Cross-Platform Co-Simulation for Vehicle Safety Analysis

Xiaomeng Tong, Isheng Yeh
Livermore Software Technology, an ANSYS Company

Abstract

Cross-platform co-simulation is gaining more popularity nowadays for vehicle safety analysis. Essential elements, such as ADAS (advanced driver-assistance system) sensors, vehicle dynamics, occupant posture, and controller, can be individually solved in each software and effectively connected to the toolchain. The concept of co-simulation well suits the vehicle integrated safety analysis, which consists of both (1) the active safety features, such as autonomous emergency braking, lane keeping, etc., and (2) the passive safety features, such as the airbag, seatbelt pretensioner, etc. The co-simulation also extends the vehicle safety analysis from the traditional in-crash to a more comprehensive inclusion of pre-crash so as to evaluate the dummy posture and injury more precisely. To achieve this purpose, LS-DYNA® develops a co-simulation feature based on the Functional-Mockup-Interface (FMI), which allows LS-DYNA to remotely exchange data with any 3rd party software supporting this standard. Two cases are demonstrated hereby: the first is a passive safety co-simulation between LS-DYNA and MATLAB, where MATLAB controls the seatbelt pretension force, timing and the airbag deployment in LS-DYNA; the second case is the integrated safety case focusing on the active seatbelt control, where ANSYS VRX Driving Simulator solves the vehicle dynamics, and MATLAB provides the controller of braking/acceleration in VRX as well as the seatbelt/airbag in LS-DYNA. Both cases reveal that a more accurate occupant posture and significant improvement of occupant injury can be achieved by optimizing the active/passive safety features through the co-simulation.

Introduction

Modern vehicles are increasingly equipped with more safety features, including both the passive restraint system, such as seatbelts, airbags, etc., and the active safety system with ADAS (advanced driver-assistant system) sensors. The integrated safety system, i.e., the combined passive and active system, could significantly improve vehicle safety and reduce the occupant injury during the vehicle crash. The classical restraint systems, including airbags, pretensioners and load limiters, are not adaptive to occupants and crash scenarios, hindering their effectiveness in the safety improvement. A fully adaptive restraint system, which can analyze sensors from both the pre-crash and in-crash stages and can dynamically adjust the seatbelt load, seat position, airbag ventilation, etc., shows enormous potential. In 2013, TRW's Active Control Retractor became the first commercially available active seatbelt system, which used an electric motor to change the pretension force [1]. Paulitz et al. showed that the adaptive seatbelt system could reduce the pelvis, chest, and head accelerations by more than 50%, and the peak lap belt force by 60% for the frontal crash cases, by adaptively controlling the seatbelt force to be a constant [1]. Holding et al. studied the effect of a moving seat through a series of physical testing and found that a 30% reduction of neck moment and 26% reduction of pelvis acceleration could be observed compared with a static seat [2].

With the equipment of ADAS sensing technology, more active safety systems are introduced for frontal collision avoidance, including collision imminent braking (CIB), autonomous emergency braking (AEB), as well as lateral collisions, such as the lane departure warning and lane keeping. These systems use sensors like radars, lidars and cameras to control the braking/acceleration, and assist the drivers for decision making to avoid collision and mitigate injury when crashes are unavoidable. To avoid occupant being out of position in the emergency braking, the seatbelt pretension is activated in time to hold the occupant in position. Tijssens et al. [3] compared the passive only system with the integrated AEB system in dozens of simulation cases and found that the majority of the injuries improved due to the pre-crash braking. Moreover, without proper seatbelt pretension, the occupant posture could be changed, leading to the change of injury mechanism. All these indicate that the crash avoidance countermeasures should be well designed to fit in the pre-crash stage, not simply the classical in-crash analysis. Parameters in the integrated safety systems should be optimized to adapt to various driving scenarios by taking sensor inputs, and can smartly adjust the seatbelt force, emergency-braking, seat position during the pre-crash, and the airbag deployment and seatbelt load limiter/force during the in-crash.

The complexity of the integrated safety system demands a more comprehensive design and optimization process in the respective software. Essential elements of the multi-physics problem often require each sub-domain to be individually solved in different software and exchange data by co-simulation. For instance, Cresnik et al. used LS-DYNA to predict the occupant injury and MATLAB to design controllers to dynamically adjust the seatbelt limiter [4]. Lee et al. investigated the AEB influence through the co-simulation of MATLAB, CarSim and PreScan in the pre-crash analysis [5]. The cross-platform co-simulation is capable to connect all software to provide a more comprehensive multi-physics toolchain. With this motivation, the co-simulation feature of LS-DYNA is developed based on the popular functional-mockup-interface (FMI) 2.0 standard [6], which is extensively supported in more than 100 engineering software. Users are allowed to import and export variables from LS-DYNA to co-simulate with any 3rd party software, which supports the FMI feature. Since the communication is based on the TCP socket, remote co-simulation across various platforms is allowed, provided that computers are in the same private network.

