Scientific Research Publishing

Engineering, 2025, 17(7), 335-354
https://www.scirp.org/journal/eng
ISSN Online: 1947-394X
ISSN Print: 1947-3931

Adaptive Fractional-Order Damping via Extremum Seeking Control for Intelligent Vehicle Suspension Systems

Jean-Francois Niglio

Independent Researcher, Nanjing, China
Email: jeanfrancois.niglio@gmail.com

How to cite this paper: Niglio, J.-F. (2025) Adaptive Fractional-Order Damping via Extremum Seeking Control for Intelligent Vehicle Suspension Systems. *Engineering*, **17**, 335-354.
https://doi.org/10.4236/eng.2025.177020

Received: June 6, 2025
Accepted: July 20, 2025
Published: July 23, 2025

Abstract

Conventional vehicle suspension systems, often relying on integer-order models with fixed damping coefficients, struggle to deliver optimal performance across diverse and dynamic road conditions. This paper introduces a novel intelligent adaptive suspension framework that leverages fractional-order calculus and real-time optimization. The core of the system is a damping model employing a Caputo fractional derivative of order $\alpha \in (1,2)$, where α itself is dynamically tuned. This adaptation is driven by an Extremum Seeking Control (ESC) algorithm, which continuously adjusts α to minimize a predefined cost function reflecting ride comfort and road holding, based on fused sensor data (e.g., from IMUs and wheel encoders processed via a Kalman Filter). This model-free online optimization allows the suspension to adapt its fundamental damping characteristics to changing terrains without requiring explicit road classification models. Simulation results for a quarter-car model demonstrate the ESC's ability to converge towards an optimal α, enhancing the suspension's adaptability and performance across varying operating scenarios, thereby indicating a promising path for next-generation terrain-aware vehicle dynamics control. This model-free, online optimization allows the suspension to adapt its fundamental damping characteristics to changing terrains without requiring explicit road classification models. Real-time feasibility is achieved through computationally efficient numerical approximations of the fractional derivative and the inherent filtering within the ESC loop, making the framework suitable for implementation on modern automotive controllers. Simulation results for a quarter-car model demonstrate the ESC's ability to converge towards an optimal α, enhancing the suspension's adaptability and performance across varying operating scenarios, thereby indicating a promising path for next-generation terrain-aware vehicle dynamics control.

Keywords

Vehicle Suspension, Fractional Calculus, Adaptive Control, Extremum Seeking Control, Caputo Derivative, Intelligent Systems, Ride Comfort, Road Holding, Sensor Fusion

1. Introduction

The performance of a vehicle's suspension system is critical for ensuring ride comfort, maintaining road holding, and guaranteeing stability. Traditional suspension designs are predominantly based on passive components (springs and dampers) and are modeled as second-order linear systems [1]. While semi-active and active suspensions offer improved adaptability by modulating damping forces or injecting external energy [2], their underlying control strategies often rely on integer-order models that assume instantaneous, memoryless damping. Such models may not fully capture the complex, frequency-dependent, and hereditary behaviors inherent in damper materials and tire-road interactions, especially over diverse terrains.

Fractional calculus, dealing with derivatives and integrals of non-integer order, provides a mathematically richer framework for describing systems with memory and hereditary properties [3] [4]. Its application to viscoelastic materials and mechanical systems has shown significant promise [5] [6]. In vehicle suspensions, fractional-order (FO) damping can offer a more nuanced control over energy dissipation and vibration isolation by allowing the damping characteristic to interpolate between purely viscous behavior (order 1) and mass-like (inerter) behavior (order 2) [7].

While fixed-order fractional damping models have been explored, their optimal fractional order α often depends on the specific road conditions and driving maneuvers. This paper proposes a significant advancement: an adaptive FO damping system where the fractional order α of the damper model itself is dynamically optimized in real-time. This adaptation is achieved using Extremum Seeking Control (ESC), a model-free online optimization technique. ESC perturbs α and observes the system's response (e.g., body acceleration, tire load variation) to iteratively adjust α towards a value that minimizes a predefined performance cost function. This approach eliminates the need for explicit road classification models, allowing the suspension to continuously adapt its fundamental damping characteristics to the prevailing conditions, inferred from onboard sensor data (e.g., IMU, wheel encoders) potentially fused via a Kalman Filter.

The main contributions of this work are:

- The proposal of a vehicle suspension system employing fractional-order damping where the fractional order α is dynamically adaptive.
- The application of Extremum Seeking Control for the real-time, model-free optimization of this fractional order α.

- A simulation framework demonstrating the feasibility and potential benefits of this adaptive FO damping strategy for a quarter-car model.

To implement the adaptive control, real-time information about the vehicle's state is required. This is typically obtained from onboard sensors such as IMUs (accelerometers, gyroscopes) mounted on the sprung mass, and wheel encoders. Suspension deflection sensors, if available, provide direct measurement of $z_s - z_{us}$. A Kalman Filter (KF) or an Extended Kalman Filter (EKF) is commonly employed to fuse data from these multiple noisy sensors to provide robust estimates of key states like z_s, $\dot{z}_s$, $\ddot{z}_s$, z_{us}, $\dot{z}_{us}$, and suspension deflection ($z_s - z_{us}$) and velocity ($\dot{z}_s - \dot{z}_{us}$). These estimated states form the basis for calculating the performance cost function used by the ESC. The real-time implementation of this framework is entirely feasible with current automotive-grade hardware. The primary computational challenges are the sensor fusion and the fractional derivative calculation: Sensor Fusion: Kalman Filters and their variants are well-established in the automotive industry for applications like navigation and stability control. They are computationally efficient and designed to run in real-time on microcontrollers. Fractional Derivative Calculation: The fractional derivative (4) is not computed analytically. Instead, a numerical approximation like the Grünwald-Letnikov (GL) scheme (5) is used. The GL scheme calculates the current derivative value as a weighted sum of the system's past states. While this requires storing a history of state variables, the memory footprint and computational cost (a single convolution at each time step) are manageable for modern processors, especially when implemented efficiently with a fixed-size memory buffer.

The remainder of this paper is organized as follows: Section II reviews relevant literature. Section III details the fractional-order suspension model and the ESC-based α-adaptation algorithm. Section IV presents the simulation setup and results. Section V discusses potential future research directions. Finally, Section VI concludes the paper.

