2019 NDIA GROUND VEHICLE SYSTEMS ENGINEERING AND TECHNOLOGY SYMPOSIUM
MODELING, SIMULATION, TEST & VALIDATION (MSTV) TECHNICAL SESSION
AUGUST 13-15, 2019 - NOVI, MICHIGAN

WAVEFORM SYNTHESIS FOR SHOCK RESPONSE SPECTRUM REPLICATION, APPLIED TO GROUND VEHICLE COMPONENT TESTING

Samuel Allen[1]

[1] Combat Capabilities Development Command (CCDC) / Ground Vehicle Systems Center (GVSC), Warren, MI

Abstract

This paper describes the motivation and process taken for developing an acceleration time history that has a shock response spectrum that matches the MIL-STD 810 defined shock response spectrum. The mathematics from the ISO standard 18431-4 are presented, as well as the procedures and graphs from the MIL-STD. The time history is synthesized from a sum of basis-functions, parameterized by variable amplitudes and delays. An optimization routine then modifies an array of these optimization parameters to find a time history that has a shock response spectrum that matches a reference shock response. All figures are presented in Appendix A for clarity. The equations developed and computer code written to perform this task are explained, and the full code is provided in Appendix B.

Citation: S. Allen, "Waveform Synthesis for Shock Response Spectrum Replication, Applied To Ground Vehicle Component Testing", *In Proceedings of the Ground Vehicle Systems Engineering and Technology Symposium* (GVSETS), NDIA, Novi, MI, Aug. 13-15, 2019.

1. MOTIVATION

The motivation for this work is to mitigate shock levels on vehicle components, through testing new designs of equipment.

The objective is to generate a time history to perform shock testing on military and commercial ground vehicle components. The time histories are used to drive hydraulic actuators coupled to a test specimen.

The Military Standard 810g Method 516.7 (MIL-STD) prescribes actual event data as the first choice, however that is usually not an option as most components haven't been tested in actual shock events. Therefore the second procedure called out in the MIL-STD is the best choice for the lab. That second procedure is Shock Response Spectrum (SRS) based testing, and was the focus of this work. The SRS was used to generate the time history.

Another procedure defined in the MIL-STD, and is called classical shock which was the method used by the lab prior to this work. It defines pre-developed acceleration time histories for one to control the test to.

An example of classical shock waveform is in Figure 1, in Appendix A.

As seen in Figure 1, the acceleration is all positive, therefore high velocities are reached when running these profiles. In order to stop the actuator from over extending after the pulse is complete, a "post pulse" negative acceleration needs to be applied. These post pulses can be just as harsh, or worse, on the actuator and the test item. The step change in acceleration is also hard to control to on hydraulic actuators.

The post pulses are one of the items desired to be eliminated by using the SRS as a testing base. The time history that's synthesized can be tailored to have no residual velocity and still match the reference SRS. This will stop the displacement from continuing to increase after the pulse is played out.

2. BACKGROUND

A shock response spectrum is a method for characterizing the shock properties of a transient time history. A SRS is similar to Power Spectral Density (PSD), in that they both involve the frequency domain, however the PSD can be transformed into, and back from, a waveform time history mathematically. While a SRS can be derived easily, there's no well-defined unique inverse method for an SRS back into a time history. Figure 2 shows a graphical depiction of how the SRS is generated from a single input waveform.

Each mass damper system is representative of each frequency of interest, and the max amplitude of each mass damper's response is a point on the SRS curve. However, through this process a lot of the information of the initial waveform is lost. Converting an SRS back into a waveform requires an iterative mathematical approach.

3. METHOD

3.1 THEORY

The method chosen for generating a time history from an SRS, is to compose a time history from a set of basis-functions parameterized by a set of variable amplitudes and delays. This time history acts as a starting point for iterations. A SRS is generated from the time history, and compared to the MIL-STD SRS, then the variables of the time history are changed to reduce the error between the optimized SRS, and the reference SRS. Through these iterations an optimized time history results. This optimized time history has an SRS nearly identical to the MIL-STD SRS.

The basis-functions were chosen to be sinusoidal, exponentially decaying pulses, as they are analogous to real world data. See Figure 3 for an example time history.

The number of these basis functions determines the fidelity of the time history, i.e. how much room for variation there is to match the reference SRS. However, if too many are used the computation time increases.

The structure of the basis function can be seen in equation (1).

$$P_i(t, A_i, b_i) = \Big(A_i \left(f(t)\, s(t)\, g(t)\right)\Big) u(t) \quad (1)$$

Note that this function shows the structure of the pulse. The actual total pulse is a sum of multiple exponential sinusoids, with different amplitudes and delays, and the Hann is added to that sum. Note that $u(t)$ is the unit step function.

Each of the equation (1) components are defined in equation (2-6), where A is the variable amplitude of the pulse in g's, b is the variable time delay of the pulse in seconds, F is the variable amplitude of the Hann function in g's, and σ is the length of the Hann pulse in seconds.

$$f(t, b) = e^{-\frac{t-b}{\tau}} \quad (2)$$

$$s(t, b) = \sin\big(\omega * (t - b)\big) \quad (3)$$

$$g(t, b) = \omega * (t - b) \quad (4)$$

$$\omega = {}^{2\pi}/_{\tau} \quad (5)$$

$$h(t) = \frac{F}{2}\left(1 - \cos\left({}^{2\pi t}/_{\sigma}\right)\right) \quad (6)$$

The sine component was chosen to excite the resonator. The exponential term was chosen to make the pulse transient. A smoothing function was added to eliminate the sharp change at the beginning of the sine wave. This is to reduce jerk on the actuator system. To counteract the final velocity of the optimized pulse, a Hann function was added. Implementation of the Hann function eliminates the need for the classical shock, post-pulse events mentioned before. Since the Hann performs the same function, but is built into the time history and is an optimization variable, it has no adverse effects on matching the reference SRS, since it is taken into account in the iterations. See Figure 4 for an image of basis function components.

Equation (7) shows the full time history equation for the basis function.

$$P(t, A, b) = h(t) + \sum_{i=1}^{N} P_i\,(t, A_i, b_i) \quad (7)$$

The τ terms are the time constants of the pulse, and were picked after some experimentation to be between 0.003 and 0.05 seconds. The number of τ terms between those values determines how many basis functions will make up the time history. This also determines how many amplitudes and delays are in the function variable array.

As discussed before another part of developing these time histories, is to synthesize an SRS from them. This is defined by the ISO standard. Since the equations are based on the response of a mass-damper system, and that topic has been decomposed and analyzed in many other papers, the full decomposition will be omitted here. Instead the method of using the equations to generate an SRS will be presented.

Equation (8) gives the time history response y, of a single mass-damper from a given input x. [3]

$$y_n = \beta_0 x_n + \beta_1 x_{n-1} + \beta_2 x_{n-2} - \alpha_1 y_{n-1} - \alpha_2 y_{n-2} \quad (8)$$

The x terms being the points from the input time history, and the y terms are the response. The alpha and beta coefficients are defined by equations (9-15). [3]

$$\beta_0 = 1 - \exp(-R) \cdot \sin(V)/V \quad (9)$$

$$\beta_1 = 2\exp(-R) \cdot \{\frac{\sin(V)}{V} - \cos(V)\} \quad (10)$$

$$\beta_2 = \exp(-2R) - \exp(-R) \cdot \frac{\sin(V)}{V} \quad (11)$$

$$\alpha_1 = -2\exp(-R) \cdot cos(V) \quad (12)$$

$$\alpha_2 = -2\exp(-2R) \quad (13)$$

$$R = \omega\, T/2Q \quad (14)$$

$$V = \omega\, T \sqrt{1 - 1/4Q^2} \quad (15)$$

From these coefficients it can be seen that the conversion from time history into SRS is a function of frequency, i.e. ω. The Q term is the resonance gain, and is normally set to 10, or 50. Both values were attempted, and 50 resulted in less error after iterating. The T term is the inverse of the sampling frequency. The sampling frequency chosen was 4,096 Hz. This was picked for use with the controllers utilized in the case study lab. Best practice for picking the sampling frequency is to multiply the max frequency of interest in the SRS by ten. This would have required a sampling frequency of over 10,000 Hz. To save on computation time, the frequency was left at 4,096 Hz, and a post optimization check was performed. This check was to take the final optimized time history and resample it by a factor of 10, and then compute a new SRS. That SRS was then compared to the reference SRS and the optimized SRS. Because no significant differences were seen, it was assumed that no excess error was being introduced, and no content was being lost.