Co-simulation Mechanism in LS-DYNA

The co-simulation feature is implemented based on the FMI 2.0, which was released in 2014 after updated from the previous version. The FMI is a free standard that wraps a combination of XML files, binaries and C code into a single FMU (functional-mockup-unit) for model exchange and co-simulation. Currently, only co-simulation is supported in LS-DYNA, which allows the generation of FMU to co-simulate with any 3rd party software supporting the FMI standard. Users need to specify input and output variables through keywords in LS-DYNA, and the IP address of the LS-DYNA computer if the co-simulation is remote, i.e., if the other software runs on a different machine. The co-simulation feature is currently supported in the lastest LS-DYNA developer version as well as R12 for SMP/MPP, Single/Double, Windows/Linux version.

Since the FMU is based on C instead of Fortran, a plugin "FMU Manager" is delivered to assist the FMU generation and co-simulation in LS-DYNA. Users can download the toolbox from [7] and find plentiful examples inside the toolbox. Depending on the operating system, a C compiler should be installed and configured to generate an FMU with the instructions detailed in the toolbox. The co-simulation is designed to be cross-platform, indicating that multiple software is allowed to run on the same/different computer with the same/different operating system, provided that both are in the same private network, i.e., the IP starts with 192 or 10 or other same numbers. The platform-independence is extremely helpful for large problems when LS-DYNA usually runs on HPC clusters with the Linux system, and the other computer could be a Windows PC. The hardware setup helps to understand the difference between the LS-DYNA explicit time step $\Delta t1$ and the co-simulation time step $\Delta t2$. Note that $\Delta t2$ is the time interval for data exchange between LS-DYNA and other software and $\Delta t2 > \Delta t1$. The greater $\Delta t2$ is, the less frequent the data exchange will be. $\Delta t1$ is controlled in LS-DYNA and can be set through *CONTROL_TIMESTEP, while $\Delta t2$ is independently set in the other software, for instance, MATLAB, according to each simulation scenario.

A dual-step procedure is followed to implement the LS-DYNA co-simulation. (1) FMU generation. Users properly define the imported and exported variables through *COSIMULATION_FMI_INTERFACE, and specify the settings in *COSIMULATION_FMI_CONTROL, such as IP of the LS-DYNA computer, the FMU mode, where 'G' is for generation, 'C' is for co-simulation. A new FMU file will be generated after running the input file with LS-DYNA. (2) FMU co-simulation. Users import the FMU into another software, such as MATLAB and properly configure the co-simulation time step, termination time, and prepare the input file for this software. Back to the LS-DYNA machine, users should switch the FMU mode from 'G' to 'C' for co-simulation in *COSIMULATION_FMI_CONTROL, and then run LS-DYNA, which will wait for the connection from the other software. Subsequentially run another software and let it connect to LS-DYNA automatically to start the co-simulation. A more detailed workflow can be found in [8] and the "FMU Manager" toolbox [7] with multiple examples for practice.

Two cases are presented in the current publication to demonstrate the co-simulation application in the integrated safety analysis, especially for Case 2, where LS-DYNA interacts with MATLAB and ANSYS VRX Driving Simulator, covering both the pre-crash and in-crash stage. Case 1 involves LS-DYNA and MATLAB, and focuses on the seatbelt control of the passive safety analysis. The complexity steps up from Case 1 to Case 2 for users to catch up with the workflow and co-simulation scheme.

Application

Case 1: Passive Safety (LS-DYNA and MATLAB/Simulink)

Case 1 demonstrates how to optimize the restraint features and reduce the occupant chest injury through co-simulation between LS-DYNA and Simulink. A simplified sled model with an LST hybrid III 50th rigid dummy is utilized in LS-DYNA for the demo purpose. The crash pulse is pre-calculated from a Yaris-Pole frontal crash at 35 mph and imposed on the sled model through *LOAD_BODY. During the co-simulation, Simulink receives the nodal acceleration, velocity, and displacement from LS-DYNA, which serves the inputs of Simulink controllers. Meanwhile, Simulink sends out signals to change the pretension force, seat velocity, and *sensor, which activates the retractor, the pretensioner as well as the airbag in LS-DYNA. Each sensor changes its status by comparing a curve value with a predefined threshold 0.5 in LS-DYNA, and the curve can be dynamically modified through the co-simulation with Simulink, from 0 to 1 to trigger the sensor. Likewise, the pretension force (type 4 pretensioner) and the seat velocity are also modified in the time domain by Simulink through the curve value to achieve the control purpose.

