



Altair

HyperWorks

Altair MotionView 2019 Tutorials

MV-8002: Multi-Maneuver Events

altairhyperworks.com

MV-8002: Multi-Maneuver Events

In this tutorial, you will learn how to:

- Define end conditions for a maneuver or a sub-event
- Write parametric expressions
- Define events as multiple sub-events executed sequentially

End conditions

- Conditions to end a particular maneuver before given simulation end time
- Examples of the end conditions can be – End maneuver when longitudinal velocity is greater than 10 m/s or when roll angle reaches steady state
- End conditions can be logically coupled (OR-ed or AND-ed) by splitting them into groups

Multi-maneuver events

- Events consisting for more than one maneuver – these maneuvers are executed sequentially
- Controllers can only be changed while switching the maneuvers
- Hence, rule of thumb – whenever need to change the controller, change the maneuver
- Driver does following while switching the maneuvers
 - Halts previous maneuver
 - Saves the signals value that acts as initial value for next maneuver in case of parametric expressions, there is a list of signals that driver monitors. Please refer to the documentation for more details.
 - Executes the change of/in controller
 - Starts new maneuver
- Examples: Fishhook, J-turn, Throttle off cornering analysis

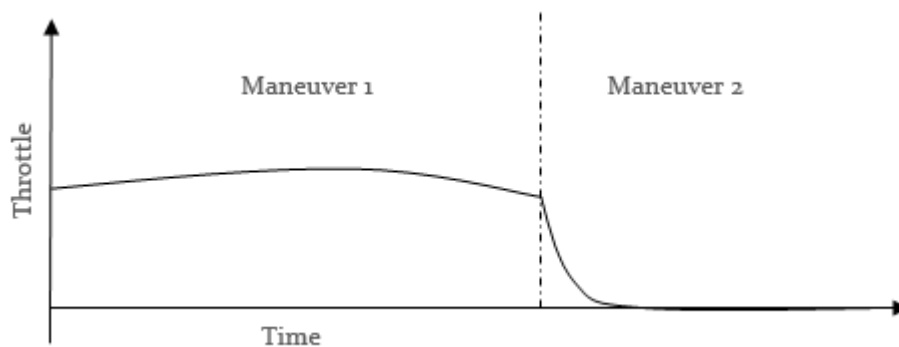
Parametric Expressions

- When in a multi-maneuver event, expressions need to be re-evaluated before the start of the maneuver in order to maintain the continuity of the signals.
- { Expression in Curly Braces } - Instruction to driver to evaluate the expression before giving it to MotionSolve
- {SIGNAL} is evaluated as VARVAL(signal solver variable id)
- {SIGNAL_0} is evaluated as Signal Value at the end of last maneuver
- {%SIGNAL} is evaluated as {SIGNAL} – {SIGNAL_0}
- Driver evaluates the expressions for the maneuver before the start of the maneuver

Example:

Throttle off cornering event

- Maneuver 1 – Constant radius cornering, constant radius path with constant velocity - until roll angle reaches its maximum and stabilizes.
- Maneuver 2 – Step down the throttle while following the same path
- In this event, Maneuver 1 would typically consist of closed loop steering and throttle controllers. In Maneuver 2, the steering controller still remains the same, however the throttle controller is open loop, type expression – ‘STEP(TIME – end time of maneuver 1 , 0, throttle value at the end of maneuver 1, 0.5, 0)’.

**Exercise****Step 1: Assembling the vehicle.**

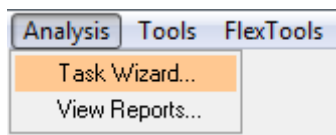
1. Follow the instructions in Step #1 of MV-8000 to create the vehicle with the topology as provided below.

Page	Label	Selection	Default (Yes/No)
1	Model type	Full vehicle with advanced driver	No
2	Driveline configuration	Front wheel drive	Yes
3	Vehicle body	Body	Yes
3	Front suspension	Frnt macpherson susp (1 pc. LCA)	Yes
3	Steering linkages	Rackpin steering	Yes
3	Rear subframe	None	Yes
3	Rear suspension	Rear quadlink susp	Yes
3	Powertrain	Linear torque map powertrain	Yes
3	Signal generator	Driver signal generator	Yes

Page	Label	Selection	Default (Yes/No)
3	Tires	FIALA/HTIRE	Yes
4	Steering column	Steering column 1 (not for abaqus)	Yes
4	Steering boost	None	Yes
5	Front struts	Frnt strut (with inline jts)	Yes
5	Front stabilizer bars	Frnt stabar with links	No
5	Rear struts	Rear strut (with inline jts)	Yes
5	Rear stabilizer bars	Rear stabar with links	No
6	Front jounce bumpers	None	Yes
6	Front rebound bumpers	None	Yes
6	Rear jounce bumpers	None	Yes
6	Rear rebound bumpers	None	Yes
7	Disk brakes	Disk brakes	Yes
7	Front driveline	Independent fwd	Yes
8		Next	No
9		Finish	No

Step 2: Adding driver analysis.

1. Use the Task Wizard to load the driver analysis.



Step 3: Specifying vehicle parameters.