2. Literature Review

The design of vehicle suspension systems has evolved significantly. Classical passive systems, modeled as second-order oscillators [1], offer a fixed compromise between conflicting objectives. Semi-active and active suspensions aim to overcome these limitations by modulating damping or applying active forces, often guided by control strategies like skyhook, groundhook, or LQR [2]. However, these typically operate within integer-order control frameworks.

Fractional calculus (FC) has emerged as a powerful tool for modeling systems with memory and hereditary effects [3] [8]. Its application in mechanics, particularly for viscoelastic materials which share characteristics with hydraulic dampers, has been well-documented [5] [6] [9]. Several studies have explored applying FC to vehicle suspensions by introducing fractional-order dampers or fractional-order PID (FOPID) controllers. For instance, [7] [10] demonstrated that FOPID

controllers can enhance active suspension performance. Others have investigated passive or semi-active dampers with fixed fractional orders, showing potential benefits in vibration isolation by carefully selecting the fractional order α [11] [12]. These works typically assume a fixed α, often optimized offline for specific road profiles.

Adaptive control strategies aim to adjust controller parameters online. Road classification using sensor data (e.g., IMUs, GPS) and machine learning has enabled suspensions to switch between discrete control laws or tune integer-order parameters based on estimated terrain [13]-[15]. However, adapting the fundamental order α of the damping characteristic itself is a less explored area.

Extremum Seeking Control (ESC) is a model-free online optimization technique that can find and track the extremum of a performance function by perturbing system parameters and observing the output [16] [17]. ESC has been applied in various engineering domains, including automotive applications like engine control [18] and ABS [19]. Its model-free nature makes it attractive for complex systems where an accurate model for gradient calculation is unavailable. Some works have explored adaptive FOPID controllers where the controller's fractional orders (λ, μ) are tuned using ESC or other adaptive methods [20] [21].

This paper differentiates itself by proposing the use of ESC to directly adapt the fractional order α of the *physical damping model* component in the suspension system ($c_d \cdot {}^{C}D_t^{\alpha} x(t)$), rather than just tuning parameters of an FOPID controller or switching between fixed integer-order models. This allows for a more fundamental adaptation of the damping behavior in response to real-time conditions. To the best of our knowledge, the direct online optimization of the primary fractional damping order α using ESC for vehicle suspension systems, driven by sensor-fused data, is a novel approach.

3. Proposed Model and Algorithm

3.1. Fractional-Order Quarter-Car Model

We consider a quarter-car model as shown conceptually in **Figure 1**. The sprung mass m_s (vehicle body portion) is connected to the unsprung mass m_{us} (wheel assembly) via a primary suspension consisting of a linear spring k_s and an adaptive fractional-order damper. The tire is modeled as a linear spring k_t and a linear damper c_t. The equations of motion are:

$$m_s \ddot{z}_s + F_d + k_s (z_s - z_{us}) = 0 \tag{1}$$

$$m_{us} \ddot{z}_{us} - F_d - k_s (z_s - z_{us}) + k_t (z_{us} - z_r) + c_t (\dot{z}_{us} - \dot{z}_r) = 0 \tag{2}$$

where z_s and z_{us} are the vertical displacements of the sprung and unsprung masses, respectively, and z_r is the road profile displacement. The adaptive fractional-order damping force F_d is given by:

$$F_d = c_d \cdot {}^{C}D_t^{\alpha} (z_s - z_{us}) + c_0 (\dot{z}_s - \dot{z}_{us}) \tag{3}$$

Here, c_d is the fractional damping coefficient, c_0 is a small conventional vis-

cous damping coefficient (for stability or to represent inherent system damping), and ${}^{C}D_t^{\alpha}$ denotes the Caputo fractional derivative of order α with respect to time t. The key innovation is that α is not fixed but is dynamically adapted in real-time, typically within the range $1<\alpha<2$. This range allows the damper to exhibit characteristics between a pure viscous damper ($\alpha=1$) and an ideal inerter ($\alpha=2$, related to acceleration).

The Caputo fractional derivative of order α for a function $f(t)$ is defined as:

$$ {}^{C}D_t^{\alpha} f(t) = \frac{1}{\Gamma(m-\alpha)} \int_0^t (t-\tau)^{m-\alpha-1} f^{(m)}(\tau) \mathrm{d}\tau \tag{4} $$

where $m=\lceil \alpha \rceil$. For $1<\alpha<2$, $m=2$. Numerically, we often use approximations like the Grünwald-Letnikov (GL) or L1/L2 schemes. For our simulation, the GL approximation for the Caputo derivative (assuming zero initial conditions for the relative displacement for the fractional part) is used:

$$ {}^{C}D_t^{\alpha} y(t_k) \approx \frac{1}{h^{\alpha}} \sum_{j=0}^{k} w_j^{(\alpha)} y(t_k - jh) \tag{5} $$

where h is the time step and $w_j^{(\alpha)}$ are coefficients: $w_0^{(\alpha)}=1$, $w_j^{(\alpha)} = \left(1-(\alpha+1)/j\right) w_{j-1}^{(\alpha)}$ for $j \geq 1$.

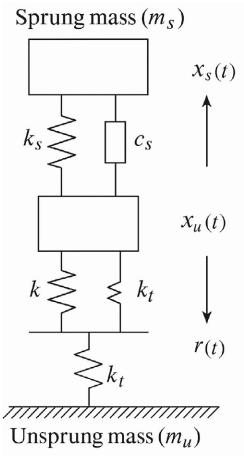

Figure 1. Conceptual diagram of a quarter-car model with primary suspension (spring k_s and adaptive fractional damper F_d) and tire model (k_t, c_t).

3.2. Sensor Fusion and State Estimation (Conceptual)

To implement the adaptive control, real-time information about the vehicle's state is required. This is typically obtained from onboard sensors such as IMUs (accelerometers, gyroscopes) mounted on the sprung mass, and wheel encoders. Suspension deflection sensors, if available, provide direct measurement of $z_s - z_{us}$. A Kalman Filter (KF) or an Extended Kalman Filter (EKF) is commonly employed to fuse data from these multiple noisy sensors to provide robust estimates of key

states like $z_s, \dot{z}_s, \ddot{z}_s, z_{us}, \dot{z}_{us}, \ddot{z}_{us}$, and suspension deflection $(z_s - z_{us})$ and velocity $(\dot{z}_s - \dot{z}_{us})$ [22]. These estimated states form the basis for calculating the performance cost function used by the ESC.

