Software Tools for Developing and Simulating the NASA LaRC CMF Motion Base

Richard B. Bryant, Jr.*
NASA Langley Research Center, Hampton, Virginia, 23681

and

David J. Carrelli†
Swales Aerospace Inc., Hampton, VA, 23681

The NASA Langley Research Center (LaRC) Cockpit Motion Facility (CMF) motion base has provided many design and analysis challenges. In the process of addressing these challenges, a comprehensive suite of software tools was developed. These tools simulate the end to end operation of the motion base system and provide facilities for software-in-the-loop testing, mechanical geometry and sensor data visualizations, and function generator setup and evaluation. This paper describes these tools and the system architecture to which they interface. The tool suite consists of five key elements. The first is the Data Visualization System (DVS) which is used to collect and graphically visualize motion base sensor data, electronically store collected data to file, provide a user interface to program function generators for creating repetitive drive signals and trajectories, and to optionally provide a permanent strip chart record. Secondly, the geometric visualization module provides an accurately scaled, shaded, three dimensional representation of the motion base structure permitting intuitive human visualization of the system's motions and geometric component clearances and interactions. Thirdly, a detailed model used to study dynamic performance and perform load analysis can also be interfaced into the architecture, providing the ability to generate complex final leg and platform trajectories while simulating closed loop operations. This same model also contains the closed loop compensator algorithms from which the embedded control law code is directly generated. Fourthly, a file replay module is included to allow the review of recoded data files using the DVS or the geometric visualization module. Finally, these elements are interfaced to one another through shared memory which also emulates the actual reflective memory hardware used to interface DVS to the embedded Digital Control Unit (DCU) during real world motion base operation.

Nomenclature

CMF	=	Cockpit Motion Facility
DCU	=	Digital Control Unit
DOF	=	Degree of Freedom
DVS	=	Data Visualization System
ESTOP	=	Emergency Stop
LaRC	=	Langley Research Center
MBM	=	Motion-Base Model

* Electrical Engineer, Flight Simulation and Software Branch, MS 125B, NASA Langley Research Center, Hampton, VA, 23681, AIAA Member.
† Engineer V, Flight Research Dept, MS 125B, NASA Langley Research Center, Hampton, VA 23681, AIAA Member.

I. Introduction

THE NASA Langley Research Center (LaRC) Cockpit Motion Facility (CMF) motion base has provided many design and analysis challenges. In the process of addressing these challenges, a comprehensive suite of software tools was developed. The software tools development began with a detailed MATLAB®/Simulink® model of the motion base which was used primarily for safety loads prediction, design of the closed loop compensator and development of the motion base safety systems[1]. A Simulink model of the digital control law, from which a portion of the embedded code is directly generated, was later added to this model to form a closed loop system model. Concurrently, software that runs on a PC was created to display and record motion base parameters. It includes a user interface for controlling time history displays, strip chart displays, data storage, and initializing of function generators used during motion base testing. Finally, a software tool was developed for kinematic analysis and prediction of mechanical clearances for the motion system. These tools work together in an integrated package to support normal operations of the motion base, simulate the end to end operation of the motion base system providing facilities for software-in-the-loop testing, mechanical geometry and sensor data visualizations, and function generator setup and evaluation.

II. Tool Integration using Shared Memory

A. Motion base computer architecture

Three computer systems are involved in the flight simulation, motion base control and motion base data recording associated with the CMF motion base. A Host computer performs the flight simulation and generates the desired motion base trajectory. A VME based Digital Control Unit (DCU) transforms the host trajectory into leg trajectories, and generates the valve commands necessary to control the actuator dynamics. A PC Data Visualization System (DVS) monitors and records motion base parameters. The Host computer transmits data to the DCU using a shared memory network, and the DCU transmits parameters to the DVS computer using a separate shared memory network.

The Host computer is actually a system of computers and is collectively referred to as the Host computer. The motion base trajectory is generated in the Host computer. The Host also includes the systems required for gathering pilot inputs, computing simulation states, performing motion cueing filtering, and driving all display systems. For the purpose of this paper, the entire Host Simulation System of computers, can be though of as one computer which generates the desired motion base trajectory. The trajectory consists of the six degree of freedom positions, velocities and accelerations which are transmitted to the DCU at typically 40 to 80 Hz. The Host computer system is not expanded upon in this paper, and is only considered as the source of motion base trajectories.

The DCU is a VME based computer using the priority based interrupt driven VxWorks® real-time operating system to coordinate the motion base control task and the various data acquisition and communication tasks. The DCU receives the Host trajectory, and generates and controls the corresponding actuator trajectories. The DCU performs the closed loop actuator control at 2 kHz, and uses the velocity and acceleration components of the 40 to 80 Hz host trajectory to predict future higher rate leg trajectory data. The DCU also generates up to seven leg space or degree of freedom (DOF) space function generators which iterate at 2 kHz. These function generators can be controlled by the Host, or used without the Host computer for motion base testing. The resulting motion base trajectory is the sum of the 2 kHz propagation of the Host trajectory and the output of the seven function generators. The DCU compensator algorithm can utilize actuator position, actuator velocity, actuator acceleration, platform acceleration, platform rotational rates, and hydraulic pressure feedbacks. The DCU sends the current motion base states back to the host, in response to receiving a host trajectory update. The DCU also transmits a record of motion base parameters to the DVS system at a nominal rate of 100Hz. For tests that require higher resolution data, the DCU is configured to transmit data to the DVS system at up to 1 kHz. The data record written by the DCU and read by the DVS computer consists of 178 floating point doubles including a DCU generated time stamp, the requested trajectory from the simulation host computer, all motion base sensors, and selected control system parameters computed by the digital control law running on the DCU. The record format is defined in the "CMF Motion Base Control System Design Description"[2].

