ENGINEERING DYNAMICS COMPANY

DyMESH

HVE Human Vehicle Environment

Markdown Edition — revised to describe the current version of the software

Contents

1. DyMESH Reference Manual
2. DyMESH Overview
3. DyMESH Collision Model
4. DyMESH Version 3
5. DyMESH — Facts of Life
6. DyMESH Wheel vs. Environment

DyMESH Reference Manual

DyMESH (Dynamic Mechanical Shell) is the three-dimensional dynamic-mesh collision and damage model used by HVE's simulation models. It computes inter-vehicle (and vehicle-environment) contact forces, moments, and damage directly from the polygon meshes that describe the exterior surfaces of the vehicles and objects in an event. The resulting collision forces are combined with the other external forces acting on each vehicle (tires, aerodynamic drag, etc.) and the six-degree-of-freedom equations of motion are integrated to produce the collision dynamics and the resulting damage (crush) profile.

DyMESH is invoked primarily by **SIMON** (and is also available to **EDSMAC4**). It replaces the idealized impulse-momentum treatment of impact with a true force-based, time-stepped collision simulation.

Provenance

This manual was organized from three 2026 HVE Forum presentations by **Allen R. York** and **Terry D. Day** of the Engineering Dynamics Company (EDC):

- **DyMESH Collision Simulation** (York & Day, March 2026) — the core technical overview: history, mesh model, contact algorithm, force/damage computation, and coupling to the vehicle dynamics.
- **DyMESH Version 3** (EDC, 2026 HVE Forum) — the Version 3 additions, chiefly the *DyMESH Wheels* collision model (forces and moments from impact at the wheels).
- **DyMESH — Important Notes / Facts of Life** (EDC, 2026 HVE Forum) — practical guidance, caveats, and limitations for users.

The technical content has been brought up to date with the current version of HVE, covering the DyMESH collision model, its use by SIMON, and the DyMESH Options dialog. Where the current program differs from the slide decks, the discrepancy is called out with an italic

(updated: ...)

note.

Contents

1. Overview — what DyMESH is and does; where it sits in HVE; which models use it; history.
2. Collision Model — the technical model: mesh representation, contact (Location) and restoration detection, force and moment computation, 3-D force-deflection stiffness, vertex displacement (crush depth), restitution, and coupling to the equations of motion.
3. Version 3 — the DyMESH Wheels collision model and other Version 3 capabilities.
4. Facts of Life — practical notes, caveats, and limitations.
5. Wheel vs. Environment — the DyMESH wheel vs. environment contact model: the wheel and terrain as DyMESH objects, the simulation loop, and the handoff between the point-contact tire model and DyMESH.

Related documentation

- **SIMON manual** — the primary vehicle dynamics model that hosts DyMESH.
- **EDSMAC4 Calculation Options**.

- **Options menu (DyMESH Options dialog)**.
- **Wheel damage / displacement** — the wheel-data inputs used by the DyMESH Wheels model.

Source: DyMESH Collision Simulation, DyMESH Version 3, and DyMESH Important Notes (2026 HVE Forum). Updated to match the current version of HVE.

DyMESH Overview

What DyMESH is

DyMESH — *Dynamic Mechanical Shell* — is a general, three-dimensional method for simulating collisions. The exterior surfaces of the vehicles and/or objects (collectively, *bodies*) taking part in an event are discretized with a triangular mesh. When two bodies interact, the meshes inter-penetrate; DyMESH detects that penetration, deforms the meshes to enforce a no-penetration condition, and computes collision forces at the nodes of the deformed mesh. The deformation of the mesh **is** the damage (crush) profile, and the same mesh is used both for the physics and for the visual display, so damage is visualized directly.

The collision forces DyMESH produces are combined with the other external forces acting on each vehicle and the equations of motion are then solved. In short:

> Collision forces at the nodes of the deformed mesh are computed and combined with other external forces to solve for the motion of the vehicle or object.

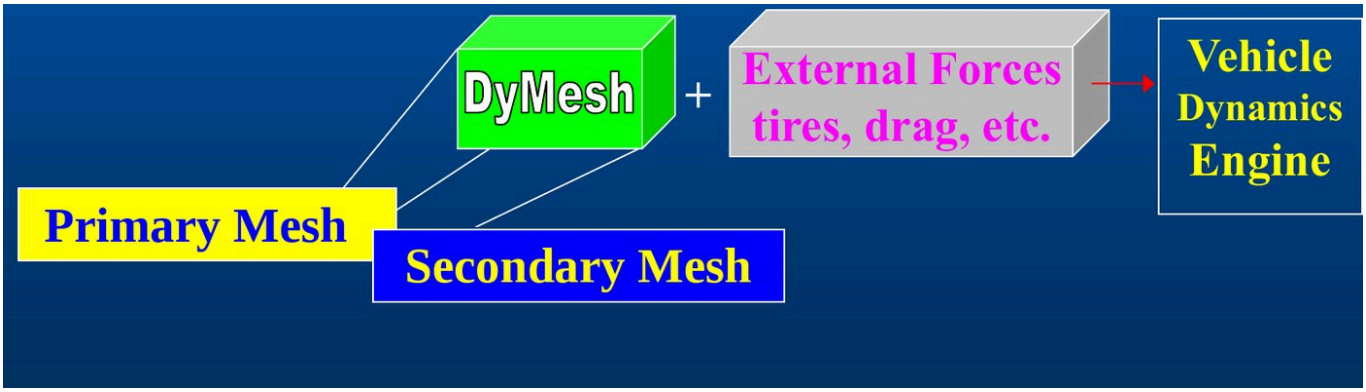


Figure: block diagram — Primary Mesh + Secondary Mesh feed "DyMesh", which is summed with "External Forces (tires, drag, etc.)" into the Vehicle Dynamics Engine.

DyMESH is integrated into a **six degree-of-freedom vehicle dynamics model** (SIMON). It is not a stand-alone code; it supplies the impact forces and moments that the host dynamics model integrates each time step.

Why DyMESH (vs. 2-D impulse-momentum methods)

DyMESH goes beyond the existing two-dimensional methods for vehicle collision simulation and allows the potential for more accurate simulations. Because it is fully three-dimensional and force-based, it can represent phenomena that planar, impulse-momentum collision models cannot:

- **Non-uniform crush in the z (vertical) direction.**
- **Override and underride.**
- **Rollover.**

Key distinctions:

- It is a **collision simulation**, *not* an idealized impulse-momentum method. Forces are developed continuously through the contact from the mesh deformation, rather than being resolved as a single instantaneous impulse.
- DyMESH simulations are **interactive**, running in a number of minutes on a typical workstation.
- DyMESH does **not** require the expense of generating a complex, three-dimensional, finite-element-like mesh. The user is not concerned with meshing the vehicle — the display mesh already present in the vehicle geometry is what DyMESH uses.

Where DyMESH sits in HVE / which models use it

DyMESH is the collision engine invoked by HVE's simulation models — primarily **SIMON**, and also available to **EDSMAC4**. It is enabled and configured per event through the **DyMESH Options** dialog (Options menu). When DyMESH is enabled, the host model calls it each collision time step for every interacting pair of bodies; the returned forces and moments are added to the vehicle's force system before the equations of motion are integrated.

DyMESH can also compute contact against the **environment** (terrain / fixed objects) when the *Include Environment* option is selected, and — as of Version 3 — against the individual **wheels** of a vehicle (see Version 3).

(updated: DyMESH is a licensed feature; the DyMESH Options dialog refuses to enable it without a license.)

History

DyMESH has a long development lineage at EDC:

Date	Milestone
6/98	Development begins
1/99	Prototype code, proof-of-concept
—	Published: SAE 1999-01-0104 and 2000-01-0844
6/99 – 1/01	Port to HVE
1/01	SIMON/DyMESH prototype
—	Commercial development
5/04	**Commercial release**
2/08	Major update
12/11	Major update
1/18	Major update — improvement to the contact algorithm: *Vertex Displacement (Crush Depth)*

The 2/08, 12/11 and 1/18 updates correspond to the modeling improvements that define DyMESH **Version 2** (improved accelerations, crush-depth simulation / general damage profile, and restitution) and **Version 3** (user-defined third-order force-deflection curve fit, and the DyMESH Wheels collision model).

*(updated: The current version of HVE implements DyMESH with selectable **Version 3** and **Version 4** behavior — the DyMESH Options dialog exposes "Version 3" and "Version 4" radio buttons. Version 4 additionally enables the "pushback reduction" restoration refinement, which is not applied under Version 3. The 2026 decks describe the model through "Version 3"; Version 4 is a further refinement not covered by the slides.)*

Source: DyMESH Collision Simulation (2026 HVE Forum). Updated to match the current version of HVE.

DyMESH Collision Model

This chapter describes the technical model implemented by DyMESH: the mesh representation, the contact (Location) and restoration algorithm, the node force–deflection law, the three-dimensional stiffness, the vertex-displacement (crush-depth) treatment, restitution, and the coupling to the vehicle equations of motion. The description follows the

DyMESH Collision Simulation

deck, updated to match the current version of HVE.

1. Mesh representation

Each body is represented by the same **triangular surface mesh** used for its display: an array of vertices and an array of triangular polygons.

- Each polygon carries its three vertex indices, an earth-fixed outward **normal**, an earth-fixed **center**, and its current and original **area**.
- Each vertex carries undamaged and current (damaged) coordinates in both the earth-fixed and vehicle-fixed frames, its current deformation vector, its accumulated maximum deformation, the force acting on it, and a per-vertex **stiffness** set and **saturation crush**.

During a run, one body is designated the **Primary** (P) and the other the **Secondary** (S). Primary *surfaces* (polygons) are tested against Secondary *nodes* (vertices): the algorithm finds, for each secondary node, the primary surface it has penetrated. The roles can be swapped every time step, or multiple times per time step, so that both bodies accumulate damage.

Because the search over all polygons would be expensive, DyMESH restricts each test to a **search box** around each secondary vertex, and to precomputed lists of neighboring surfaces. The neighbor lists are only recomputed every few time steps, for efficiency. The box size is either computed automatically from the vehicle extents or set by the user (**Search Option**: Automatic or Set Box Size).

2. Contact algorithm

DyMESH uses a contact algorithm similar to that used in explicit, dynamic finite-element simulation codes. The contact algorithm has **two main functions**:

1. **Location** — determining mesh inter-penetration (which secondary nodes are inside which primary surfaces).
2. **Restoration** — altering the positions of the nodes of the mesh to enforce the no-penetration condition.

The sequence over a contact is:

1. The bodies approach one another.
2. **Location phase:** the meshes inter-penetrate; the region of overlap is identified.
3. **Restoration phase:** the meshes are adjusted (nodes pushed back to the contact surface).
4. Forces are applied due to the deformations; these forces produce accelerations that separate the bodies.