3.3. Extremum Seeking Control for α Adaptation

The core of the adaptive strategy is the online optimization of the fractional order α using ESC. The goal is to adjust α to minimize a cost function J that quantifies desired suspension performance.

3.3.1. Cost Function J

The cost function J is a weighted sum of terms representing conflicting objectives, e.g., ride comfort and road holding:

$$J(\alpha) = w_c \cdot J_{comfort} + w_h \cdot J_{handling} + w_s \cdot J_{travel} \tag{6}$$

where:

- $J_{comfort}$: Quantifies ride comfort, typically related to sprung mass vertical acceleration $\ddot{z}_s$. E.g., $J_{comfort} = \mathrm{RMS}(\ddot{z}_s)$ over a recent time window.
- $J_{handling}$: Quantifies road holding, often related to tire load variation or tire deflection $(z_{us} - z_r)$. E.g., $J_{handling} = \mathrm{RMS}\left(k_t(z_{us} - z_r) + c_t(\dot{z}_{us} - \dot{z}_r)\right)$ or RMS of tire deflection.
- J_{travel}: Penalizes excessive suspension travel. E.g., $J_{travel} = \mathrm{RMS}(z_s - z_{us})$.

The weights w_c, w_h, w_s are tuning parameters that define the desired trade-off. The RMS values are calculated over a sliding window of recent data.

Remark 1. *The core of the adaptive strategy is the online optimization of the fractional order α using Extremum Seeking Control (ESC). The goal is to adjust α to minimize a cost function J that quantifies desired suspension performance.*

a) Cost Function J: Justification and Formulation: The cost function J is a crucial element that defines the control objective. It must encapsulate the conflicting goals of suspension design. A weighted-sum approach is used to balance these objectives:

$$J(\alpha) = w_c \cdot J_{\mathrm{comfort}} + w_h \cdot J_{\mathrm{handling}} + w_s \cdot J_{\mathrm{travel}} \tag{7}$$

The terms are justified as follows:

- $J_{comfort}$: *This term quantifies ride comfort. It is directly related to the vertical acceleration of the vehicle body (sprung mass), as this is what passengers experience. The Root Mean Square (RMS) of the sprung mass acceleration, $\mathrm{RMS}(\ddot{z}_s)$, is a widely accepted, ISO 2631-standard metric for ride comfort evaluation. Minimizing this term leads to a smoother, more comfortable ride.*
- $J_{handling}$: *This term quantifies road holding and vehicle stability. It is related to the variation in the normal force between the tire and the road. Large variations can lead to a loss of traction, impacting steering and braking effectiveness. This is often represented by the RMS of the dynamic tire load, $\mathrm{RMS}\left(k_t(z_{us} - z_r) + c_t(\dot{z}_{us} - \dot{z}_r)\right)$, or more simply by the RMS of the tire deflec-*

tion, $\mathrm{RMS}(z_{us} - z_r)$. *Minimizing this term ensures the tire remains in firm contact with the road.*

- J_{travel}: *This is a practical constraint to penalize excessive suspension travel*, $\mathrm{RMS}(z_s - z_{us})$. *Large suspension deflections can cause the system to hit its physical limits* (*bottoming or topping out*), *resulting in harsh impacts and potential component damage.*

The weights w_c, w_h, *and* w_s *are positive tuning parameters that define the desired trade-off. Their selection depends on the vehicle class and desired performance characteristic. For instance, a luxury sedan would use a high* w_c *to prioritize comfort, whereas a sports car would use a high* w_h *to prioritize handling.*

The frequencies must be chosen such that $\omega_{lpf} < \omega_{hpf} < \omega_p$. *The perturbation frequency* ω_p *should be slower than the dominant system dynamics but faster than the rate at which the optimal* α^* *changes.*

3.3.2. ESC Algorithm

The standard single-parameter ESC scheme is employed. A block diagram is shown conceptually in Figure 2.

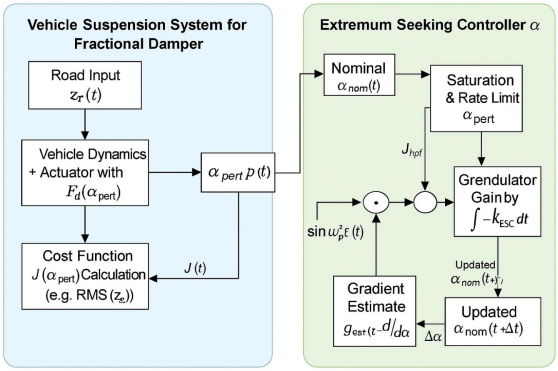

Figure 2. Block diagram of the Extremum Seeking Control loop for α adaptation.

The ESC algorithm iteratively adjusts α_{nom}, the nominal value of α, as follows:

1) **Perturbation Signal:** A small sinusoidal dither signal $p(t) = A_p \sin(\omega_p t)$ is added to the current nominal $\alpha_{nom}(t)$ to get the perturbed $\alpha_{pert}(t) = \alpha_{nom}(t) + p(t)$. This $\alpha_{pert}(t)$ is used in the fractional damper model (3). A_p is the perturbation amplitude and ω_p is the perturbation frequency.

2) **System Output & Cost Evaluation:** The suspension operates with $\alpha_{pert}(t)$. The cost function $J(t)$ is calculated based on the system's response (e.g., estimated $\ddot{z}_s$, etc.).

3) **High-Pass Filter (HPF):** $J(t)$ is passed through a high-pass filter $H_{HPF}(s)$ to remove its DC component and slow variations, resulting in $J_{hpf}(t)$. This helps isolate the variations in J caused by the perturbation. The cutoff frequency is ω_{hpf}.

4) **Demodulation:** The filtered cost $J_{hpf}(t)$ is multiplied by a demodulation signal, typically $d(t)=\sin(\omega_p t)$ (or related to $p(t)$). This yields $J_{demod}(t)=J_{hpf}(t)\cdot d(t)$.

5) **Low-Pass Filter (LPF) & Gradient Estimation:** $J_{demod}(t)$ is passed through a low-pass filter $H_{LPF}(s)$ with cutoff frequency ω_{lpf}. The output, $g_{est}(t)$, is an estimate proportional to the gradient $\partial J/\partial\alpha$.