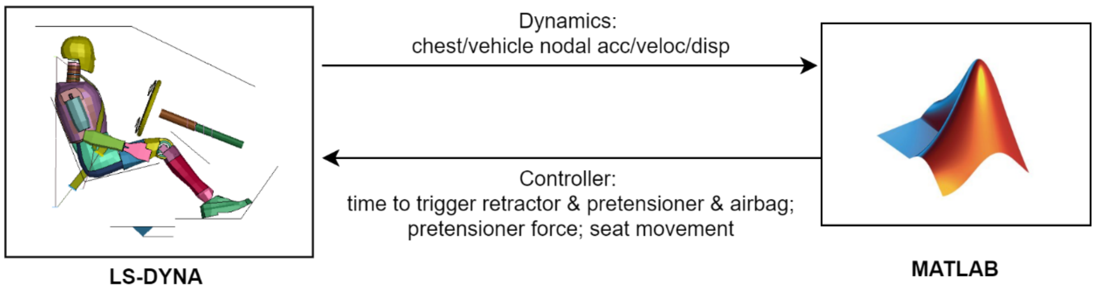
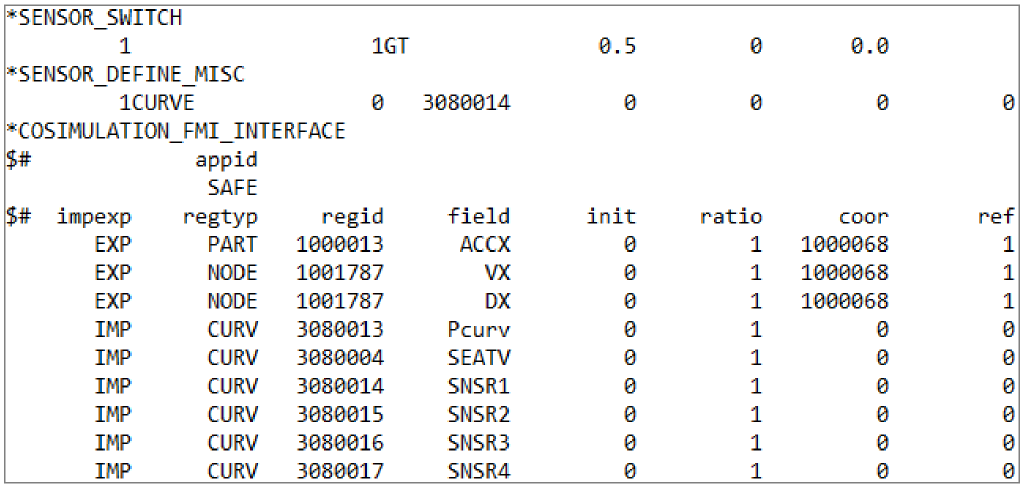

Dynamics:
chest/vehicle nodal acc/veloc/disp
Controller:
time to trigger retractor & pretensioner & airbag;
pretensioner force; seat movement
LS-DYNA
MATLAB

Fig. 1 Co-simulation structure of Case 1

The interface variables in LS-DYNA are defined in Fig. 2. Note that the first 3 variables are to be exported from LS-DYNA local coordinate, where the 1st one is the x acceleration of the rigid chest part, the 2nd and 3rd are the chest nodal velocity and displacement. All the other variables are imported from Simulink to change the specified curves, which are referred to by *element_seatbelt_pretensioner to transiently alter the pretension force, by *boundary_prescribed_motion_rigid to change the seat velocity, and by *sensor to switch the airbag from rigid to deformable bodies, to deploy the airbag, and to activate the pretensioner and the retractor.



```
*SENSOR_SWITCH
         1              1GT              0.5         0       0.0
*SENSOR_DEFINE_MISC
         1CURVE            0   3080014         0         0         0         0
*COSIMULATION_FMI_INTERFACE
$#             appid
                SAFE
$#  impexp    regtyp     regid     field      init     ratio      coor       ref
       EXP      PART   1000013      ACCX         0         1   1000068         1
       EXP      NODE   1001787        VX         0         1   1000068         1
       EXP      NODE   1001787        DX         0         1   1000068         1
       IMP      CURV   3080013     Pcurv         0         1         0         0
       IMP      CURV   3080004     SEATV         0         1         0         0
       IMP      CURV   3080014     SNSR1         0         1         0         0
       IMP      CURV   3080015     SNSR2         0         1         0         0
       IMP      CURV   3080016     SNSR3         0         1         0         0
       IMP      CURV   3080017     SNSR4         0         1         0         0
```

Fig. 2 Interface variables of Case 1

The FMU is then imported into Simulink for the controller design and co-simulation with LS-DYNA, following the first step, i.e., “generation” of FMU, and a schematics of the Simulink diagram is shown in Fig. 3. Note that it is used for demonstration purposes with less focus on the complexity of the control system itself. The FMU block is marked by “LSTC”, note that the interface variables on the left are to be sent to LS-DYNA including the pretension force and four sensors, and the variables on the right are imported from LS-DYNA, including the chest acceleration, velocity and displacement to design the controller. The co-simulation time step is set to be 0.01 ms in the Simulink before the co-simulation (double click the FMU block to set this value), and the smallest LS-DYNA time step is 3.4E-3 ms by checking the d3plot after the co-simulation. Note that the co-simulation time step is much larger than the LS-DYNA explicit time step.

