2026 NDIA MICHIGAN CHAPTER
GROUND VEHICLE SYSTEMS ENGINEERING
AND TECHNOLOGY SYMPOSIUM
MODELING, SIMULATION, PROTOTYPING, AND VALIDATION (MSPV) TECHNICAL SESSION
AUG. 11-13, 2026 - NOVI, MICHIGAN

MOTION APPROACH FOR TURRET SYSTEM PHYSICAL SIMULATION

VICTOR PAUL[1], ANDREW HOELSCHER[1], AHMED TIGUERT[2], AND HARRY ZYWIOL[2]

[1]US Army DEVCOM GVSC MI
[2]DCS Corporation, Warren, MI

ABSTRACT

Physical simulation permits government and contractor engineers to characterize, validate and test a complex weapon system's many components and subsystems. This is particularly important when new sub-systems and components are in the prototype development stage and integrated into a full weapon system for the first time. This paper presents a comprehensive methodology in creating a motion environment to adequately test a functional turret system in a lab environment using the GVSC's Crew Station Turret Motion Base Simulator. The simulator is a high-capacity, 6-degrees-of-freedom test device that utilizes computer-controlled hydraulic actuators and a platform to reproduce dynamic conditions encountered by a combat vehicle turret system traversing off-road terrain. The paper presents an iterative technique using the simulator's measured frequency response functions to achieve a targeted response. Platform error and repeatability are presented which is key for "baseline vs. modified" studies.

Citation: Victor Paul, Andrew Hoelscher, Ahmed Tiguert, and Harry Zywiol "Motion Approach for Turret System Physical Simulation," In *Proceedings of the Ground Vehicle Systems Engineering and Technology Symposium* (GVSETS), NDIA, Novi, MI, Aug. 11-13, 2026.

1. INTRODUCTION

In a significant strategic pivot for its ground forces, the U.S. Army is overhauling its approach to modernizing its fleet of combat vehicle systems. This new strategy moves away from continuous incremental upgrades on legacy platforms and instead prioritizes the development of next-generation systems. This shift is heavily influenced by observations from recent global conflicts, which have highlighted the urgent need for vehicles that are more survivable, mobile, and sustainable on the modern battlefield. Incorporating advanced features such as sophisticated gun-turret drives, weapon stabilization, and automated ammunition handling systems, these next-generation vehicles represent a substantial leap in capability. Consequently, their development will demand intensive prototyping and testing, supported by advanced physical simulations in controlled

laboratory environments to validate system-level performance before field deployment.

The Army's Ground Vehicle Systems Center (GVSC) located in Warren, MI has been conducting physical simulation of military vehicles using motion base simulators since the 1960's [1]. In 1990, a unique capability for the Army, the Crew Station/Turret Motion Base Simulator (CS/TMBS) was designed and installed at GVSC. This simulator is a high-capacity, 6-degrees-of-freedom (DOF) test device that utilizes computer-controlled hydraulic actuators and a platform to reproduce dynamic conditions encountered by combat vehicle turret systems traversing off-road terrain. It has a payload capacity exceeding 25 tons thus enabling the characterization of even heavy turret systems. The simulator can be programmed with different operating modes that permit field-recorded data playback and analytical model-derived motion. This gives engineers the capability to simulate concept and prototype subsystem designs through production turrets.

Over the years, simulation and turret system engineers have developed methodologies and techniques to determine how to best use the simulator for gun/turret testing and analysis. Engineers provide precise six degree-of-freedom motion commands to the CS/TMBS simulating hull motion at the simulator platform interface that mimics vehicle hull motion over selected terrain/speed dynamic scenarios. This enables gun turret drive (GTD) engineers to measure and analyze on-the-move GTD performance in key axes, both elevation and azimuth. Autoloader performance can also be observed and recorded.

The main advantages of laboratory testing over field testing are flexibility, repeatability, efficiency, and affordability. Xu [2] et al enumerates six advantages. The ability to rapidly change test profiles provides significant flexibility and test efficiency that cannot be found in field experiments. Laboratory testing is repeatable and reproducible over time. [3] Stabilization component and subsystem developers can insert new prototype components and observe early results using laboratory testing well before full system development. Analytical models of these components and subsystems can be tuned and validated. The advantage of multi-physics modeling and simulation is its ability to leverage experimental mechanical investigations into the interconnectivity of the boundary conditions

2. MOTION DRIVE CREATION

As illustrated in **Figure 1**, the CS/TMBS can simulate the tank-turret dynamics while traversing a wide range of terrain surfaces and their impact on the system response.