The PC based DVS computer is used by the motion base operator to view selected motion base parameters on a graphical time history display, to save motion base data to a local or networked drive and to optionally drive a paper strip chart recorder with real-time parameters. For maintenance operations, the DVS is also used to set up motion base trajectories using the DCU function generators without the use of a Host computer. The DVS system is described in more detail in Section III.

B. Shared memory in the system architecture

The shared memory communication mechanism for the all of the CMF software applications grew from the communication design for the DCU and the DVS. In this design, the DCU writes records into a shared memory ring buffer, updating a head point to indicate the newest record, and the DVS reads records, updating a local tail pointer to track the last record processed. In this architecture the DCU is the only writer and the DVS is the only reader. Additional software tools were developed which behaved as record writers or readers. Record writing applications write to shared memory, and only one writing application can run at a time. Record reader applications read from shared memory, and multiple readers can run simultaneously.

The DCU uses two shared memory interfaces. One shared memory interface is used to communicate with the Host computers. This memory network may consist of over a dozen computers, and includes mechanisms for synchronized, bidirectional interrupt driven communication. The second shared memory is used to communicate with the DVS system, and is the shared memory which is used by the motion base software tools. The DCU to DVS shared memory in used for a ring buffer for the one way flow of time history data from the DCU to the DVS and for a single structure used to initialize the DCU function generators. The DCU to DVS ring buffer allows for non-deterministic behavior of the PC based DVS computer. The DCU to DVS communication uses its own shared memory interface, so that motion base data processing and Host computer network communications do not interfere.

As far as the DCU is concerned, the ring buffer is simply a storage location for data records. The DCU does not track the processing of the ring buffer data by any other program. Once the buffer is full, the DCU will begin overwriting the buffer from the beginning. When the DCU is powered up, it will initialize the shared memory region, and is the only system to initialize the shared memory. Since it does not track the processing of the data, the DCU will run with or without the presence of the DVS system. This enhances the operational reliability of the motion base system since the DVS system is not a necessity for normal operations.

The function generator data structure is initialized by the DCU with all function generators in the off state and all waveform parameters set to zero. The DVS system includes a user interface allowing the motion base operator to initialize the function generators. When the function generator user interface is accessed, the DVS will initialize the function generator controls by reading the data currently stored in the shared memory. While the DVS is generally a reader of the ring buffer, it will write the function generator structure when the motion base operator applies new function generator settings and maintenance (non Host driven trajectory) motion base operation is selected. Under these conditions the DCU reads the function generator setup buffer once, each time the operator requests that the motion base transition to the drive state. The DCU then generates the trajectory using the function generator settings. The DCU will not read function generator setup buffer again until the motion base is recyled by the operator.

Given the fixed size of the ring buffer, the fixed size of the data records and the constant sample rate the ring buffer holds a fixed duration of data. With the current configuration of a 64 MB shared memory buffer, 178 floating point doubles in each record which are collected at a nominal 100Hz rate, the ring buffer holds about seven minutes of data. When the system experiences an unexpected Emergency Stop (ESTOP), the DCU will write the ring buffer to disk as a record of the previous seven minutes of motion base operations. The DCU will actually collect data for an additional 30 seconds after the ESTOP event so that the time prior to and immediately after an unexpected shutdown is always recorded.

The DCU is configured to boot, operate and save ESTOP data using a solid state disk configured on a 256 MB compact flash memory. This limits the amount of disk space for ESTOP files. With the application code and the operating system saved on the flash disk, there is space for up to three 64 MB ESTOP files. The DCU will keep the last three ESTOP data files and always overwrite the oldest file. If detailed analysis of the ESTOP event is warranted, the ESTOP data is transferred off of the VxWorks® based DCU to the DVS system for additional analysis.

The ring buffer communication mechanism is encapsulated in a C++ class designed for a single writer (e.g. the DCU) application which uses a shared memory head pointer, and any number of reader applications (e.g. the DVS) which maintain their own tail pointers. When the DCU writes records to the shared memory, it maintains a head pointer to the next available record within the circular buffer. This pointer is stored within the shared memory data structure. The DVS system maintains a tail pointer to the last record that it processed. The tail pointer is stored in non shared local memory. When the DVS initializes, it sets the tail pointer equal to the shared memory head pointer. The DVS then uses a timer to invoke a method to test the pointers for equality. If the pointers are not equal, the DVS knows that the DCU has stored a new record, and the DVS then processes the data at the tail pointer, which is then incremented. It repeats this process until the pointers are equal. A key element of this communication is that the two

application programs use a common C++ class allowing the DCU to initialize and update the shared memory head pointer, and the DVS to initialize and update its own local tail pointer.

Correct order of shared memory initialization and writer failure detection is ensured by the reader application. The reader implementation includes a watchdog function allowing the readers to detect if the shared memory writer (DCU or model) has initialized, and to detect if the writer has stopped updating the head pointer. If the writer has not initialized, the reader (DVS) will not initialize its tail pointer until the writer has initialized and started incrementing its head pointer. If a timeout value is exceeded, the reader detects that the writer has stopped updating records and the reader reverts to waiting for initialization.

C. Shared memory emulation by the tools

The shared memory communication class was developed to configure a local block of system memory to emulate the shared memory if the shared memory hardware is not detected in the system in order to facilitate unit testing of the DVS software. All subsequent software tools using the shared memory communication class use the same ring buffer protocol and technique of emulating shared memory hardware in local system memory when the shared memory hardware is not found. This allows all of the software tools to be used on the motion base computers, employing shared memory hardware, or on any desktop computer, emulating the shared memory hardware with a named block of system memory. The first additional software tool was a simple DCU emulator used for unit testing the DVS software that initialized the shared memory partition and wrote records to the shared memory partition at 100Hz. This shared memory emulation technique allowed the DVS application to be unit tested on a desktop computer without tying up hardware resources. As additional tools were developed the shared memory emulation allowed all of the software to run on any desktop computer, allowing anyone to use the actual system software on their desktop computer for review of recorded data or parameter selection in preparation for a motion base test. The main application programs will report, in their title bar, if the shared memory hardware is in use or if system memory is being used to emulate the hardware. Figure 7 shows a title bar with shared memory hardware in use and Figure 1 shows a title bar with the system memory is emulating shared memory hardware.