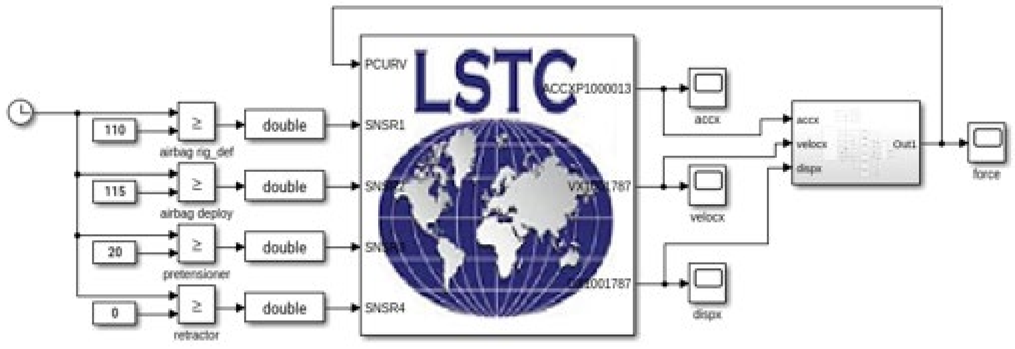
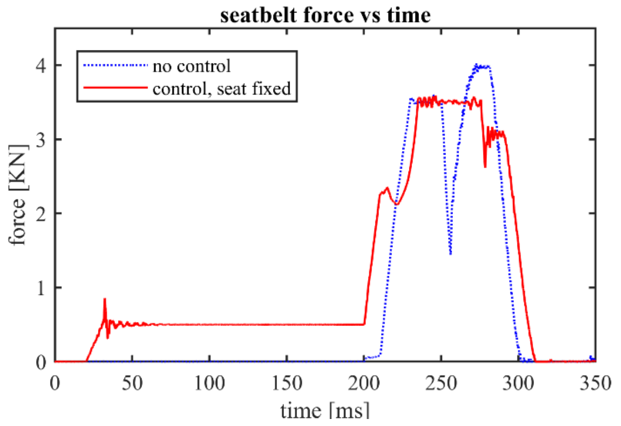


Fig. 3 Controller demo in Simulink of Case 1

The crash is assumed to occur at t=200 ms, v= 35 mph before the vehicle brakes for a duration of 200 ms with an acceleration of 0.5g. The uncontrolled case has the seatbelt retractor activated at t=200 ms and the pretentioner at 210 ms, with its force vs time curve predefined in LS-DYNA. In the controlled case, the ADAS sensors are capable to detect the unavoidable crash in an early stage (assumed, not modeled) and thus activate the pretensioner at t=20 ms, exerting a low-level force to hold the occupant in position and preventing it from sliding forwards due to the vehicle brake. The seatbelt force can dynamically alter its level to accommodate the occupant and vehicle motions, aiming to reduce the seatbelt force and occupant chest injury. The results are shown in Fig. 4, note that the seatbelt starts to exert forces during the pre-crash stage, i.e., t<200ms. Also, the occupant's chest compression has dramatically reduced from around 60 mm to 40 mm with a static seat, indicating that the chest injury is effectively reduced by optimizing the seatbelt force controller. The chest compression decreases to less than 30 mm if the seat is allowed to move backwards to increase the safety space between the occupant and the steering wheel.



