f(t) + \frac{2(\Delta t)^2f''(t)}{2} + O((\Delta t)^4). \]

So

\[ f''(t) = \frac{f(t - \Delta t) - 2f(t) + f(t + \Delta t)}{(\Delta t)^2} + O((\Delta t)^2) \]

which coincides with the second order approximation \([1/(\Delta t)^2](1,-2,1)\) to the second order differentiator.

In general we can set up the equation

\[ \sum_{i=-k}^{k} c_i f(t + i\Delta t) = \sum_{i=-k}^{k} c_i \sum_{j=0}^{\infty} \frac{(i\Delta t)^j}{j!} f^{(j)}(t) = f^{(2)}(t) + O((\Delta t)^2k). \]

We already know some constraints. That is, \(c_{-i} = c_i\) (for the central difference approximations) so we only need to find \(k + 1\) coefficients. Let us reverse indices above and find

\[ \sum_{j=0}^{\infty} \frac{f^{(j)}(t)}{j!} \sum_{i=-k}^{k} c_i (i\Delta t)^j = f^{(2)}(t) + O((\Delta t)^{2k}). \]

We match coefficients based on the derivative order.

\[ j = 0 \quad \Rightarrow \quad c_0 + 2 \sum_{i=1}^{k} c_i = 0 \quad \text{balance of coefficients.} \]

\[ ^4 \text{Note that we use the positive sign in the argument “}t + i\Delta t\text{”. It does not matter here because the filter is symmetric. However for the first order, which is anti-symmetric we should respect the minus “} - \text{” sign that acknowledges the convolution process. See equation 4.29.} \]
5.2 Second Order

This provides the interesting property that the middle coefficient balances twice the rest of the coefficients. That is \( c_0 = -2 \sum_{k>0} c_k \). For \( j = 1 \),

\[
j = 1 \quad \Rightarrow \quad \sum_{i=-k}^{k} c_i i \Delta t = 0. \quad \text{balance of first momentum. Center of mass.}
\]

This equation does not provide any new contribution, since except for the \( c_0 \) term, the other terms cancel by pairs (since \( i \) is odd with respect to 0). Now for \( c_0 \) the coefficient is \( i = 0 \).

For \( j = 2 \),

\[
j = 2 \quad \Rightarrow \quad \sum_{i=-k}^{k} \frac{c_i i^2 (\Delta t)^2}{2} = 1 \quad \text{normalization of second momentum.}
\]

That is

\[
\sum_{i=1}^{k} c_i i^2 = \frac{1}{(\Delta t)^2}.
\]

For higher order we make all coefficients \( j > 2 \) equal to zero. That is, let \( j > 2 \) then

\[
\sum_{i=-k}^{k} c_i (i \Delta t)^j = 0,
\]

or simply

\[
\sum_{i=-k}^{k} c_i i^j = 0.
\]

If \( j \) is odd, then this equation does not provide new information, so let us assume \( j = 2p \), then

\[
j = p, \quad p > 1 \quad \Rightarrow \quad \sum_{i=1}^{k} c_i i^{2p} = 0.
\]

We can summarize this section with the following simultaneous equations

\[
c_0 = -2 \sum_{i=1}^{k} c_i \quad \text{Balance of coefficients}
\]

\[
\sum_{i=1}^{k} c_i i^2 = \frac{1}{(\Delta t)^2} \quad \text{Second Momentum Conservation.}
\]

\[
p > 1 \quad \Rightarrow \quad \sum_{i=1}^{k} c_i i^{2p} = 0 \quad \text{High order momentums vanish.}
\]
for the coefficients $c_0, \ldots, c_k$ of a second derivative filter which approximate a polynomial function perfectly up to the $k$ power. Note that the first equation is decoupled from the rest of equations. We can always solve the $k$ lower equations with $k$ unknowns and then find $c_0$ from the first equation.

Let us apply this equation to two low order approximations

- $k = 1$.
  