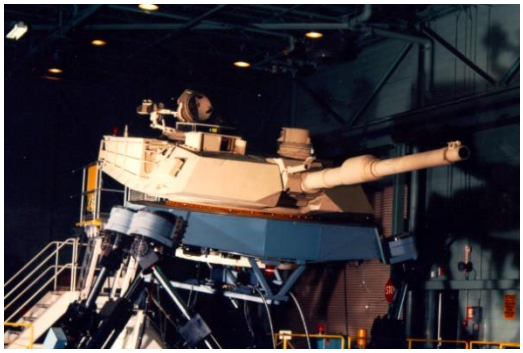

Figure 1: Crew Station Turret Motion Base Simulator outfit with heavy turret system.

[4]. Six electro-hydraulic actuators support a platform that enables a wide range of tank turrets with payloads of more than 25 tons. Electronic controls and controller software [5], [6] permit the closed-loop control in a safe and controlled manner.

This paper presents a methodology utilized on a recent heavy turret test to achieve accurate target acceleration and velocity data for use on a motion base simulator. It presents an iterative method to refine the simulator response, reduce errors, and improve accuracy. The results of this methodology are presented. Repeatability of the simulator response over time is also presented. This permits turret system engineers to accurately assess baseline vs. modified designs without other

undesirable factors such as weather, course changes and driver variability.

In this effort, it is desired to reproduce field-recorded accelerations and velocities of a tank turret system in the simulator lab facility. The goal of this portion of the project was to create drive files for the simulator that would result in the closest representation of the motion profile of a combat vehicle traversing a proving ground course.

The drive files are necessary as they contain the format required by the simulator's control system. The drive file process essentially consisted of two steps. First, manipulation of field data sets is needed to create target or desired acceleration profiles for the simulator. Secondly, an iteration process is formulated to manipulate the drive file until the simulator response closely represents the desired acceleration response.

Payload Analysis

The authors conducted an analysis to determine if the CS/TMBS could be used in an overladen capacity of 55,000 lb. while being operated safely and retain the designed safety-factor for the structural integrity of the platform. The CS/TMBS was originally designed for a 50,000 lb. payload maximum with displacements, velocities, and accelerations shown in **Table 1**.

Table 1: Simulator Maximum Capability at 50,000 lb. Payload

Axis	±Displ	± Velocity	± Acceleration
Long (Tx)	0.76 m	1.76 m/s	17.6 m/s²
Lat (Ty)	0.76 m	1.76 m/s	17.6 m/s²
Vert (Tz)	0.76 m	1.76 m/s	58.8 m/s²
Roll (Rx)	0.35 rad	1.22 rad/s	51 rad/s²
Pitch (Ry)	0.35 rad	1.22 rad/s	31 rad/s²
Yaw (Rz)	0.35 rad	1.22 rad/s	34 rad/s²

The CS/TMBS structural a Design Criteria is 50,000 lbs payload with a maximum acceleration of 5g and safety factor of 2.0.

Additional design criteria include a safety factor of 4.0 with the worst-case loads of 4 g vertical and 3 g arbitrarily directed with simultaneous 10 rad/sec^2 acceleration.

Statistics were calculated and evaluated from the field data sets provided. Maximum accelerations from these data were found to be considerably less than the limits of the motion base shown in Table 1. Even with the increased payload, the motion profiles produce accelerations and forces well below the simulator design limits thus preserving the 2X safety factor.

Target File Creation

The authors were provided with representative data of a vehicle traversing various terrain profiles. The data were recorded from real vehicle runs commonly referred to as field data. These sets contained three (3) gyro signals for angular rate (roll, pitch and yaw) of the vehicle along with three (3), three-axis accelerometer signals to give longitudinal, lateral and vertical accelerations in 3 distinct turret locations. The full data set is listed in **Table 2Error! Reference source not found.** and the accelerometer location and orientation in **Figure 2** below.

Table 2: Field Data Acquisition Channels

Channel Name	Orientation
Turret Pitch Gyro	+down
Turret Roll Gyro	+right
Hull Yaw Gyro	+cw
Gun Trun Accel (Tri-axial)	X Fwd
	Y Down
	Z Left
Cmdr Accel (Tri-axial)	X Fwd
	Y Right
	Z Down
Loader Accel (Tri-axial)	X Fwd
	Y Up
	Z Right

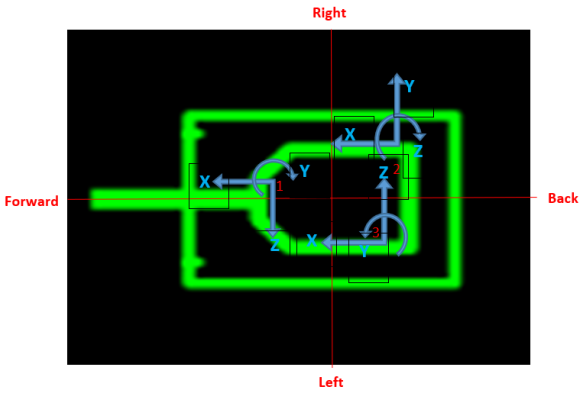

Figure 2: Accelerometer locations and orientations.