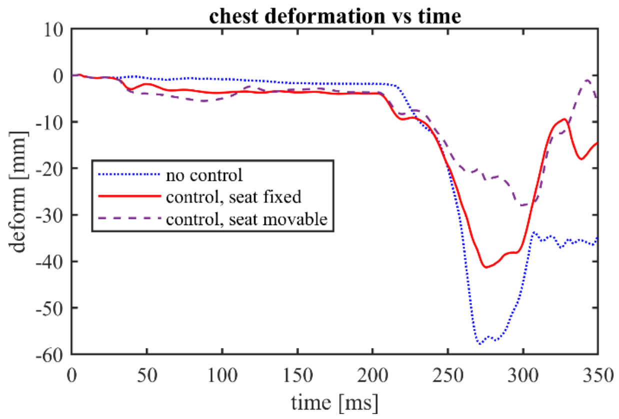
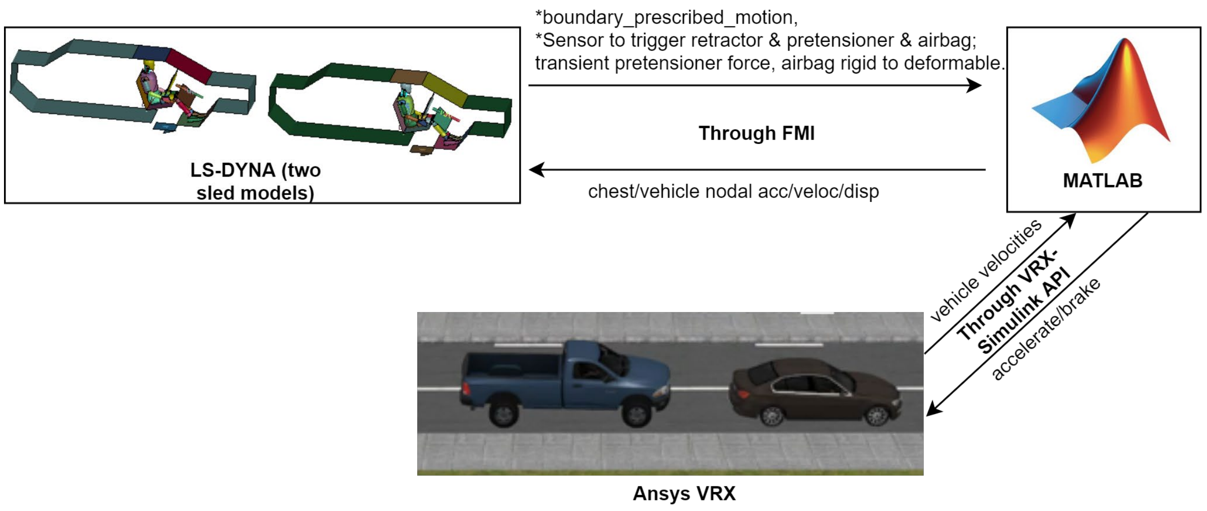


Fig. 4 Seat belt force and chest deformation of Case 1

Case 2: Integrated Safety (LS-DYNA, Simulink and ANSYS VRX Driving Simulator)

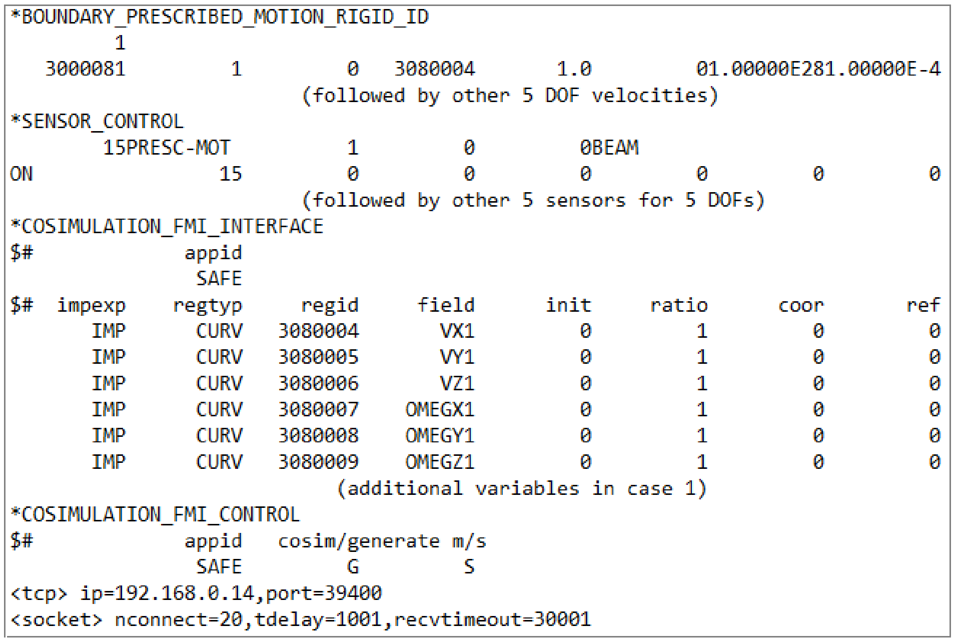

*boundary_prescribed_motion,
*Sensor to trigger retractor & pretensioner & airbag;
transient pretensioner force, airbag rigid to deformable.
Through FMI
chest/vehicle nodal acc/veloc/disp
LS-DYNA (two sled models)
MATLAB
vehicle velocities
Through VRX-Simulink API
accelerate/brake
Ansys VRX
Fig. 5 Co-simulation structure of Case 2

