VisSim Tutorial Series

Fundamentals of Mathematical Modeling and Simulation

***Peter Darnell**, president, Visual Solutions, Inc.*

***Arun Mulpur,** product manager**,** Visual Solutions, Inc.*

VisSim Tutorial Series

Fundamentals of Mathematical Modeling and Simulation

Excerpted with permission from *Modeling and Visual Simulation in Industry*, A. Mulpur and P. Darnell, International Thomson Computer Press, Boston, MA, 1997.

Other books in the VisSim Tutorial Series include:

- *Biomedical Systems: Modeling and Simulation of Lung Mechanics and Ventilator Controls Design*. Mike Borrello, Metran America, Inc.
- *Heating, Ventilation and Air Conditioning (HVAC) Controls: Variable Air Volume (VAV) Systems*. Nebil Ben-Aissa, Johnson Controls, Inc.
- *Introduction to 6-DOF Simulation of Air Vehicles*. Robert Josselson, ITT Aerospace Systems Group.
- *Simulation of Communication Systems*. Eugene Estinto, Eritek, Inc.
- *Simulation of Motion Control Systems*. William Erickson, Indramat-Rexroth.

Table of Contents

Problem Statement

To better illustrate the principles of modeling and simulation, a system comprising two masses m_1 and m_2, shown in Figure 1, is considered as a case study. In this system, the motion of the system components is assumed to be restricted to one-dimension, along the x-axis. The masses slide on a flat surface, and the coefficients of friction between the two masses and the flat surface are given by μ_1 and μ_2, respectively.

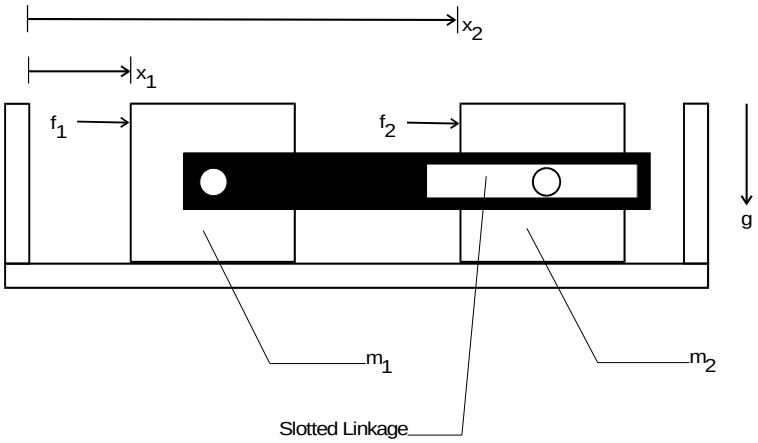

Figure 1. Schematic representation of the two mass system connected by a slotted linkage

As shown in the figure, the motion of the two masses is constrained by travel stops on either end of the flat surface and by a slotted linkage that limits the relative displacement of the masses. Gravity g is the normal force acting between the two masses and the surface.

An external force $f_1(t)$ acts on m_1. The travel stop, represented by the left wall, constrains the position of m_1, represented by x_1, such that it is not less than a minimum value $x_{left\text{-}wall}$. When m_1 collides with the left wall, the block should rebound with a coefficient of restitution $r_{left\text{-}wall}$.

The mass m_2 is subjected to an external force of $f_2(t)$. The right wall serves as a travel stop and constrains the position of m_2, represented by x_2, to be not more than $x_{right\text{-}wall}$. Upon collision with the right wall, m_2 bounces back with a coefficient of restitution $r_{right\text{-}wall}$.

The slotted linkage connecting the two masses is assumed to be mass-less. It constrains the relative displacement of the two masses

$$x_{diff} = x_2 - x_1$$

such that it is not less than $x_{diff\text{-}min}$ and not more than $x_{diff\text{-}max}$. When

$$x_{diff} \leq x_{diff-min}$$

the two blocks collide with a coefficient of restitution, $r_{slot\text{-}min}$. When

$$x_{diff} \geq x_{diff-max}$$

the two blocks collide with a coefficient of restitution, $r_{slot\text{-}max}$.

A model that describes the motion of the two masses subject to given constraints must be developed. In the event of a collision, the mass velocities after the collision must be computed, depending on the nature and type of collision. The model must be tested with various forcing inputs, $f_1(t)$ and $f_2(t)$, and different system parameter values.