Finally the optimization problem needs to be defined, with the goal being a time history with an SRS identical to the reference SRS. The starting point for the time history has been defined. The method for generating a SRS from that time history is defined. The optimization is to minimize the error between two SRS. See equation (16) for the error function.

$$SRS_{error} = \left(S - \hat{S}\right)^T (S - \hat{S}) \quad (16)$$

Additionally, three regularization terms were added to the error to keep the optimization variables minimized as well. Again each array of the amplitudes and delays were separately multiplied by their transpose to get a scalar. The third regularization term was set up to compare the final velocity of the time history to the desired final velocity, i.e. zero. These three regularization terms were multiplied by scaling factors to give their values a significant enough impact on the optimization function, compared to the SRS error.

The final optimization problem takes the form of equation (17, 18). Where a, d, and f, are the regularization terms for amplitude, delay, and Hann amplitude respectively. The C terms are the scaling factors to define importance of each regularization term.

$$\min_{A,b,F}\left[SRS_{error} + C_1 A^t A + C_2 b^t b + C_3 \int_0^{t_f} P\, dt\right] \quad (17)$$

Subject to

$$A_i \in [0, A_{max}], b_i \in [0, b_{max}], F \in \mathbb{R}^+ \quad (18)$$

3.2 IMPLEMENTATION

All the functions and data processing were written and performed in MATLAB. The final code is shown in its entirety in the Appendix. All necessary computation and inputs are in a single script, with imbedded functions. The user inputs are entered at the beginning of the script. While the code is applicable to any SRS being used as a reference, the SRS_Reference function would need to be changed to run with a different objective SRS, than the one defined in the MIL-STD. The reference SRS can be seen as the blue line in Figure 5.

Figure 5 shows the SRS of the final optimized time history. The dashed lines above and below the reference SRS are a +/- 3dB range of acceptable variation. As can be seen, the optimized time history falls well within the 3dB bounds.

The velocity and displacement of the acceleration time history were calculated to verify that the displacement was captured, and it was within the limits of the actuator. Figure 6 shows the acceleration, velocity, and displacement time histories. Figure 7 shows a zoomed in view of the first 0.200 seconds. That time range shows the major peak of the acceleration pulse.

It can be seen from these graphs that the velocity was brought to zero at the end of the pulse, and therefore the displacement was held at its final value. The max displacement is 0.0546 meters (2.15 in), and the max velocity is 1.3339 m/s (52.52 in/s). Both are well within the capabilities of the actuators in the CCDC/GVSC lab.

Figure 8 shows the kinematics of the optimized history, without the Hann function added in. The error between the ref SRS and the optimized was unaffected however, by the presence of the Hann function.

3.3 LAB SETUP & TESTING

After the final optimized time history was generated a test on an actuator in the lab was planned. The test would involve a single axis hydraulic actuator. The actuator, the control system, and the software for running it, were all built by MTS [4]. See Figure 9 for an image of the test rig.

Data recorded would be displacement of the actuator shaft, and two accelerometers mounted on the head of the shaft. Two accelerometers were used on separate data acquisition systems. One being the MTS system and the other was a National Instruments cRIO.

The test would be controlled simultaneously to both the displacement time history and the acceleration time

history. While the displacement control works well for lower frequency content, the high frequency of interest required the accelerometers.

The MTS system has software, called RPC, designed for developing the drive file based off of the desired response. The first step is to bring the desired response into the software and convert it into the MTS data structure. Next is to calculate a transfer function between the input and output of the actuator. This is done by playing random white noise into the actuator, and recording the response. The system then has a relationship between input and output. Figure 10 shows a screenshot of the white noise model process.

Now that the system knows the relation between input and output, it can calculate the drive needed for the optimized time history. To get the needed drive, an iterative approach is used. Figure 11 shows a screen shot of the MTS iterative process.

The software starts with a low amplitude starting drive file, which is played out and the response, collected. The error between the two is calculated and a new slightly higher amplitude drive is generated. This process of running and adjusting levels continues until the error is within acceptable limits. Displacement error is preferred to be under 1%. In Figure 11 the time history overlay of desired and response can be seen on the left and the error between the desired and the iteration response, on the right. The two lines are the acceleration response error and the displacement response error.

Once a drive with low enough error is developed it can be loaded and played out to get final results and collect test data. Figure 12 shows the MTS window for running a test and collecting response data.

The response data in Figure 12 shows the response collected by the MTS system. The NI data acquisition system was collected by a separate computer.

4. RESULTS

The data was pulled from the two data acquisition systems and loaded back into MATLAB for analysis. The time history results were synced to compare. Figure 13 shows an overlay of the results from the two data acquisition systems and the optimized, desired pulse.

Note that the NI results needed to be filtered to compare to the MTS results. The MTS controller and the NI data acquisition systems were both set to sample at the same rate. The MTS system, however, actually up-samples and then filters back to the set frequency. The NI system just samples at the rate set. Because of this, the NI system picked up high frequency accelerations that didn't show up on the MTS output. While it looked like noise at first, compared to the MTS data, the frequency of the additional content was the same as some of the content in the control signal that was sent to the actuator. So, most likely the NI system was picking up real accelerations, and the MTS picked it up

too, but filtered it out. Future testing is planned to try and eliminate this extra content, either through fine tuning of the actuator, or additional iterations in the RPC Simulate to remove error from the drive signal.

After the testing was complete the acquired data was passed through the SRS synthesis function and compared to the reference SRS. Figure 14 shows that comparison.

While some high frequency content has higher error between the reference and tested SRS, it is still within the 3dB bounds set. This higher frequency content may be regained through additional iterations of the drive signal, and/or tuning of the actuator.

5. References

[1] DOD, U. A. (2014, April 15). MIL-STD-810G w/Change 1, Method 516.7 Shock.

[2] Irvine, T. (2012, October 13). *Vibrationdata*. Retrieved from Synthesize an Acceleration Time History to Satisfy a Shock Response Spectrum: https://vibrationdata.wordpress.com/2012/10/13/synthesize-an-acceleration-time-history-to-satisfy-a-shock-response-spectrum/

[3] ISO 18431-4. (2007, March 01). Switzerland.

[4] MTS. (2018). *RPC® Pro Software*. Retrieved from MTS: https://www.mts.com/en/products/producttype/test-components/software/rpc/index.htm

[5] Physics, D. (2018). *Understanding the Shock Responce Spectrum*. Retrieved from DataPhysics: http://www.dataphysics.com/web-seminar-understanding-the-shock-response-spectrum-on-thursday-july-11-2013/

[6] Wikipedia. (2018). *Hann Function*. Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Hann_function

Appendix A: Figures

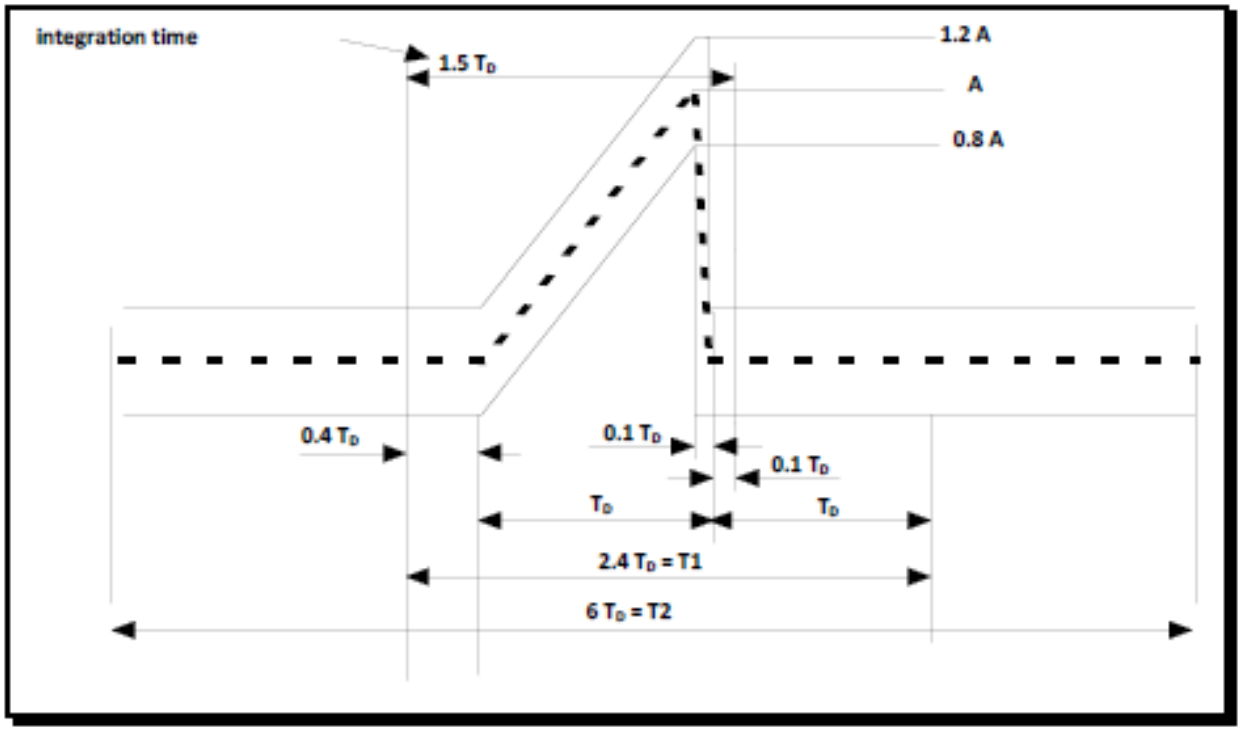

Figure 1:"Sawtooth" Classical Shock [1]