The CS/TMBS utilizes an SAE® coordinate system (**Figure 3**) and International System of units (SI). The axis are defined as: x is fore-aft (positive forward); y is lateral (positive to the right); z is up/down (positive down); roll is rotation around the x-axis (positive to the right) ; pitch is rotation around the y-axis (positive gun tube up); and z is rotation around the z-axis (positive clockwise).

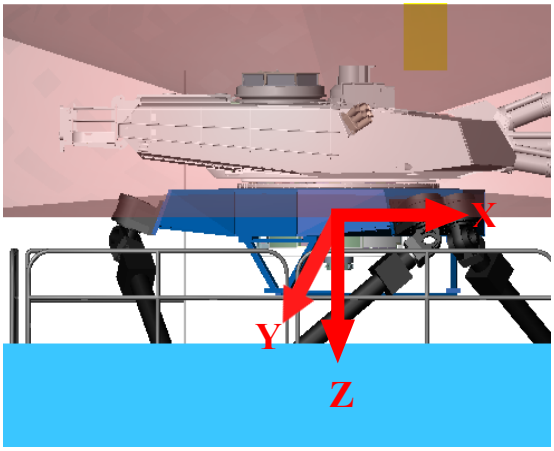

Figure 3: CS/TMBS Coordinate System and Orientation.

The next step was to convert the field data to the correct coordinate system and units that the simulator accepts to create drive files. Ultimately the goal of this data manipulation was to create a 6-DOF acceleration time history of the field data oriented about the center of the turret basket at the turret/platform interface plane. To do this, simulation engineers performed the following operations utilizing MATLAB® [7] scripting and existing MATLAB based processing tools:

1. Convert field gyro and acceleration signals into correct units and orientation.
2. Calculate three angular accelerations by taking two-sided derivatives of converted rate signals.
3. Translate all acceleration signals from their original location to the center of the turret race ring at the hull/turret interface plane.
4. Average the translated linear accelerations and create 3 new acceleration signals (Average longitudinal, lateral and vertical accelerations).
5. Create new file with average of translated accelerations (X, Y, Z) and calculated angular accelerations (roll, pitch, yaw).
6. Trim the beginning and end of the datafile and fade in/out all channels with 1s sinusoidal fade. This step is taken to delete unnecessary data (data before and after the actual drive) and to ensure that all channels start/stop at zero.
7. Filter all the channels at 10Hz with a 3rd Order Butterworth Low-Pass Filter. A filter cutoff frequency of 10Hz was chosen due to the acceleration response bandwidth of the longitudinal and lateral axis of the simulator. All channels are filtered at the same frequency to avoid different phase shifts across the channels.
8. Compare created drive files with field data.

These new files now become the "Target" responses that simulation engineers would try to replicate with the CS/TMBS. The MATLAB® script that was used to process steps 1-5 above is included in the appendix.

Collision Analysis

Due to the size of the unit under test—particularly the length of the cannon—it was critical to identify any platform positions that could cause the turret system to collide with the laboratory's infrastructure. To assess this risk, a CAD model of the platform and turret system was developed. This model incorporated simulator kinematics alongside full turret rotation, elevation, and depression capabilities.

By simulating expected platform positions and orientations, the model evaluated potential impact risks with existing laboratory equipment. The analysis confirmed that most platform positions and dynamic motion profiles posed little to no risk of a collision. One of the platform positions analyzed is depicted in **Figure 4**. However, composite platform orientations involving simultaneous roll and pitch presented a risk of gun tube interference. Consequently, these specific angular positions required modification in the test plan.

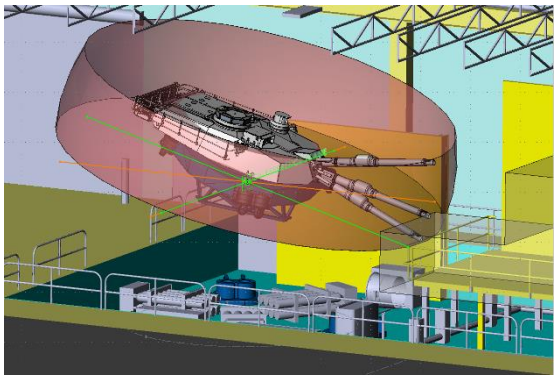

Figure 4: CAD Model Depicting Full Turret Rotation/Gun Tube Travel while Platform is Pitched Forward 15 Degrees

Drive File Iteration

Once the target file is generated, the second major step in drive file creation is the iteration process, which incorporates control approaches formally known as Time Waveform Replication (TWR) and Iterative Learning Control (ILC). As the name suggests, this is an iterative process in which the input command or "drive files" are generated to closely match the response of our target files. To achieve this, simulator engineers utilized Moog™ Inc.'s Replication software [8].