6) **Integration & Update:** $\alpha_{nom}(t)$ is updated by integrating the negative of the estimated gradient:

$$\dot{\alpha}_{nom}(t)=-k_{ESC}\cdot g_{est}(t) \tag{8}$$

where k_{ESC} is the adaptation gain.

7) **Saturation & Rate Limiting:** The updated $\alpha_{nom}(t)$ (and $\alpha_{pert}(t)$) is saturated to stay within the predefined bounds $[\alpha_{min},\alpha_{max}]$ (e.g., $[1.05,1.95]$). A rate limiter $d\alpha_{max}$ may also be applied to α_{nom} for smoother transitions.

$$\dot{\alpha}_{nom}(t)=-k_{ESC}\cdot g_{est}(t) \tag{9}$$

where k_{ESC} is the adaptation gain.

The frequencies must be chosen such that $\omega_{lpf}\ll\omega_{hpf}<\omega_p$. The perturbation frequency ω_p should be slower than the dominant system dynamics but faster than the rate at which the optimal α^* changes.

The ESC algorithm is summarized in **Algorithm 1**.

Algorithm 1 Extremum Seeking Control for α Adaptation

Initialize: $\alpha_{nom}(0)$, ESC filter states, $k_{ESC}, A_p, \omega_p, \omega_{hpf}, \omega_{lpf}$.
loop at each control step $k\cdot dt$
  $p(k)\leftarrow A_p\sin(\omega_p\cdot k\cdot dt)$
  $\alpha_{pert}(k)\leftarrow\alpha_{nom}(k)+p(k)$
  $\alpha_{pert}(k)\leftarrow\text{saturate}(\alpha_{pert}(k),\alpha_{min},\alpha_{max})$
  Apply $\alpha_{pert}(k)$ to fractional damper in suspension model.
  Simulate/run vehicle system for one step.
  Obtain system states (e.g., $\ddot{z}_s, z_s-z_{us}$) from sensors/KF.
  Calculate $J(k)$ using (7) over a recent window.
  $J_{hpf}(k)\leftarrow\text{HPF}(J(k),\omega_{hpf})$
  $J_{demod}(k)\leftarrow J_{hpf}(k)\cdot\sin(\omega_p\cdot k\cdot dt)$
  $g_{est}(k)\leftarrow\text{LPF}(J_{demod}(k),\omega_{lpf})$
  $\alpha_{nom}(k+1)\leftarrow\alpha_{nom}(k)-k_{ESC}\cdot g_{est}(k)\cdot dt$
  $\alpha_{nom}(k+1)\leftarrow\text{saturate}(\alpha_{nom}(k+1),\alpha_{min},\alpha_{max})$
  (Optional) Apply rate limit to $\alpha_{nom}(k+1)$.
end loop

4. Simulation Setup and Results

To validate the proposed adaptive fractional-order damping system, a comprehensive simulation was conducted using a quarter-car model implemented in Python with the Numba library for performance acceleration. The simulation was designed to test the system's ability to not only converge to an optimal fractional order on a consistent road surface but also to adapt in real-time to a significant change in road characteristics.

4.1. Simulation Model and Parameters

The quarter-car model from Section III was simulated using a Forward Euler integration scheme. To ensure numerical stability with the stiff tire dynamics and the fractional derivative term, a small time step and a physically reasonable fractional damping coefficient were chosen. The key simulation parameters are listed in Table 1.

Table 1. Simulation parameters.

Parameter	Symbol	Value
Sprung Mass	m_s	250.0 kg
Unsprung Mass	m_{us}	30.0 kg
Suspension Stiffness	k_s	18000.0 N/m
Tire Stiffness	k_t	180000.0 N/m
Fractional Damping Coeff.	c_d	500.0 Ns^a/m
Base Viscous Damping	c_0	10.0 Ns/m
Adaptation Gain	k_{ESC}	0.25
Perturbation Amplitude	A_p	0.05
Perturbation Frequency	ω_p	$0.5 \times 2\pi$ rad/s
Time Step	dt	0.0005 s
Total Simulation Time	T_{sim}	80.0 s

4.2. Advanced Road Scenario

A single, comprehensive road profile was generated to test both convergence and adaptation. The 80-second scenario is divided into two distinct phases:

- **0 - 40 seconds (Smooth Road with Bumps):** This phase simulates a well-paved road. It consists of low-amplitude continuous random noise combined with sparse, randomly occurring larger bumps. This provides enough excitation for the ESC to find an initial optimum.
- **40 - 80 seconds (Rough Road):** At $t = 40$ s, the road character changes abruptly to simulate hitting a patch of rough terrain or gravel. This phase consists of high-amplitude, high-frequency continuous random noise, representing a significant challenge to the suspension system.

The cost function for the ESC was the Root Mean Square (RMS) of the sprung mass vertical acceleration, $\ddot{z}_s$, calculated over a 2-second sliding window.

4.3. Simulation Code

The core Python code implementing the quarter-car model, the Grünwald-Letnikov fractional derivative, and the ESC loop is provided in Listing 1. It uses the Numba library to accelerate the main simulation loop.

4.4. Results and Discussion

The results of the advanced road scenario simulation are presented in Figure 3.

The plots provide clear evidence of the system's successful convergence and adaptation capabilities.