  Only two equations are required
  
  $$c_0 = -2c_1$$
  $$c_1 = \frac{1}{(\Delta t)^2}$$
  
  So obviously the second order central difference operator $[1/(\Delta t)^2](1, -2, 1)$ results.

- $k = 2$.
  
  Here we need three equations
  
  $$c_0 = -2(c_1 + c_2)$$
  $$c_1 + 4c_2 = \frac{1}{(\Delta t)^2}$$
  $$c_1 + 16c_2 = 0.$$  
  
  From the third equation $c_1 = -16c_2$, plug this in the second equation and find
  
  $$-16c_2 + 4c_2 = \frac{1}{(\Delta t)^2} \Rightarrow c_2 = -\frac{1}{12(\Delta t)^2}$$
  
  So
  
  $$c_1 = -16c_2 = \frac{16}{12(\Delta t)^2} = \frac{4}{3(\Delta t)^2}$$
  $$c_0 = -2(c_1 + c_2) = -\frac{2}{(\Delta t)^2} \left( \frac{-1}{12} + \frac{4}{3} \right) = -\frac{30}{12(\Delta t)^2} = -\frac{5}{2(\Delta t)^2}.$$  
  
  So the 5 coefficients are
  
  $$\frac{1}{\Delta t} \left( -\frac{1}{12}, \frac{4}{3}, -\frac{5}{2}, \frac{4}{3}, -\frac{1}{12} \right).$$
  
  This coincides with expression [5.39] derived from the Lagrange polynomial interpolation.

As indicated above, both the Lagrange and the Taylor series method should provide the same results, since they are both yield exact approximations to polynomials of order $k$.

The methods outline above could be used to find representations of higher order derivatives (3 and above) but, of course, the algebra becomes more difficult as the order of the derivatives increase. Next we discuss the case of fractional differentiation.
5.3 Fractional Derivatives

The frequency domain is a good platform to explain fractional derivatives. As $-i\omega$ represents a first derivative filter and $-\omega^2$ a second derivative filter, we can represent any order $p$, of differentiation as $(-i\omega)^p$.

We will use the properties of fractional calculus as shown in Oldham and Spanier, [5]. Mainly that

$$D^p D^q = D^{p+q}$$

(5.40)

where $D^x$ means fractional derivative of order $x$. The definition of fractional derivative is given in equation (5.58). I suggest the interested reader on fractional calculus to look at Oldham and Spanier reference, I found this book quite clear and complete.

I will first find $D^{-1/2}$ and then by taking the derivative find $D D^{-1/2} = D^{1/2}$. The half integrator $D^{-1/2}$ on the frequency domain is defined as

$$D^{-1/2}(\omega) = \frac{1}{\sqrt{-i\omega}}$$

(5.41)

To find the half integrator in the time domain we compute the inverse Fourier transform. This is given by

$$D^{-1/2}(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} d\omega \frac{e^{-i\omega t}}{\sqrt{-i\omega}}$$

(5.42)

Let us do the mathematical exercise of computing this integral

$$D^{-1/2}(t) = \frac{1}{2\pi} \int_{-\infty}^{0} d\omega \frac{e^{-i\omega t}}{\sqrt{-i\omega}} + \frac{1}{2\pi} \int_{0}^{\infty} d\omega \frac{e^{-i\omega t}}{\sqrt{-i\omega}}$$

$$= \frac{1}{2\pi} \int_{0}^{\infty} d\omega \frac{e^{i\omega t}}{\sqrt{i\omega}} + \frac{1}{2\pi} \int_{0}^{\infty} d\omega \frac{e^{-i\omega t}}{\sqrt{-i\omega}}$$

$$= \frac{1}{2\pi} \int_{0}^{\infty} d\omega \frac{e^{i(\omega t - \pi/4)} + e^{-i(\omega t - \pi/4)}}{\sqrt{\omega}}$$

$$= \frac{2}{2\pi} \int_{0}^{\infty} d\omega \frac{\cos(\omega t - \pi/4)}{\sqrt{\omega}}$$

so from

$$\cos(\omega t - \pi/4) = \cos(\omega t) \cos(\pi/4) + \sin(\omega t) \sin(\pi/4) = \frac{1}{\sqrt{2}} (\cos \omega t + \sin \omega t)$$

we obtain

$$D^{-1/2}(t) = \frac{1}{\sqrt{2\pi}} \int_{0}^{\infty} d\omega \cos(\omega t) + \sin(\omega t) \sqrt{\omega}.$$

