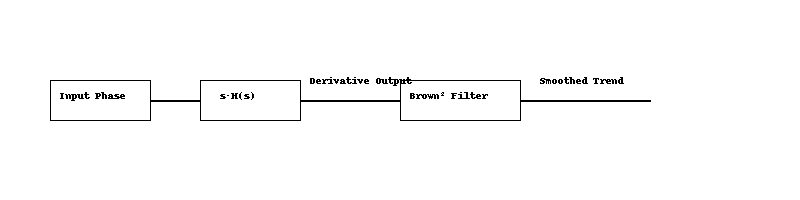
# PLL Derivative and Brown Filter Analysis

This document illustrates the structure and frequency responses for a second-order PLL system's derivative output, with and without a second-order Brown low-pass filter applied.

## Block Diagram

The diagram below shows the signal flow and where each transfer function is measured:



• The input phase on the picture is the PLL input signal.  
• The pre-integrator output is modeled by s·H(s) where H(s) is defined as:

A black background with white text

AI-generated content may be incorrect.

The unit response for z=1, is:

A screen shot of a graph

AI-generated content may be incorrect.

And the frequency response and phase plot are:

A screenshot of a computer

AI-generated content may be incorrect.

• This output is optionally passed through a second-order Brown filter to produce a smoothed trend signal.

## Bode Plot

The plot below compares the frequency responses of the derivative output (red) and the filtered version (blue).

A screenshot of a computer

AI-generated content may be incorrect.

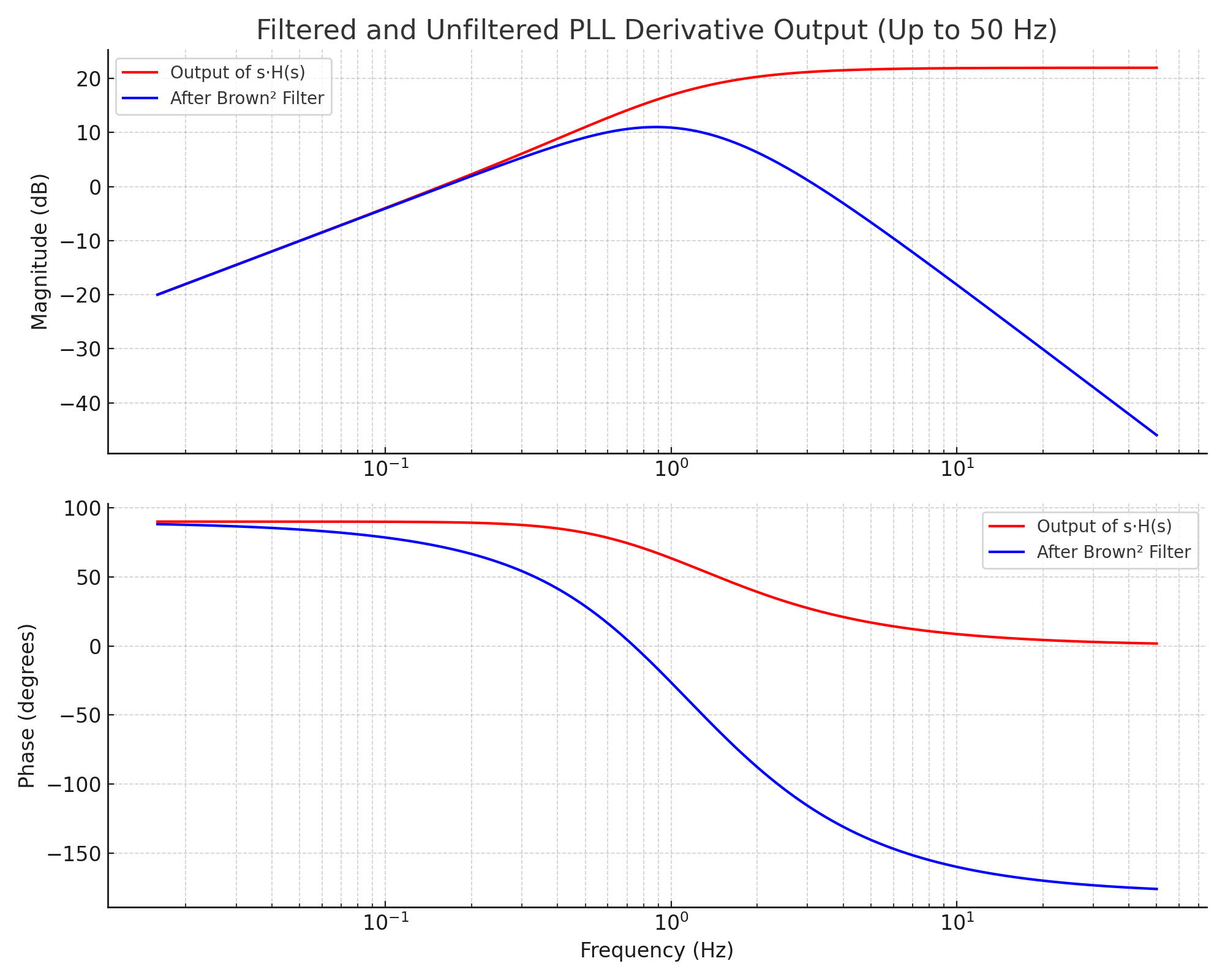
If the output derivative is filtered with a second order Brown low pass filter the system full response become a 4 order system with the following characteristic:

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.



• The raw derivative (s·H(s)) shows significant high-frequency amplification.  
• The Brown² filter reduces high-frequency noise while preserving trend responsiveness near the PLL's natural frequency.

**Title: Relationship Between Alpha and Cutoff Frequency in First-Order Exponential Filters**

**1. Introduction** A first-order exponential filter (also called a one-pole IIR low-pass filter or Brown's exponential smoother) is commonly used in digital signal processing for its simplicity and computational efficiency. The key design parameter, alpha (α\alpha), controls the responsiveness or smoothing level of the filter. Understanding the relationship between α\alpha and the filter's 3 dB cutoff frequency is important for both filter design and interpretation.

**2. Filter Definition** The filter is defined by the recursive equation:

Where:

* x[n]: input signal
* y[n]: output signal
* : smoothing parameter

**3. Frequency Response and Cutoff Definition** The frequency response in the Z-domain is:

Taking the squared magnitude gives:

The cutoff frequency is defined as the frequency where the gain is reduced by 3 dB:

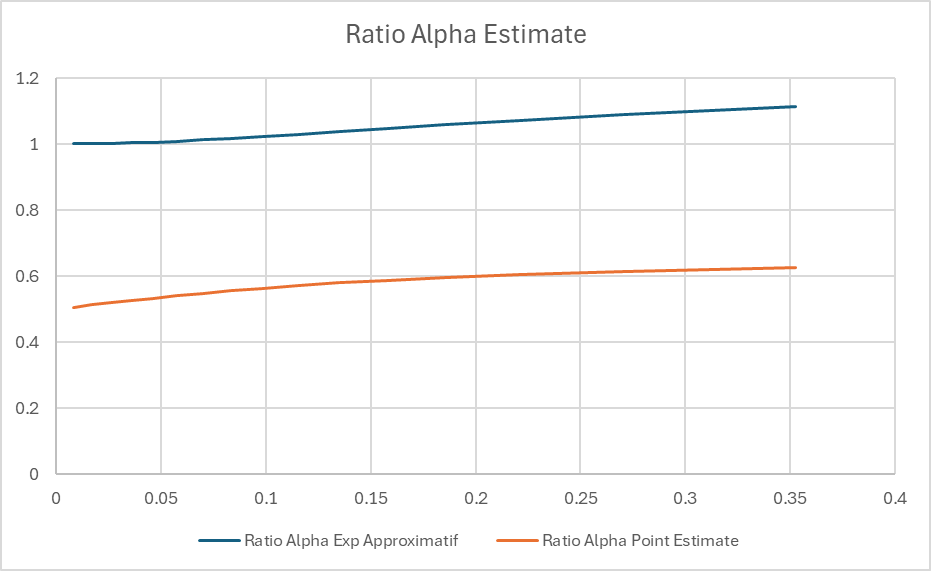
Solving this gives the **exact expression**:

**4. Approximate Formula** A commonly used approximation based on continuous-time filter equivalence via the bilinear transform is:

Where:

* : desired cutoff frequency in Hz
* : sampling rate in Hz

Inverting this gives:



**7. Time to Reach 90% of Final Value (Step Response)** When a unit step is applied to a first-order exponential filter, the output follows:

To determine how many samples it takes for the response to reach 90% of its final value:

Solving for n:

**Examples:**

|  | **Samples to 90% ()** |
| --- | --- |
| 0.01 | 230 |
| 0.05 | 45 |
| 0.1 | 22 |
| 0.3 | 8 |
| 0.5 | 4.3 |
| 0.9 | 1.05 |

**5. When to Use Each** **Exact Expression:**

* Use when precise control of cutoff is required
* Recommended for analytical work or design specifications where exact cutoff mapping is critical

**Approximation:**

* Sufficient for most practical filter implementations
* Easier to compute, especially when
* Useful when deriving from a desired analog cutoff frequency

**6. Conclusion** The relationship between α and the 3 dB cutoff frequency is fundamental to the effective use of exponential smoothing filters. Both the exact and approximate expressions have their place, depending on the required precision and computational constraints.

**Appendix: Common Approximate Values**

| α | **Approx** |
| --- | --- |
| 0.01 | 0.063 |
| 0.1 | 0.628 |
| 0.5 | 1.81 |
| 0.9 | 2.81 |
| 0.99 | 3.08 |

**5. Equivalent Noise Bandwidth (ENBW)** The Equivalent Noise Bandwidth (ENBW), also called represents the bandwidth of an ideal rectangular filter that passes the same noise power as the actual filter. For the exponential filter:

This is derived by integrating the squared frequency response over the Nyquist interval.

6. Comparison Between and

* Both are bandwidth measures, but serve different purposes:
  + : frequency at which the gain drops to -3 dB
  + : area-equivalent bandwidth for noise
* For small , both increase approximately linearly with
* < for small , but they converge for to 1

**Appendix: Common Approximate Values**

|  | **Approx. (rad/sample)** |
| --- | --- |
| 0.01 | 0.063 |
| 0.1 | 0.628 |
| 0.5 | 1.81 |
| 0.9 | 2.81 |
| 0.99 | 3.08 |

# Frequency Response of Brown's Double Exponential Smoothing Filter

Brown’s Double Exponential Smoothing filter is a two-stage filter designed to capture both the level and the trend in a time series. Unlike a simple exponential filter (single smoothing), it applies exponential smoothing twice, enabling it to adapt to linear trends more effectively.

## Filter Definition

The filter operates in two stages:  
 • First smoothing: S₁(t) = α x(t) + (1 - α) S₁(t-1)  
 • Second smoothing: S₂(t) = α S₁(t) + (1 - α) S₂(t-1)  
The forecast is computed using: x̂(t+1) = 2 S₁(t) - S₂(t)

## Transfer Function (Z-domain)

The Z-domain transfer function of the second smoother is:  
 H(z) = α² / (1 - (1 - α) z⁻¹)²  
This is equivalent to a second-order IIR low-pass filter.

## Frequency Response

To understand the filter's behavior in the frequency domain, we evaluate the transfer function on the unit circle:  
 z = e^{jω}, ω ∈ [0, π]  
The magnitude response becomes:  
 |H(e^{jω})| = α² / [1 - 2(1 - α)cos(ω) + (1 - α)²]

This formula shows that low frequencies (ω ≈ 0) pass through with near-unit gain, while high frequencies (ω ≈ π) are increasingly attenuated. The smaller the α, the stronger the smoothing and the steeper the attenuation.