III. Data Visualization System

The Data Visualization System (DVS) is used to collect and graphically visualize motion base sensor data, electronically store collected data to file, provide a user interface to program function generators for creating repetitive drive signals and trajectories for maintenance operations, and to optionally provide a permanent strip chart record. The DVS system was used for all data recording while testing the performance of motion base safety systems[3].

The Data Visualization System is a Microsoft Windows based computer which is used by the motion base operator. In a typical scenario, the motion base operator views the trajectory requested by the Host computer prior to engaging the motion base with a new simulation program. Once valid trajectories are being generated, the motion base operator engages the motion system and can observe selected trajectory, feedback or control parameters using the DVS computer. The DVS computer allows the operator to selectively display up to 36 parameters on graphical time histories, configure and drive up to 16 channels on a paper strip chart recorder and write all parameters to disk. The DVS application display is shown in Figure 1 with three time history displays containing six channels each. The DVS system generates comma separated value files that can be reviewed using the DVS time history displays, or loaded into any data analysis software package which accepts comma separated value files.

The DVS system is the motion base operator's interface for viewing motion base parameters and generating files which are used for motion base performance analysis. As such, the DVS system requires a user friendly graphical interface, the ability to write and access networked files, and perform file and plotter IO. These requirements are well suited to a Windows operating system. The DVS should generally maintain continuous update rates, but is not generally considered to be or required to be deterministic. The shared memory communication allows the hard real-time VxWorks® based DCU to record and timestamp the data and then store it in shared memory for the non real-time Windows based computer to display and process. As long as the buffer is large enough to store records during non deterministic interruptions of the DVS, the data stream remains intact. The Windows based DVS typically processes DCU records at 100 Hz, transmitting up to 16 channels of strip chart data through a digital interface, and 178 doubles to disk, and up to 36 channels of data on graphical time history displays, without noticeable

interruptions. The ring buffer allows for approximately 7 minutes of data storage before the oldest records are overwritten. As long as the windows based DVS can keep up with data processing, the oldest data is always processed, before it is overwritten.

For some tests, the data recording rate is increased from 100 Hz to 1 kHz. The DCU is already processing data at 2 kHz, and it is only a slight increase is demand to transmit 1 kHz data to the shared memory. The Windows based DVS can also process 1 kHz data without any noticeable degradation in performance. The data files get proportionally larger, and the ring buffer only holds approximately 45 seconds of data once the records are spaced one millisecond apart. The main implication of this higher data rate is that the Emergency Stop data recording no longer records 7 minutes of data prior to the event and 30 seconds after the ESTOP event. The higher rate causes only 15 seconds of data prior to the event and 30 seconds after the event. High data rate recording is only used on system tests which require the higher resolution data.

The DVS application is multi-threaded to allow the various IO processes to take place without holding each other up. The main thread handles the user interface and drawing of the time history displays. Another thread performs the writing of data to disk, and an additional thread manages the communication with the strip chart recorder. This multi-threaded design prevents pauses in the processing of data due to disk block transfers or plotter issues including power off, paper out or paper jams.