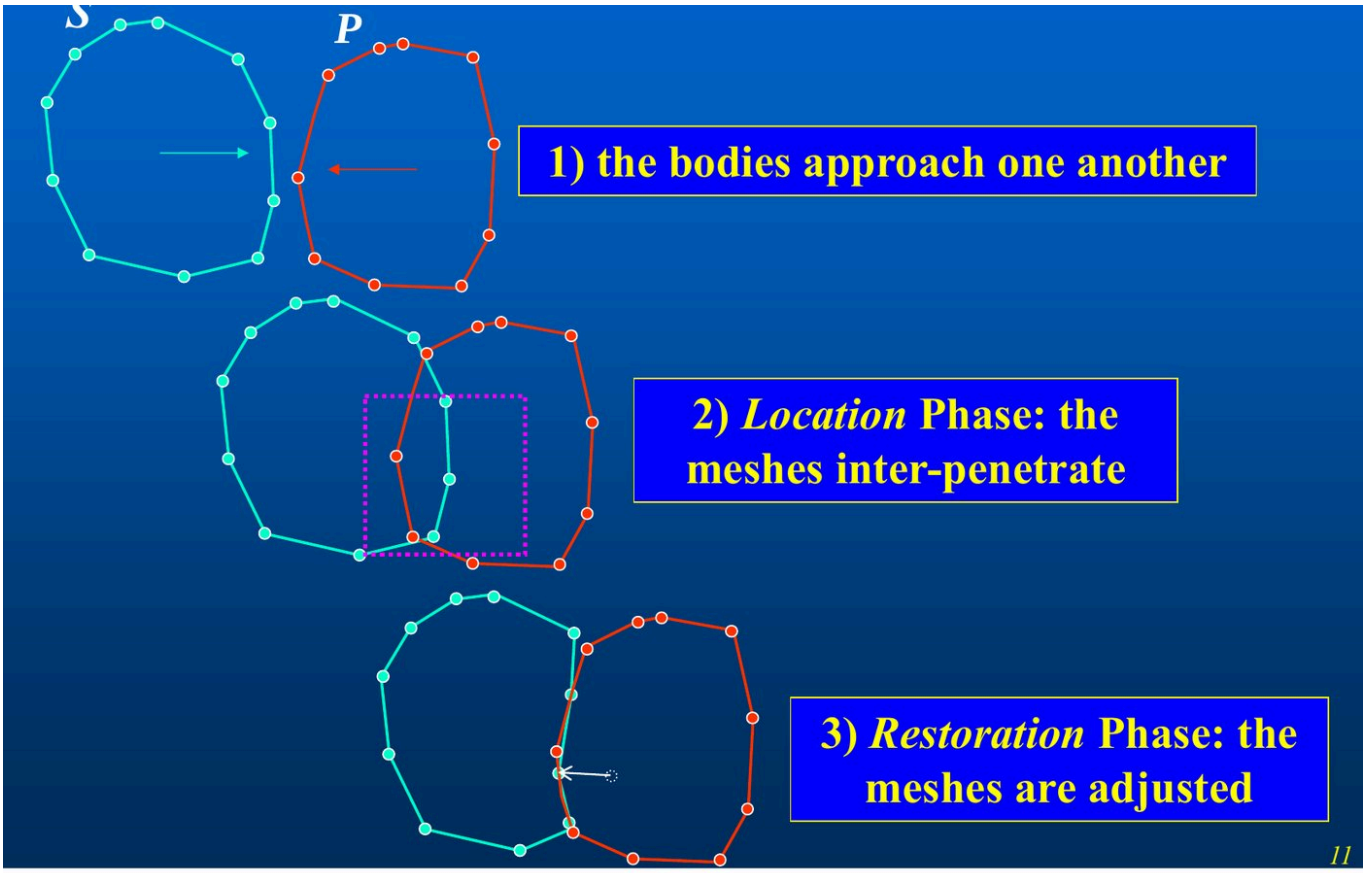


Figure: three-panel contact-algorithm example — (1) bodies approach, (2) Location phase overlap box, (3) Restoration phase adjustment.

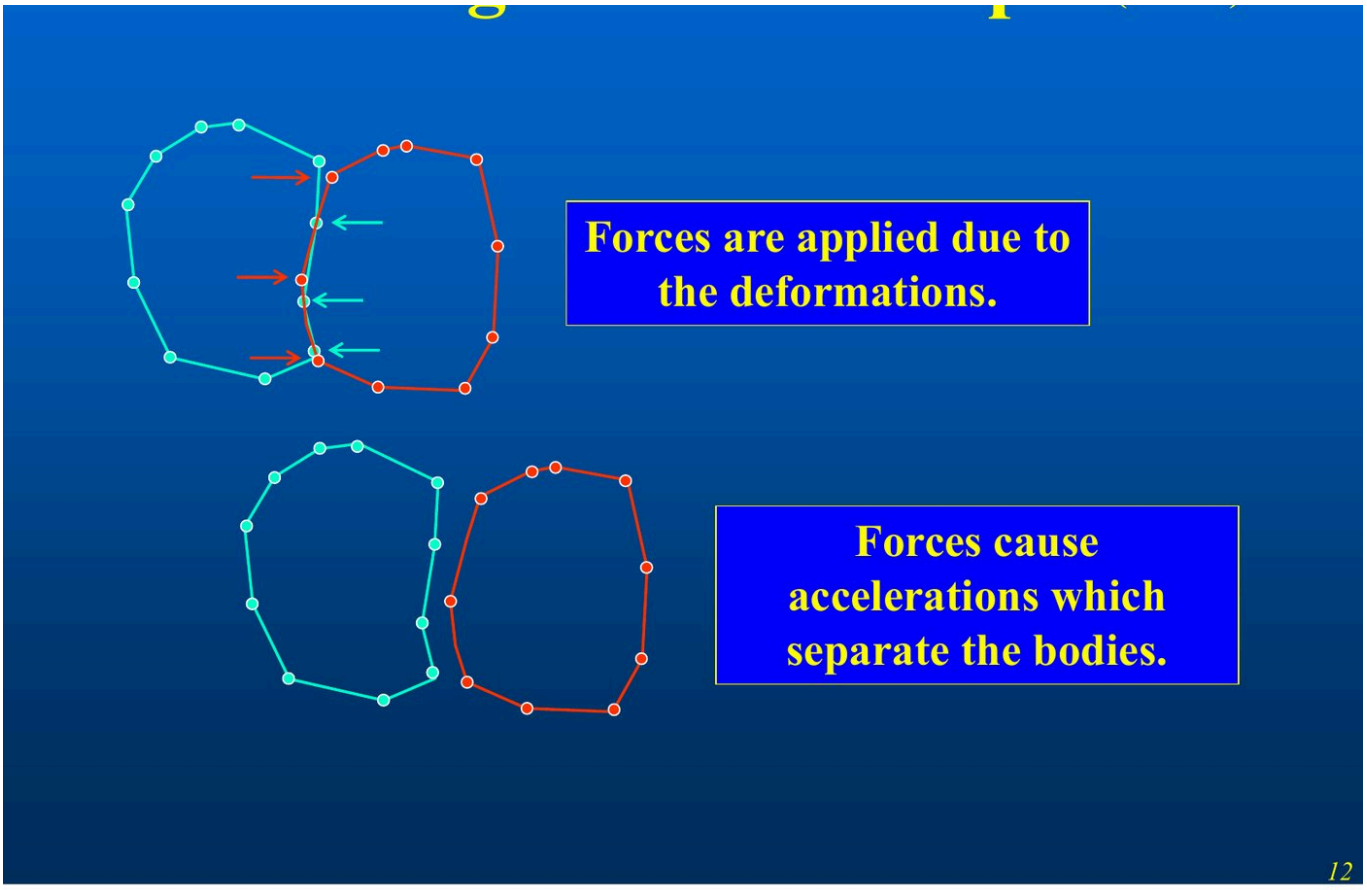


Figure: force application and separation — deformation forces at overlapping nodes drive the bodies apart.

Locating the primary surface for a secondary node

A secondary node is matched to the primary surface it has penetrated. A candidate contact point is the intersection of a search ray from the secondary node with the primary triangle; an **inside-polygon test**, using the cross-product method, confirms the intersection lies within the triangle. The direction of the search ray can be chosen among several methods — the local average normal, the velocity direction, the polygon normal, or a specified direction (a vertex-normal method also exists but is disabled) — and the local search may stop after the first, best-of-two, or best-of-all candidates. These are exposed on the **Advanced** tab of the DyMESH Options dialog.

Restoration (pushback)

Once a secondary node is found interior to a primary surface, its penetration depth is the distance from the node to the primary surface along the restoration direction. The node is restored (pushed back) toward the surface; the amount of that pushback, accumulated over time, is the node's permanent deformation. To prevent numerical instabilities, the pushback can be reduced by a limited amount, and is constrained not to restore across implausible distances.

*(updated: the pushback reduction is only active when DyMESH **Version 4** is selected. Under Version 3 the reduction is not applied.)*

3. Node force–deflection law

A central design point is that **any force-deflection relationship can be used**; the force–deflection behavior is prescribed *a priori* per surface. The standard A and B crush coefficients prescribe the three-dimensional force-deflection

relationship. **DyMESH Version 3 extends the force-deflection relationship to a third-order polynomial** by adding C and D coefficients together with a **saturation deflection** δ_{sat}.

The force on a single node is computed as follows. Let δ be the node's current total deflection (penetration depth), A_v be the area associated with the node, and δ_0 be a small **null-band** offset (0.5 in) below which only the constant term acts. For increasing penetration ($\delta > \delta_{max}$) the node force per unit area follows a cubic:

$$F = [\, A + B\,(\delta - \delta_0) + C\,(\delta - \delta_0)^2 + D\,(\delta - \delta_0)^3 \,]\, A_v \qquad \text{(Eq. 1)}$$

where A (constant, lb/in²), B (linear, lb/in³), C (quadratic, lb/in⁴), and D (cubic, lb/in⁵) are the four per-node stiffnesses. Below the null band the force is simply $F = A\,A_v$. Once the deflection reaches the node's **saturation crush** δ_{sat}, the force is held at its saturation value and does not increase further:

$$F_{sat} = [\, A + B\,(\delta_{sat} - \delta_0) + C\,(\delta_{sat} - \delta_0)^2 + D\,(\delta_{sat} - \delta_0)^3 \,]\, A_v \qquad \text{(Eq. 2)}$$