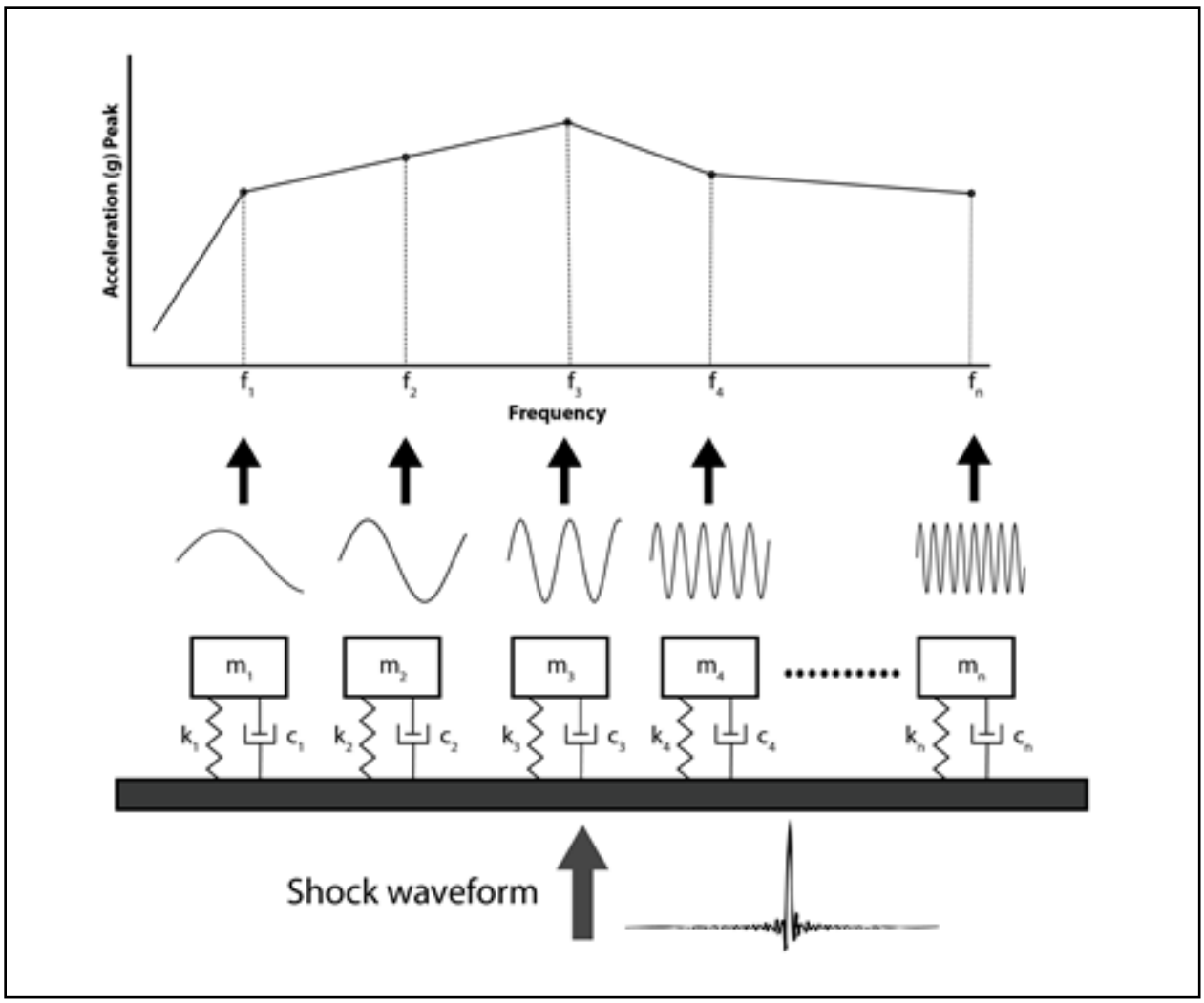

Figure 2: SRS Synthesis [5]

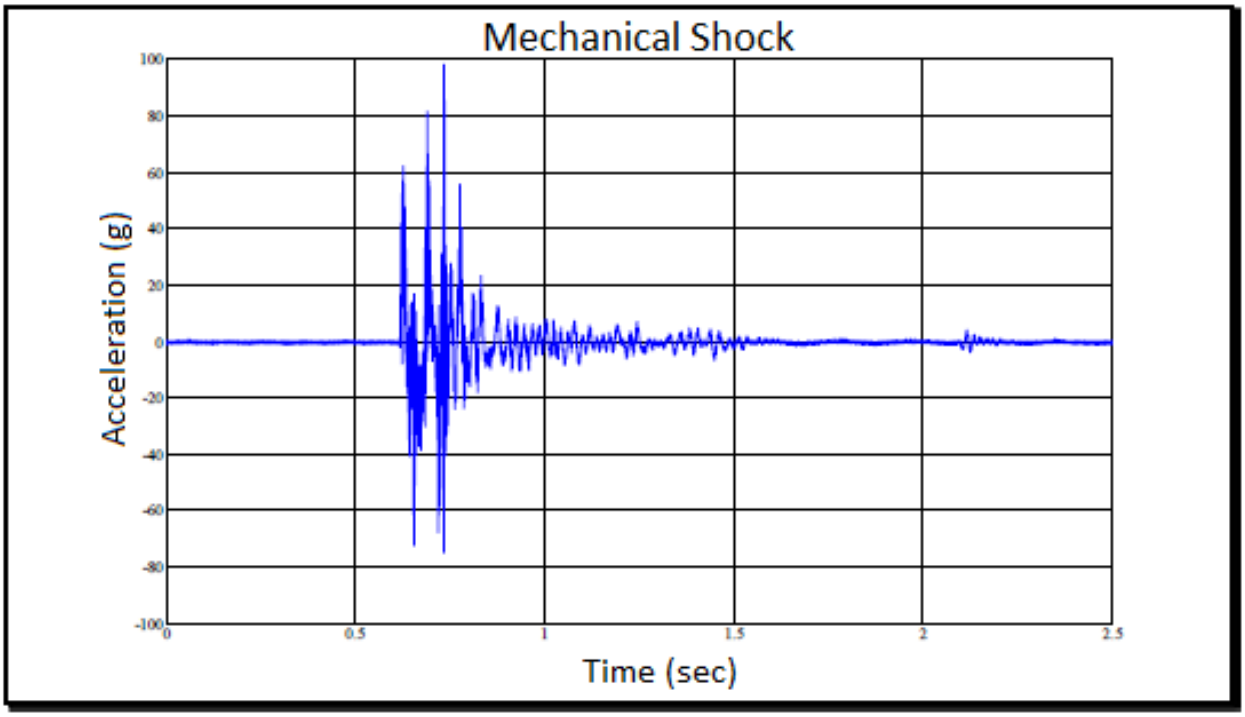

Figure 3: Example Time History [1]