(5.43)
We now make the convenient change of variables $\omega t = u^2$, so

$$\frac{d\omega}{\sqrt{\omega}} = \frac{2 du}{\sqrt{t}}$$

and equation (5.43) turns out to be

$$D^{-1/2}(t) = \frac{1}{\pi} \sqrt{\frac{2}{t}} \int_0^\infty du (\cos u^2 + \sin u^2).$$

This integral is of the family of the Fresnel integrals. I take the following formulas from Abramowitz and Stegun [1]

$$C(z) = \int_0^z dt \cos \frac{\pi t^2}{2}$$
$$S(z) = \int_0^z dt \sin \frac{\pi t^2}{2}$$

and

$$\lim_{z \to \infty} C(z) = \lim_{z \to \infty} S(z) = \frac{1}{2}$$

Now from the change of variables $u^2 = \pi t^2/2$ on equation (5.45) we find that

$$D^{-1/2}(t) = \frac{1}{\pi} \sqrt{\frac{2}{t}} \sqrt{\frac{\pi}{2}} \int_0^\infty du (\cos \frac{\pi}{2} u^2 + \sin \frac{\pi}{2} u^2),$$

so that finally,

$$D^{-1/2}(t) = \begin{cases} 0 & \text{if } t < 0 \\ \frac{1}{\sqrt{\pi t}} & \text{if } t > 0 \end{cases}$$

That is,

$$D^{-1/2}(t) = \frac{H(t)}{\sqrt{\pi t}},$$

with $H(t)$ being the Heaviside (step) function. Note that $D^{-1/2}(t)$ diverges at $t = 0$ and so it does its derivative being “extremely discontinuous there”. Therefore we could be able to tune the filter outside of $t = 0$.

Equation (5.49) was derived by Deregowski and Brown [4]. Deregowski and Brown also used equation (5.40) to derive the half differentiator. That is, by taking the derivative of equation (5.49) we find

$$D^{1/2}(t) = \frac{1}{\sqrt{\pi}} \left( \frac{\delta(t)}{\sqrt{t}} - \frac{1}{2} \frac{H(t)}{t^{3/2}} \right)$$

(5.50)
The question here is: what is the ideal realization of this function on the discrete world? We know beforehand that the half derivative operator has all frequencies so any sampling on this function will certainly introduce aliasing. By using the same theory employed to derive the realization of the derivative filter on the previous section we find that the band–limited filter on the time domain is achieved by the convolution with the sinc function (2.2).

Convolution of $5.50$ with the sinc function (2.2) is given by

$$
\hat{D}^{1/2}(t) = \int_{-\infty}^{\infty} \frac{1}{\sqrt{\pi}} \left( \frac{\delta(\tau)}{\sqrt{\tau}} - \frac{1}{2} \frac{H(\tau)}{\tau^{3/2}} \right) \Delta t \sin\left[\omega_N (t - \tau) \right] \pi \frac{t - \tau}{\pi},
$$

(5.51)

This integral diverges for $\tau = 0$. For $\tau \neq 0$ the $\delta$ term vanishes so that the result is,

$$
\hat{D}^{1/2}(t) = -\frac{\Delta t}{2\sqrt{\pi}} \int_{0}^{\infty} \sin\left[\omega_N \left(t - \tau\right) \right] \pi \frac{t - \tau}{\pi(t - \tau) \tau^{3/2}}.
$$

(5.52)

The problem here is that this integral is difficult to evaluate in closed form. I am not aware of any solution to this integral.