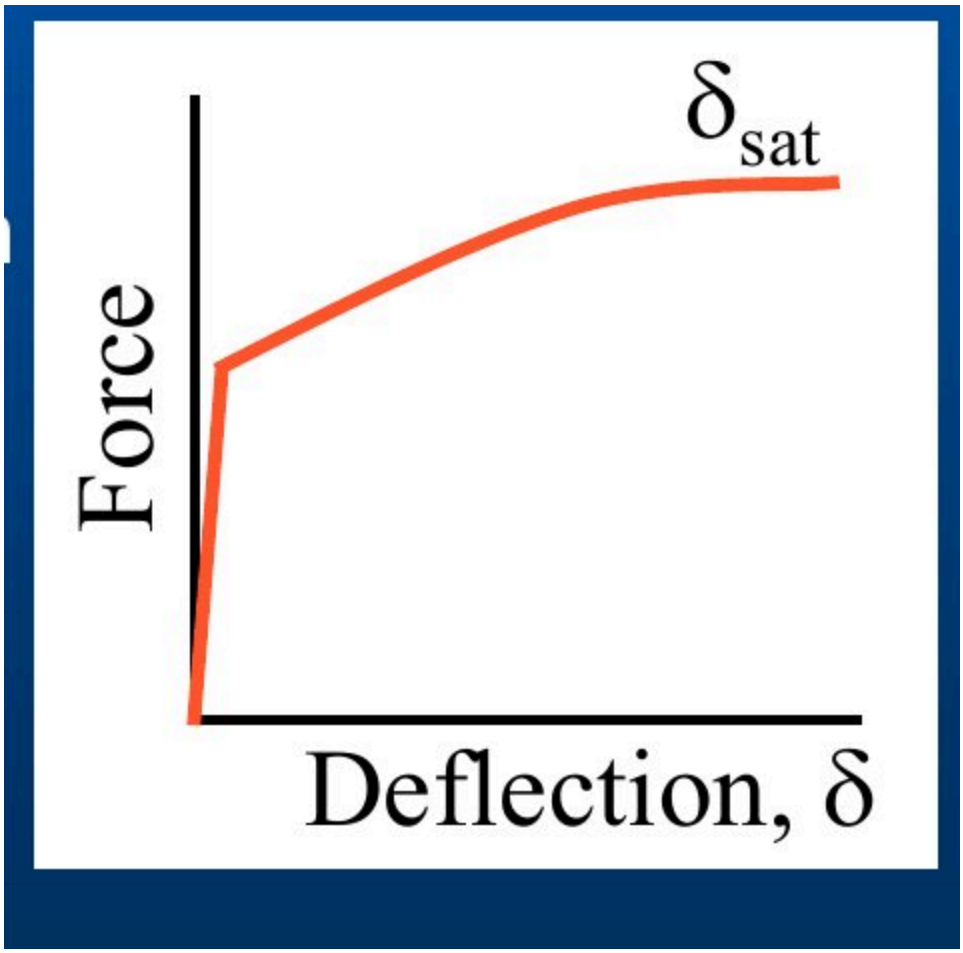


Figure: Force vs. Deflection curve rising then flattening at the saturation deflection δ_{sat}.

The stiffnesses are assigned per body **surface** — front, right, back, left, top and bottom — so different faces of a vehicle can have different crush behavior.

4. Three-dimensional (per-area) force-deflection

The published crush coefficients (A, B) are two-dimensional — force per unit crush **width**. DyMESH needs force per unit crush **area**. The 3-D stiffness coefficients are obtained by dividing the standard (2-D) coefficients by the **conversion height** H of the crush region that was used to derive the 2-D coefficients:

$$\tilde{A} = \frac{A}{H}, \qquad \tilde{B} = \frac{B}{H} \qquad \text{(Eq. 3)}$$

Here A has units of force per width (lb/in) and $\tilde{A}$ has units of force per area (lb/in²); likewise B (lb/in²) becomes $\tilde{B}$ (lb/in³). The default conversion height is 30 in, matching the **Crush Height For Stiffness** dialog default.

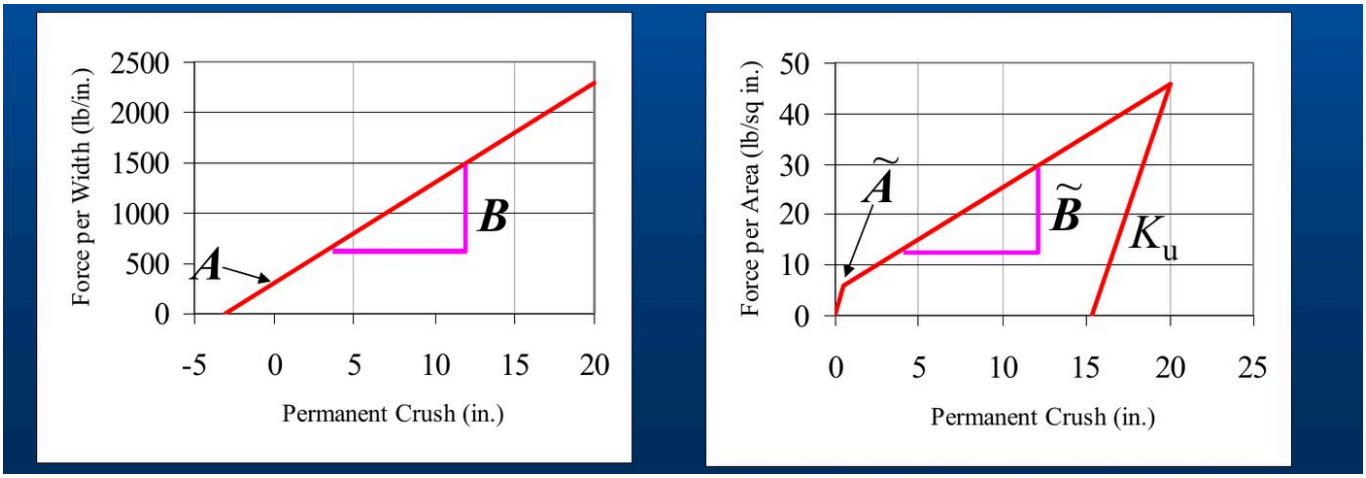


Figure: two force-deflection plots — Force per Width (lb/in) vs. Permanent Crush with intercept A and slope B; and Force per Area (lb/in²) vs. Permanent Crush with $\tilde{A}$, $\tilde{B}$ and unloading slope K_u.

3-D stiffness dialog

The 3-D stiffness dialog lets the user enter A, B, C and D coefficients and a **Saturation Crush**, and to "tune" the coefficients using the **Conversion Height** to arrive at accurate 3-D stiffnesses. The dialog fields are: K_v (lb/in³, unloading), Constant A (lb/in²), Linear B (lb/in³), Quadratic C (lb/in⁴), Cubic D (lb/in⁵), Saturation Crush (in), Maximum Crush (in), and Conversion Height (in, default 30). The user can now define their own third-order curve fit.

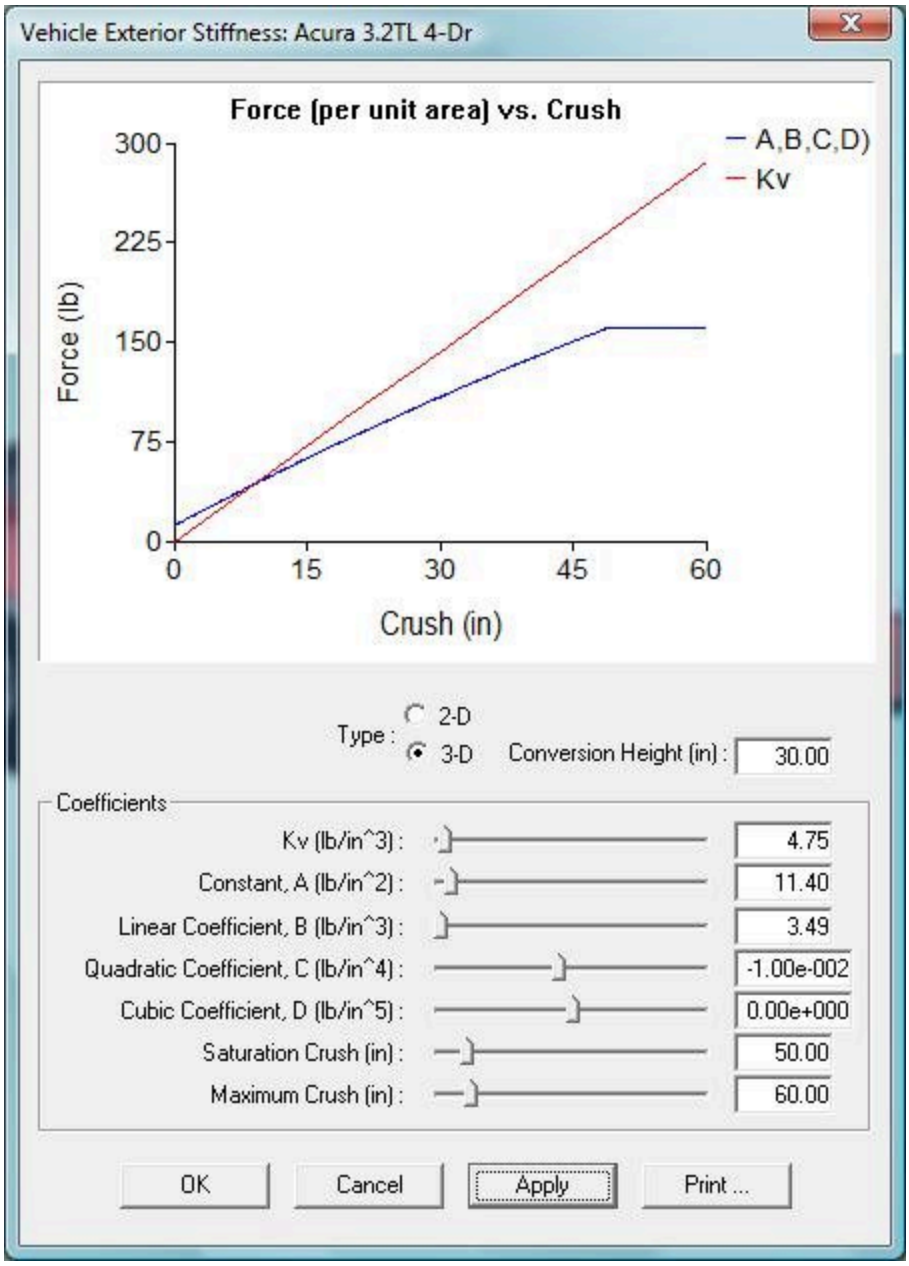


Figure: "Vehicle Exterior Stiffness" dialog showing the Force-per-area vs. Crush plot (A,B,C,D curve vs. K_v line) and the coefficient sliders.

Note on occupant response

The **selection** of stiffness coefficients is important for simulations involving occupant response: two acceleration pulses (e.g. a tall triangular pulse and a lower square pulse) can share the same velocity change Δv yet represent very different loading severities for the occupant. Choosing physically representative stiffness/saturation values therefore matters beyond matching the gross Δv.