Before running "iterations", the simulator is tuned to optimize simulator performance, stability, and repeatability. However, properties such as oil type, mass, mechanical wear, and resonance attributes influence simulator responses non-linearly. This is where iterations are especially useful. Instead of tuning the entire system for a very specific set of signals, we can provide the system with a "target file" and incrementally modify the input or "demand file" to achieve a drive file that, when played into the simulator, has a measured response that closely resembles the desired response (target file). This allows simulation engineers to maintain a robust tuning set while closely mimicking unique target files.

Iteration Process

To begin the iteration process, a target file and a frequency model of the system are provided to the system's software application. The model (simulator frequency response) is obtained through the following process:

1. Play an input signal (multi-spectral white noise) into each of the simulator DOF's.
2. Measure the response of the simulator to the input from step 1.
3. Calculate a 6x6 transfer function matrix (H(f)) based on measured vs demanded acceleration. This transfer function (H(f)) not only measures the direct transfer function for each axis (diagonal of matrix) but also considers the cross-coupling of the simulator and measures the transfer functions of each axis due to an input in the other axis (off-diagonal matrix elements).

The critical step in obtaining an appropriate model is determining the frequency range of the input signal. The goal is to ensure that the frequency band of the target signal is well within the frequency range of the excitation (input) signal without including frequencies that are beyond the frequency region of interest. This ensures that frequencies of the target signal are represented within the input signal while excluding irrelevant frequencies that will only result in longer model generation and iteration times. Being that the target files were filtered at 10Hz, it was determined that the upper range of the system identification would be 30 Hz. The resultant 6x6 transfer function Matrix is used to obtain the system model. The diagonal elements of the transfer function matrix

are illustrated in **Figure 5**.

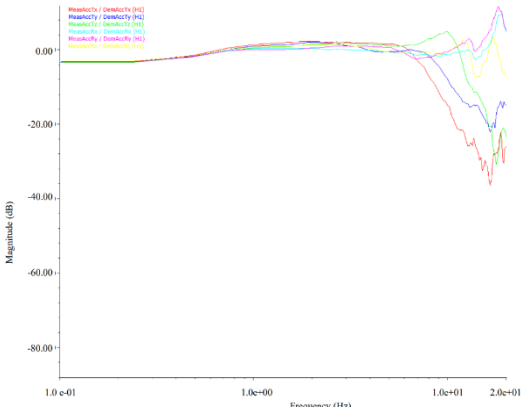

Figure 5: Frequency response of each degree of freedom.

Once the target file and system model have been generated, the iteration process (iterations) can be performed to refine the drive file. The core of the ILC algorithm is answering the inverse question: "To achieve a desired output, what is the required input?". The iterative loop undergoes the following process:

1. Error Calculation: During iterations, the target and system response are compared to generate an "error signal" that is fed into the inverse of the system model previously generated.
2. Corrective Signal Generation: The inverse model is a mathematical representation of the system's input over its output. By utilizing the system's inverse model, output signals can be multiplied with the inverse model in the frequency domain to obtain input signals. Here, the controller utilizes the error signal to calculate a correcting signal that is based off the system model.

3. Drive File Update: The resultant signals are then scaled and added to the drive of the previous iteration. Typically, the correction is attenuated by a factor between 0.1 and 0.7 to prevent overcorrection and instability. This results in a drive file that considers the system model, target signal, and previous response. The system response is then used as feedback for the next iteration. A loop diagram of the iteration process can be seen in **Figure 6**.

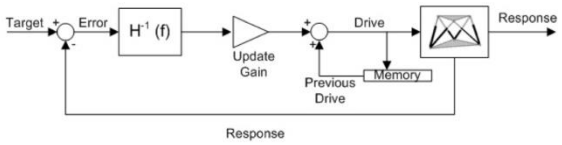

Figure 6: Iteration Process.

This process was continued until the root-mean-square (RMS) of the error signal (measured vs target) for the primary DOFs (Vertical, Roll, Pitch) fell below 10% and the subsequent signal for the secondary DOFs (Longitudinal, Lateral, Yaw) dropped to acceptable values. The values of the secondary signals are orders of magnitude smaller than the primary values which greatly diminish their effect on the system's fidelity. Hence, they are of lesser importance. The target percentages were reached after iterating 27 times. Root mean squared (RMS) error plots of the six degrees of freedom for each iteration are shown in **Figure 7**.

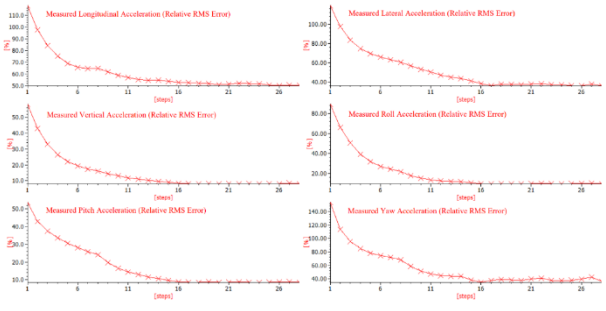

Figure 7: Error plots for each iteration.