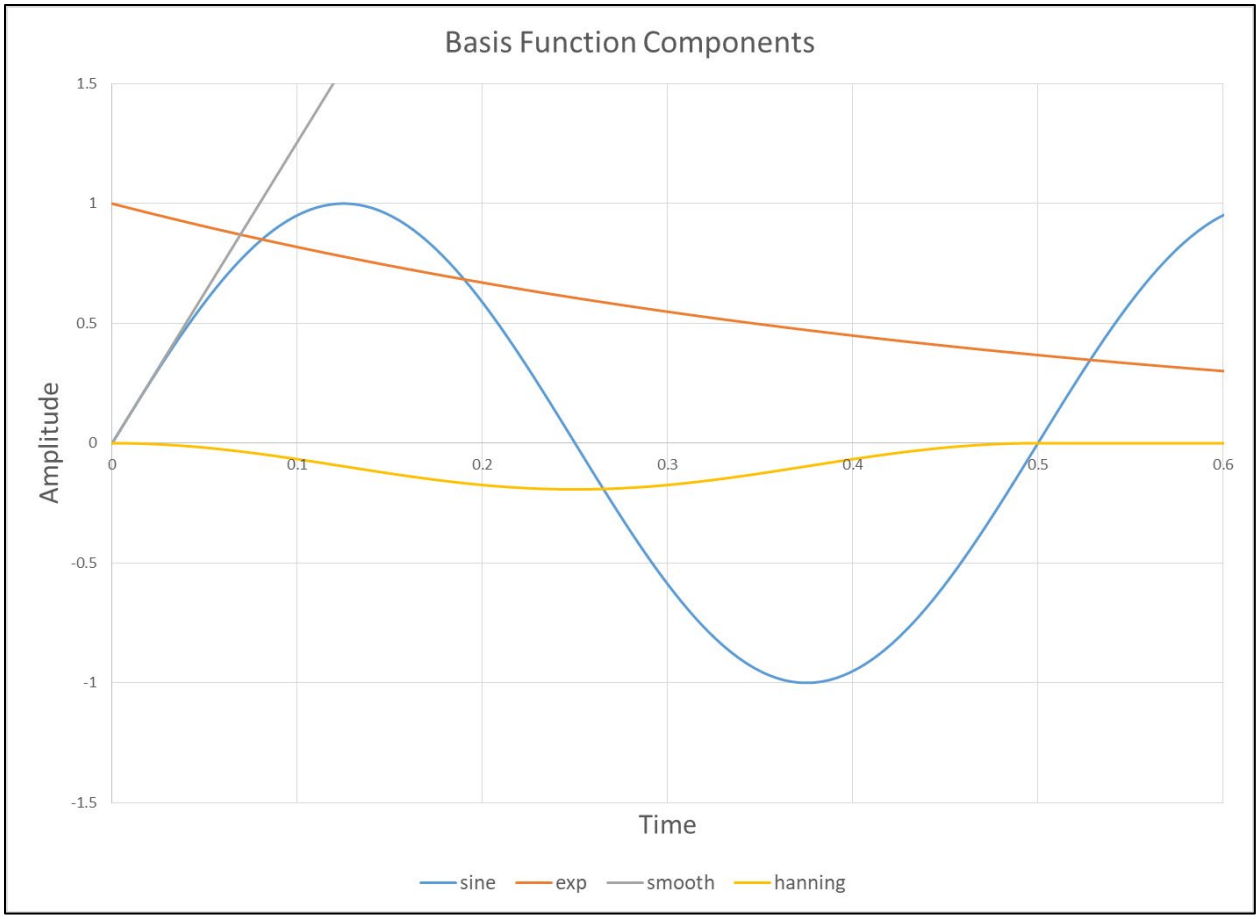

Figure 4: Basis Function Components

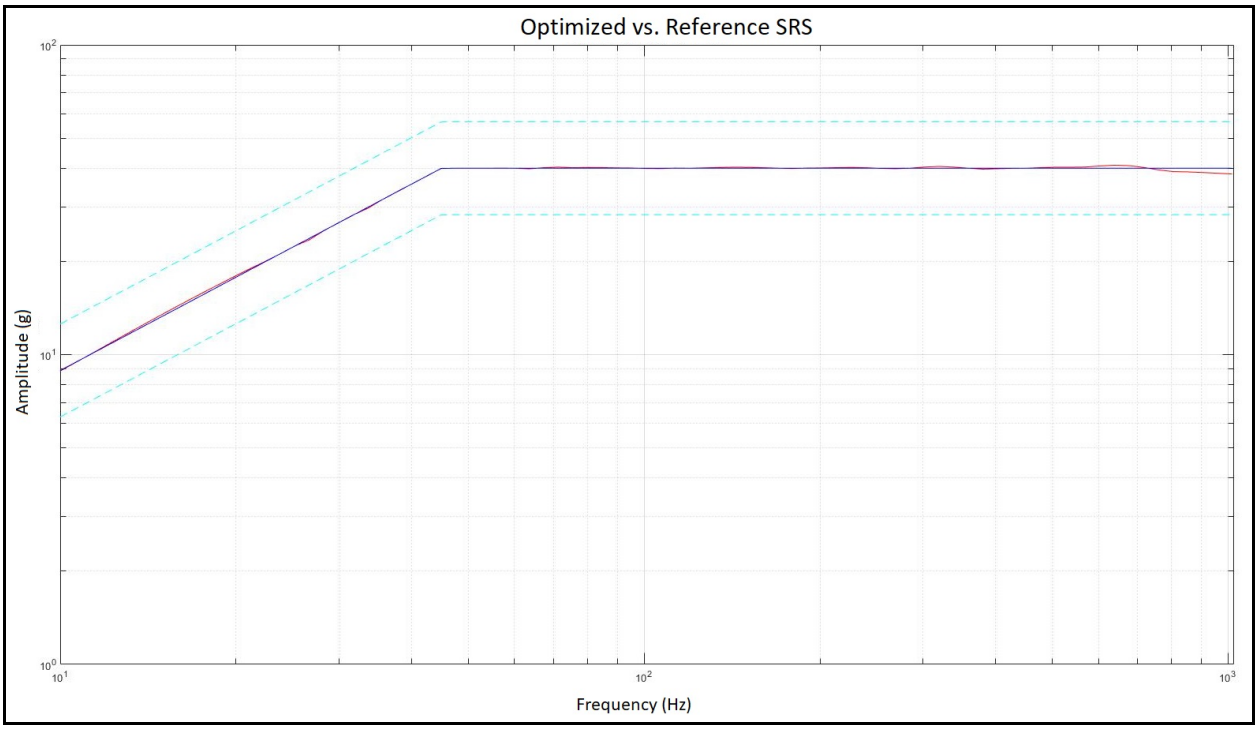

Figure 5: SRS of the Optimized Time History

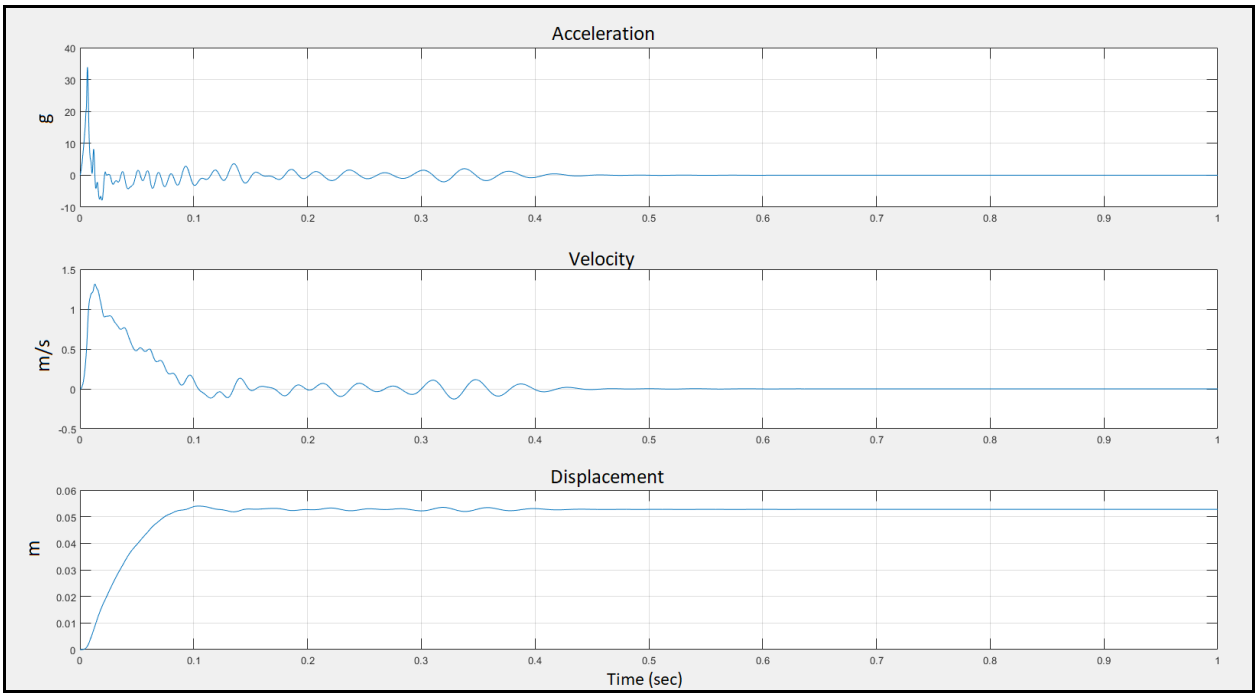

Figure 6: Optimized Kinematic Time Histories

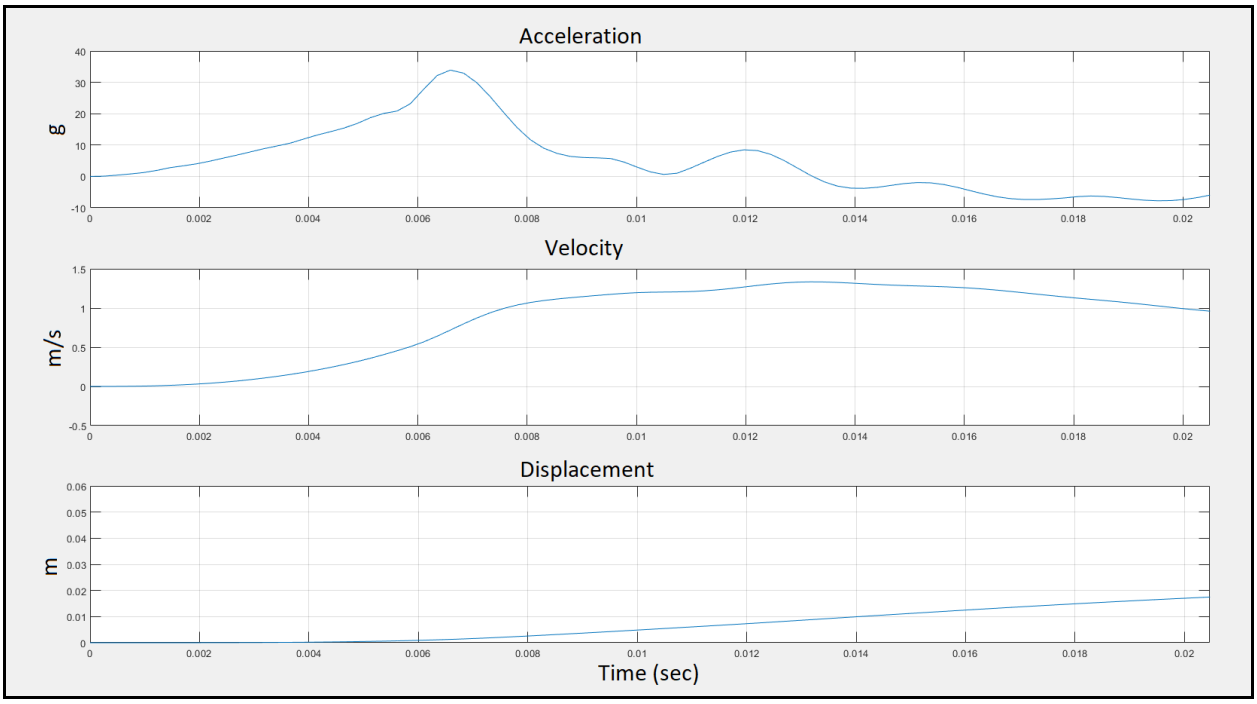

Figure 7: Kinematics Zoomed In

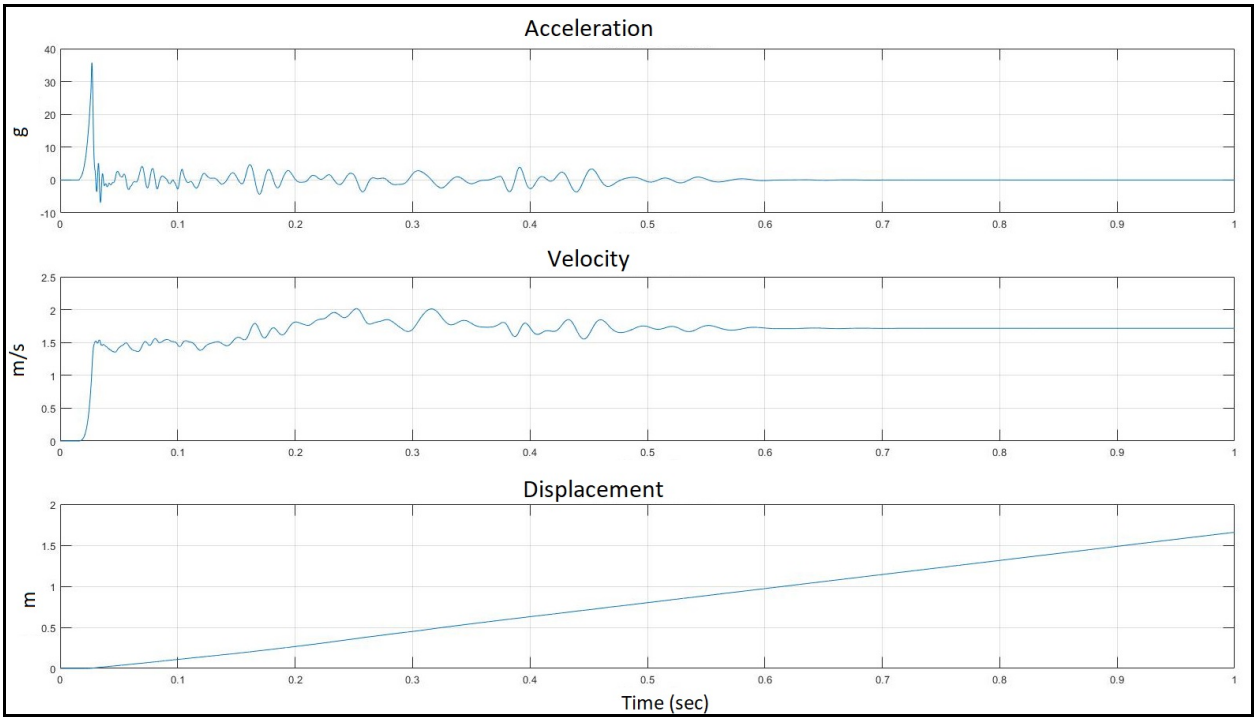

Figure 8: Time Histories w/o Hann

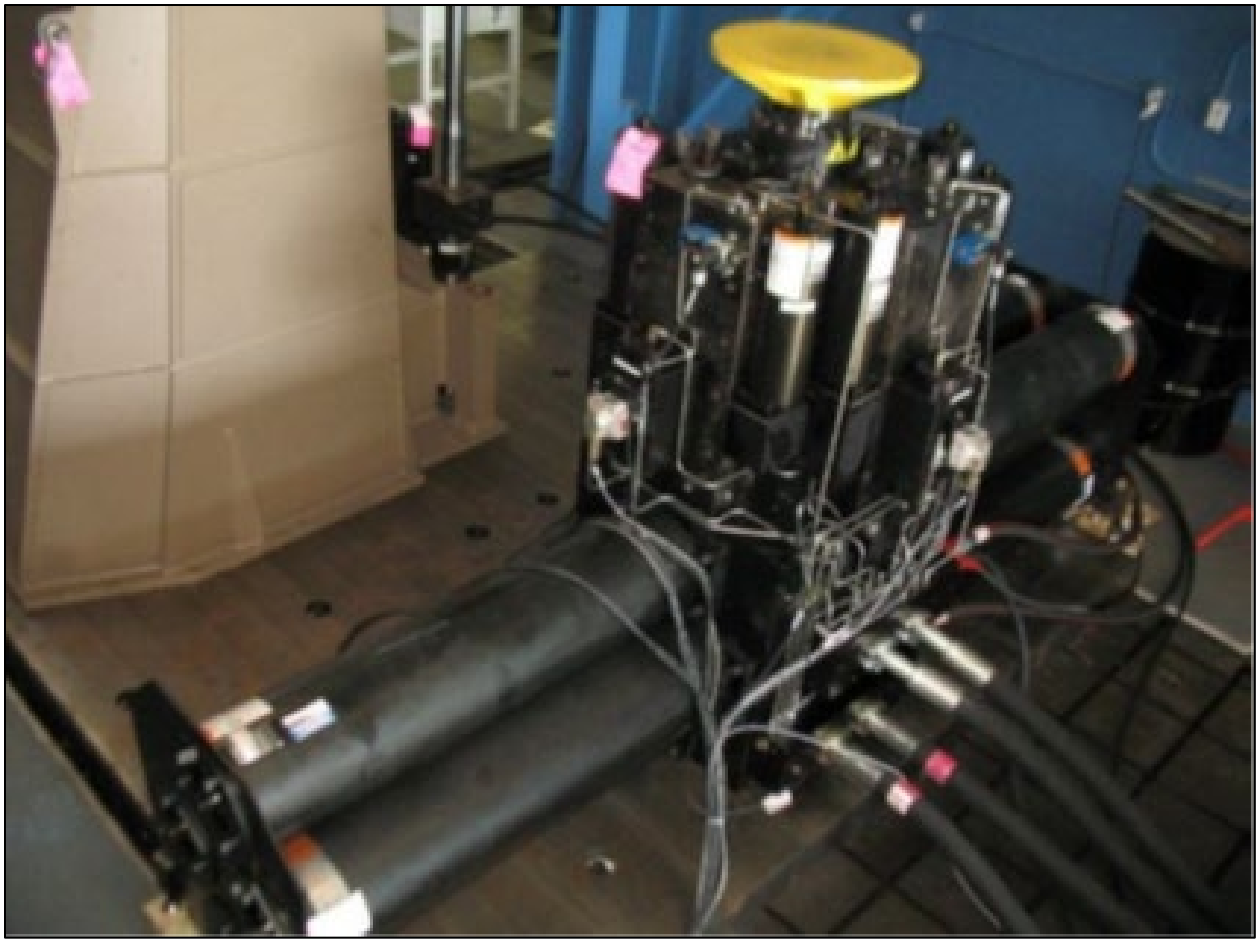

Figure 9: BATS Rig

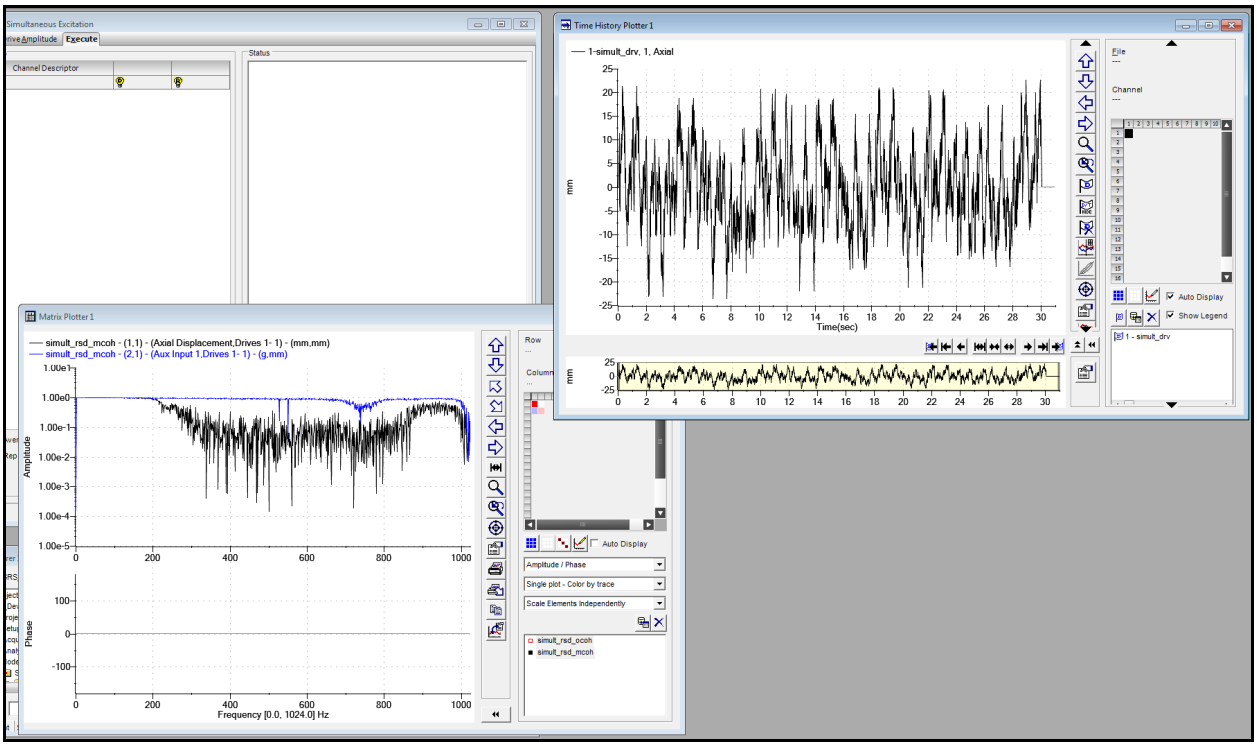

Figure 10: RPC Model Window

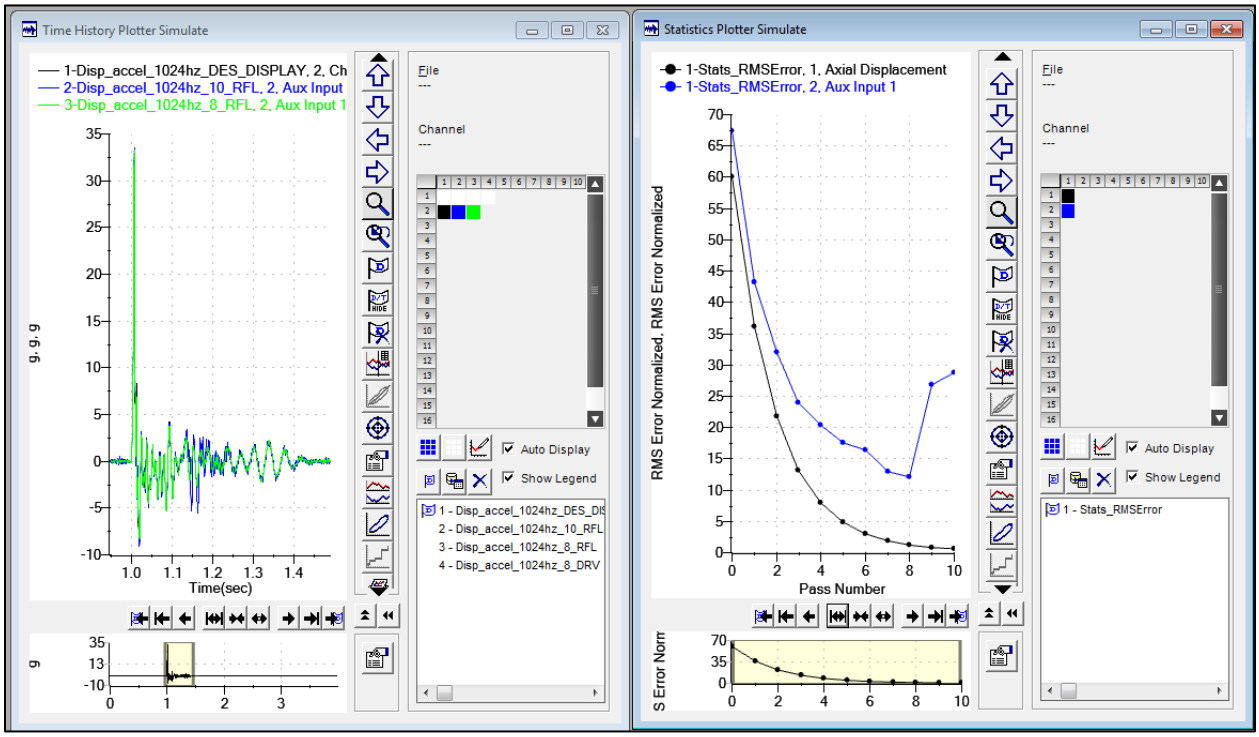

Figure 11: RPC Simulate Window

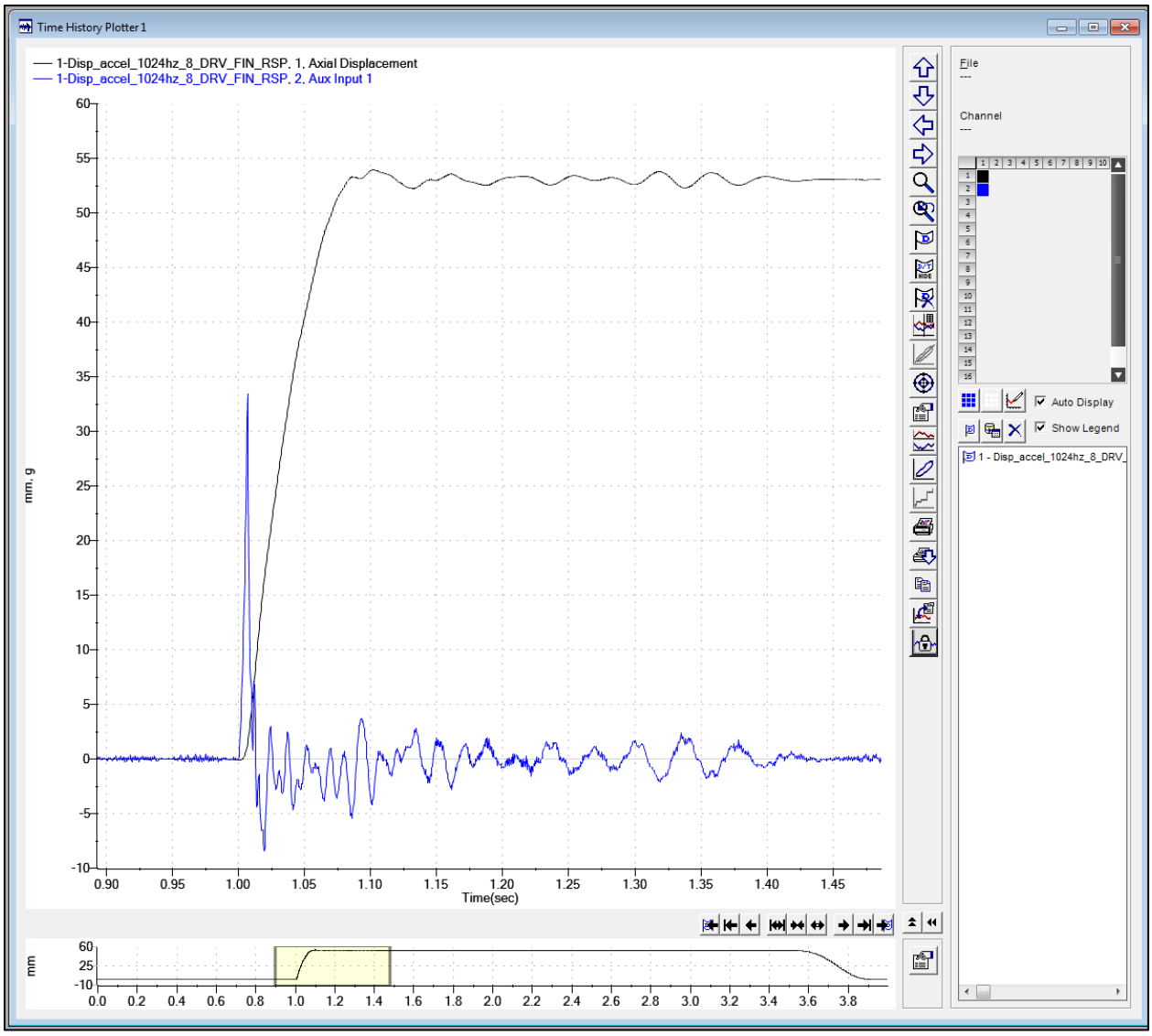

Figure 12: RPC Test Window

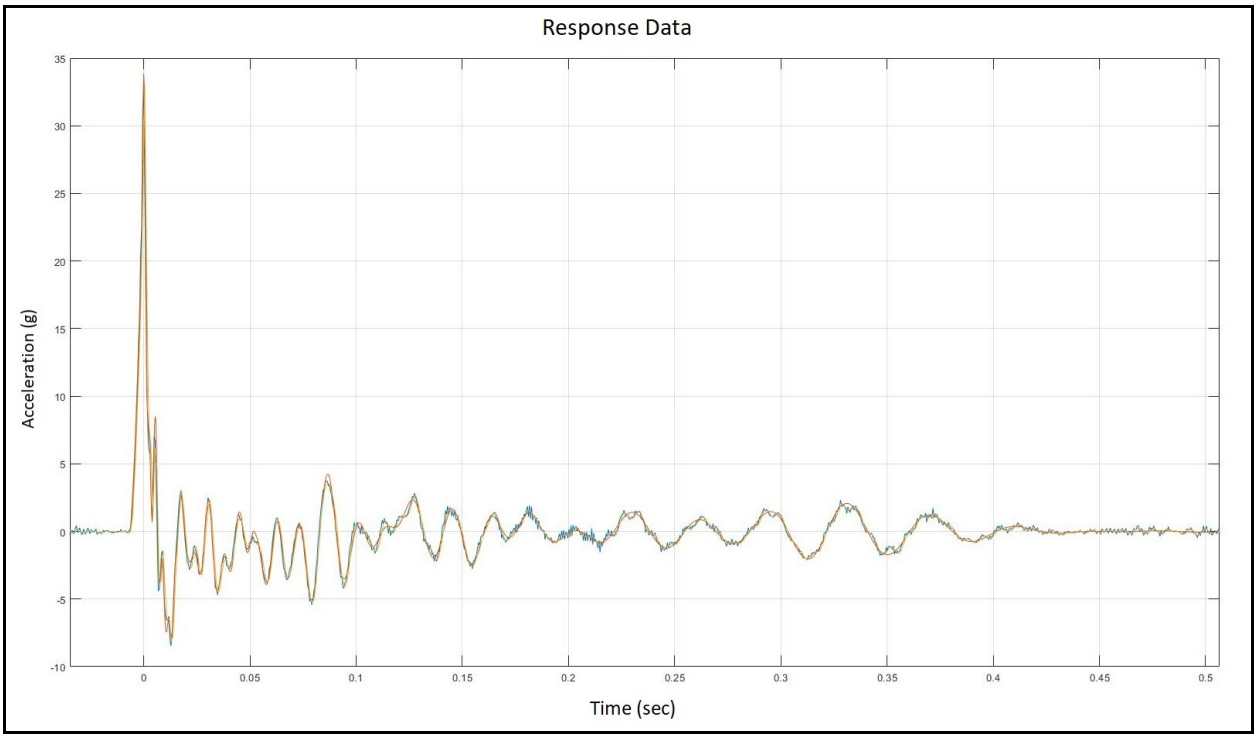

Figure 13: Testing Data

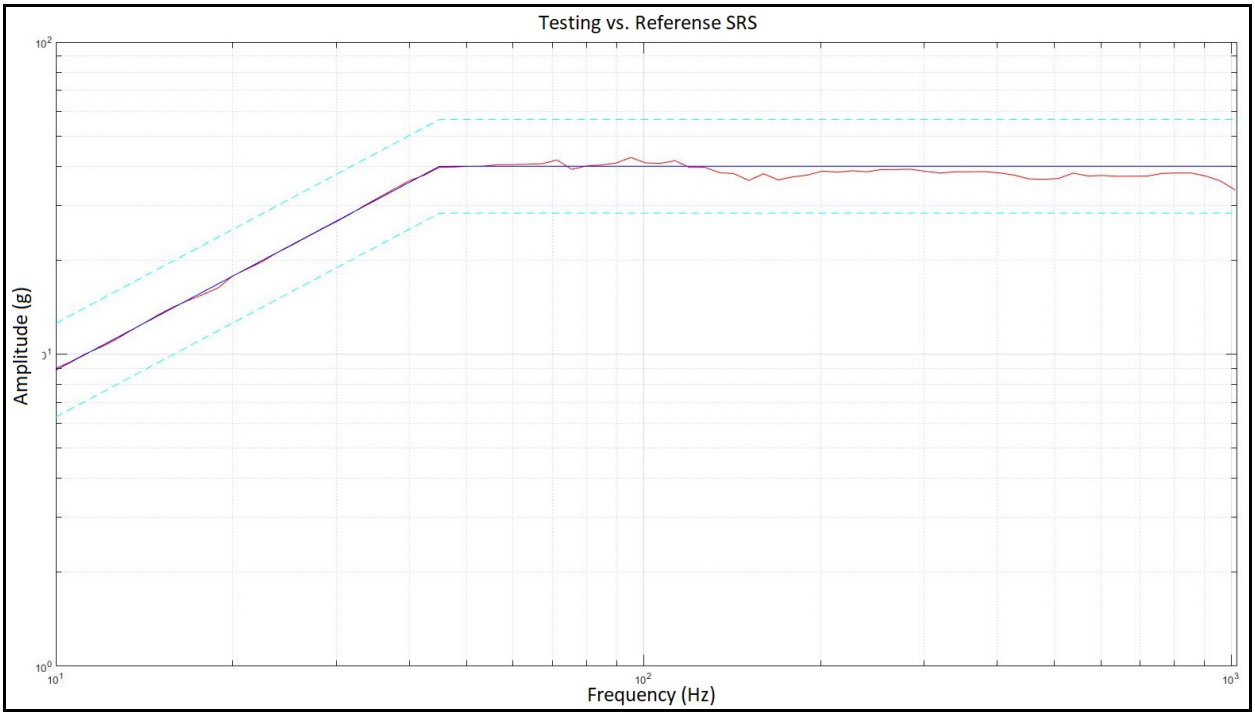

Figure 14: SRS of Testing Results

Appendix B: MATLAB Code

```
% Samuel C Allen
% US Army - CCDC / Ground Vehicle Systems Center - PST
% March 11, 2018

    % User input
f_s=4096; % Sampling frequency (Hz)
Q=50; % Resonance gain (Standard is either 10 or 50)
tau=0.003:0.001:0.05; % Range of time constants for basis functions
f_min=10; % Minimum of frequency range computed
f_max=1024; % Maximum of frequency range computed
dB=1; % +/- dB error range allowed for SRS