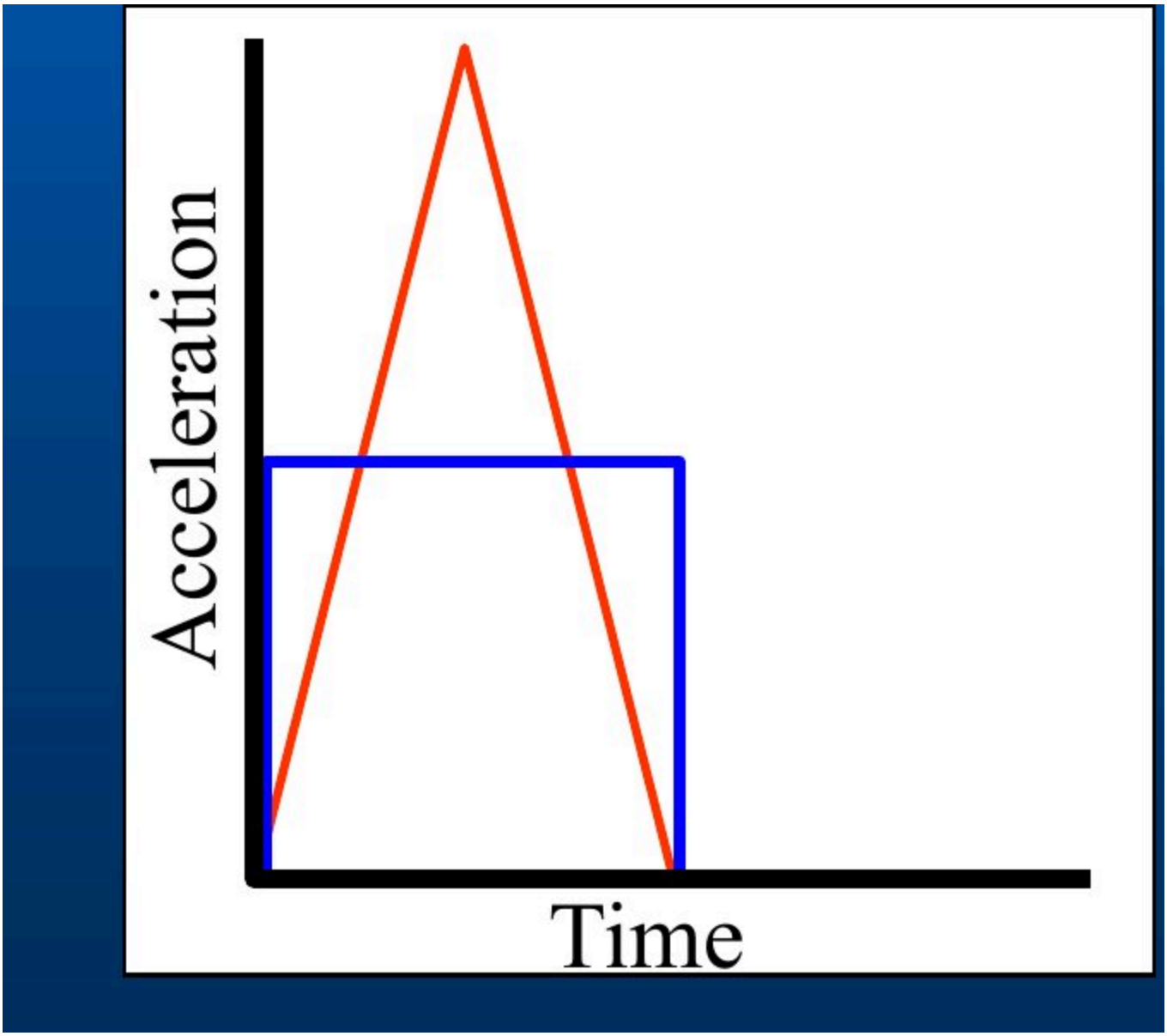


Figure: Acceleration vs. Time — a triangular pulse and a rectangular pulse with equal area (equal Δv).

5. Vertex displacement (crush depth) — sharing crush between bodies

The 1/18 "Crush Depth" update refined how the total penetration is *partitioned* between the two vehicles so that the **forces on the two bodies are equal and opposite** (Newton's third law) even when their stiffnesses differ.

Given the primary vehicle stiffness A_{primary}, B_{primary}, the secondary vehicle stiffness $A_{\text{secondary}}$, $B_{\text{secondary}}$, and the total secondary vertex displacement δ_{total} (the total penetration — **always computed using the secondary vehicle**), the force each vehicle *would* carry at the full penetration is:

$$F_{\text{primary}} = A_{\text{primary}} + B_{\text{primary}}\,\delta_{\text{total}} \qquad \text{(Eq. 4)}$$

$$F_{\text{secondary}} = A_{\text{secondary}} + B_{\text{secondary}}\,\delta_{\text{total}} \qquad \text{(Eq. 5)}$$

To equalize the forces, define the sharing fraction

$$\theta = \frac{F_{\text{primary}}}{F_{\text{primary}} + F_{\text{secondary}}} \qquad \text{(Eq. 6)}$$

and assign the secondary vehicle its share of the crush:

$$\delta_{\text{secondary}} = \delta_{\text{total}}\,\theta \qquad \text{(Eq. 7)}$$

There is no need to compute δ_{primary} separately — the primary's share is the remainder, $\delta_{\text{total}}(1-\theta)$. The softer body receives the larger share of crush, and the contact force is consistent for both bodies.

(updated: the primary triangle's effective stiffness used at a contact is the average of the A and B stiffnesses of its three corner vertices, consistent with this force-equalization approach.)

6. Restitution (unloading)

After peak penetration the node **unloads** along an unloading branch. Unloading is governed by a restitution coefficient e : a node only carries force while its current deflection exceeds $(1-e)\,\delta_{\max}$, i.e. force goes to zero once the node has relaxed by the restitution fraction of its maximum deflection:

$$F = 0 \quad \text{when} \quad \delta \le (1-e)\,\delta_{\max} \qquad \text{(Eq. 8)}$$

On the rebound branch the force is evaluated at the **rebound deformation** $\delta_r = \delta - (1-e)\,\delta_{\max}$ using the same cubic law as Eq. 1 (with δ_r in place of $\delta - \delta_0$), while the stored maximum deflection $\delta_{\max}$ is *not* updated. The unloading state of each node is tracked through a small sequence of states — cannot unload, beginning to unload, able to unload, now unloading, and finished unloading — governed by deflection thresholds and by a limit on how much the unloading can grow in a single time step. A separately stored unloading slope K_u defines the steeper unloading line (see the K_u line in the force-per-area plot).

DyMESH Version 2 "greatly improved" the restitution model together with the accelerations and the crush-depth (general damage profile) simulation.

7. Friction

Tangential (frictional) forces at the contact are computed from a material friction coefficient μ — the **Inter-vehicle Friction** value for vehicle-vehicle interactions, and the wheel material's own friction value for wheels — with a friction null band that suppresses spurious tangential force at very low relative sliding velocity.

8. Coupling to the equations of motion

For each interacting pair, DyMESH sums the node forces and moments into a resultant force and moment on each body's center of gravity: the earth-fixed node forces are accumulated into a vehicle-fixed summed force and a summed moment about the CG. These collision forces and moments are then **added to the vehicle's other external forces** (tires, drag, gravity, suspension) and the six-degree-of-freedom equations of motion are integrated by the host model (SIMON) over the collision time step. Because DyMESH is called every collision time step and the meshes are advanced with the vehicle motion, the collision force develops continuously through the contact rather than as a single impulse.

The vehicle orientation used to transform between the vehicle-fixed and earth-fixed frames is the standard Euler-angle direction-cosine matrix A, built from roll ϕ, pitch θ, and yaw ψ.

Source: DyMESH Collision Simulation (2026 HVE Forum). Updated to match the current version of HVE.

DyMESH Version 3

DyMESH Version 3 (Engineering Dynamics Company, 2026 HVE Forum) adds a collision model for the vehicle **wheels** — *DyMESH Wheels* — so that impact forces and moments developed at the tires/wheels are computed and fed into the vehicle dynamics. Version 3 also finalizes the user-defined third-order force-deflection stiffness described in the Collision Model chapter.

Problem statement

- DyMESH **Version 2** calculated 3-D collision forces and moments **only for the sprung mass** (the vehicle body).
- However, **significant collision forces and moments can occur during impact at the wheel(s)** — e.g. an angled wheel-to-wheel or wheel-to-body strike.
- **Ignoring these forces affects the collision dynamics** (post-impact heading, spin, and trajectory).

Solution criteria

The wheel collision model had to:

- Calculate forces and moments from the collision at **each individual wheel**.
- Integrate the results **directly into the equations of motion**.
- Be **robust**, accounting for:
- **Wheel shape** — a round cylinder.
- **Wheel location / displacement** (the wheel can be moved by the impact).
- **Wheel steering and camber change.**
- **Wheel spin** (tangential friction).

Solution

The wheel model reuses the DyMESH machinery by treating each wheel as its own DyMESH body:

- **Treat each wheel as an independent DyMESH object.**
- **Create the wheel mesh based on tire size.** A cylindrical wheel mesh is built from the tire radius and width, using 80 angular increments. Wheel material stiffnesses are set to their own values — A / B of 500 / 100 for truck wheels, or 250 / 50 for passenger-car wheels — and a saturation deflection tied to the wheel half-width is assigned per vertex.
- **Place transforms in front of and following the DyMESH calculation** to carry the wheel's location, steer, camber, and spin into and out of the collision computation.
- **The user can select individual wheels for analysis** (for computational efficiency — only the wheels expected to be involved need be simulated).

Flow charts

Version 3 adds a wheel loop nested within the existing sprung-mass collision loop.

Sprung Mass loop — for each pair of vehicles:

1. Collision Test → if no collision, skip.
2. Swap Vehicles (assign Primary / Secondary).
3. Get Vehicle Matrices.
4. **DyMESH (Primary, Secondary)** — the body-to-body collision.
5. **DyMESH Wheels (Primary, Secondary)** — the wheel collisions.
6. Smooth Collision Pulse.
7. Separation Test.