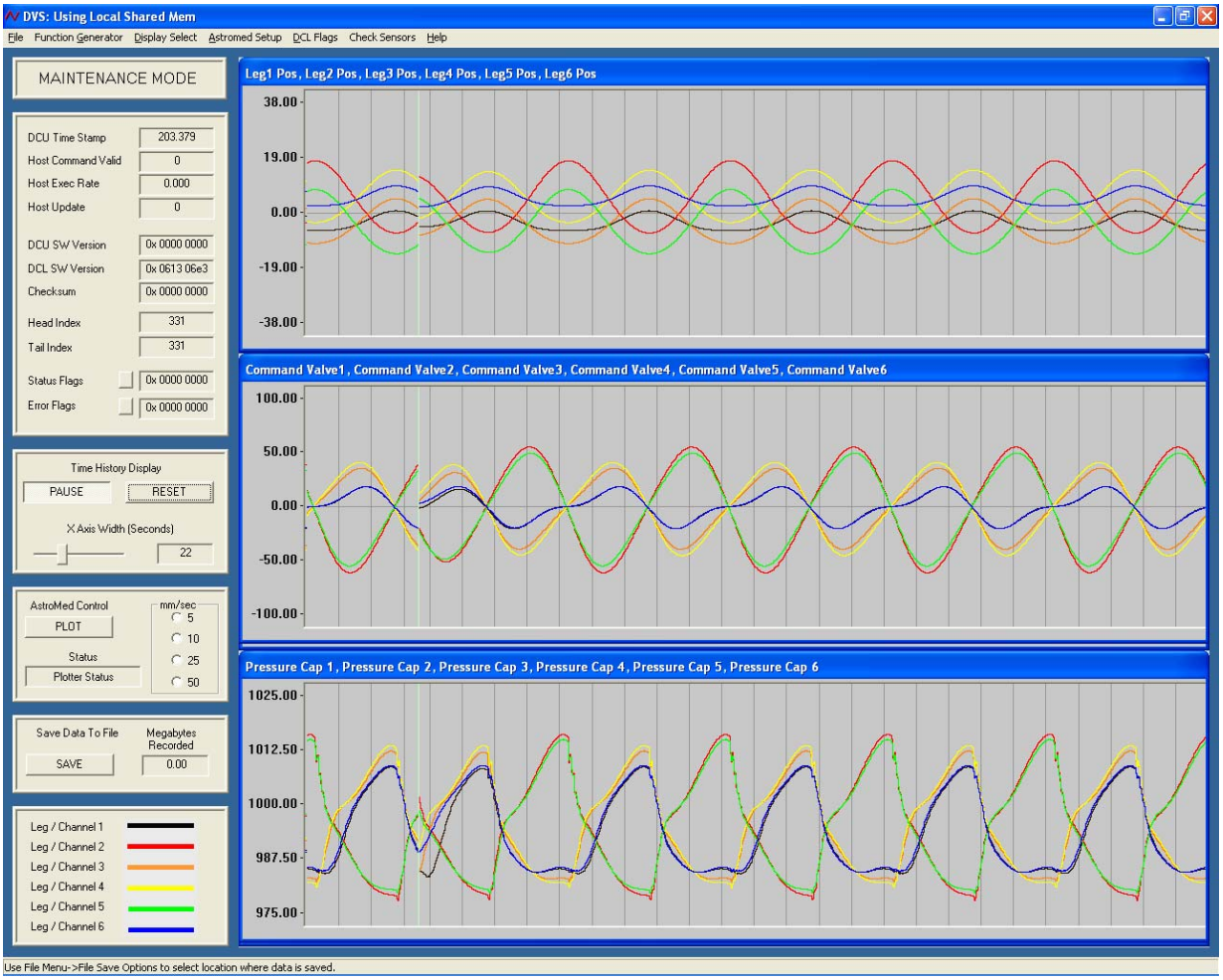

Figure 1 DVS Application showing model parameter time history

IV. Detailed System Model

A MATLAB®/Simulink® system model, consisting of a detailed open loop motion-base model (MBM), digital control law (DCL) model, and a simplified DCU model, was also integrated into the tool suite. This allows the complete closed loop system to be simulated, utilizing the same DVS interface and visualization tools as would be used when running the actual hardware.

The system model depicted in Figure 2 may be run within the Simulink® environment, if interactive adjustment of model parameters is desired, or a Windows PC compatible stand-alone executable may be generated using the Real-Time Workshop® toolbox. This permits the use of the system model in environments which do not have a MATLAB license. This is useful for visualizing and checking potential motion-base test trajectories on non development machines. As the simulation runs in soft real-time, i.e. one second of simulation takes approximately one second of real time, the DVS displays and function generator setup interactions are identical to those of the real system.

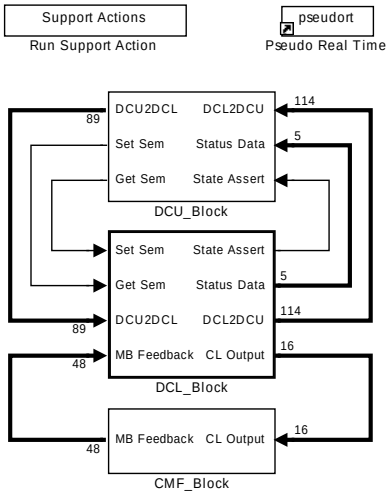

Figure 2 System model block diagram

The simplified DCU model is responsible for providing the shared memory interface and routing simulated outputs into the shared memory structure exactly as in the real DCU. The DCU model employs the same technique of using a block of system memory if the shared memory hardware is not detected, thus allowing the model to be run in concert with the other tools over a shared memory network, or locally on a single machine. This model is obviously a writer of shared memory data. As such it is run in conjunction with the DVS or other applications that read the shared memory ring buffer. Since only one writer of ring buffer data can run at any given time the system model is never run on any of the hardware connected to the DCU when the VxWorks® DCU software is also running.

To avoid having to model the operator control console the DCU model also includes state transitions which emulate motion base operator inputs. When the system model starts, the DCU model immediately sends the DCL a command to transition from the full down position to the standby position (all actuators at mid stroke) and then finally commands the DCL to the drive state. The DCU model then monitors the shared memory function generator data structure. When a structure update is detected it will automatically cycle the DCL to standby and immediately back into drive thus forcing the DCL into using the new function generator setups and following the trajectory as if the motion base operator had command the actions.

The DCL model is responsible for controlling the motion-base transitions based on DCU input, collecting system data, providing function generator trajectories, and performing control loop compensation. Using Real-Time Workshop® this model provides, without modification, the algorithm source for the direct generation of the embedded DCL code which runs on the VxWorks® based DCU computer. This facilitates rapid and easy modification of the control law, while ensuring that the algorithms used in simulation are identical to those used in the embedded code. The operation of the DCL is for the most part transparent to the user however; its inclusion into the system model allows software-in-the-loop testing of the function generators and compensator algorithms.

The MBM simulates the open loop motion-base hydraulic, and payload systems[4]. The model contains a full non-linear open loop model of the motion base actuators including hydraulic cushion effects. It has been used extensively in dynamic analysis for the design of the motion-base control system, prediction of maximum loads, and determining factors of safety in the mechanical design. Its main role in the full system model is to simulate feedback data for use in the other tools of the software suite, particularly for viewing in DVS, and to drive the geometric visualization module discussed in the next section.

By way of an example, note that Figure 1 shows coordinated position, valve command and pressure time histories, which are all generated from the system model. The valve commands are generated from the desired trajectories which are a combination of sinusoidal degree of freedom function generators. The position trajectories are the results of the MBM, being driven closed loop with the shown DCL generated valve commands. Similarly the pressures shown, are the pressure outputs of the MBM, and clearly demonstrate its non-linear characteristics.

V. Geometric Visualization Module – CMF Kinematics

The CMF Kinematics application calculates all bearing and actuator angles for a given pose and provides a geometric visualization using an accurately scaled, shaded, three dimensional representation of the motion base. This application was used to define the requirements for the universal joint bearings and has been used for an intuitive visualization of the system's motions and geometric component clearances and interactions.