## Slope and Prediction Insight

The difference between the first and second smoothing stages, S₁(t) - S₂(t), represents the recent trend or slope. This difference is scaled by α / (1 - α) to estimate the slope, allowing the filter to make a one-step forecast that incorporates both the level and trend of the signal:  
 Trend ≈ (α / (1 - α)) \* (S₁ - S₂)

This trend-following behavior is particularly useful in forecasting applications where the underlying data is expected to follow a consistent linear trajectory in the short term.

Figure: Frequency response (in dB) of Brown’s double exponential filter for various α values.

# Relation Between Laplace and Z Transform in Exponential Filters

This document explains how the Laplace-domain representation of an exponential filter relates to its discrete-time Z-transform equivalent, particularly in the context of Brown's exponential smoothing.

## Continuous-Time (Laplace Domain)

A first-order low-pass filter in the Laplace domain is represented as:  
 H(s) = 1 / (τs + 1)  
Where:  
 • τ is the time constant,  
 • s is the Laplace transform variable.

## Discrete-Time (Z Domain)

The equivalent Z-domain representation of a single exponential smoother is:  
 H(z) = α / [1 - (1 - α) z⁻¹]  
Where:  
 • α ∈ (0, 1] is the smoothing factor,  
 • (1 - α) = β is the decay factor (memory).

## Mapping Between Domains

The approximate transformation from Laplace to Z-domain using backward Euler is:  
 s ≈ (1 - z⁻¹) / T  
Substituting into the Laplace transfer function yields:  
 H(z) ≈ 1 / [ (τ/T)(1 - z⁻¹) + 1 ]  
Letting α = T / (τ + T), this simplifies to:  
 H(z) = α / [1 - (1 - α) z⁻¹]

## Parameter Equivalence

To relate α and τ directly:  
 α = T / (τ + T)  
 τ = T \* (1 - α) / α  
Thus, larger τ means slower response and smaller α (heavier smoothing).  
Smaller τ gives faster response and larger α (less smoothing).

## Interpretation of β

The term β = 1 - α represents the filter's memory. A larger β means the filter gives more weight to older samples, resulting in stronger smoothing.

## Example: Frequency Response

The graph below shows the frequency response of the Brown double exponential filter in dB for several α values. It demonstrates that smaller α values result in stronger attenuation of high-frequency components.

A graph of different colored lines

AI-generated content may be incorrect.

Figure: Frequency response (in dB) of Brown’s double exponential filter for various α values.

Generated by ChatGPT · OpenAI

**Brown Filter Response for the trend measurement:**

A screenshot of a computer

AI-generated content may be incorrect.

**Brown Filter Response for the Amplitude measurement:**

****

**Brown Filter Response for the Amplitude with no trend component:**

A screenshot of a computer

AI-generated content may be incorrect.

**Equivalent model for a derivative Brown Filter Response:**

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

**Frequency Response of PLL Equivalent digital Filter**

**What is the PLL Digital Frequency?**

**First let define a signal *x*(*t*) with frequency *fo* signal period *To*. The**

**digital sequence *x*[*k*] is sampled at discrete times = , where is the sampling period.**

**Now the signal can be written as:**

***x*(k) = *A* cos(ω+ *φ*) = *A* cos(ω+ *φ*)**

**If we define a new parameters called the digital frequency we can now write the above equation as:**

***x*(k) =*A* cos(+ *φ*) = *A* cos(+ *φ*)**

**In other word, we are simply normalizing the frequency of the physical signal to how fast it was sampled. Note also that the digital frequency Ω is the angular frequency (rad/s) times the sampling period (s/sample), so the units of digital frequency are rad/sample.**

**As a concept one major reason for using Ω instead of is that signal processing techniques fundamentally take in a sequence of numbers *x*[*n*] and operate on them in the same way regardless of the sampling period. By generalizing the sequences to the parameters k the time become an integer index and we can generalize signal processing techniques from that.**

**Going back to the equation of a second order ADPLL as presented in**

A close-up of a computer screen

AI-generated content may be incorrect.