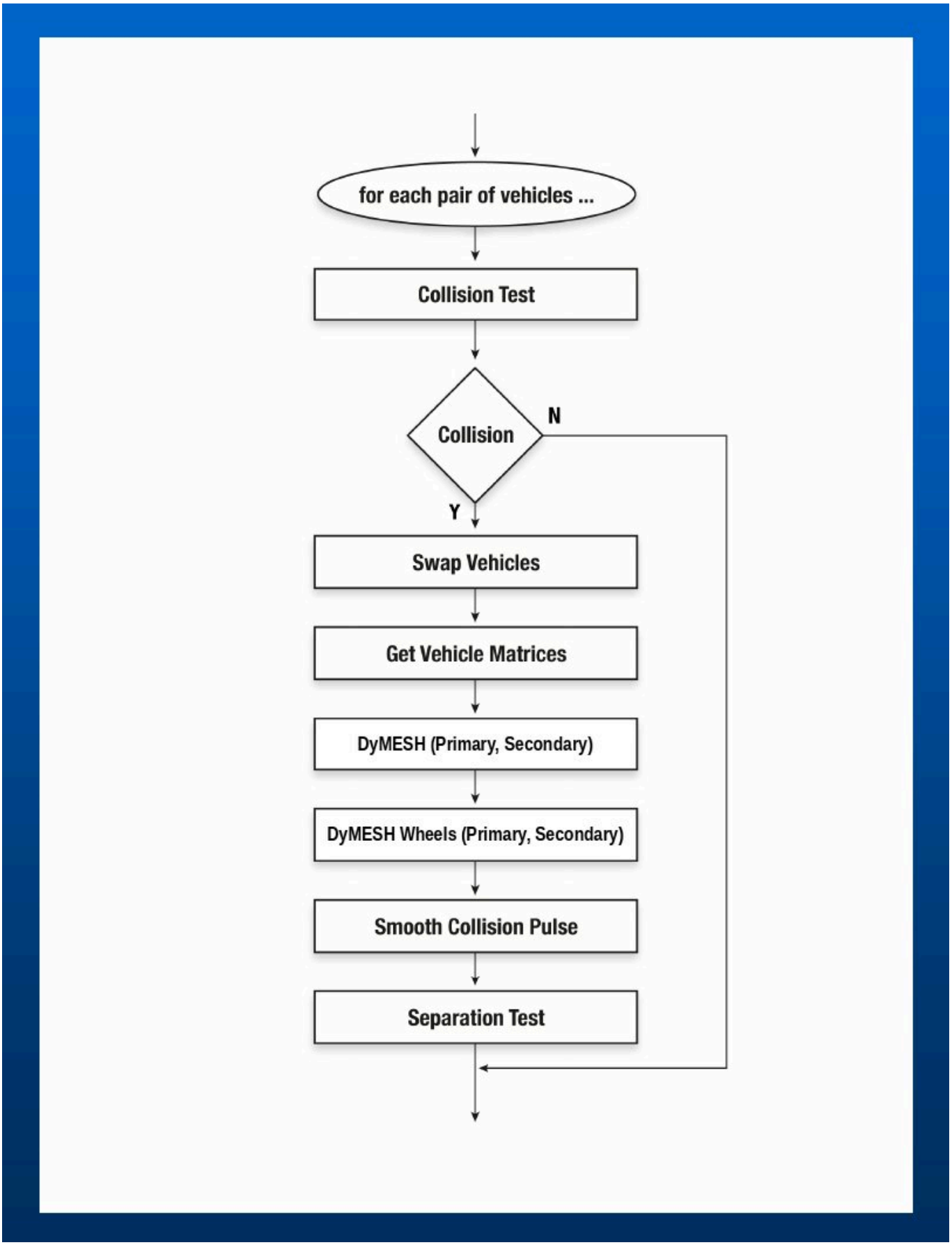

for each pair of vehicles ...
Collision Test
Collision
N
Y
Swap Vehicles
Get Vehicle Matrices
DyMESH (Primary, Secondary)
DyMESH Wheels (Primary, Secondary)
Smooth Collision Pulse
Separation Test

Figure: Sprung Mass flow chart.

Wheel loop — for each wheel:

1. Collision? → if no, skip.
2. Get Wheel Matrices.
3. Update Wheel Mesh (apply location/steer/camber/spin transform).
4. **DyMESH (Primary, Secondary)** — run the contact algorithm on the wheel mesh.
5. Add Wheel Force to Sprung Mass.
6. Update Wheel Mesh Damage.
7. Calculate Wheel Displacement.

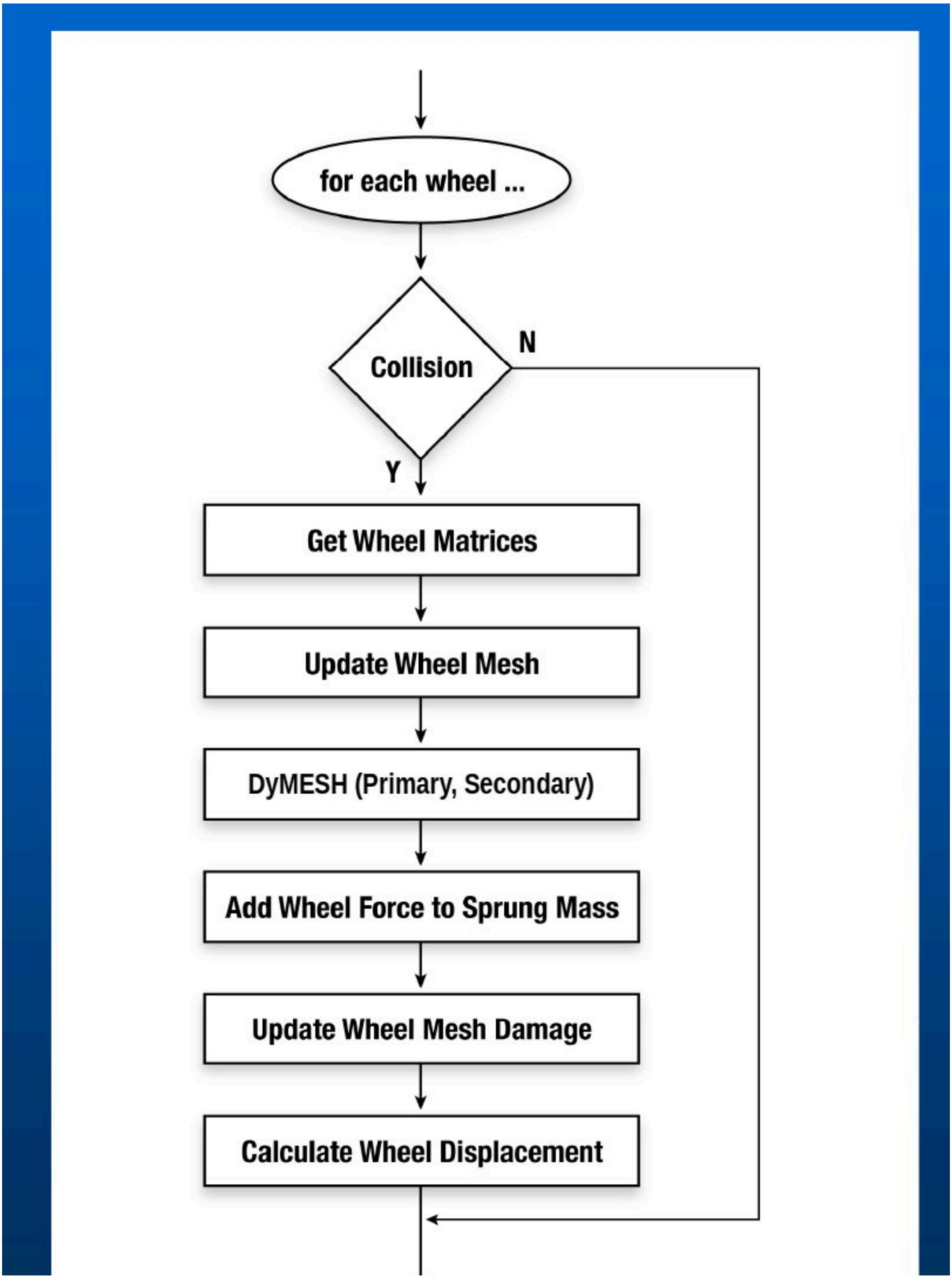

for each wheel ...
Collision
N
Y
Get Wheel Matrices
Update Wheel Mesh
DyMESH (Primary, Secondary)
Add Wheel Force to Sprung Mass
Update Wheel Mesh Damage
Calculate Wheel Displacement



Figure: Wheels flow chart.

(updated: in the current version of HVE these steps are carried out by SIMON, which updates each wheel mesh, adds the resulting wheel forces to the sprung mass, updates the wheel-mesh damage, and computes the resulting permanent wheel displacement.)

DyMESH Wheels user interface

The wheel model is driven by three areas of the interface: the **DyMESH Options**, the per-wheel **Set-up**, and the **Output vs. Time** results.

DyMESH Options dialog

The DyMESH Options dialog (Options menu) contains:

- **Use DyMESH** (turns DyMESH on or off).
- **Include Environment** — also run DyMESH contact against the environment (see Wheel vs. Environment for the wheel side of this feature and the tire-model handoff).
- **Force To x-y Plane** — keep the DyMESH deformation direction horizontal (not applied to contact with the environment).
- **Tow Vehicle / Trailer Contact** — allow contact between connected vehicles.
- **DyMESH Version No** — *Version 3* or *Version 4*.
- **DyMESH Start Time** and **DyMESH Environment Start Time** — when vehicle-to-vehicle and vehicle-to-environment contact detection begin.
- **Search** — *Automatic* or *Set Box Size* (a user-entered box size, e.g. 20.00 in).
- **Force Smoothing** — *Version 1 and 2* or *Version 3 and Later*.

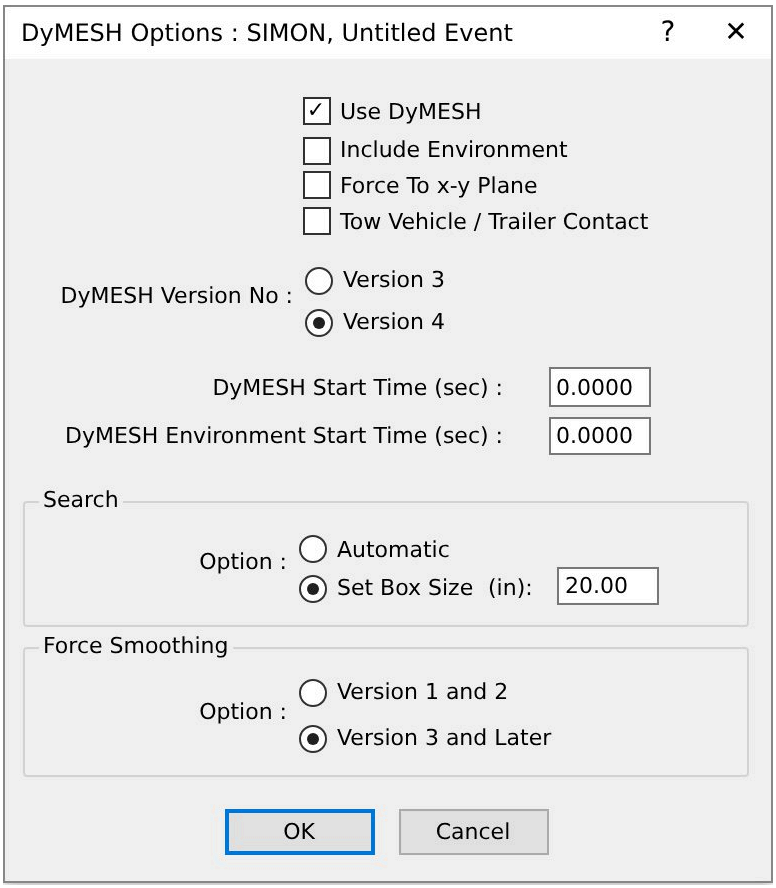


Figure: "DyMESH Options" dialog.

(updated: the deck shows an earlier version of this dialog, with Force Smoothing labelled Version 1 / Version 2 and a Timesteps group, and without the version, connection and start-time controls. Timesteps are now set in the Simulation Controls dialog. The low-level search and pushback settings offered on an Advanced tab in some earlier releases are fixed in the current release; see the HVE User's Manual, Options Menu, DyMESH, Fixed settings.)