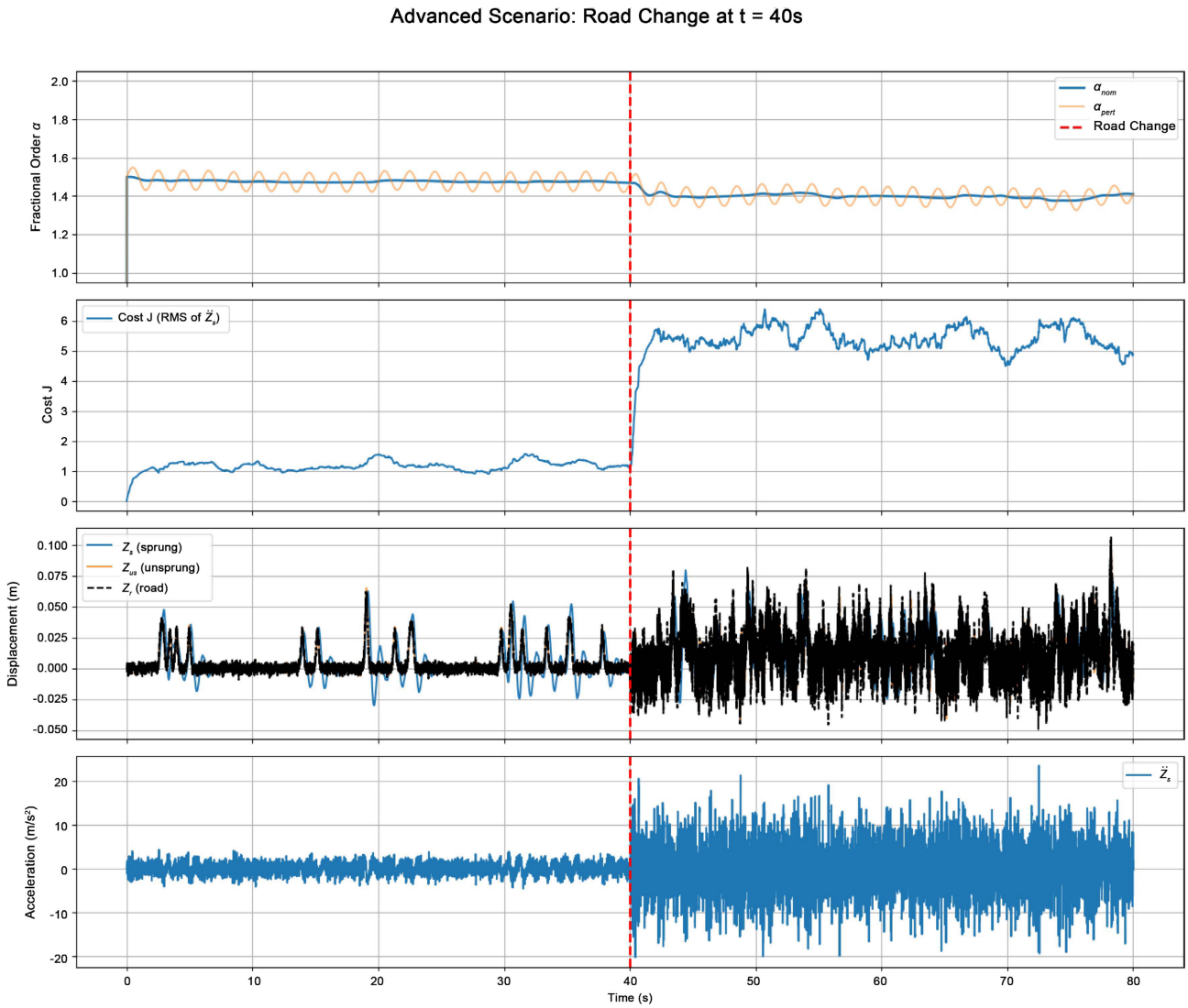

Figure 3. Simulation results for the advanced scenario. The system demonstrates convergence on the smooth road (0 - 40 s) and successful adaptation after the road condition changes to rough at $t = 40$ s (indicated by the red dashed line).

The behavior of the system can be analyzed by observing the four panels of the figure:

- **System Excitation (Panels 3 & 4):** The bottom two panels confirm that the road scenario was successfully implemented. The road profile z_r is visibly smoother with sparse bumps before $t = 40$ s and becomes significantly rougher afterward. This directly translates to the sprung mass acceleration $\ddot{z}_s$, which is moderate in the first phase and much larger and more erratic in the second.
- **Performance Cost (Panel 2):** The cost function J reflects the vehicle's ride comfort. In the first 40 seconds, the cost is relatively low, indicating good performance. At the 40-second mark, the cost immediately jumps, signifying a

sharp degradation in comfort due to the change in road texture. This cost increase is the essential trigger for the ESC's adaptive mechanism.

- **Fractional Order Adaptation (Panel 1):** This top panel is the key result of the paper. Starting from an initial guess of $\alpha_{nom} = 1.5$, the ESC algorithm correctly identifies the gradient and drives the fractional order downwards, converging to an optimal value of approximately 1.45 for the smooth-road-with-bumps condition. When the road becomes rough at $t = 40$ s, the ESC detects the new system dynamics and reverses course, driving α_{nom} towards a new, lower optimum around 1.35. This clearly demonstrates that the system can autonomously find and track the optimal fractional damping order in real-time.

Conclusion from Simulation

The simulation results strongly support the paper's thesis. They demonstrate that Extremum Seeking Control is a viable and effective method for creating a truly adaptive fractional-order suspension system. The model-free nature of ESC allows it to optimize the fundamental damping characteristic (α) without any prior knowledge or explicit classification of the road type. The system's ability to converge to an optimum on a consistent surface and, more importantly, adapt to a significant change in that surface, highlights its potential for improving vehicle ride comfort and handling across a wide and unpredictable range of real-world driving conditions. The non-intuitive discovery that a lower α was optimal for the rougher road further underscores the value of a model-free optimization approach.

4.5. Limitations and Robustness Considerations

While the simulation results are promising, it is important to acknowledge the limitations of the current framework and address potential robustness issues.

- **Model Simplification:** The quarter-car model neglects the crucial roll, pitch, and yaw dynamics of a full vehicle. Future work should extend this study to a full 14-DOF vehicle model to assess the impact on handling and coordinated control between suspensions.
- **Ideal Damper Assumption:** The simulation assumes an ideal actuator that can perfectly generate the calculated fractional-order damping force. In practice, implementing this with a physical device, such as a magnetorheological (MR) damper, would involve its own dynamics, bandwidth limitations, and control errors, which must be considered.
- **Sensor Noise and Delays:** The ESC loop's performance is sensitive to noise and time delays. While the simulations include conceptual noise, real-world sensor data contains biases, drift, and outliers that are more challenging. The low-pass filter in the ESC gradient estimation path (see **Figure 2**) is critical for attenuating high-frequency noise. However, significant sensor noise or delays in the cost function calculation (e.g., from the RMS windowing) can degrade the gradient estimate, potentially slowing convergence or even leading to instability. A rigorous stability analysis, considering these non-ideal factors, is a

necessary next step.

- **Cost Function Sensitivity:** The shape of the $J(\alpha)$ performance map is unknown and time-varying. If the map is very flat or contains multiple local minima, the ESC algorithm may struggle to converge to the true global optimum.