t_final=1; % Length of time history in seconds
F=-1; % Amplitude of Hann function (Initial guess)

    % Calculated from user input
t=linspace(0,t_final,f_s); % Time history range
freq_1=log2(f_min):1/12:log2(f_max); % Frequency range
freq=2.^freq_1;
T=1/f_s; % Sampling time interval in seconds
amp=5.0*ones(size(tau)); % Amplitudes (Initial guess)
del=0.01*ones(size(tau)); % Delays (Initial guess)

    % Optimization parameters
x0=[amp,del,F]; % Initial guess for optimizing

% Linear inequality constraints
A=[];
b=[];
% Linear equality constraints
Aeq=[];
beq=[];
% Lower bound of optimization parameters
lb=[-150.0*ones(size(amp)),zeros(size(del)),-10];
% Upper bound of optimization parameters
ub=[150.0*ones(size(amp)),0.5*ones(size(del)),10];
% Nonlinear constraints
nlcon=[];
% Algorithm options for fmicon
options=optimoptions('fmincon', 'MaxFunctionEvaluations', 100000,...
    'MaxIterations', 10000, 'Display', 'iter','Algorithm', 'sqp');

    % Optimization function; minimize the error
[x,fval,ef,output,lambda]=fmincon(@(x) opt_err(x,tau,t,freq,T,Q,dB),...
    x0,A,b,Aeq,beq,lb,ub,nlcon,options);

    % Calculate optimized SRS and kinematics
amp_opt=x(1:(length(x)-1)/2);
del_opt=x((length(x)-1)/2+1:(length(x)-1));
F_opt=x(end);
[r,r_upper,r_lower]=SRS_Reference(freq,dB);