An easier way to solve the problem is to band–limit the half derivative operator with a filter other than a box car and then sample it. A running average is a low pass filter. The discretization formula

$$
\hat{D}^{1/2}(n) = \frac{1}{\Delta t} \int_{(n-1/2)\Delta t}^{(n+1/2)\Delta t} dt D^{1/2}(t)
$$

(5.53)

provides a sampling of the function after a low pass filter (the running average). This filter should attenuate high frequencies and therefore the aliasing will be reduced. Still this is an approximation. The evaluation of this integral is presented next.

As pointed out before, this integral diverges on any interval around 0. So we will assume $t > 0$. So our first value would be $n = 1$. We have

$$
\hat{D}_{1/2}(n) = \frac{1}{\Delta t} \int_{(n-1/2)\Delta t}^{(n+1/2)\Delta t} dt \frac{1}{\sqrt{\pi}} \left( \frac{\delta(t)}{\sqrt{t}} - \frac{1}{2} \frac{H(t)}{t^{3/2}} \right)
$$

$$
= \frac{-1}{2\sqrt{\pi} \Delta t} \int_{(n-1/2)\Delta t}^{(n+1/2)\Delta t} \frac{1}{t^{3/2}}
$$

$$
= \frac{1}{\sqrt{\pi} \Delta t \Delta t^{1/2}} \left[ \frac{1}{(n+1/2)\Delta t} - \frac{1}{(n-1/2)\Delta t} \right]
$$

$$
= \sqrt{\frac{2}{\pi}} \frac{1}{\Delta t^{3/2}} \left[ \frac{1}{2n+1} - \frac{1}{2n-1} \right]
$$

(5.54)

That is

$$
D_{1/2}(n) = \begin{cases} 
0 & \text{if } n < 0 \\
\text{undefined} & \text{if } n = 0 \\
\sqrt{\frac{2}{\pi}} \frac{1}{\Delta t^{3/2}} \left[ \frac{1}{2n+1} - \frac{1}{2n-1} \right] & \text{if } n > 0 
\end{cases}
$$

(5.55)
The fact that this function is not defined at \( t = 0 \) leaves us in a waving hand argument about how to implement this function. Deregowski and Brown also derive a discretization of the half derivative operator (see their equation (18)). I want to point out that in their derivation, the operator is missing the sampling rate factor \( (\Delta t) \) and also they found that at \( t = 0 \) the operator is 1. I do not agree with that. Next is Deregowski and Brown version of the discrete time series for the half derivative operator:

\[
D^{1/2}(n) = \begin{cases} 
0 & \text{if } n < 0 \\
1 & \text{if } n = 0 \\
\left[ \frac{1}{\sqrt{2^{n+1}}} - \frac{1}{\sqrt{2^{n-1}}} \right] & \text{if } n > 0.
\end{cases}
\] (5.56)

I found an approximation where I define the value of the half derivative filter as 1 if \( n = 0 \) but then I had to tune up the series (by an overall scaling value). This is,

\[
D^{1/2}(n) = \begin{cases} 
0 & \text{if } n < 0 \\
1 & \text{if } n = 0 \\
-0.025 \sqrt{\frac{2}{\Delta t}} \left[ \frac{1}{2^{n+1}} - \frac{1}{2^{n-1}} \right] & \text{if } n > 0
\end{cases}
\] (5.57)

Before comparing the different representations of the discrete half derivative operator, I will discuss one more representation. This representation is given by Oldham and Spanier, [5] and it is the topic of the next section.

**The Oldham and Spanier fractional derivative**. Oldham and Spanier, [5] define the \( q \)-fractional derivative of a one dimensional function at the point \( a \) as

\[
\left[ \frac{d^q}{d(x-a)^q} \right] f(x) = \lim_{N \to \infty} \left\{ \frac{x-a}{N} \right\}^{-q} \frac{\Gamma(j-q)}{\Gamma(j+1)} \sum_{j=0}^{N-1} \Gamma(j-q) \Gamma(j+1) f \left( x - j \left[ \frac{x-a}{N} \right] \right) \}
\] (5.58)

This definition is well supported on an extension of the concept of a forward difference operator from derivatives of positive integer order to those of negative integer order (integrals). Then by going to a continuum through the extension of factorials using the function \( \Gamma \).