Case 2 intends to demonstrate a more comprehensive workflow involving LS-DYNA, Matlab and Ansys VRX driving simulator, with double sleds in LS-DYNA to model the crash scenario. VRX can effectively assist engineers to set up a realistic simulation for autonomous vehicles with millions of driving scenarios including traffic, weather, physical sensors [9]. Through the co-simulation with Simulink, VRX can take advantage of the extensive control features in Simulink and the physical ADAS sensors in VRX itself to more realistically predict the vehicle motions before the collision occurs. For instance, Simulink can control the vehicle braking/acceleration in VRX after performing the sensor fusion. In Case 2, VRX also passes the vehicle dynamics to LS-DYNA through Simulink to prescribe the vehicle velocity with *boundary_prescribed_motion, as shown in Fig. 5, recall that Simulink can dynamically modify the curve value in LS-DYNA, hence, changing its velocity. To save the execution cost, the vehicle and airbag models in LS-DYNA are initially rigidized in the pre-crash stage and are switched to deformable bodies when the crash is about to occur. The vehicle and occupant dynamics are collected from LS-DYNA and sent to the Simulink to control the *sensor in LS-DYNA to active the pretensioner, airbag, etc., and to dynamically alter the seatbelt force level. Once the vehicle distance is close to zero, i.e., the crash is about to occur, the *boundary_prescribed_motion is turned off in LS-DYNA, and the simulation moves from the pre-crash to the in-crash stage seamlessly. After this moment, VRX will be disconnected from the co-simulation toolchain, leaving only LS-DYNA and Simulink to play an active role. Recall that in Case 1, the crash pulse is pre-calculated and imposed through *load_body, and hereby

Case 2 demonstrates a more complete toolchain of the integrated safety analysis since the crash is implemented through the physical contact of two sleds. Replacing the sleds with full-vehicle models is feasible but could be extremely time-consuming with the current strategy and will be discussed in the last section.

Besides the interface variables defined in Case 1, additional variables are listed in Fig. 6 to generate the FMU. Note that the six translational and angular velocities are imported into LS-DYNA and imposed as velocity boundary conditions on the vehicle sled model. Additional *sensor is needed in LS-DYNA to turn off the velocity boundary conditions when the crash is about to occur. During the pre-crash stage, the six velocities are actually sent out by VRX to LS-DYNA through Matlab, considering that LS-DYNA can only co-simulate with one software currently, i.e., only one single FMU is supported. Since Matlab is already involved here, VRX thus plays a trick to let Matlab deliver its message to LS-DYNA. The multi-FMU capability is in progress and will be delivered in a future release. The Simulink model is demonstrated in Fig. 7, note that the speeds of both vehicles are acquired in Simulink from VRX, and are sent to LS-DYNA together with the pretension force and sensors, etc. LS-DYNA outputs occupant chest acc/v/s to Simulink for control purposes. Based on the data collected from VRX and LS-DYNA, Simulink will output brake/acceleration command to control the VRX vehicles in the pre-crash stage, such as the AEB. Note that the presented control diagram focuses on the demonstration of the workflow and users may implement their own controller in a more complex way.

The crash scenario is as follows: the initial speed of the ego and front vehicle is 65 kph and 25 kph, respectively, and the vehicle gap is 5.6 m at t=0 ms. By detecting that the crash is unavoidable, Simulink sends out the emergency braking signal to VRX, and the ego car brakes to 47 kph at t=660 ms before it hits the front car, which maintains a constant speed of 25 kph. In the controlled case, controllers in Simulink activate the pretensioner at t=0 ms and start to exert a low-level force on the seatbelt to hold the dummy in position before collision, and the uncontrolled case disables such feature. Fig. 8 compares the occupant posture with/without control in the pre-crash stage, i.e., t=420 ms, and the in-crash stage, i.e., t=720 ms, where “red” is the uncontrolled result. It is observable that without pretension force exerted in the pre-crash stage, the occupant tends to slide/tilt forwards due to the emergency braking, which adversely decreases the safety distance between the occupant and the steering wheel.



```
*BOUNDARY_PRESCRIBED_MOTION_RIGID_ID
         1
   3000081         1         0   3080004       1.0         01.00000E281.00000E-4
                         (followed by other 5 DOF velocities)
*SENSOR_CONTROL
        15PRESC-MOT          1         0         0BEAM
ON              15         0         0         0         0         0         0
                         (followed by other 5 sensors for 5 DOFs)
*COSIMULATION_FMI_INTERFACE
$#            appid
               SAFE
$#  impexp    regtyp     regid     field      init     ratio      coor       ref
       IMP      CURV   3080004       VX1         0         1         0         0
       IMP      CURV   3080005       VY1         0         1         0         0
       IMP      CURV   3080006       VZ1         0         1         0         0
       IMP      CURV   3080007    OMEGX1         0         1         0         0
       IMP      CURV   3080008    OMEGY1         0         1         0         0
       IMP      CURV   3080009    OMEGZ1         0         1         0         0
                            (additional variables in case 1)
*COSIMULATION_FMI_CONTROL
$#            appid   cosim/generate m/s
               SAFE         G         S
<tcp> ip=192.168.0.14,port=39400
<socket> nconnect=20,tdelay=1001,recvtimeout=30001
```