a_opt=pulse(amp_opt,del_opt,tau,t,F_opt);
s_opt=SRS(a_opt,freq,T,Q);
```

```
e_opt=SRS_Error(s_opt,r);
[accel,vel,disp]=kinematics(a_opt,t);
Actuator_TH_disp=[t',disp'];
Actuator_TH_vel=[t',vel'];
Actuator_TH_accel=[t',a_opt'];

    % Plot optimized SRS, the reference SRS, and the upper and lower
    % bounds of the SRS reference
figure
loglog(freq,s_opt','r',freq,r','b',freq,r_upper','c--',freq,r_lower','c--')
title('Optimized vs. Reference SRS')
xlabel('Frequency (Hz)')
ylabel('Amplitude (g)')
axis([f_min f_max 1 100]);
grid on

    % Plot the acceleration, velocity and displacement of the
    % optimized time history
x1=0;
x2=t_final;
figure
subplot(3,1,1)
plot(t,a_opt)
title('Optimized Acceleration')
xlabel('Time (sec)')
ylabel('Acceleration (g)')
xlim([x1 x2]);
grid on
subplot(3,1,2)
plot(t,vel)
title('Optimized Velocity')
xlabel('Time (sec)')
ylabel('Velocity (m/s)')
xlim([x1 x2]);
grid on
subplot(3,1,3)
plot(t,disp)
title('Optimized Displacement')
xlabel('Time (sec)')
ylabel('Displacement (m)')
xlim([x1 x2]);
grid on

    % Function for calculating the system kinematics
function [a,v,s]=kinematics(accel_g,t)
    a=accel_g*9.81;
    v=cumtrapz(t,a);
    s=cumtrapz(t,v);
end

    % Function for generating the basis function time histories