The CMF Kinematics application was initially developed to compute the cascaded universal joint bearing angles and the unit independent dexterity[5] based on user input degree of freedom positioning, leg lengths or calculated trajectories. This model was used to sweep the 64 extreme combinations of fully extended and retracted actuators in order to identify the maximum bearing angle requirements. For each input pose, all angles are calculated, displayed, and optionally written to a record in a data file. When the application is driven with a calculated trajectory, the data file contains corresponding trajectories of bearing angles, actuator angles, pose dexterity, and rotation between the rod and cylinder of each actuator.

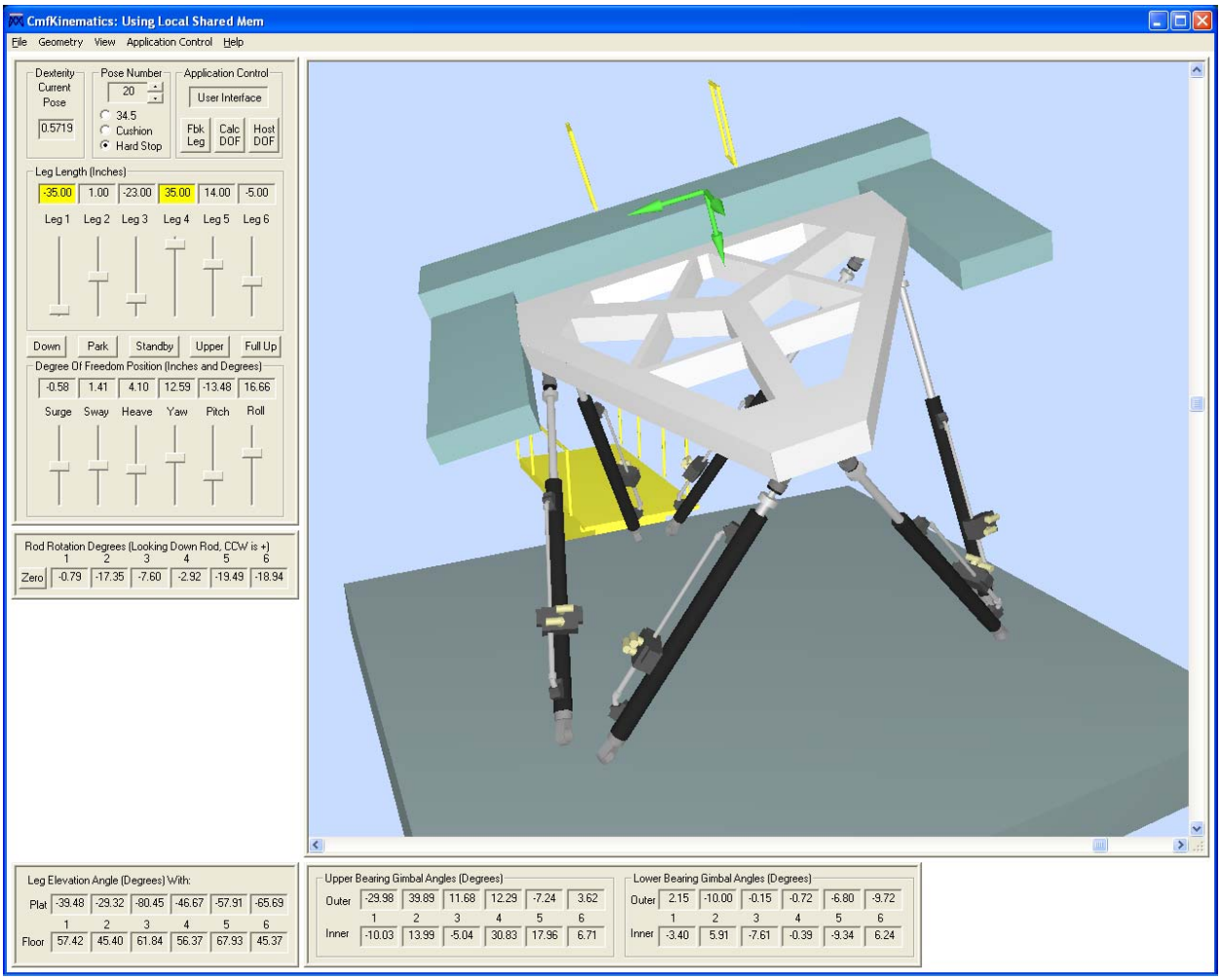

Figure 3 CMF Kinematics Application shown with user controls and parameter displays

The CMF Kinematics application was extended to include mesh based animation of the motion base actuators, hydraulic manifolds, hard pipes and the outer mold lines of the catwalk which is installed on the moving platform. For catwalk clearance issues, the animation was viewed as the motion base was swept through the 64 extreme poses, and areas of concern were identified. The catwalk hardware was modified based on the animation results and areas which were extremely close (<1" clearance) were identified and tested on the motion base during system shakedown.

The CMF Kinematics application was finally extended to be used as an operational visualization tool by including the shared memory interface class and a user interface option to take the input motion base pose from the shared memory records or from the application controls. When taking the pose from the shared memory ring buffer, the pose can be selected from the measured leg lengths, the calculated DOF positions or the Host computer's demanded DOF positions. These input options allow real-time calculation of the kinematic parameters and visualization of the motion base in operation, from a recorded data file, or from the Host requested DOF prior to running a motion base simulation. CMF Kinematics is a shared memory reader maintaining its own tail pointer, so it can run in conjunction with any other motion base software either on the motion base hardware or any desktop computer.

CMF Kinematics can be used as a visualization tool for the system model running in soft real time. The ability to visualize a motion base trajectory on a desktop computer is useful when observing the trajectories resulting from combinations of function generators, such as when setting up function generator parameters for a specific test.

VI. Miscellaneous modules - File Replay, Split, Simple DVS Simulation

A. Purpose of additional modules

Three additional relatively minor software applications complete the software suite. They were developed to use the common shared memory interface class, so they work with or without shared memory hardware installed in the system.