Fig. 6 Co-simulation keyword of Case 2

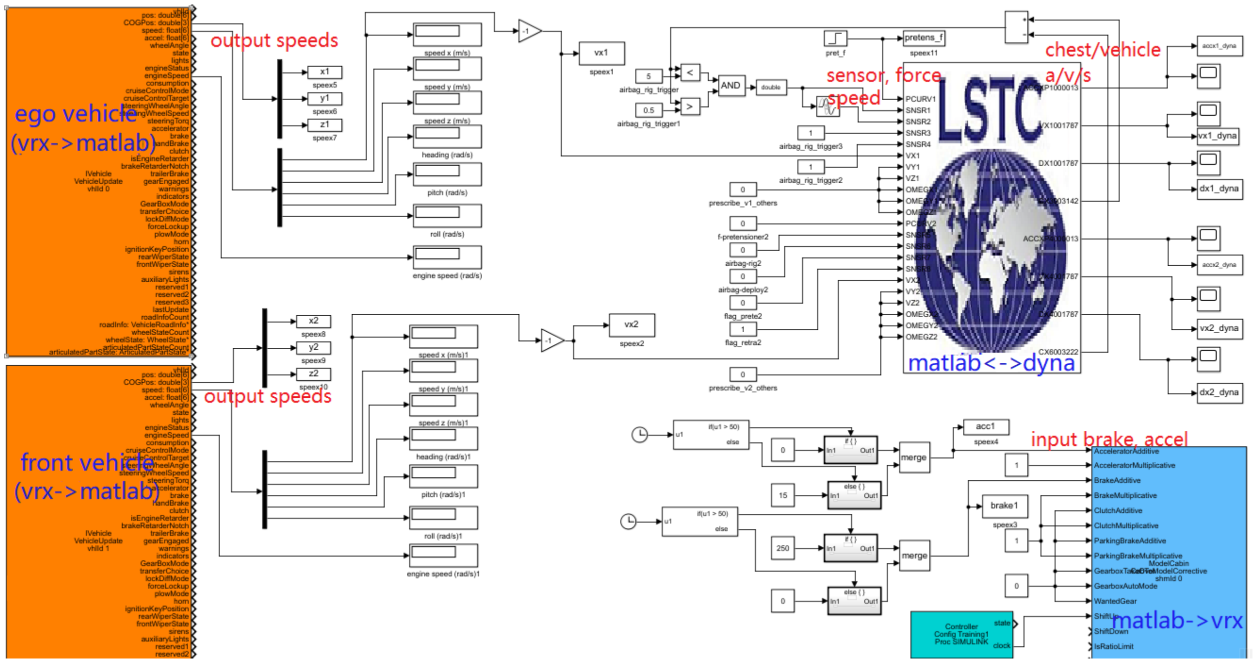


Fig. 7 Simulink control diagram of Case 2

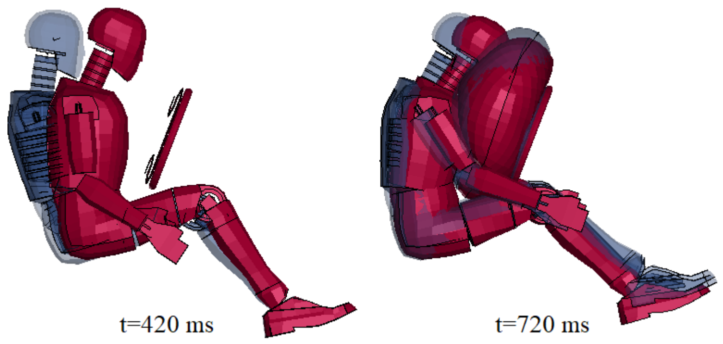


Fig. 8 Comparison of dummy posture with/without the active seatbelt (blue: with, red: wo.)

The dummy chest compression and head acceleration are plotted in Fig. 9. Recall that the crash occurs at t = 660 ms, and in the controlled case, the chest compression is non-zero in the pre-crash stage due to the pretension force from the active seatbelt. The maximum chest compression falls from around 30 to 25 mm compared with the uncontrolled case. The head acceleration also sees a 30% reduction in its peak value, indicating that the occupant suffers less injury with a proper controller implemented through the co-simulation.