We have the following transfert fonction et parameters defined as:

A math equations and formulas

AI-generated content may be incorrect.

Another aspect to consider for a ADPLL is that there is three frequencies of concern for system performance consideration. We have the resonance frequency , which appear directly as a parameters in the transfer function H(s), the 3 dB bandwidth and the noise bandwidth equivalent related to the measure in the energy of the noise at the output relative to a perfect windows type bandwidth this parameter (NEB) can be called Bn but it is generally better to define it as as to distinguish it from the parameter .The relation for or can be shown to be dependent of and the damping factor ζ as:

A mathematical equation with numbers and symbols

AI-generated content may be incorrect.

**Or**

=

The reference from the equation above is R. Best, Phase Locked Loops - Design, Simulation and Applications (6th Edition), McGraw Hill, 2007

Also the the 3dB bandwidth can be approximated with this formula extractr from a visual graphic interpretation from ChatGPt:

# 3 dB Bandwidth vs. Damping in Second-Order ADPLL

This document presents an improved model for estimating the 3 dB bandwidth of a second-order analog or digital phase-locked loop (ADPLL) system as a function of the damping factor ζ.

## Measured Bandwidth Data

Measured 3 dB bandwidths (normalized by natural frequency fn):

|  |  |
| --- | --- |
| Damping Factor (ζ) | 3 dB Bandwidth / fn |
| 0.70 | 2.04 |
| 1.00 | 2.46 |
| 1.25 | 2.86 |
| 1.50 | 3.28 |

## Improved Analytical Fit

Rather than relying on a linear approximation, the following analytical model provides a better match:

f₃dB ≈ 2 · fₙ · (ζ + 1 / (4ζ))

This expression reflects the natural shape of the transfer function and improves accuracy across a wider range of damping factors compared to simple linear regression.

## Comparison Graph

The following graph compares the measured 3 dB bandwidths to values predicted by the improved formula:

A graph with a red line

AI-generated content may be incorrect.

Generated by ChatGPT · OpenAI

**Frequency Response Curves**

**This figure shows the frequency response of the second-order ADPLL system for different damping factors ζ. The 3 dB points are visually confirmed in each case.**

**A diagram of a frequency response

AI-generated content may be incorrect.**

# Estimating PLL Parameters from Unit Step Response

This sextion explains how to estimate the parameters of a second-order system or phase-locked loop (PLL) based on its response to a unit step input. This method is useful for validating filter performance or estimating internal settings from external observations.

## Applicable System

The method applies to systems with the following transfer function:  
  
 H(s) = (2ζωₙs + ωₙ²) / (s² + 2ζωₙs + ωₙ²)  
  
This represents the output of a PLL just after the integrator. It is the 'velocity-type' output of a second-order system that can track changing input related to speed.

## Step Response Features to Measure

You can extract the following key features from the unit step response:

* • Peak Value:
* • Final Value:
* • Overshoot:
* • Time to Peak: when the peak occurs

## Formulas to Estimate Parameters

Estimate the damping factor (ζ) from overshoot:

ζ ≈ -ln() / sqrt(π² + ln²())

Estimate the natural frequency (ωₙ) from time to peak:

ωₙ ≈ π / ( \* sqrt(1 - ζ²))

If needed, convert ωₙ to frequency (Hz):

fₙ = ωₙ / (2π)

## Example

Measured response:  
• Overshoot ≈ 16%  
• Time to Peak ≈ 6 seconds  
  
→ ζ ≈ 0.503  
→ ωₙ ≈ 0.53 rad/s  
→ fₙ ≈ 0.085 Hz

## Usage in Code

You can automate this analysis in VB.NET by:  
• Collecting step response data  
• Computing overshoot and time to peak  
• Applying the above formulas  
  
This is useful for validating PLL behavior in live systems or during development.

Generated by ChatGPT · OpenAI