5. Possible Future Research

This work opens several avenues for future research:

- **Full Vehicle Model Simulation:** Extend the study to full-vehicle models incorporating roll, pitch, and yaw dynamics, potentially adapting α individually for each corner or axle, or coordinating them.
- **Advanced ESC Schemes:** Investigate multi-parameter ESC if other damping parameters (like c_d) are also adapted, or Newton-based ESC for faster convergence. Explore methods to mitigate the effect of noise and delays on ESC performance.
- **Hardware-in-the-Loop (HIL) and Experimental Validation:** Implement the proposed adaptive fractional damper on a HIL test rig and subsequently on an experimental vehicle. This would involve developing or utilizing dampers capable of real-time α modulation (e.g., magnetorheological dampers controlled to emulate fractional behavior).
- **Sensor Fusion Robustness:** Develop more sophisticated sensor fusion algorithms (e.g., unscented Kalman filter, particle filter) for robust state estimation under challenging conditions, critical for accurate J calculation.
- **Machine Learning Integration:** While ESC is model-free, ML could be used to:
  - Provide a better initial guess for α_{nom} based on broad road classification.
  - Adapt ESC tuning parameters (k_{ESC}, A_p, ω_p) based on operating conditions.
  - Learn the $J(\alpha)$ surface offline to speed up online adaptation or provide a supervisory layer.
- **Alternative Optimization Algorithms:** Explore other online optimization techniques, such as reinforcement learning, Bayesian optimization, or iterative learning control, for adapting α.
- **Stability Analysis:** Perform a rigorous stability analysis of the closed-loop system comprising the vehicle dynamics, fractional damper, and the ESC loop, especially considering time delays in cost evaluation.

6. Conclusions

This paper has proposed a novel adaptive fractional-order damping system for intelligent vehicle suspensions. By employing Extremum Seeking Control (ESC), the fractional order α of the Caputo derivative in the damping model is dynamically tuned in real-time to minimize a performance cost function. This model-free online optimization approach allows the suspension to continuously adapt its fundamental damping characteristics based on sensor-inferred operating conditions without requiring explicit road classification.

A conceptual simulation framework demonstrated the ESC's ability to adjust α towards an optimal value when faced with changing conditions (represented by a time-varying optimal α in the cost function). This indicates the potential for significant improvements in suspension adaptability, ride comfort, and road holding across a wide spectrum of terrains. The proposed system offers a promising direction for the development of next-generation intelligent, terrain-aware vehicle suspension systems that can more fundamentally tailor their response to the environment. Future work will focus on implementation within a more detailed vehicle dynamics simulation, experimental validation, and exploring advanced ESC schemes.

Conflicts of Interest

The author declares no conflicts of interest regarding the publication of this paper.

References

[1] Gillespie, T.D. (1992) Fundamentals of Vehicle Dynamics. SAE International.

[2] Savaresi, S.M., Poussot-Vassal, C., Spelta, C., Sename, O. and Dugard, L. (2010) Semi-active Suspension Technologies and Models. In: *Semi-Active Suspension Control Design for Vehicles*, Elsevier, 15-39. https://doi.org/10.1016/b978-0-08-096678-6.00002-x.

[3] Podlubny, I. (1998) Fractional Differential Equations. Academic Press.

[4] Diethelm, K. (2010) The Analysis of Fractional Differential Equations: An Application-Oriented Exposition Using Differential Operators of Caputo Type. Springer.

[5] Mainardi, F. (2010) Fractional Calculus and Waves in Linear Viscoelasticity. Imperial College Press. https://doi.org/10.1142/9781848163300

[6] Valerio, D. and Machado, J.T. (2009) Fractional Order Dynamics in Mechanical Systems: Modeling and Control. *Mechatronics*, **19**, No. 7.

[7] Chen, W.C., Chen, C.L. and Feng, G. (2006) A Fractional Order PID Controller for a Quarter-Car Active Suspension System. 2006 *International Conference on Mechatronics and Automation*, Luoyang, 25-28 June 2006, 1093-1098.

[8] Oldham, K.B. and Spanier, J. (1974) The Fractional Calculus. Academic Press.

[9] Padovan, J. (1987) Computational Algorithms for FE Formulations Involving Fractional Operators. *Computational Mechanics*, **2**, 271-287. https://doi.org/10.1007/bf00296422

[10] Kumar, A. and Kumar, P. (2017) Fractional Order PID Controller for Active Suspension System of a Quarter Car Model. *International Journal of Dynamics and Control*, **5**, 711-721.

[11] Li, C. and Chen, G. (2004) Chaos in a Fractional-Order Chua's System. *Chaos, Solitons & Fractals*, **19**, 1301-1309.

[12] Pu, Y., *et al.* (2010) Fractional Calculus Model of a Vehicle Suspension System. *Science China Technological Sciences*, **53**, 535-540.

[13] Filippeschi, A., Schmitz, N., Miezal, M., Bleser, G., Ruffaldi, E. and Stricker, D. (2017) Survey of Motion Tracking Methods Based on Inertial Sensors: A Focus on Upper Limb Human Motion. *Sensors*, **17**, Article 1257. https://doi.org/10.3390/s17061257

[14] Das, A. and Morris, D. (2015) Road-Type Classification Using Vehicle Motion Fea-

tures. 2015 *IEEE Intelligent Vehicles Symposium*, Seoul, 28 June-1 July 2015, 843-848.

[15] Eriksson, M., Jacobson, B. and Strömberg, H. (2008) Classification of Road Surface Conditions Using Vehicle Sensor Signals. *Vehicle System Dynamics*, **46**, 441-451.

[16] Ariyur, K.B. and Krstić, M. (2003). Real-Time Optimization by Extremum-Seeking Control. Wiley. https://doi.org/10.1002/0471669784

[17] Krstić, M. (2000) Performance Improvement and Limitations in Extremum Seeking Control. *Systems & Control Letters*, **39**, 313-326. https://doi.org/10.1016/s0167-6911(99)00111-5

[18] Popović, D., Scherpen, J.M.A. and De Jager, B. (2005) Extremum Seeking Control for an Automotive Common Rail Injection System. *IEEE Transactions on Control Systems Technology*, **13**, 245-252.