B. DVS Replay

A file replay module is included to allow the review of recoded data files using the DVS or the CMF Kinematics geometric visualization module. The user interface is depicted in Figure 4. DVS replay will initialize the shared memory data structures and then allow the user to select a data file for replay. The file is parsed in order to determine the total time interval of the data file and then it is loaded into system memory. The replay user interface includes a slider which allows a quick review of the data and a quick selection of the starting point for a replay session. The controls include standard Play, Rewind and Pause controls as well as the ability to select any subset of the file for looped playback.

When replaying a data file or when dragging the slider control to a specified location with the file, the recorded data is replayed into the shared memory ring buffer allowing the data to be viewed using either DVS or CMF Kinematics. The file replay application is a standalone executable which can be executed from the File menu of DVS or CMF Kinematics. The DVS replay module is a shared memory writer, so it should not be run with the other shared memory writers.

When Play is selected, the records are written into the shared memory ring buffer at a nominal rate of one record every 10 ms. This results in real-time like playback of the 100 Hz data rates. DVS replay can also optionally play back data at a slower rate of one record every 100ms, to allow closer observation of the data.

DVS, CMF Kinematics and DVS Replay are intended for quick look or presentation types of playback of recorded data. Detailed analysis is better performed using other commercial packages.

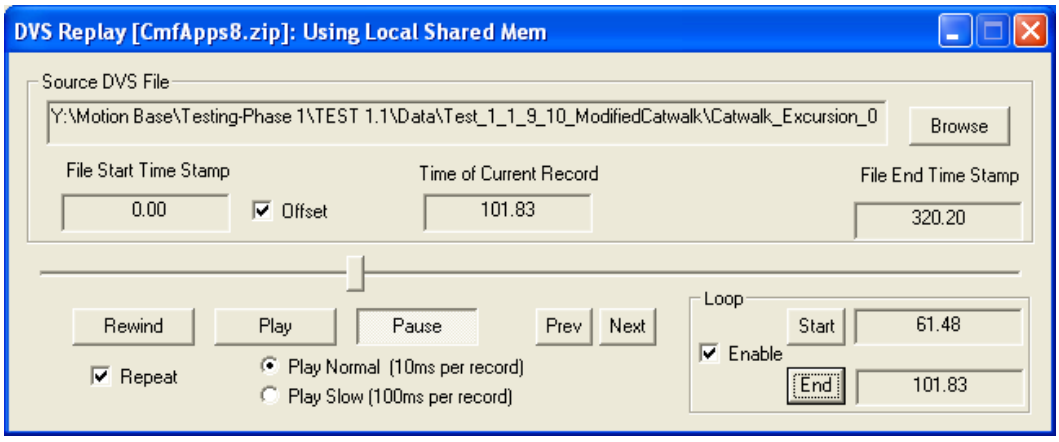

Figure 4 DVS Replay user interface

C. Simple DCU Simulation

The simple DCU simulation uses the common shared memory class to initialize and fill shared memory with records of unique waveforms. Figure 5 shows the Simple DCU Simulation running DVS configured with six time history plots, each containing six signals. The waves are in no way representative of actual motion base data, but are unique to allow unit testing of the DVS and CMF Kinematics software applications.