Set-up (Wheels)

Each wheel is configured on the **Damage** tab of the Wheel Data dialog: Axle No., Side (Left / Right), *Wheel is Damaged*, *Auto Start*, Start Time, Duration (e.g. 0.1 s), and Peak Lock-up Torque. A **Use DyMESH** section supplies the wheel force-deflection and moment-deflection parameters:

Parameter	Example	Meaning
Max No-Damage Force	3000 lb	Force threshold below which the wheel is not permanently displaced
Displacement Rate	300 lb/in	Force per unit wheel displacement (stiffness)
Max Force	30000 lb	Saturation force
Max No-Damage Moment	3000 ft-lb	Moment threshold below which no permanent reorientation
Moment Displacement Rate	300 ft-lb/deg	Moment per unit angular change
Max Moment	30000 ft-lb	Saturation moment

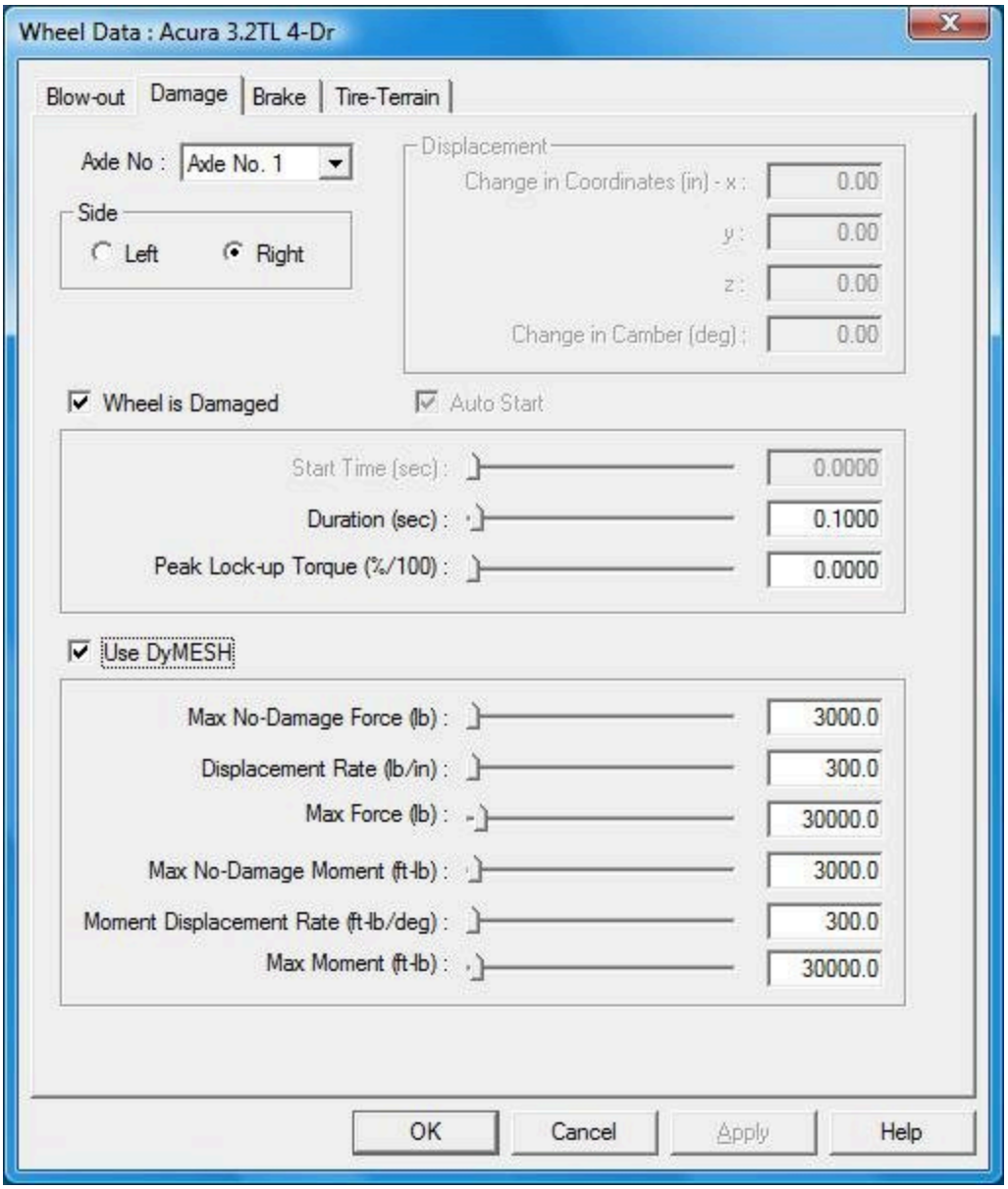
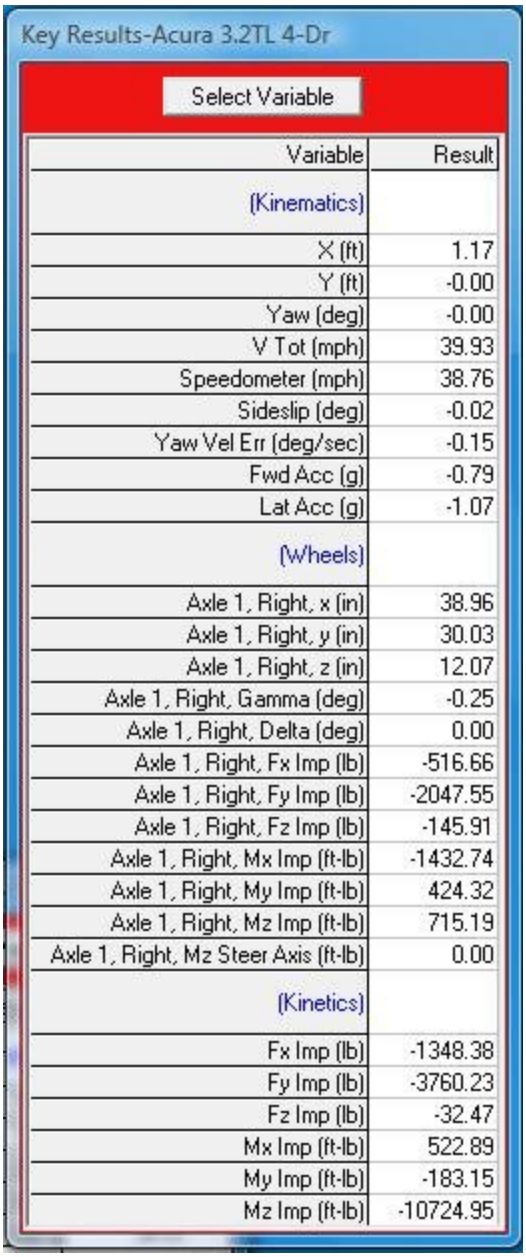


Figure: "Wheel Data" dialog, Damage tab, with the Use DyMESH force/moment fields highlighted.

See also the wheel-data / wheel-displacement inputs documented in the Wheel Data dialog reference.

Output vs. time

The Key Results output reports, per wheel, the collision impulses and moments in addition to the usual kinematics and kinetics — e.g. for "Axle 1, Right": **Fx Imp**, **Fy Imp**, **Fz Imp** (lb) and **Mx Imp**, **My Imp**, **Mz Imp** (ft-lb), alongside the whole-vehicle (Kinetics) impulses. This lets the user see how much of the collision load each wheel carried.



Key Results-Acura 3.2TL 4-Dr

Select Variable

Variable	Result
(Kinematics)	
X (ft)	1.17
Y (ft)	-0.00
Yaw (deg)	-0.00
V Tot (mph)	39.93
Speedometer (mph)	38.76
Sideslip (deg)	-0.02
Yaw Vel Err (deg/sec)	-0.15
Fwd Acc (g)	-0.79
Lat Acc (g)	-1.07
(Wheels)	
Axle 1, Right, x (in)	38.96
Axle 1, Right, y (in)	30.03
Axle 1, Right, z (in)	12.07
Axle 1, Right, Gamma (deg)	-0.25
Axle 1, Right, Delta (deg)	0.00
Axle 1, Right, Fx Imp (lb)	-516.66
Axle 1, Right, Fy Imp (lb)	-2047.55
Axle 1, Right, Fz Imp (lb)	-145.91
Axle 1, Right, Mx Imp (ft-lb)	-1432.74
Axle 1, Right, My Imp (ft-lb)	424.32
Axle 1, Right, Mz Imp (ft-lb)	715.19
Axle 1, Right, Mz Steer Axis (ft-lb)	0.00
(Kinetics)	
Fx Imp (lb)	-1348.38
Fy Imp (lb)	-3760.23
Fz Imp (lb)	-32.47
Mx Imp (ft-lb)	522.89
My Imp (ft-lb)	-183.15
Mz Imp (ft-lb)	-10724.95

Figure: "Key Results" output table highlighting the per-wheel impulse rows.

Wheel displacement capping *(updated)*

(updated: the current version of HVE adds a refinement not present in the 2026 decks. During each timestep the wheel-mesh damage update tracks the largest actual movement of any wheel-mesh vertex while in contact — the magnitude of the damaged-minus-undamaged vertex displacement, referred to below as the maximum wheel-mesh contact movement for that timestep. The permanent wheel displacement is then ***capped by that physical contact motion****. The displacement derived from the collision force,*

$$\delta_{Wheel} = \frac{F_{Total} - F_{Const}}{F_{Linear}}$$

where F_{Total} is the collision force on the wheel and F_{Const} and F_{Linear} are the constant (no-damage) force threshold and the linear force-per-displacement rate from the wheel's damage data, is limited to no more than the maximum wheel-mesh contact movement, and the per-time-step change in the x/y wheel displacement is likewise scaled so it cannot exceed that movement. This prevents the wheel from being "bent" further than the collision physically moved it.)

Validation

Validation is described as under way, starting from the initial conceptual design (see the Dial Engineering white-paper presentation at the 2012 HVE Forum), covering vehicle-to-vehicle collisions and other configurations.

Examples

The deck shows the wheel model applied to: an **intersection collision**, a **truck tractor**, a **trailer under-ride**, and (memorably) **monster trucks**. The main technical deck additionally shows angled collisions — a Ford pickup vs. Explorer on a graded road, and the RICSAC 1 (20 mph) and RICSAC 2 (32 mph) staged tests — as sample DyMESH validations.



Figure: Angled collision, Ford pickup vs. Explorer.