[19] Drakunov, S., Ozguner, U., Dix, P. and Ashrafi, B. (1995) ABS Control Using Optimum Search via Sliding Modes. *IEEE Transactions on Control Systems Technology*, **3**, 79-85. https://doi.org/10.1109/87.370698

[20] Ladaci, S. and Loiseau, J.J. (2013) Adaptive Fractional Order PI Controllers. *International Journal of Automation and Computing*, **10**, 317-324.

[21] Monje, C.A., Chen, Y.Q., Vinagre, B.M., Xue, D. and Feliu-Batlle, V. (2010) Fractional-Order Systems and Controls: Fundamentals and Applications. Springer. https://doi.org/10.1007/978-1-84996-335-0

[22] Simon, D. (2006) Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches. Wiley. https://doi.org/10.1002/0470045345

Appendix

Python Simulation Code (Conceptual)

The core Python simulation code implementing the fractional oscillator with GL approximation and the ESC loop is provided below.

```
import numpy as np
import matplotlib.pyplot as plt
from numba import njit

#
    ================================================================

# 1. PARAMETERS (STABILIZED)
#
    ================================================================


# --- Vehicle Parameters ---
m_s = 250.0   # Sprung mass (kg)
m_us = 30.0   # Unsprung mass (kg)
k_s = 18000.0 # Suspension stiffness (N/m)
k_t = 180000.0# Tire stiffness (N/m)
c_t = 100.0   # Tire damping (Ns/m)
c_0 = 10.0    # Small base viscous damping (Ns/m)
c_d = 500.0   # Fractional damping coefficient (Ns^a/m) - REDUCED
    FOR STABILITY

# --- ESC Parameters ---
Ap = 0.05               # Perturbation amplitude
wp = 2 * np.pi * 0.5    # Perturbation frequency (rad/s)
k_esc = 0.25            # Adaptation gain
alpha_min, alpha_max = 1.05, 1.95

# --- Filter & Cost Function Parameters ---
whpf_hz = 0.1
wlpf_hz = 0.05
L_cost_mem = 2.0
L_frac_mem = 2.0

# --- Simulation Parameters ---
T_sim = 80.0    # Total simulation time (s) - INCREASED FOR
    SCENARIO
dt = 0.0005     # Time step (s) - REDUCED FOR STABILITY

#
    ================================================================

# 2. ROAD PROFILE GENERATION (ADVANCED SCENARIO)
#
    ================================================================

def generate_road_profile(duration, dt, switch_time=40.0,
                          smooth_noise_level=0.002,
    rough_noise_level=0.012,
                          bump_rate_smooth=0.5, bump_rate_rough
    =1.5,
                          bump_height=0.03):
    """
    Generates an advanced road profile that switches from a smooth
    road
    with sparse bumps to a rough road with more frequent bumps.
    """
```

```
    num_steps = int(duration / dt)
    time = np.linspace(0, duration, num_steps)
    z_r = np.zeros(num_steps)

    # --- Generate base continuous noise ---
    b, a = np.array([1.0]), np.array([1.0, -0.95])
    noise = np.random.randn(num_steps)
    road_base = np.zeros(num_steps)
    for i in range(1, num_steps):
        road_base[i] = b[0] * noise[i] - a[1] * road_base[i-1]
    road_base = road_base / np.std(road_base)

    switch_idx = int(switch_time / dt)
    z_r[:switch_idx] = smooth_noise_level * road_base[:switch_idx]
    z_r[switch_idx:] = rough_noise_level * road_base[switch_idx:]

    # --- Add discrete, randomly timed bumps ---
    def generate_bump(t, center, height, width=0.1):
        return height * np.exp(-((t - center)**2) / (2 * width**2))

    num_bumps_smooth = int(switch_time * bump_rate_smooth)
    if num_bumps_smooth > 0:
        bump_times_smooth = np.random.uniform(1.0, switch_time -
    1.0, num_bumps_smooth)
        for t_bump in bump_times_smooth:
            z_r += generate_bump(time, t_bump, bump_height)

    num_bumps_rough = int((duration - switch_time) *
    bump_rate_rough)
    if num_bumps_rough > 0:
        bump_times_rough = np.random.uniform(switch_time + 1.0,
    duration - 1.0, num_bumps_rough)
        for t_bump in bump_times_rough:
            z_r += generate_bump(time, t_bump, bump_height)

    return time, z_r

#
    ================================================================

# 3. CORE SIMULATION FUNCTION (NUMBA-ACCELERATED)
#
    ================================================================

@njit
def run_simulation(z_r, alpha_nom_init, T_sim, dt):
    """
    Main simulation loop accelerated with Numba.
    """
    num_steps = len(z_r)

    # --- History buffer sizes in steps ---
    cost_mem_steps = int(L_cost_mem / dt)
    frac_mem_steps = int(L_frac_mem / dt)

    # --- Pre-calculate filter coefficients for IIR implementation
    ---
```

```
rc_lpf = 1.0 / (2 * np.pi * wlpf_hz)
alpha_lpf = dt / (rc_lpf + dt)
rc_hpf = 1.0 / (2 * np.pi * whpf_hz)
alpha_hpf = rc_hpf / (rc_hpf + dt)

# --- Initialize state arrays ---
time = np.linspace(0, T_sim, num_steps)
z_s, z_us = np.zeros(num_steps), np.zeros(num_steps)
dot_z_s, dot_z_us = np.zeros(num_steps), np.zeros(num_steps)
ddot_z_s, ddot_z_us = np.zeros(num_steps), np.zeros(num_steps)

alpha_nom_hist = np.zeros(num_steps)
alpha_pert_hist = np.zeros(num_steps)
J_hist = np.zeros(num_steps)

lpf_out, hpf_out, hpf_prev_in = 0.0, 0.0, 0.0
rel_disp_history = np.zeros(frac_mem_steps)
accel_history = np.zeros(cost_mem_steps)

alpha_nom = alpha_nom_init

# --- Main Loop ---
for k in range(num_steps - 1):
    # 1. ESC: Generate perturbed alpha
    perturb_signal = Ap * np.sin(wp * time[k])
    alpha_pert = alpha_nom + perturb_signal

    if alpha_pert > alpha_max:
        alpha_pert = alpha_max