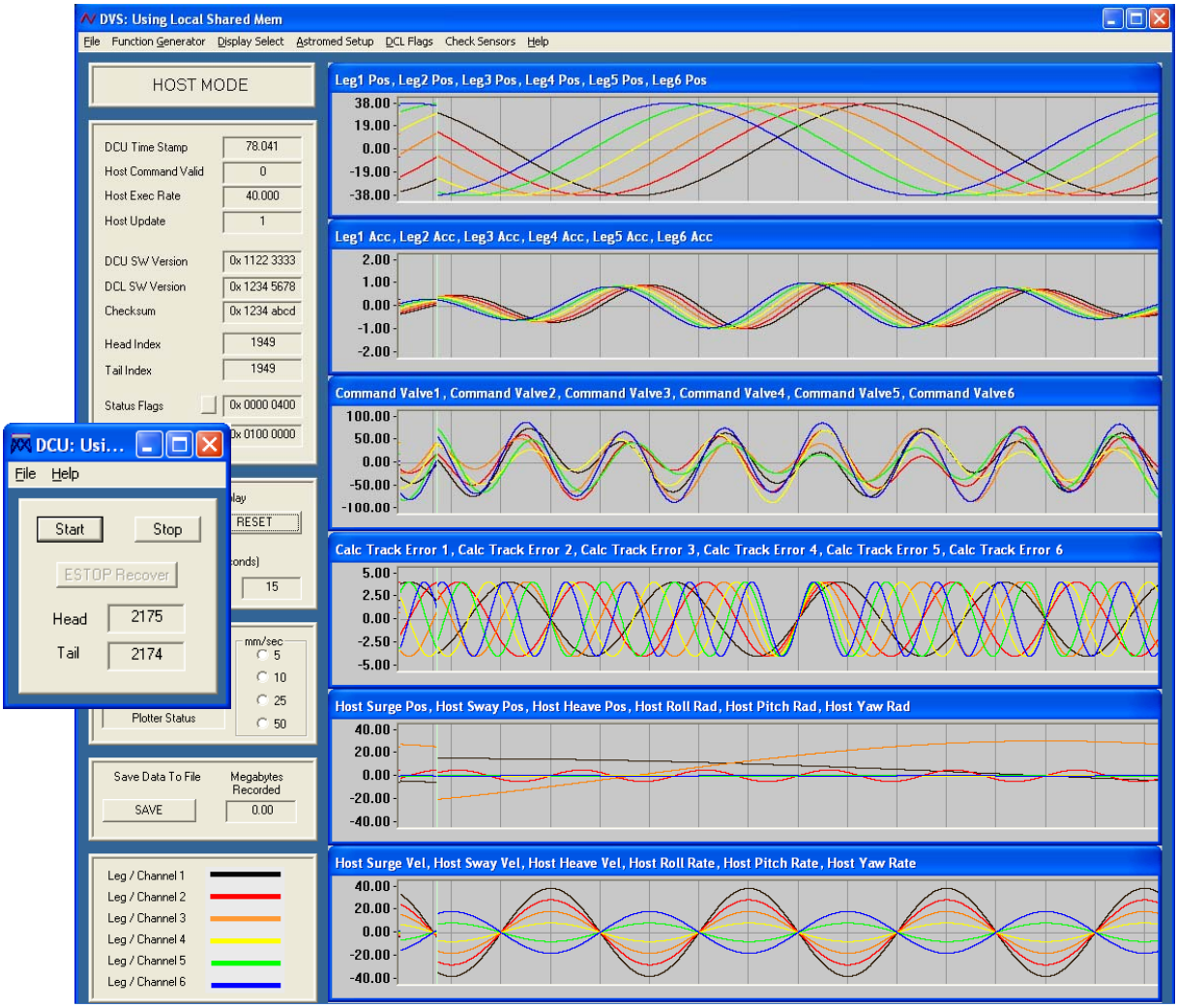

Figure 5 Simple DCU Simulation driving DVS unit testing

D. DVS Split

The DVS Split module is used to break up a large data file into smaller files. It can be used to remove fields from each record and to break one large file containing many records into a number of smaller files with each containing fewer records than the original data file. This was initially developed for Excel based analysis, since MS Excel uses a maximum of 64K records. This can also ease data analysis processing and memory usage when only a small portion of the data file is of interest.

The DVS Split user interface, Figure 6, is set up to allow the quick selection of all parameters grouped by actuator or sensor type. Actuator groups are selected using the column of "Leg N" controls. Sensor or calculation groups are selected using the row of controls starring with the "Leg Position" control. The resulting files will include records beginning with the time stamp and containing only the parameters of interest over the time range of interest.

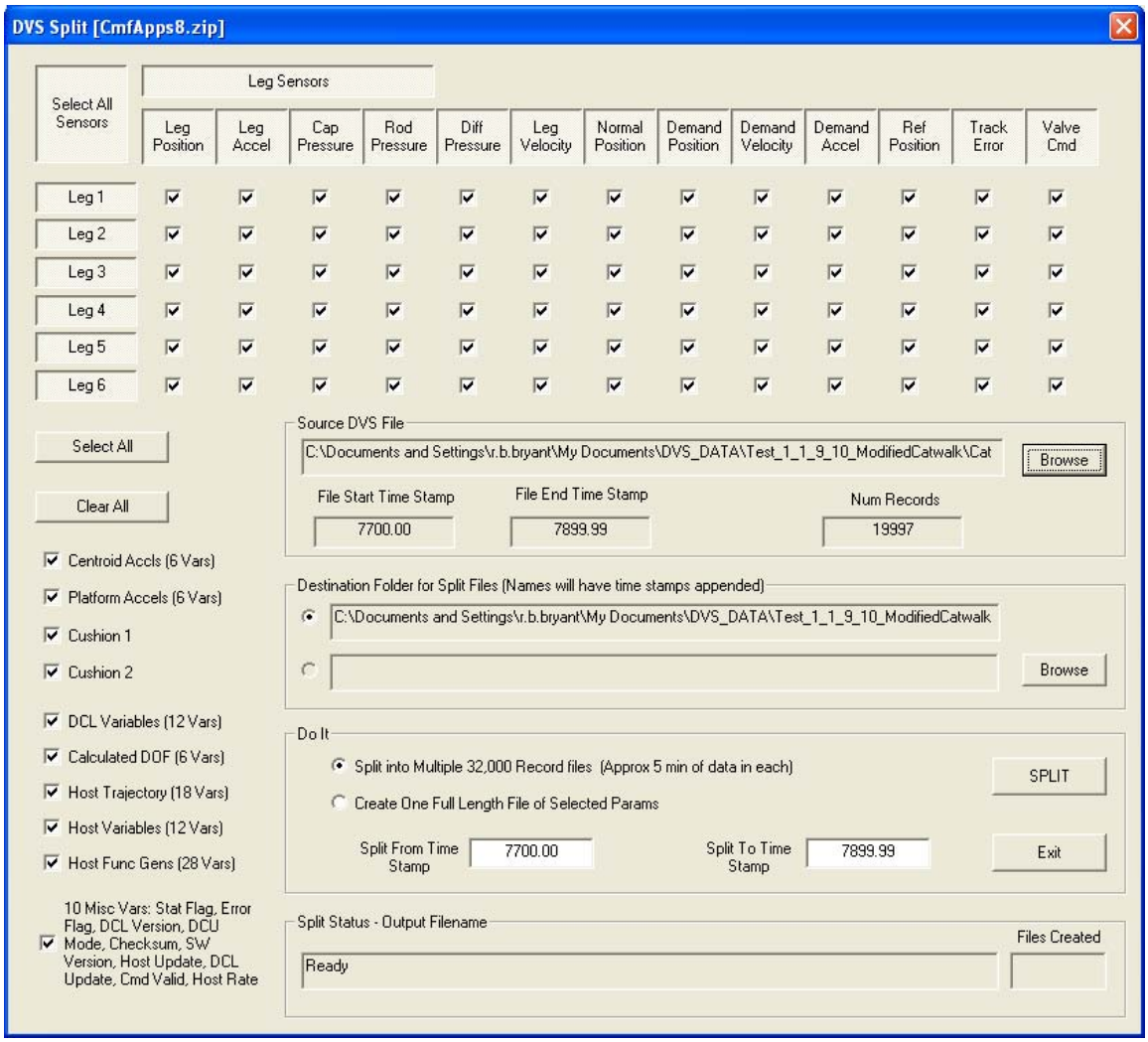

Figure 6 – CMF Split user interface

VII. Conclusion

The CMF control system design team has successfully developed software tools to model motion base dynamics, visualize the geometry and kinematic motion, and to display and record data, during simulation, at run time, and during post-analysis. These tools are all integrated using a common shared memory interface which allows them to operate seamlessly on the motion base hardware or on any desktop computer.