The main parameters of interest are gravity g, coefficients of friction μ_1 and μ_2; coefficients of restitution $r_{left\text{-}wall}$, $r_{right\text{-}wall}$, $r_{slot\text{-}min}$ and $r_{slot\text{-}max}$; masses m_1 and m_2; positions of the left and right walls $x_{left\text{-}wall}$ and $x_{right\text{-}wall}$; and relative block displacement limits $x_{diff\text{-}min}$ and $x_{diff\text{-}max}$.

In each case, the pertinent system variables that need to be observed are time t, position of first mass x_1, velocity of first mass v_1, position of second mass x_2, velocity of second mass v_2, and the total kinetic energy is

$$KE = \tfrac{1}{2} m_1 v_1^2 + \tfrac{1}{2} m_2 v_2^2 .$$

Modeling and simulation principles required to solve this problem are examined next.

Physics of Objects Colliding in One Dimension

The coefficient of restitution is usually defined as the ratio of the forces that resist deformation R, to the expansion forces R'

$$r = R / R'$$

In a perfectly elastic collision, $r = 1$ and in a perfectly inelastic collision, $r = 0$. In the latter case, the colliding objects stick together after collision and can be considered a single larger object.

In many cases, including the case at hand, the positions and velocities of the colliding objects after impact are the points of interest. To determine these parameters, it is not necessary to know the exact magnitude, time variation, and duration of the resistive and restoring forces. Instead, an arbitrary model for the resistive and restoring forces can be assumed to be as follows.

Considering a two-object system, prior to collision, the two objects are moving towards each other. It is assumed that the first object has a mass of m_1 and a velocity of v_1. Similarly the second object is assumed to have a mass of m_2 and velocity of v_2. Let us assume that the collision occurs at time $t = 0$ and the duration of the collision is τ seconds. In this discussion, R_0 is the peak resistive force at the time of maximum compression. The arbitrary collision model assumes that there are two phases during the collision: compression and expansion. It is further assumed that the two phases are of equal duration $\tau/2$. Finally, after collision, the two objects move away or move together, depending on the type of the collision. The various stages of the two-object collision scenario are shown in Figure 2.

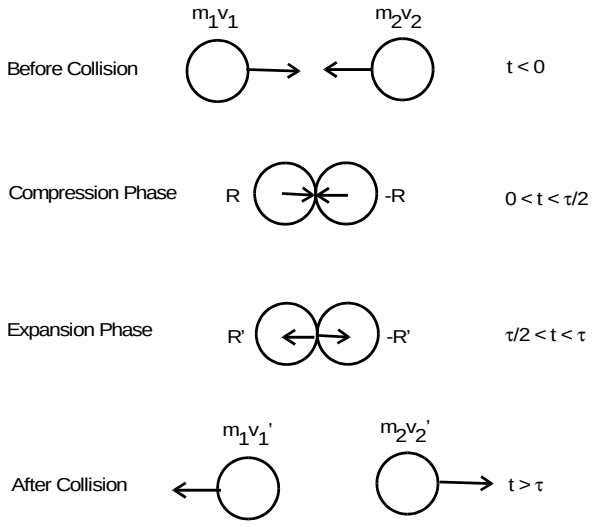

Figure 2. Collision of two objects in one dimension

During the compression phase of the collision, the resistive force is given by

$$R(t) = R_0\, 2t / \tau$$

and, during the expansion phase, the restoring force is given by

$$R'(t) = rR_0\, 2(t - \tau) / \tau$$

Integration of these equations separately from $t = 0$ to $\tau/2$ and $t = \tau/2$ to τ, respectively, yields

$$m\left(v_0 - v_1\right) = \int_0^{\tau/2} R dt$$

$$m\left(v_0 - v_2\right) = -\int_0^{\tau/2} R dt$$

and

$$m\left(v_1' - v_0\right) = \int_{\tau/2}^{\tau} R' dt$$

$$m\left(v_2' - v_0\right) = -\int_{\tau/2}^{\tau} R' dt$$

Next, two impulse forces ρ and ρ' are defined as

$$\rho = \int_0^{\tau/2} R dt = R_0 \tau$$

$$\rho' = \int_{\tau/2}^{\tau} R' dt = rR_0 \tau$$

The above result, $\rho' = r\rho$ is independent of the collision model assumed, and other models of this form give the same result. At the time of maximum compression, $t = \tau/2$, there is no relative motion for either object. At this instant, the two objects move together at the same velocity v_0 for an instant. This allows one to write the following two equations:

$$m_1 \frac{dv_1}{dt} = m_1 \frac{d^2 x_1}{dt^2} = R(t)$$

$$m_2 \frac{dv_2}{dt} = m_2 \frac{d^2 x_2}{dt^2} = -R(t)$$

Rewriting the momentum equations in terms of ρ and ρ' yields

$$m_1(v_0 - v_1) = \rho$$

$$m_2(v_0 - v_2) = -\rho$$

$$m_1(v_1' - v_0) = r\rho$$

$$m_2(v_2' - v_0) = -r\rho$$

Eliminating the two unknowns ρ and v_0, a solution is obtained for the post-collision velocities v_1' and v_2' in terms of m_1, m_2, v_1, and v_2 as

$$v_1' = \frac{(m_1 - rm_2)v_1 + (1+r)m_2 v_2}{m_1 + m_2}$$

$$v_2' = \frac{(m_2 - rm_1)v_2 + (1+r)m_1 v_1}{m_1 + m_2}$$

For the sake of completeness, in the case of a perfectly inelastic collision, $r = 0$ and consequently

$$v_1' = v_2' = \frac{m_1 v_1 + m_2 v_2}{m_1 + m_2}$$

If the two masses are equal, and the collision is perfectly elastic, it follows that

$$v_1' = v_2$$

and

$$v_2' = v_1$$

Based on this knowledge of colliding objects, a mathematical model for the entire system can be developed.

Development of Mathematical Model

A practical approach to developing a mathematical model involves developing models of sub-systems and then connecting them together to model a larger system. This section presents the development of the mathematical model and its VisSim representation in small, incremental, and interrelated steps.

Force Balance

Considering m_1, it is clear that the external applied force $f_1(t)$ facilitates the mass to move. Forces acting on m_1 that enable it to resist motion are the net frictional force and the gravitational force. Similar observations can be made about m_2.

Net Frictional Force

If the velocity of the object is zero, the Coulomb friction opposes the applied force until the applied force is sufficiently large to overcome the Coulomb friction. When the object is in motion, the frictional force experienced by the object is the sliding frictional force. In this example, it is assumed that dry friction (static friction) has the same magnitude as sliding friction (dynamic friction), given by the product of the coefficient of friction and the normal force

$$f_f = m_1 \mu_1 g$$

If the above requirements are converted into a programming structure, the following rules are obtained:

- If the velocity is not equal to zero, net frictional force is equal to the sliding friction force: $m_1 * \mu_1 * g$
- If the velocity is equal to zero, the net frictional force is equal to the smaller of the following two quantities – static friction force $m_1 * \mu_1 * g$ and the external applied force $f_1(t)$ – min ($m_1 * \mu_1 * g$, $f_1(t)$)
- The direction of the net frictional force is always such that it opposes the external applied force $f_1(t)$

In many practical cases, the sliding and static friction forces have different magnitudes. For such cases, the appropriate values for the sliding and static friction coefficients must be used in the above structure. The net frictional force sub-system can now be developed in VisSim. At the top level, it is a compound block with three inputs and one output as shown in Figure 3.

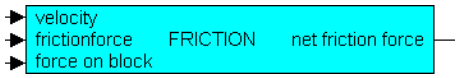

Figure 3. Top-level FRICTION block

The inputs to the compound block are velocity, friction force, and force on block; the output of the block is the net friction force. In Figure 4, the contents of the `Friction` block are shown.