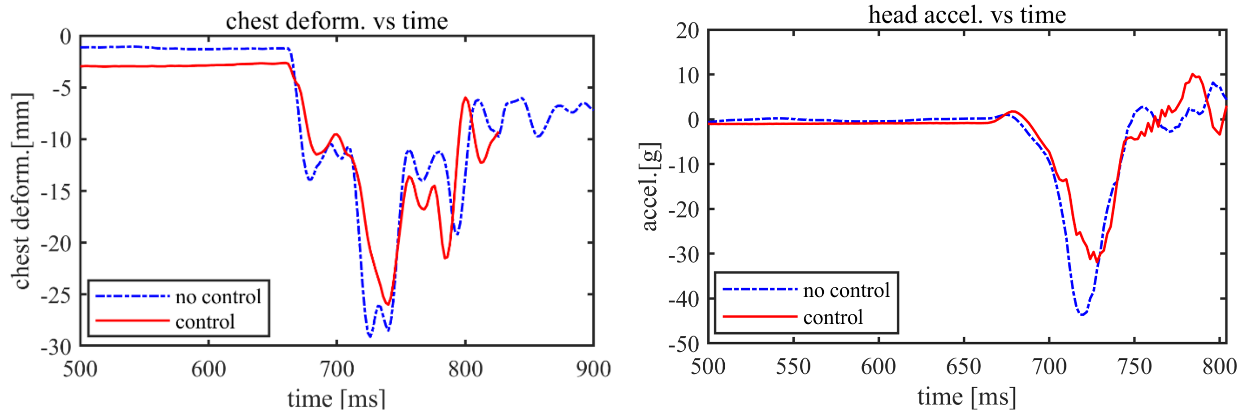


Fig. 9 Comparison of chest deformation and head acceleration in Case 2

Discussion and Future Work

Experienced users of the integrated safety analysis are aware of the much longer duration in the pre-crash stage compared with the in-crash stage. The in-crash typically lasts about 200 ms, while the pre-crash can be as long as several seconds, including the responding time of the drivers, and the vehicle motions, such as the emergency braking, lane changing, etc. To reduce the computational cost in the pre-crash stage, earlier studies [10] adopt a reduced model with only the dummy, seatbelt and necessary seat parts around the dummy, which are similar to the sled model used in this publication. At the end of the pre-crash simulation, necessary node velocities, coordinates, stress of the reduced model are then mapped to the in-crash model, which usually includes a full vehicle and dummy model with millions of DOFs, as initial conditions. The mandatory data mapping due to the model change is very challenging and usually requires to export all results into a LS-DYNAIN file at the end of the pre-crash stage with LS-PrePost® and then include it into the in-crash input files. The joint nodes, seatbelt node/element definitions need to be calibrated to avoid errors due to the mismatched joint nodes and seatbelt motion. Using the LS-PrePost to automatically snap these node joints could save lots of effort but still requires a double-check. This strategy is currently achievable within the current LS-DYNA and LS-PrePost capabilities and by exercising caution, however, has several limitations, such as, it cannot consider complex vehicle motions such as dramatic lateral dynamics which may induce vehicle instability, i.e., yawing or even rolling-over of the vehicle.

A more user-friendly and robust alternative would integrate the pre-crash and in-crash in one single run without manual interaction. It is similar to Case 2 in this paper, however with a more realistic full-vehicle model throughout the entire simulation in LS-DYNA. To reduce the computational intensity in the pre-crash stage, the majority of vehicle parts will be rigidized and automatically switched to deformable bodies when the crash occurs. The rigidization process is proposed to be automatically implemented in LS-DYNA, once users specify the desired part set ID in the keyword file, or spell out which part should be left for deformable. This strategy requires no intermediate files such as LS-DYNAIN or manual interaction to snap the joint nodes or correct the seatbelt settings, and is dedicated by LS-DYNA developers in the future release.

References

[1] Paulitz, T.J., Blackketter, D.M. and Rink, K.K., 2005, June. *Fully-adaptive seatbelts for frontal collisions*. In Proceedings of the 19th International Technical Conference on the Enhanced Safety of Vehicles (ESV) (Vol. 127, pp. 6-9).
[2] Holding, P.N., Chinn, B.P. and Happian-Smith, J., 2001. *An evaluation of the benefits of active restraint systems in frontal impacts through computer modeling and dynamic testing* (No. 2001-06-0094). SAE Technical Paper.
[3] Tijssens, M., Bosma, F. and Kietlinski, K., 2015. *A Methodology and Tool Chain to Develop Integrated Safety Systems*. In 24th International Technical Conference on the Enhanced Safety of Vehicles (ESV) National Highway Traffic Safety Administration (No. 15-0329).
[4] Cresnik, R., Rieser, A. and Schluder, H., 2009. *Dynamic simulation of mechatronic systems*. In 7th European LS-DYNA Conference, Graz, Austria.
[5] Lee, J.K., Chu, H.J. and Hurh, K.R., 2017. *A Development of the CAE Process for the AEB-Occupant Integrated Safety System*. In 25th International Technical Conference on the Enhanced Safety of Vehicles (ESV) National Highway Traffic Safety Administration.
[6] FMI Version 2.0, *FMI for Model Exchange and Co-Simulation*, http://www.fmi-standrd.org
[7] http://ftp.lstc.com/anonymous/outgoing/xiaomeng/deliver/FMU_Manager_release_note.txt
[8] Dong, K., Tong, X. and Yeh I., 2020, Coupled Crash Live Development Simulation using LS-DYNA Functional Mock-up Interface, In 16th International LS-DYNA User Conference.
[9] https://www.ansys.com/products/systems/ansys-vrxperience/vrxperience-capabilities#cap1
[10] Öztürk, A., Mayer, C., Kumar, H., Ghosh, P., Mishra, A., Chitteti, R.K. and Fressmann, D., *A Step Towards Integrated Safety Simulation Through Pre-Crash to In-Crash Data Transfer*, In Proceedings of the 19th International Technical Conference on the Enhanced Safety of Vehicles (ESV).