Equation (5.58) indicates that to obtain the fractional derivative of a function \( f \) at the point \( a \) is approximated we should convolve the filter coefficients \( a_j \) with the the samples of the discretized function \( f \). Here the coefficients \( a_j \) are defined as:

\[
a_j = \frac{\left[ \frac{x-a}{N} \right]^{-q}}{\Gamma(-q)} \frac{\Gamma(j-q)}{\Gamma(j+1)}
\] (5.59)

Let us simplify this expression. First, we set up \( q = 1/2 \) and \( \Delta t = (x-a)/N \). Let us evaluate the factors with the function \( \Gamma \). First we find \( \Gamma(-1/2) \). We use the following identities of the \( \Gamma \) function:

\[
\Gamma(1/2) = \sqrt{\pi}
\]
\[
\Gamma(x-1) = \Gamma(x)/(x-1).
\] (5.60)
5.3 Fractional Derivatives

From here it is clear that $\Gamma(-1/2) = -\sqrt{\pi}$. So

$$a_j = -\frac{\Delta t^{-1/2}}{2\sqrt{\pi}} \frac{\Gamma(j - 1/2)}{\Gamma(j + 1)}$$  \hspace{1cm} (5.61)

We could find an explicit simplification of this coefficients in terms of factorials. However for computational purposes recursive coefficients are more efficient. We use the property $\Gamma(x) = (x - 1) \Gamma(x - 1)$, and find that

$$\frac{a_j}{a_{j-1}} = \frac{\Gamma(j - 1/2) \Gamma(j)}{\Gamma(j + 1) \Gamma(j - 3/2)} = \frac{j - 3/2}{j}.$$  \hspace{1cm} (5.62)

The computer implementation reduces to the recursive sequence:

$$a_0 = -\Delta t^{-1/2}$$
$$a_1 = a_0/2$$
$$a_j = a_{j-1} \frac{j - 3/2}{j} \quad \text{for} \quad j \geq 2.$$  \hspace{1cm} (5.63)

To fit the overall amplitude and phase, I defined $a_0$ as

$$a_0 = 0.0405617/\sqrt{\Delta t}.$$  

Figure 36 shows a comparison between the analytical half derivative of the sin function and the numerical implementation in the frequency domain, the time domain method following Deregowski and Brown’s approach, Oldham and Spanier and also my own approach. Except for the Deregowski and Brown approach, all other methods look accurate.

- Analytical
- Frequency domain
- Eq. (5.49)
- Deregowski and Brown
- Oldham and Spanier

Figure 36: Comparison for the half derivative of a sinusoidal function. Here are the analytical solution, the Deregowski and Brown filter, and the implementation from equation 5.57.
5.3 Fractional Derivatives

Figure (37) shows a comparison of the error between the analytical half derivative of the sin function and the numerical implementation displayed in Figure (36).

It is clear from this figure that Oldham and Spanier followed by the implementation from equation 5.57 are more accurate (at least for the picked function) than that of Deregowski and Brown. However except for the edge effect at the beginning of the function, the frequency domain solution fits better the analytical solution. The fact that I had to hard coded scaling factors to fit amplitudes leaves me on a waving hand argument for the time domain implementations. There is still much uncertainty on the time domain filter for the half derivative. The conclusion here should be evident: A frequency domain implementation of the half derivative filter is not only easier to code but easier to understand from the theoretical point of view. We do not yet know the ideal half derivative (2D and 2.5D) filter in the time domain. This is a source of theoretical research. In any case it is unlikely that a fractional derivative implementation could be use for finite difference algorithms of wave equations. Hence the frequency evaluation should be the chosen method for correct for the rho filter in integration approaches.