A "demand" file was generated using the command signals from iteration 27. Statistics recorded from the 27th iteration of the 6-graph chart above help illustrate the accuracy of the final drive as shown in **Table 3**.

Table 3: Comparison of Desired and Actual Platform Response

Channel Units	Target Min Max	Response Min Max	Target RMS Std Dev	Response RMS Std Dev
Long (+fwd) m/s^2	-2.72 3.57	-2.99 3.66	0.84 0.84	0.92 0.92
Lat (+right) m/s^2	-2.51 2.06	-2.57 0.45	0.45 0.45	0.46 0.46
Vert (+down) m/s^2	-6.92 5.31	-7.00 1.78	1.78 1.78	1.77 1.77
Roll (+right) r/s^2	-3.23 2.87	-3.20 0.54	0.54 0.54	0.54 0.54
Pitch (+up) r/s^2	-2.57 2.47	-2.63 0.79	0.79 0.79	0.78 0.78
Yaw (+cw) r/s^2	-0.89 0.47	-0.94 0.11	0.11 0.11	0.12 0.12

A time plot of the simulator's response (measured) vs target (command) acceleration is shown in **Figure 8** along with the error signal.

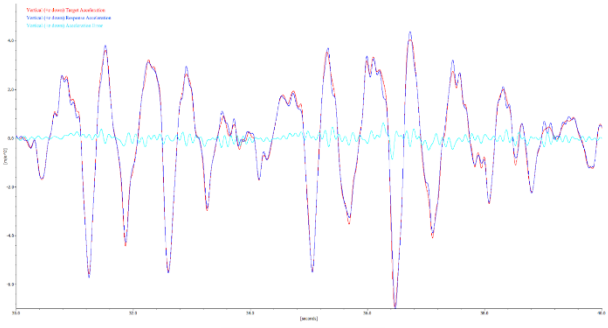

Figure 8: Time plot of simulator response, command, and error.

The demand file from this last iteration becomes the final drive file and can be played as an instruction is a sequence file. A sequence file is a set of serial and/or parallel instructions that are executed by the system's motion controller. This sequence file included the final drive file repeated several times and a data recording. This allowed the simulation engineers to create a longer test period from the shorter (as little as 45sec) field data files. The data recordings were especially useful in substantiating the high degree of repeatability of the simulator.

SIMULATOR REPEATABILITY

The ability of a motion base simulator to repeat a response over time is a very desirable and useful feature. A repeatable motion response enables comparison of baseline (original) to modified (changed) scenarios. Highly repeatable simulators thus ensure that variations in turret system test results are due to intentional modifications rather than uncontrolled experimental errors. High repeatability strengthens the validity of findings and makes them more likely to be acceptable to the US Army development community.

The CS/TMBS is a very repeatable platform. To document the simulator's repeatability for this testing, simulation engineers looked at data from several runs taken throughout the test period. Visual and statistical analysis shows that there is little difference in the acceleration data between runs. As seen in **Figure 9** the vertical response for a sampling of 6 runs completed during the test over a two-month period are repeatable (9/17-11/18). As one can see in the plots the vertical acceleration across these runs is nearly identical.

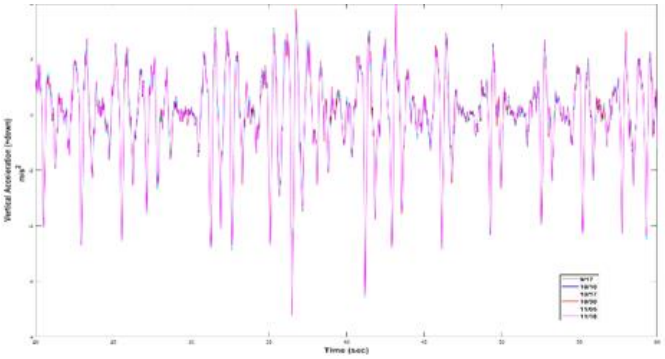

Figure 9: Vertical acceleration repeatability over time.

In addition to the visual comparison, simulation engineers statistically analyzed simulator response over a period of 2 months. Using the first data recording of this period as the baseline data, an Error for the Root-Mean-Square of the platform

acceleration response was calculated for each of the six axes. The percentage difference from the baseline was also calculated and averaged across 21 additional runs to determine the average percent error per axis. Now these axis errors were averaged across the primary axes of interest (vertical, roll and pitch) and the secondary axes for an average percentage acceleration RMS Error of 0.99% (primary) and 6.76% (secondary). **Table 4** shows the average per axes statistics across the runs analyzed and further highlights the repeatability of the simulator. One should note that the RMS errors in the X, Y, and Yaw axis are higher than the other axes because these axes contain very low amplitude accelerations. Therefore, residual noise in those response data creates high error values.