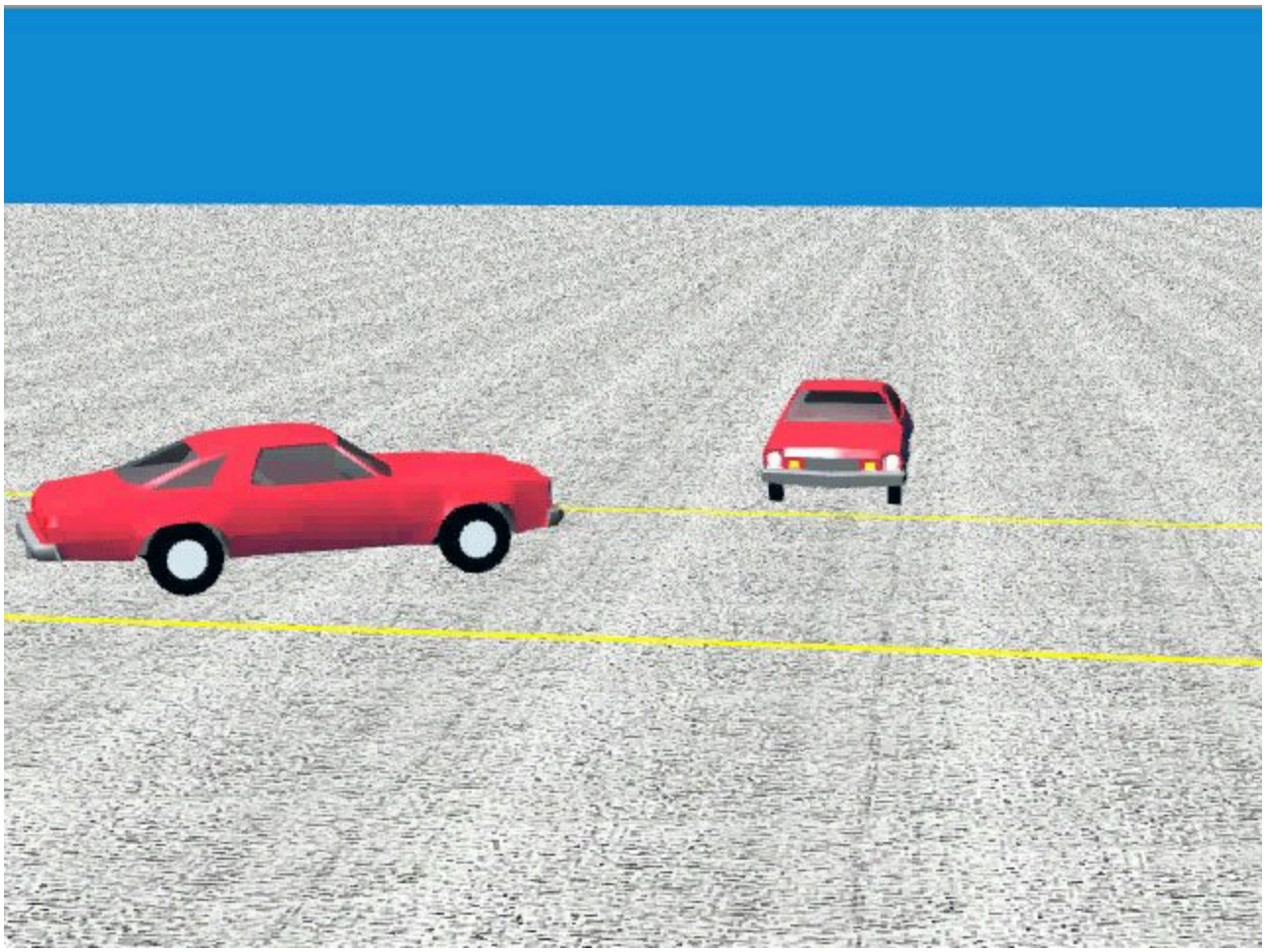


Figure: RICSAC 1 (20 mph) angled collision.

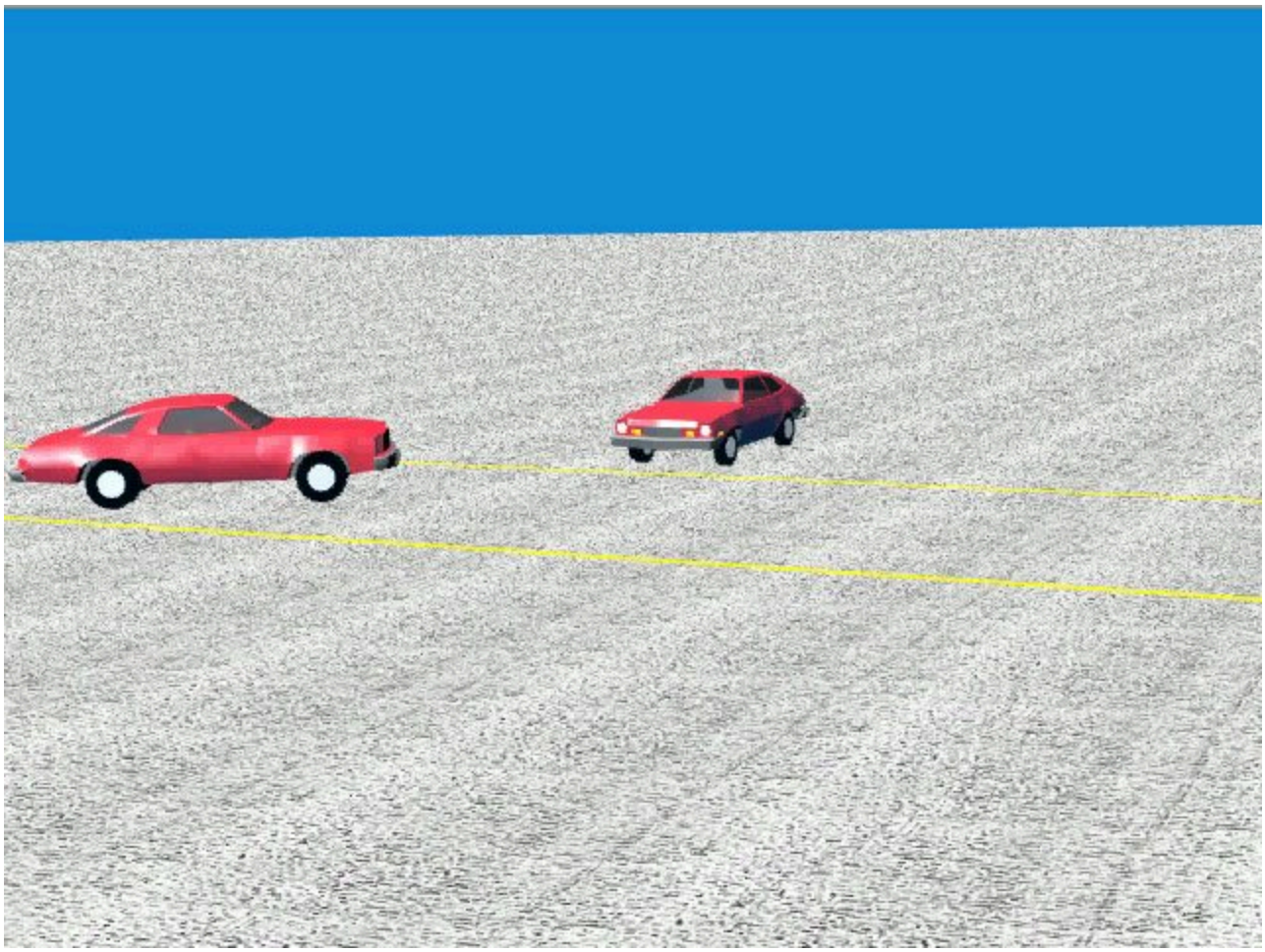


Figure: RICSAC 2 (32 mph) angled collision.

Summary

- **DyMESH Wheel Impact is robust.**
- **Clean integration into the existing vehicle dynamics model (SIMON).**
- **More validation is needed.**
- **Initially released as a Beta option, now included in the commercial release.**

Source: DyMESH Version 3 (2026 HVE Forum). Updated to match the current version of HVE.

DyMESH — Facts of Life

Practical guidance, caveats, and limitations for users of DyMESH, drawn from the EDC

DyMESH — Important Notes

presentation (2026 HVE Forum). These are the things to keep in mind before, during, and after a DyMESH run.

1. You are doing 3-D collision simulation

When using DyMESH, you're doing three-dimensional collision simulation. This is technically a very difficult task. **Don't expect it to be simple.** Setting up a credible DyMESH case takes care with geometry, stiffness, and impact configuration.

2. DyMESH will not work on every crash

DyMESH is a powerful general-purpose tool, but **it is not going to work on every crash.** Some impact configurations, geometries, or data quality problems will not yield a good simulation. Be prepared to fall back to other methods when a case is unsuitable.

3. Computer horsepower matters

Computer horsepower means everything to DyMESH. The contact search and force computation run every collision time step over the vehicle meshes, so CPU speed and memory directly determine how quickly (and how finely) a case can be run.

4. Be patient during a run

Be patient. HVE is *resting* in between integration time steps while DyMESH is executing, so quickly punching a bunch of buttons during a run may result in unexpected behavior. Let the run proceed rather than clicking through the interface while DyMESH is working.

5. No induced damage

Induced damage is not included. No damage will be displayed unless one or more vertices on one vehicle were *actually in contact* with the other vehicle. DyMESH represents only direct-contact (direct) damage — it does not model the secondary/induced structural deformation that propagates away from the contact region. (This is the same limitation noted in the technical overview: "local behavior of structure not captured in all cases; no induced damage.")

(updated: this is consistent with how the damage profile is built — a vertex only enters the damage profile once it has been marked as damaged, which happens only after it has penetrated a primary surface and been restored.)

6. Mesh quality is a factor

The quality of the vehicle mesh directly affects the result:

- **Poorly tessellated meshes do not work well.** The contact algorithm relies on reasonably uniform, well-formed triangles.
- **Early EDC meshes do not meet all criteria**, especially on the **bottom** of the vehicle.
- **Use the *Tessellation* option** to improve a mesh before running DyMESH.

(updated: The Tessellation / "Vehicle Mesh Options" controls in the current version include a Tessellate toggle, a Vertex Weld Distance, and a Max Side Length — these are the levers referred to here for cleaning up a mesh.)

7. Generic vehicles must be tessellated

Generic Vehicles' meshes *must* be tessellated before use with DyMESH. Their as-delivered meshes are not suitable for the contact algorithm without tessellation.

Source: DyMESH Important Notes (2026 HVE Forum). Updated to match the current version of HVE.

DyMESH Wheel vs. Environment

This chapter describes the **wheel vs. environment** extension of the DyMESH Wheels collision model: each selected wheel's cylindrical mesh collides with the **environment (terrain) mesh** as a full DyMESH contact pair. Wheel-to-curb strikes, ditch impacts, and ground contact during rollover are then computed by the DyMESH contact algorithm rather than by the point-contact tire model alone, which loses validity at large camber angles and large radial deflections.

Because the tire is now touching the ground through *two* models at once — the classic radial-spring tire model and the DyMESH wheel mesh — the model includes an explicit **handoff** between the two so the ground force is not double-counted. That handoff is described in detail in The handoff between the tire model and DyMESH, which is the heart of this chapter.

The description below reflects the behavior of the current version of HVE.

Enabling the model

Wheel vs. environment contact runs for a given wheel only when **all** of the following are true:

Level	Setting	Where
Event	**Use DyMESH**	DyMESH Options dialog
Event	**Include Environment**	DyMESH Options dialog
Event	**Environment start time** reached	DyMESH Options dialog
Wheel	**Wheel is Displaced**	Set-up → Wheels → Damage tab
Wheel	**Use DyMesh**	Set-up → Wheels → Damage tab
Wheel	**Use DyMesh Environment**	Set-up → Wheels → Damage tab
Wheel	Wheel damage start time reached	Set-up → Wheels → Damage tab

Notes:

- If *Include Environment* is off, or the environment has no tessellated mesh, wheel-environment contact never occurs.
- With **Auto Start** checked on the Damage tab, the wheel participates immediately; with an explicit start time the wheel joins the collision model only after that time.
- The *Use DyMesh Environment* checkbox is enabled in the dialog only when the wheel is displaced, DyMESH is on, *Use DyMesh* is checked, and the event's *Include Environment* is on.