1. We are going to use feedforward controllers for velocity profile following. Feedforward controllers model the vehicle and hence, require vehicle parameters. Vehicle parameters need not be precise for controllers to work. Most of the vehicle parameters required by the driver can be automatically calculated from the vehicle model.

Step 4: Writing an Altair Driver File (ADF) driving event.

Example #1 Fish Hook Event

We will model this event in three maneuvers.

1. Open any text editor and copy and paste the following text into it. **Important: All blank lines must be removed prior to saving the file!** Be sure to read through the comments for a better understanding on what is written in the ADF.

```

$-----ALTAIR_HEADER
[ALTAIR_HEADER]
FILE_TYPE           = 'ADF'
FILE_VERSION        = 1.0
FILE_FORMAT         = 'ASCII'
$-----UNITS
[UNITS]
(BASE)
{length force      angle      mass      time}
'meter' 'newton' 'radians' 'kg' 'sec'
$-----VEHICLE_IC
[VEHICLE_INITIAL_CONDITIONS]
VX0      = -17.5
VY0      = 0.0
VZ0      = 0.0
$-----STEERING_STANDARD
[STEER_STANDARD]
$Upper and lower bounds are kept to match the event requirement of saturating at
$270 deg and -540 deg respectively
MAX_VALUE      = 4.712
MIN_VALUE      = -9.425
SMOOTHING_FREQUENCY = 5
INITIAL_VALUE   = 0.0
$-----THROTTLE_STANDARD
[THROTTLE_STANDARD]
MAX_VALUE      = 1
MIN_VALUE      = 0
SMOOTHING_FREQUENCY = 5
INITIAL_VALUE   = 0.0
$-----BRAKING_STANDARD
[BRAKE_STANDARD]
MAX_VALUE      = 1
MIN_VALUE      = 0
SMOOTHING_FREQUENCY = 5
INITIAL_VALUE   = 0.0
$-----MANEUVERS_LIST
[MANEUVERS_LIST]
{name          simulation_time      h_max      print_interval}
'GO_STRAIGHT'      2.0          0.01        0.1
'LEFT_TURN'        12.0         0.001       0.1
'RIGHT_TURN'       10.0         0.001       0.1
[GO_STRAIGHT]

```

TASK = 'STANDARD'

(CONTROLLERS)

```
{DRIVER_SIGNAL PRIMARY_CONTROLLER ADDITIONAL_CONTROLLER}
STEER OL_CONSTANT_STEER NONE
THROTTLE FEED_FORWARD_TRACTION NONE
BRAKE FEED_FORWARD_TRACTION NONE
```

\$-----MANEUVER_2

[LEFT_TURN]

TASK = 'STANDARD'

(CONTROLLERS)

```
{DRIVER_SIGNAL PRIMARY_CONTROLLER ADDITIONAL_CONTROLLER}
STEER OL_LEFT_STEER NONE
THROTTLE FEED_FORWARD_TRACTION NONE
BRAKE FEED_FORWARD_TRACTION NONE
```

\$We want to end the maneuver if the roll rate reaches steady state i.e. $d(\text{Roll rate})/dt = 0$

\$(tolerance = 0.005) for 0.5 seconds

(END_CONDITIONS)

```
{SIGNAL GROUP ABS OPERATOR VALUE TOLERANCE WATCH_TIME}
ROLL_RATE 0 Y SS 0 0.005 0.5
```

\$-----MANEUVER_3

[RIGHT_TURN]

TASK = 'STANDARD'

(CONTROLLERS)

```
{DRIVER_SIGNAL PRIMARY_CONTROLLER ADDITIONAL_CONTROLLER}
STEER OL_RIGHT_STEER NONE
THROTTLE FEED_FORWARD_TRACTION NONE
BRAKE FEED_FORWARD_TRACTION NONE
```

\$-----STEER for Maneuver 1

[OL_CONSTANT_STEER]

TAG = 'OPENLOOP'

TYPE = 'CONSTANT'

VALUE = 0

\$-----STEER for Maneuver 2

\$Ramp up the steering wheel @ 360 deg per second

[OL_LEFT_STEER]

TAG = 'OPENLOOP'

TYPE = 'EXPRESSION'

SIGNAL_CHANNEL = 0

EXPRESSION = '{STEER_0} + {%TIME}*PI*2'

\$-----STEER for Maneuver 3

[OL_RIGHT_STEER]

TAG = 'OPENLOOP'

TYPE = 'EXPRESSION'

SIGNAL_CHANNEL = 0

EXPRESSION = '{STEER_0} - {%TIME}*PI*2'


\$-----THROTTLE and BRAKE controller for entire event

[FEED_FORWARD_TRACTION]

TAG = 'FEEDFORWARD'

TYPE = 'FOLLOW_VELOCITY'

```
LOOK_AHEAD_TIME = 0.5
DEMAND_SIGNAL = 'DEMAND_VEL'
$-----Demand Velocity
[DEMAND_VEL]
TYPE          = 'CONSTANT'
VALUE         = 17.5
```

2. **Run** the simulation .
3. Observe the results.

Maneuver 2 stops when roll rate is consistently 0 (with mentioned tolerance) for 0.5 seconds.