Table 4: Repeatability Statistics

Axis	Baseline RMS	Ave RMS	Std Dev	Ave Axis RMS Error (%)
X (m/s^2)	1.3716	1.2713	0.1015	6.90%
Y (m/s^2)	0.7527	0.7710	0.0466	4.55%
Z (m/s^2)	1.5779	1.5778	0.0071	0.34%
Roll (r/s^2)	0.5397	0.5352	0.0063	2.28%
Pitch (r/s^2)	0.7282	0.7261	0.0023	0.35%
Yaw (r/s^2)	0.2554	0.2391	0.0178	8.83%

3. CONCLUSIONS

This paper presented a comprehensive methodology that produces accurate motion environment to a gun turret drive weapon system in a laboratory environment. It provides results of such methodology using time and frequency plotting and analysis. Error and repeatability data and results are presented. The GVSC's Crew Station Turret Motion Base Simulator features six degree-of-freedom motion that permits weapon systems characterization in a moving vehicle environment.

4. REFERENCES

[1] J. M. Dasch and D. J. Gorsich, The TARDEC Story: Sixty-five Years of Innovation, 1946-2010, Warren, MI: U.S. Army Tank Automotive Research Development and Engineering Center, 2012.

[2] Xu, D. Wong, P. LeBlanc and G. Peticca, "Road Test Simulation in Light Vehicle Development and Durability Evaluation," in SAE World Congress, Detroit, 2005.

[3] B. LaRose, M. Morgan, K. Stark, D. Kosinski, K. Fischer, M. Brudnak, G. Schultz, A. Christino, M. Wayne, I. Baseski, "Testing in a Complex World", In Proceedings of the Ground Vehicle Systems Engineering and Technology Symposium (GVSETS), NDIA, Novi, MI, Aug. 13-15, 2019.

[4] D'Onofrio, D., and Kuang, Z., "Multi-physics Approach in Sub-System Modeling & Simulation (M&S) For a Gun Turret Drive Weapon System.," SAE Technical Paper 2024-01-3205

[5] Ground Vehicle Systems Center, "Crew Station Turret Motion Base Simulator Safety Assessment Report", Dec 15, 2022.

[6] Moog Inc., "System Design Report, RMS and CSTMBS Controller Upgrade v2", 2022

[7] The MathWorks, Inc. (2025). MATLAB version:(R2025b). https://www.mathworks.com.

[8] Moog B.V., "User Manual Moog Replication 3.32," Nieuw-Vennep, The Netherlands, 2021.

5. CONTACT INFORMATION

Victor J. Paul
Technical Expert
Immersive Simulation

Ground Vehicle Systems Center
6501 E Eleven Mile Rd
FCDD-GVS-MSP MS157Warren MI 48397-0001
USA
victor.j.paul2.civ@army.mil

6. ACKNOWLEDGMENT

The authors recognize the support of the GVSC Durability Test Laboratory without whom this work would not have been possible.

7. ACRONYMS

DEVCOM	U.S. Army Combat Capabilities Development Command
GVSC	Ground Vehicle Systems Center
GTD	Gun Turret Drive
CS/TMBS	Crew Station/Turret Motion Base Simulator
DOF	Degree of Freedom
RMS	Root Mean Squared
PID	Proportional Integral Derivative

DEVCOM	U.S. Army Combat Capabilities Development Command
GVSC	Ground Vehicle Systems Center
GTD	Gun Turret Drive
CS/TMBS	Crew Station/Turret Motion Base Simulator
DOF	Degree of Freedom
RMS	Root Mean Squared
PID	Proportional Integral Derivative