When used in conjunction with the motion base hardware, these tools facilitate operational and maintenance operations. A typical configuring is shown in Figure 7 using DVS and CMF Kinematics. The configuration of application windows includes the DVS setup with fourteen parameters shown on three time history displays and the CMF Kinematics visualization with all parameter windows turned off.

When used on the desktop, these tools allow for the visualization of recorded data, the setup and prediction of motion base trajectories, including time histories of dynamic parameters, and the ability to perform software-in-the-loop testing of modules used in the operation of the CMF motion base. These tools have proven to be very effective in testing software, predicting motion base dynamic and kinematic performance and reviewing motion base test data.

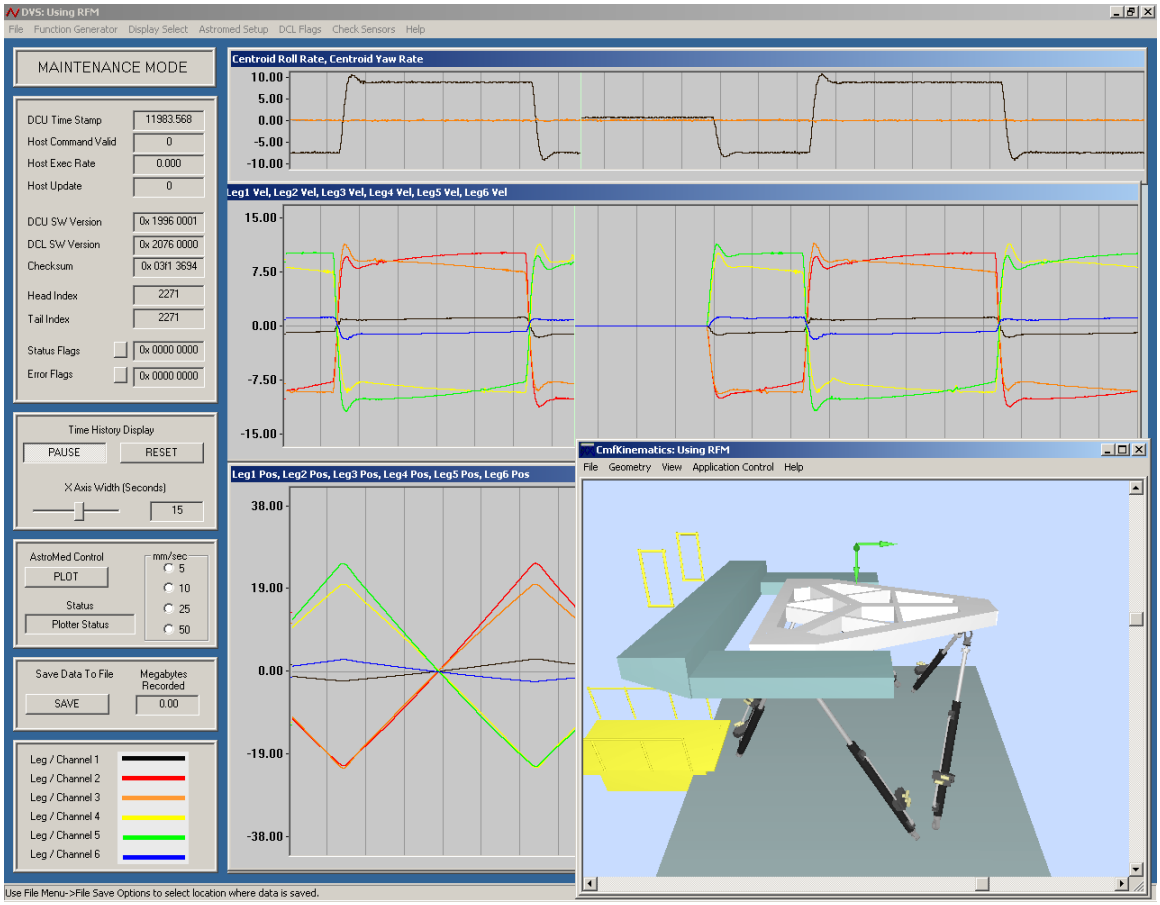

Figure 7 – Typical configuration of DVS and CMF Kinematics

References

[1] Bryant, R. Barry; Gupton, Lawrence E.; Martínez, Debbie; Carrelli, David J., "Mitigating Motion Base Safety Issues – The NASA LaRC CMF Implementation", AIAA-2005-6107, AIAA Modeling and Simulation Technologies Conference, San Francisco, CA, August 2005

[2] Bryant, R. Barry, "CMF Motion Base Control System Design Description", Version C, October 31, 2001 (To Be Published as NASA TM at completion of shakedown testing.)

[3] Gupton, Lawrence E., Bryant, R. Barry, Carrelli, David J.," Evaluating the Performance of the NASA LaRC CMF Motion Base Safety Devices", AIAA-2006-6364, AIAA Modeling and Simulation Technologies Conference, Keystone, CO, August 2006.

[4] Carrelli, David J. "Detailed Dynamic Modeling of the NASA LaRC CMF Motion Base", AIAA-2006-6362, AIAA Modeling and Simulation Technologies Conference, Keystone, CO, August 2006.

[5] Carrelli, David J.; Bryant, R. Barry, "A Proposed Unit Independent Dexterity Calculation for use in Motion Base Design", AIAA-2004-5152, AIAA Modeling and Simulation Technologies Conference, Providence, RI, August 2004.