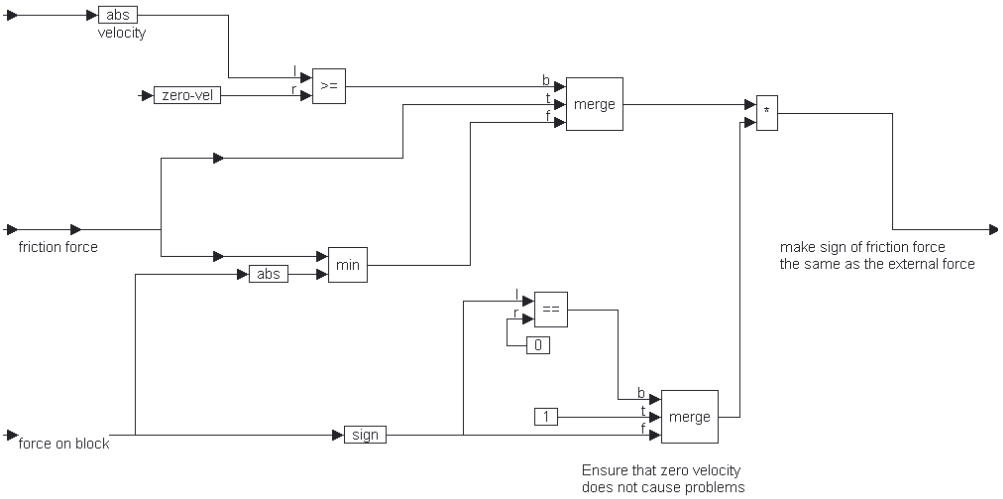

Figure 4. Computing net frictional force

As with programming in any language, it is generally not a good idea to perform Boolean equality comparisons involving floating point values in VisSim. All equality comparisons, such as {`IF velocity IS EQUAL TO zero, THEN`…} must first be converted to Boolean inequality comparisons such as {`IF velocity IS LESS THAN OR EQUAL TO` <a small value>, `THEN` …}.

The reason for this is that floating point variables such as *velocity* are rarely exactly equal to zero, if they are obtained by solving one or more equations. Consequently, the velocity is compared with a zero velocity threshold value that is set externally. It is clear that the three rules developed earlier for computing and applying frictional force are enforced in the VisSim diagram segment shown in Figure 4.

A word of caution is in order here. The output of the `sign` block in VisSim is +1 if the input is positive, -1 if the input is negative, and 0 if the input is equal to zero. Since the output of the `sign` block is multiplying the magnitude of the friction, we must ensure that when the velocity is zero, the force is unchanged.

Collision Detection

The collision detection logic is considered next. Recalling that x_{diff} is the difference between the positions of the two masses

$$x_{diff} = x_2 - x_1$$

According to the problem specification, if

$$x_{diff} < x_{diff\text{-}min}$$

or if

$$x_{diff} > x_{diff\text{-}max}$$

the blocks undergo a link collision. Similarly, if

$$x_1 < x_{left\text{-}wall}$$

m_1 collides with the left wall; and if

$$x_2 > x_{\text{right-wall}}$$

m_2 collides with the right wall. The collision detection logic that performs these functions is shown in Figure 5.

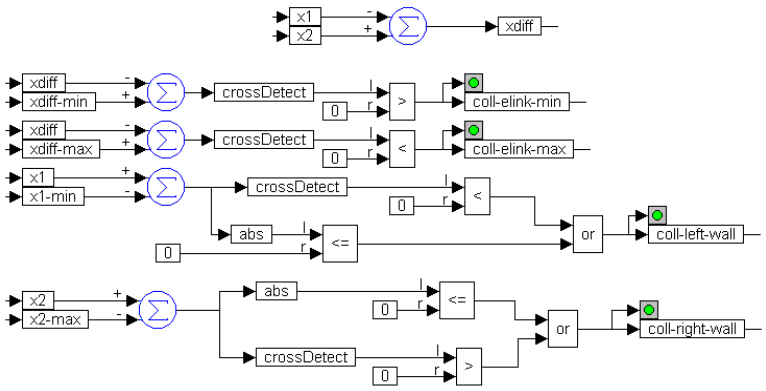

Figure 5. Collision detection logic

In order to precisely detect the instant of occurrence of a collision, `crossDetect` blocks are used. To set the `crossDetect` blocks up to perform *zero crossing* detection, the difference between the two values to be compared, is computed. This difference is then fed to the `crossDetect` block such that when the two variables being compared are equal, their difference becomes zero, which is then detected by the `crossDetect` block.