FF	Feed Forward

Appendix

The following script was used to process the field data to be used on the simulator. The recorded field data was imported into a data structure used by a data processing tool created by GVSC-IS utilizing a collection of Matlab Scripts. The variable DataFile represents the top level of this structure.

```
function processDriveFile()

% Function to take field data from General Dynamics and create a drive file
% to run on the GVSC Motion Base Simulators.

% Assumptions:
% 1. The data is already readable by MUNCH and has been loaded.
% 2. The channels are in the following order:
%      1.  Turret Pitch Gyro
%      2.  Turret Roll Gyro
%      3.  Hull Yaw Gyro
%      *********Tri-axial Accel #1********
%      4.  Gun Trun Accel X FWD
%      5.  Gun Trun Accel Y DOWN
%      6.  Gun Trun Accel Z Left
%      *********Tri-axial Accel #2********
%      7.  Cmdr Accel X FWD
%      8.  Cmdr Accel Y RIGHT
%      9.  Cmdr Accel Z DOWN
%      *********Tri-axial Accel #3********
%     10.  Loader Accel X FWD
```

```
%      11.  Loader Accel Y UP
%      12.  Loader Accel Z RIGHT
%
%% Processing steps:
% 1. Translate all of the data to motion base coordinate system (x:fwd, y:right, z:down)
% 2. Convert to motion base units (m/s, rad/s^2)
% 3. Integrate the each of the rate gyros
% 4. Translate the Accelerometer data to the center of the race ring @ the
% hull turret interface plane
% 6. Filter the data
% 7. Average the 3 x, y and z accels.
% 8. Create the drive and save as .fstrm

global DataFile Active NumFiles angAccel;

inch2meter = 0.0254; %inches to meters
g2mpss = 9.80665; %Gs to m/s^2
mils2rad = 0.000982; %NATO mils to rad

vecAB(1,:) = [-40, -18, 22]*inch2meter; % Vector from Trunion to Race Ring
vecAB(2,:) = [44, -30, 36.6]*inch2meter; % Vector from Cmdr's Pos to Race Ring
vecAB(3,:) = [28, 41, 8.5]*inch2meter; % Vector from Ldr's Pos to Race Ring

axesName{1,:} = {'Tr X1 Ac','Tr Y1 Ac','Tr Z1 Ac'};
axesName{2,:} = {'Tr X2 Ac','Tr Y2 Ac','Tr Z2 Ac'};
axesName{3,:} = {'Tr X3 Ac','Tr Y3 Ac','Tr Z3 Ac'};

aveName = {'Av X Ac','Av Y Ac','Av Z Ac'};

%Copy file to a new working file
prevFile = Active;
NumFiles = NumFiles+1;
DataFile(NumFiles) = DataFile(Active);
Active=NumFiles;

%**************Convert Gyro Units and put in Roll,Pitch, Yaw order**********************
roll = -DataFile(Active).Chaninfo(2).Data*mils2rad;
pitch = -DataFile(Active).Chaninfo(1).Data*mils2rad;
yaw = DataFile(Active).Chaninfo(3).Data*mils2rad;

% Put the data in Roll, Pitch, Yaw order
temp = DataFile(Active).Chaninfo(1);  %put the pitch Channel info into a temp variable
DataFile(Active).Chaninfo(1).Data = roll;
DataFile(Active).Chaninfo(1).LongName = DataFile(Active).Chaninfo(2).LongName;
```

```
DataFile(Active).Chaninfo(1).ShortName = DataFile(Active).Chaninfo(2).ShortName;

DataFile(Active).Chaninfo(2).Data = pitch;
DataFile(Active).Chaninfo(2).LongName = temp.LongName;
DataFile(Active).Chaninfo(2).ShortName = temp.ShortName;

DataFile(Active).Chaninfo(3).Data = yaw;

for i=1:3
DataFile(Active).Chaninfo(i).UnitName = 'rad/s';
end

%Get All Accel channels in the proper orientation and units
%*********Tri-axial Accel #1********
% X Axis (Longitudinal) - Positive Fwd
DataFile(Active).Chaninfo(4).Data = DataFile(prevFile).Chaninfo(4).Data*g2mpss;
DataFile(Active).Chaninfo(4).LongName = 'Gun Trun X Accel';
DataFile(Active).Chaninfo(4).ShortName ='GT X ACC';
% Y Axis (Lateral) - Positive Right
DataFile(Active).Chaninfo(5).Data = -DataFile(prevFile).Chaninfo(6).Data*g2mpss;
DataFile(Active).Chaninfo(5).LongName = 'Gun Trun Y Accel';
DataFile(Active).Chaninfo(5).ShortName ='GT Y ACC';
% Z Axis (Vertical) - Positive Down
DataFile(Active).Chaninfo(6).Data = DataFile(prevFile).Chaninfo(5).Data*g2mpss;
DataFile(Active).Chaninfo(6).LongName = 'Gun Trun Z Accel';
DataFile(Active).Chaninfo(6).ShortName ='GT Z ACC';
%*********Tri-axial Accel #2********
% X Axis (Longitudinal) - Positive Fwd
DataFile(Active).Chaninfo(7).Data = DataFile(prevFile).Chaninfo(7).Data*g2mpss;
DataFile(Active).Chaninfo(7).LongName = 'Cmdr X Accel';