In addition, there is no room for Lagrange or Taylor series methods under fractional derivatives. We are not including here either spectral matching methods, and the error analysis is done only for one frequency. This opens some room for future research on the topic.
6 Multiple Dimensions

For multiple dimensions we consider Cartesian coordinate systems where the axis are orthogonal. In this ways the derivatives are decoupled and we could consider each axis as a piece of the one–dimensional space where we know how to compute already the derivatives. That is, the evaluation of the partial derivative

$$\frac{\partial f}{\partial x_i}$$

numerically will be implemented with the same algorithms used for the computation of the total derivative

$$\frac{df}{dx}.$$  

By the way of an example let us consider the Laplacian operator

$$\nabla = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}. \quad (6.64)$$

The finite difference discretization of the Laplacian \[6.64\] could be given by

$$\nabla^2 u \approx a_{xyz0} \ u_{j,k,l}^i + \sum_{n=1}^{n=4} a_{xn} (u_{j+n,k,l}^i + u_{j-n,k,l}^i)$$

$$+ a_{yn} (u_{j,k+n,l}^i + u_{j,k-n,l}^i)$$

$$+ a_{zn} (u_{j,k,l+n}^i + u_{j,k,l-n}^i) \quad (6.65)$$

$$= \mathcal{L}(u^i)$$

with

$$a_{xyz0} = \frac{a_0}{\Delta x^2} + \frac{a_0}{\Delta y^2} + \frac{b_0}{\Delta z^2}$$

$$a_{xn} = \frac{a_n}{\Delta x^2}$$

$$a_{yn} = \frac{a_n}{\Delta y^2}$$

$$a_{zn} = \frac{b_n}{\Delta z^2}$$

where the symbol

$$u_{j,k,l}^i$$

represents the function \(u\) evaluated at the point three dimensional space point \((x_0 + j\Delta t, y_0 + k\Delta y, z_0 + l\Delta z)\) with grid origin \((x_0, y_0, z_0)\) and spatial sampling rates \(\Delta x, \Delta y, \Delta z\).
The super index $i$ notes the temporal evaluation at $t_0 + i\Delta t$, where $t_0$ is the zero time and $\Delta t$ the time sampling rate. For the particular case of the Laplacian all derivatives are spatial derivatives, but in the wave equation the time derivative is required.

The coefficients $a_{xn}$, $a_{yn}$ and $a_{zn}$ could be computed using the methods for the one-dimensional differentiation explained previously.

Figure 38 shows the finite difference star for the spatial coordinates.

Figure 38: Illustration of the spatial finite difference star for the 3D wave equation. The blue atoms represent the coefficients along the $x$ and $y$ directions. The green atoms are the coefficients along the $z$ direction, which are computed by a least square method (if non-uniform sampling along this direction). The red atom is the center, common to all grid propagation directions and carry a unique (combined) coefficient. The volume of the atoms somehow represent the size of the coefficient, and the signs of the coefficients are all alternating along each direction with the center (red) being negative.

In the frequency/wave number domain

$$\frac{\partial^p}{\partial x_i^p} \xrightarrow{FT} (ik_i)^p,$$

where $p$ can be a rational number and $k_i$ is the wave number along a dimension axis $x_i$ or $-\omega$ if $x_i$ represents time $t$. 
7 Conclusions

We showed a set of algorithms that can be used to differentiate digital data under different contexts. The algorithms are both in the time/space and frequency/wave number domains. We found that the frequency/wave number domains have a better cost–benefit relationship than the time/space algorithms for modeling/migration of seismic data under integration methods such as the Kirchhoff algorithms. For finite differences a small set of coefficients should be used. The minimum number of coefficients are 2 for first order and 3 for second order. Fractional derivatives require a large number of filter coefficients but in the frequency domain seems easy and accurate to implement. In addition, together with the windowing methods in the frequency/wave number domains, we studied the Lagrange interpolation algorithm methods, the Taylor series methods, and least square spectral matching methods. While the windowing method (using Hanning widows) yields better accuracy, it is not appropriate for finite difference algorithms. We illustrate the application of fractional derivative filters both in time/space domains and frequency/wavenumber domains. Finally we showed that the analysis for one dimension is ready to apply to multi–dimensional differentiation and illustrate this with an example of a Laplacian operator in a three–dimensional real space.

References