Since the `crossDetect` block outputs a +1 or a -1, depending on whether the signal crossed the set point (in this case, the set point is zero), with a positive slope or a negative slope, one can use the appropriate Boolean logic to convert the `crossDetect` output to a detected collision. So, whenever a collision occurs, the detection logic generates a pulse and assigns it to the appropriate variable.

Further, two special cases need to be addressed. When m_1 is in constant contact with the left wall, the collision detection logic must output a series of collision pulses – one for every time step, for the duration for which m_1 is contact with the left wall. Similar logic must be set up for the case where m_2 is in constant contact with the right wall. These cases are addressed by considering that if the difference between x_1 and $x_{left\text{-}wall}$ or x_2 and $x_{right\text{-}wall}$ is equal to zero, the appropriate collision pulse must be generated, independent of whether a zero crossing of the x_{diff} variable was observed.

Setting Boolean Variables

The next step is to use the collision logic to determine the course of action when collisions of different types occur. From the problem statement, it follows that v_1 must be reset when m_1 undergoes a left wall collision, or a slot link collision with m_2. Similarly, v_2 must be reset when m_2 undergoes a right wall collision, or a slot link collision with m_1. Consequently two Boolean variables `rst-x1dot` and `rst-x2dot` are defined as shown in Figure 6.

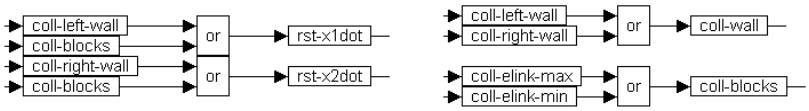

Figure 6. Boolean variables for collision detection

Further, since the physics involved in wall collisions is different from that in block collisions, block velocities and positions must be reset differently, depending on the type of collision. To facilitate this, two more Boolean variables `coll-wall` and `coll-blocks` are defined. The variable `coll-wall` is true if { m_1 collides with the left wall OR m_2 collides with the right wall}; `coll-blocks` is true if {a link minimum collision occurs OR a link maximum collision occurs}. These four Boolean variables are shown in Figure 6.

Computing Post-Collision Velocities

It is assumed that a $slot_{min}$ collision and a $slot_{max}$ collision cannot occur at the same instant. Consequently, in the event of a slot collision, the corresponding coefficient of restitution is computed as shown in Figure 7.

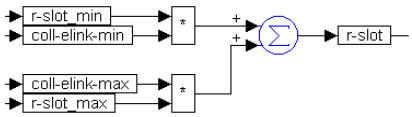

Figure 7. Computing effective coefficient of restitution in the event of a slot link collision

As depicted earlier in Figure 5, `coll-elink-min` and `coll-elink-max` are Boolean variables that are "high" when the corresponding collision, $slot_{min}$ or $slot_{max}$, is detected. From Figure 7, it is seen that `r-slot` is equal to `r-slot_min` or `r-slot_max`, depending on whether a $slot_{min}$ or a $slot_{max}$ collision was detected.

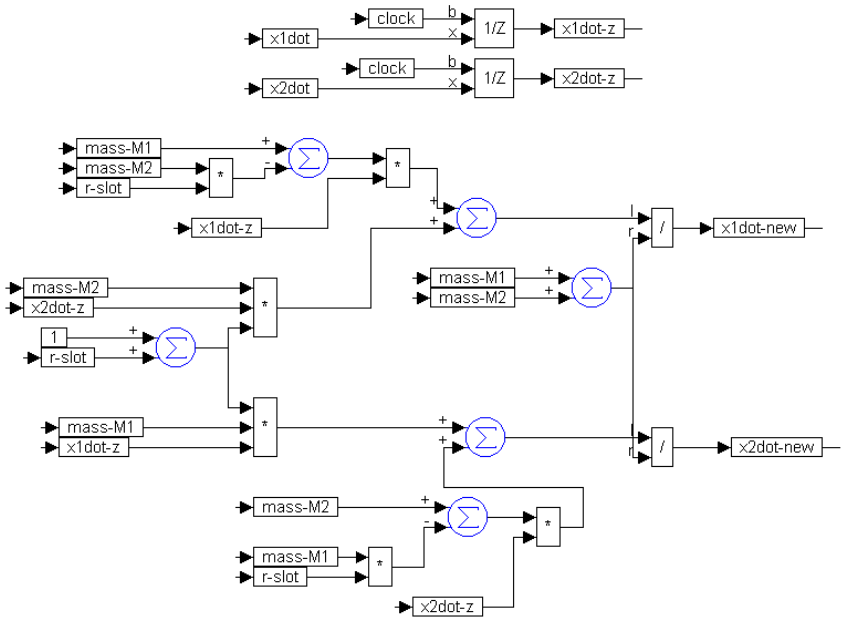

Figure 8. Computing link collision bounce velocities

From Figure 8, it is clear that `x1dot-z` and `x2dot-z` contain the "previous" (or, one-step delayed) velocities v_1 and v_2. In the event of a link collision, the post-collision velocities are given by