function p=pulse(amp,del,tau,t,F)
    p=t*0;
    for i=1:length(tau)
        gain_1=ones(size(t));
```

```
        w=2*pi/tau(i);
        dt=t(2)-t(1);
        ind_1=min(floor(del(i)/dt+1),length(t));
        gain_1(1:ind_1)=0;
        A=amp(i);
        b=del(i);
        expon=exp(-(t-b)/tau(i));
        sinusoid=sin(w*(t-b));
        smooth=w*(t-b);
        p=p+(A.*expon.*sinusoid.*smooth).*gain_1;
    end
    ntc=0.1;
    gain_2=ones(size(t));
    ind_2=ceil(ntc/dt);
    gain_2(ind_2:end)=0;
    Hann=F*0.5*(1-cos(2*pi.*t/ntc)).*gain_2;
    p=p+Hann;
end

    % Function for fmincon to optimize
function e=opt_err(x,tau,t,freq,T,Q,dB)
    C1=50000;
    C2=0;
    C3=10000;
    x_1=x(1:(length(x)-1)/2);
    x_2=x((length(x)-1)/2+1:(length(x)-1));
    x_3=x(end);
    % Generate pulses
    a=pulse(x_1,x_2,tau,t,x_3);
    % Calculate Kinematics
    [~,vel_1,~]=kinematics(a,t);
    vel_ref=0;
    % Calculate SRS
    s=SRS(a,freq,T,Q);
    % Calculate reference SRS
    r=SRS_Reference(freq,dB);
    % Calculate error between reference and optimized SRS
    e=SRS_Error(s,r)+...
        C1*(x_2*x_2')+...
        C2*(x_1*x_1')+...
        C3*abs(vel_1(end)-vel_ref);
end

    % Function for generating the reference SRS and the bounds