    elif alpha_pert < alpha_min:
        alpha_pert = alpha_min

    # 2. VEHICLE DYNAMICS: Calculate forces
    rel_disp = z_s[k] - z_us[k]
    rel_vel = dot_z_s[k] - dot_z_us[k]

    rel_disp_history = np.roll(rel_disp_history, 1)
    rel_disp_history[0] = rel_disp

    w = np.zeros(frac_mem_steps)
    w[0] = 1.0
    for j in range(1, frac_mem_steps):
        w[j] = w[j - 1] * (1 - (alpha_pert + 1) / j)

    D_alpha_y = np.sum(w * rel_disp_history) / (dt**alpha_pert)
    F_frac = c_d * D_alpha_y

    F_d = F_frac + c_0 * rel_vel
    F_s = k_s * rel_disp
    F_t = k_t * (z_us[k] - z_r[k]) + c_t * (dot_z_us[k] - 0)

    # 3. VEHICLE DYNAMICS: Calculate accelerations
    ddot_z_s[k] = (-F_d - F_s) / m_s
    ddot_z_us[k] = (F_d + F_s - F_t) / m_us

    # 4. VEHICLE DYNAMICS: Integrate state (Euler method)
    dot_z_s[k+1] = dot_z_s[k] + ddot_z_s[k] * dt
```

```
        dot_z_us[k+1] = dot_z_us[k] + ddot_z_us[k] * dt
        z_s[k+1] = z_s[k] + dot_z_s[k+1] * dt
        z_us[k+1] = z_us[k] + dot_z_us[k+1] * dt

        # 5. COST FUNCTION: Calculate J
        accel_history = np.roll(accel_history, 1)
        accel_history[0] = ddot_z_s[k]
        current_J = np.sqrt(np.mean(accel_history**2))

        # 6. ESC: Update alpha_nom
        hpf_out_new = alpha_hpf * (hpf_out + current_J -
    hpf_prev_in)
        hpf_prev_in = current_J
        hpf_out = hpf_out_new

        demod_signal = hpf_out * np.sin(wp * time[k])
        lpf_out = lpf_out + alpha_lpf * (demod_signal - lpf_out)
        grad_est = lpf_out

        alpha_nom = alpha_nom - k_esc * grad_est * dt

        if alpha_nom > alpha_max:
            alpha_nom = alpha_max
        elif alpha_nom < alpha_min:
            alpha_nom = alpha_min

        # --- Store history ---
        alpha_nom_hist[k+1] = alpha_nom
        alpha_pert_hist[k+1] = alpha_pert
        J_hist[k+1] = current_J

    return {
        "time": time, "z_s": z_s, "z_us": z_us, "z_r": z_r,
        "ddot_z_s": ddot_z_s, "alpha_nom": alpha_nom_hist,
        "alpha_pert": alpha_pert_hist, "J": J_hist
    }

#
    ================================================================

# 4. PLOTTING FUNCTION
#
    ================================================================

def plot_results(results, title):
    """Plots the simulation results in a 4-panel figure."""
    fig, axs = plt.subplots(4, 1, figsize=(14, 12), sharex=True)
    fig.suptitle(title, fontsize=16)

    # Panel 1: Alpha Adaptation
    axs[0].plot(results["time"], results["alpha_nom"], label=r'$\
    alpha_{nom}$', lw=2)
    axs[0].plot(results["time"], results["alpha_pert"], label=r'$\
    alpha_{pert}$', alpha=0.5)
    axs[0].set_ylabel(r'Fractional Order $\alpha$')
    axs[0].grid(True)
    axs[0].set_ylim(alpha_min - 0.1, alpha_max + 0.1)
```

```
    # Panel 2: Cost Function J
    axs[1].plot(results["time"], results["J"], label=r'Cost J (RMS
    of $\ddot{z}_s$)')
    axs[1].set_ylabel('Cost J')
    axs[1].legend()
    axs[1].grid(True)

    # Panel 3: Displacements
    axs[2].plot(results["time"], results["z_s"], label=r'$z_s$ (
    sprung)')
    axs[2].plot(results["time"], results["z_us"], label=r'$z_{us}$
    (unsprung)', alpha=0.8)
    axs[2].plot(results["time"], results["z_r"], 'k--', label=r'
    $z_r$ (road)')
    axs[2].set_ylabel('Displacement (m)')
    axs[2].legend()
    axs[2].grid(True)

    # Panel 4: Sprung Mass Acceleration
    axs[3].plot(results["time"], results["ddot_z_s"], label=r'$\
    ddot{z}_s$')
    axs[3].set_xlabel('Time (s)')
    axs[3].set_ylabel('Acceleration (m/s  )')
    axs[3].legend()
    axs[3].grid(True)

    plt.tight_layout(rect=[0, 0, 1, 0.96])
    return fig, axs

#
    ================================================================

# 5. MAIN EXECUTION
#
    ================================================================

if __name__ == '__main__':
    print("Running Advanced Scenario: Smooth road with bumps ->
    Rough road...")

    switch_t = T_sim / 2

    # Generate the road profile
    time_vec, road_profile = generate_road_profile(
        duration=T_sim,
        dt=dt,
        switch_time=switch_t
    )

    # Run the simulation
    results = run_simulation(road_profile, alpha_nom_init=1.5,
    T_sim=T_sim, dt=dt)

    # Plot the results, adding a line to show where the road
    changes
```

```
fig, axs = plot_results(results, f"Advanced Scenario: Road
Change at t={switch_t:.0f}s")

for ax in axs:
    ax.axvline(x=switch_t, color='r', linestyle='--', linewidth
=2, label='Road Change')

# Redraw legend on the first plot to include the new line
handles, labels = axs[0].get_legend_handles_labels()
# Add the new label only if it's not already there
if 'Road Change' not in labels:
    line = [line for line in axs[0].get_lines() if line.
get_label() == 'Road Change'][0]
    handles.append(line)
    labels.append('Road Change')
axs[0].legend(handles, labels)

plt.show()
```

Listing 1. Python simulation for adaptive fractional damping with ESC.

Note: The Python code provided is a conceptual demonstration of the ESC mechanism adapting α towards a time-varying optimum. A full quarter-car model simulation with the fractional damper would require a numerical ODE solver incorporating the Grünwald-Letnikov approximation for the fractional derivative term within the force calculation at each time step. The cost J would then be derived from the simulated vehicle states (e.g., RMS of sprung mass acceleration).