The wheel as a DyMESH object

During initialization one mesh is built per selected wheel:

- **Object list.** The DyMESH objects are ordered vehicles first, then the environment, then the wheels. Each selected wheel keeps a stable object index for the whole run.
- **Geometry.** A closed cylinder with **80 angular slices**: $6 \times 80 + 2 = 482$ vertices and $12 \times 80 = 960$ triangles. From the hub outward: two hub-center vertices, two **rim disk** rings at the rim radius r_{rim} (parsed from the tire size designation), two **near-rim** rings at $r_{\text{near}} = [\, r_{\text{rim}} + 4\,(0.98\, r_{\text{tire}})\,] \,/\, 5$ — that is, four-fifths of the way out from the

rim radius to the tread radius — and two **tread-edge** rings at $0.98\,r_{\text{tire}}$, where r_{tire} is the tire **static loaded radius**. Dual tires are modeled as one wide cylinder whose width is the tire width plus the dual-tire spacing.

- **Vertex materials.** Each vertex carries an A (constant) and B (linear) stiffness, a friction coefficient, two restitution coefficients, and a saturation deflection:

Region	B stiffness	Friction	Restitution vs. vehicle	Restitution vs. environment	Saturation
Hub / rim disks	Wheel *Displacement Rate* ÷ wheel disc area	0.55	0.05	0.05	wheel width
Near-rim rings	average of rim and tread stiffness	0.75	0.05	0.05	wheel width
Tread ring	tire *Initial Ride Rate* ÷ contact-patch area	0.75	0.05	**1.00**	static loaded radius ÷ 4

Restitution here controls **permanent deformation**: a coefficient of 1.0 springs the vertex fully back each step (elastic), while 0.05 retains ~95% of the crush (plastic). Against the environment the tread is therefore fully elastic — the tire does not accumulate crush from ground contact — while the rim deforms permanently, as a real wheel does. The vs.-environment restitution value is selected automatically whenever the contact partner is the environment. Friction for any wheel contact is a fixed 0.75 for the tread and near-rim vertices (0.55 for the rim disks); the environment material's own friction value is not used for wheels.

The environment as a DyMESH object

- The GUI tessellates the environment geometry into an environment mesh; at initialization its vertices and faces are copied into the environment object slot, with per-vertex A/B stiffness taken from the environment surface materials.
- The environment is **immutable**: it never deforms, and the rebound (unloading) pass skips it entirely.
- A flat terrain is treated as a **solid block of earth**, not a thin shell: its bounding extents are padded downward — 60 in when the object extents are updated, and 120 in in the polygon-search bounding boxes — so a wheel cannot pass through a zero-thickness ground plane between timesteps.

The simulation loop

Each derivative evaluation of the vehicle dynamics model treats the environment as the last collision partner of every vehicle, in this order:

1. **Proximity test.** Bounding spheres are built — one for the sprung mass and one per active DyMESH wheel — and each wheel sphere that touches the partner is flagged as interfering.
2. **Body contact.** The standard DyMESH contact runs for the sprung-mass mesh against the environment.
3. **Wheel contact.** The wheel loop runs over the interfering wheels that pass the enable and time gates. For each wheel:
4. The wheel's transform is formed from its vehicle-fixed center position, camber angle, spin angle (the negative of the accumulated wheel rotation), and steer angle, plus their rates — the mesh physically spins, so vertex velocities include tread surface speed and friction acts in the correct direction.
5. The damaged and undamaged wheel meshes are placed in vehicle space.

6. The standard DyMESH contact algorithm is run between the environment and the wheel mesh (contact search, penetration, node force $F = (A + B\,\delta + C\,\delta^2 + D\,\delta^3)A_v$, restitution, friction), with two wheel-specific rules:
   - a blown tire's stiffness multiplier is applied to the wheel mesh;
   - **hub, rim, and near-rim vertices produce no vertical force** — their earth-fixed z force component is zeroed, so only the tread ring carries vertical load. Longitudinal and lateral forces still come from every vertex, so a rim striking a curb face pushes back horizontally.
7. The resulting wheel force is accumulated into the vehicle's wheel collision force totals.
8. The deformed mesh is mapped back into wheel space (persisting damage across steps) and the wheel-fixed force and the camber/spin/steer moments are summed. The largest movement of any wheel-mesh vertex during the step — the **maximum wheel-mesh contact movement** — is also recorded.
9. The permanent wheel displacement/bending model is applied, capped by that contact movement (see the Version 3 chapter); displacement is horizontal (x, y) only, since vertical wheel motion is a suspension degree of freedom.
10. **Smoothing.** Each wheel's force and moment is averaged with the previous step's values (two-point moving average). The smoothed previous vertical force is what the tire-model handoff (below) reads.

Force paths into the vehicle dynamics

Quantity	Path
Wheel collision Fx, Fy	Directly onto the sprung mass
Wheel collision Mz (yaw)	Directly onto the sprung mass
Wheel collision **Fz**	Into the **unsprung-mass equation of motion**, i.e., through the suspension
Spin moment (My′)	Into the wheel-spin equation as a collision torque
Camber / steer moments	Permanent camber/steer change via the wheel displacement model
Permanent x/y displacement	Added to the wheel's vehicle-fixed center position each step, with solid-axle side coupling

Because collision loads can legitimately exceed the normal suspension limits, the excessive-deflection, excessive-velocity, and excessive-force suspension error stops are suppressed for DyMESH wheels, as is the excessive-tire-deflection stop in the tire/terrain model.

The handoff between the tire model and DyMESH

With wheel vs. environment active, the same tire touches the ground through two models. Left uncorrected, the radial-spring tire force and the DyMESH tread force would both push the vehicle up and the total would be roughly double the correct load. The model therefore hands authority from one model to the other in three coordinated pieces, all gated on the wheel having

Use DyMesh Environment

selected and on both the event's environment start time and the wheel's damage start time having been reached.

1. Tire deflection rate limiting

The point-contact model computes the radial tire deflection δ_{tire} geometrically from the terrain under the wheel center. During a hard vertical event (curb face, ditch wall, rollover touchdown) that geometric deflection can jump in a single

timestep. When the wheel is a DyMESH environment wheel, the growth of δ_{tire} is limited per timestep:

- δ_{tire} may not exceed last step's deflection plus the tire's secondary-stiffness breakpoint deflection;
- if the tire carried **no** deflection on the previous step, δ_{tire} is forced to zero — a suddenly deep penetration produces *no* point-contact force at all, and the DyMESH wheel mesh carries the impact instead.

The tire radius used by the rest of the tire model is recomputed from the limited deflection, keeping the two models geometrically consistent.

2. Vertical force blending

After the radial tire force F_{tire} is computed, it is blended with the DyMESH result using three reference quantities:

- F_{DyMESH} — the previous step's **smoothed DyMESH vertical wheel force** (taken with the sign convention that upward ground reaction is positive), divided by the number of tires at the wheel;
- W_{static} — the **static load** carried at that wheel (static suspension force minus wheel weight, and minus half the solid-axle mass where applicable);
- F_{spring} — last step's deflection times the current radial stiffness (the force the tire spring "should" be carrying).

The blend then works case by case:

Case	Resulting radial force
Tire force below both the DyMESH force and static load	$F_{\text{DyMESH}} + F_{\text{tire}}$
Tire force below DyMESH force, at/above static load	$F_{\text{DyMESH}} + W_{\text{static}}$
Tire force ≥ DyMESH force, DyMESH force ≥ static load	$F_{\text{DyMESH}} + \min(W_{\text{static}}, F_{\text{spring}})$
Tire force ≥ DyMESH force, DyMESH force < static load (and > 0)	$F_{\text{DyMESH}} + \min(W_{\text{static}}, F_{\text{tire}})$
Deflection > 15% of tire radius while DyMESH is carrying load	$\min(F_{\text{tire}}, F_{\text{DyMESH}})$, or 0 if the DyMESH force ≤ 0

The intent in every case is the same: **the DyMESH force is authoritative, and the point-contact tire model contributes at most its static share on top of it.** In quiet rolling (DyMESH force ≈ 0) the ladder reduces to the ordinary tire force; in a hard strike the DyMESH force dominates and the tire spring is prevented from stacking a second full reaction on top.

Two additional rules close the loop:

- beyond the maximum roll angle the radial tire force is zeroed entirely — the DyMESH mesh is then the only ground contact;
- the final blended force is converted back into an equivalent deflection, $\delta_{\text{tire}} = F_{\text{tire}} / K_t$ using the radial tire stiffness K_t, capped at the tire's maximum deflection, so that outputs and the next step's rate limiter see a consistent state.

3. Shear-force fade near rollover

The friction-circle tire model produces Fx/Fy from slip at the contact patch — meaningless once the wheel plane approaches the ground plane. For a non-DyMESH wheel, tire forces are simply zeroed when the cosine of the wheel-to-ground inclination angle γ_g falls below the minimum-inclination threshold $c_{\min} = 0.17365$ (inclination beyond ≈80°).

For a DyMESH environment wheel they are instead **faded smoothly**: each of the contact-patch forces F_x, F_y and F_z, the plough forces, the slips, and the roll moment is scaled by

$$f = \frac{2}{\frac{\cos\gamma_g}{c_{\min}} + \frac{c_{\min}}{\cos\gamma_g}} \qquad \text{(Eq. 1)}$$

which equals 1.0 exactly at the threshold and falls toward 0 as the tire goes flat. This avoids the force discontinuity of the hard cutoff while the DyMESH tread/rim contact takes over the shear loads through its own friction model.

Summary of the handoff

Regime	Ground normal force	Ground shear force
Normal rolling	Tire model (DyMESH ≈ 0)	Tire model
Curb/obstacle strike	DyMESH tread force + capped tire share	Tire model + DyMESH friction
Near/at rollover (>~80° inclination)	DyMESH tread force (tire force faded/zeroed)	DyMESH friction (tire forces faded)

Outputs

- The per-wheel collision impulses (**Fx Imp**, **Fy Imp**, **Fz Imp**, **Mx Imp**, **My Imp**, **Mz Imp**) appear in the Key Results output, and the smoothed wheel collision forces and wheel-fixed moments drive the Output vs. Time wheel channels.
- The program-data output tables list each DyMESH wheel's mesh size and its first/last vertex A and B stiffnesses (the rim and tread values).

Quick reference

Parameter	Value / source	Meaning
Angular slices	80	Angular increments in the wheel mesh
Wheel mesh size	482 verts / 960 triangles	Per wheel
Tread radius	0.98 × static loaded radius	Outer ring radius
Tread restitution vs. environment	1.0	Fully elastic (no permanent tire crush)
Rim restitution	0.05	Plastic (permanent rim deformation)
Wheel contact friction	0.75 (tread/near-rim), 0.55 (rim disks)	Fixed values
Rim B stiffness	*Displacement Rate* ÷ wheel disc area	From Damage tab
Tread B stiffness	*Initial Ride Rate* ÷ contact-patch area	From tire data
Minimum-inclination threshold	0.17365 (cos ≈ 80°)	Shear-fade threshold
Environment depth padding	60–120 in	Prevents ghosting through flat terrain
Environment start time	DyMESH Options dialog	Earliest wheel/body vs. environment contact