function [r,r_upper,r_lower]=SRS_Reference(freq,dB)
    b=log(8.888889/40)/log(10/45);
    a=40/(45^b);
    i=1;
    S=freq*0;
    u=freq*0;
    l=freq*0;
    for f=freq
        if f<45
            S(i)=a*f^b;
        else
```

```
            S(i)=40;
        end
        u(i)=S(i)*10^(dB/20);
        l(i)=S(i)*10^(-dB/20);
        i=i+1;
    end
    r=S';
    r_upper=u';
    r_lower=l';
end

    % Function for calculating the error between the optimized and
    % reference SRS
function e=SRS_Error(s,r)
    v=s-r;
    e=v'*v;
end

    % Function for calculating the SRS of a time history
function s=SRS(a,freq,T,Q)
    N=length(a);
    n=1;
    y=freq*0;
    srs=freq*0;
    for f=freq
        omega_n=2*pi*f;
        P=(omega_n*T)/(2*Q);
        R=omega_n*T*(1-(4*Q^2)^-1)^0.5;
        beta_0=1-exp(-P)*sin(R)/R;
        beta_1=2*exp(-P)*(sin(R)/R-cos(R));
        beta_2=exp(-2*P)-exp(-P)*sin(R)/R;
        alpha_1=-2*exp(-P)*cos(R);
        alpha_2=exp(-2*P);
        i=1;
    while i<N+1
        if i<3
            y(i)=beta_0*a(i);
        else
            y(i)=beta_0*a(i)+beta_1*a(i-1)+beta_2*a(i-2)-...
                alpha_1*y(i-1)-alpha_2*y(i-2);
        end
        i=i+1;
    end
        maximax=max(abs(y));
        srs(n)=maximax;
        n=n+1;
    end
    s=srs';
end
```