DataFile(Active).Chaninfo(7).ShortName ='Cmd X AC';
% Y Axis (Lateral) - Positive Right
DataFile(Active).Chaninfo(8).Data = DataFile(prevFile).Chaninfo(8).Data*g2mpss;
DataFile(Active).Chaninfo(8).LongName = 'Cmdr Y Accel';
DataFile(Active).Chaninfo(8).ShortName ='Cmd Y AC';
% Z Axis (Vertical) - Positive Down
DataFile(Active).Chaninfo(9).Data = DataFile(prevFile).Chaninfo(9).Data*g2mpss;
DataFile(Active).Chaninfo(9).LongName = 'Cmdr Z Accel';
DataFile(Active).Chaninfo(9).ShortName ='Cmd Z AC';
%*********Tri-axial Accel #3********
% X Axis (Longitudinal) - Positive Fwd
DataFile(Active).Chaninfo(10).Data = DataFile(prevFile).Chaninfo(10).Data*g2mpss;
DataFile(Active).Chaninfo(10).LongName = 'Ldr X Accel';
DataFile(Active).Chaninfo(10).ShortName ='Ldr X AC';
```

```
% Y Axis (Lateral) - Positive Right
DataFile(Active).Chaninfo(11).Data = DataFile(prevFile).Chaninfo(12).Data*g2mpss;
DataFile(Active).Chaninfo(11).LongName = 'Ldr Y Accel';
DataFile(Active).Chaninfo(11).ShortName ='Ldr Y AC';
% Z Axis (Vertical) - Positive Down
DataFile(Active).Chaninfo(12).Data = -DataFile(prevFile).Chaninfo(11).Data*g2mpss;
DataFile(Active).Chaninfo(12).LongName = 'Ldr Z Accel';
DataFile(Active).Chaninfo(12).ShortName ='Ldr Z AC';

for i=4:12
   DataFile(Active).Chaninfo(i).UnitName ='m/s^2';
end

% Calculate Angular Acceleration

      delta_t = 2*DataFile(Active).Step;

       for chan = 1:3

          for samp = 1:DataFile(NumFiles).Nsamp

             if (samp == 1) delta_x = DataFile(Active).Chaninfo(chan).Data(3) -
DataFile(Active).Chaninfo(chan).Data(1);
             elseif (samp == DataFile(Active).Nsamp) delta_x = DataFile(Active).Chaninfo(chan).Data(samp) -
DataFile(Active).Chaninfo(chan).Data(samp-2);
             else delta_x = DataFile(Active).Chaninfo(chan).Data(samp+1) -
DataFile(Active).Chaninfo(chan).Data(samp-1);
             end

             angAccel(chan,samp) = delta_x/delta_t;

          end

       end

%Translate the accelearation to the center of the race ring
for k = 1:3
% Translate the accelearation to the center of the race ring for each of the three accel positions
       for i =1:DataFile(Active).Nsamp
          localAngVel = [roll(i) pitch(i) yaw(i)];
          localAngAccel = [angAccel(1,i) angAccel(2,i) angAccel(3,i)];

          accelA(i,1:3) = [DataFile(Active).Chaninfo(3*k+1).Data(i)
DataFile(Active).Chaninfo(3*k+2).Data(i) DataFile(Active).Chaninfo(3*k+3).Data(i)];
```

```
        accelB(i,1:3) = translateAccelByOffset( accelA(i,1:3), vecAB(k,:), localAngVel, localAngAccel);
    end

    % Create new translated accel channels for each of the accel
    % positions and assign correct units, LongName, ShortName
  for j = 1:3
    DataFile(Active).Chaninfo(DataFile(Active).Nchan + 1).Data = accelB(:,j);
    DataFile(Active).Chaninfo(DataFile(Active).Nchan + 1).LongName = append('Translated ',DataFile(Active).Chaninfo(3*k+j).LongName);
    DataFile(Active).Chaninfo(DataFile(Active).Nchan + 1).ShortName = axesName{k}{j};
    DataFile(Active).Chaninfo(DataFile(Active).Nchan + 1).UnitName = DataFile(Active).Chaninfo(3*k+j).UnitName;

    DataFile(Active).Nchan = DataFile(Active).Nchan + 1;

  end

end
  % Average all of the Translated X, Y & Z accels and create 3 new
  % "Average" channels
  for j = 1:3

    X = [DataFile(Active).Chaninfo(12+j).Data,DataFile(Active).Chaninfo(15+j).Data,DataFile(Active).Chaninfo(18+j).Data];
    DataFile(Active).Chaninfo(DataFile(Active).Nchan+1).Data = mean(X,2);

    switch j
      case 1
        DataFile(Active).Chaninfo(DataFile(Active).Nchan + 1).LongName = append('Ave Translated X Accel');
      case 2
        DataFile(Active).Chaninfo(DataFile(Active).Nchan + 1).LongName = append('Ave Translated Y Accel');
      case 3
        DataFile(Active).Chaninfo(DataFile(Active).Nchan + 1).LongName = append('Ave Translated Z Accel');
    end
    DataFile(Active).Chaninfo(DataFile(Active).Nchan + 1).ShortName = aveName{j};
    DataFile(Active).Chaninfo(DataFile(Active).Nchan + 1).UnitName = DataFile(Active).Chaninfo(12+j).UnitName;

    DataFile(Active).Nchan = DataFile(Active).Nchan + 1;
```

end