$$v_1^{new(slot)} = \frac{(m_1 - m_2 r_{slot})v_1^{prev} + (m_2 v_2^{prev})(1 + r_{slot})}{(m_1 + m_2)}$$

$$v_2^{new(slot)} = \frac{(m_2 - m_1 r_{slot})v_2^{prev} + (m_1 v_1^{prev})(1 + r_{slot})}{(m_1 + m_2)}$$

where r_{slot} is the corresponding coefficient of restitution that is applicable for the given collision, as computed in Figure 7.

As mentioned before, the physics that governs the collision of a mass with an immovable wall is different from that which describes the slot collisions. On colliding with a wall, a mass must rebound with a velocity equal to

$$v_1^{new(wall)} = -v_1^{prev} r_{left-wall}$$

$$v_2^{new(wall)} = -v_2^{prev} r_{right-wall}$$

A note of caution is required at this point. The case when a wall collision immediately follows a link collision poses special problems. In this case, to avoid incorrect results, one must ensure that the velocity at the instant preceding the wall collision is given by `x1dot-z` or `x2dot-z` under normal conditions; and by `x1dot` or `x2dot` if at the instant preceding the wall collision, a slot collision occurred.

The diagram segment for computing new velocities following wall collisions and routing the correct reset values to the `rst-x1dot-val` and `rst-x2dot-val` variables is shown in Figure 9.

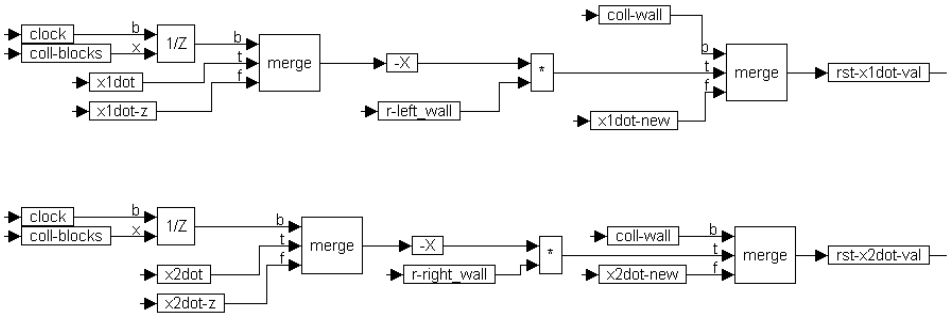

Figure 9. Determination of block velocity reset values

From Figure 9, several subtle but important points are apparent. If no wall collisions occurred in the previous instant, the reset values are `x1dot-new` and `x2dot-new` (link collision reset velocities) as computed in Figure 8. If left or right wall collisions occurred in the previous instant, the corresponding velocity is reset to $-r_{left\text{-}wall} * v_1^{prev}$ or $-r_{right\text{-}wall} * v_2^{prev}$, as appropriate. If left or right wall collisions occurred in the previous instant, and in the instant prior to that, a block collision occurred, the reset velocity is computed using v_1 and v_2 instead of v_1^{prev} and v_2^{prev}. Obviously, if no collisions occurred in the previous instant, no adjustments are made to the block velocities.

Enforcing No Movement on Zero Velocity

Similar to the Boolean equality evaluations described previously, another possible source of errors in dynamic simulations is the spurious movements produced by near-zero velocities and forces. This is particularly important in cases where exact simulation of nonlinear frictional behavior is required.

Figure 10. Velocity threshold to enforce zero movement (top level)

Figure 10 represents a velocity thresholding operation at a higher level. The actual signal and a zero threshold value are provided as inputs to the compound block, and a corrected value is returned as the output.

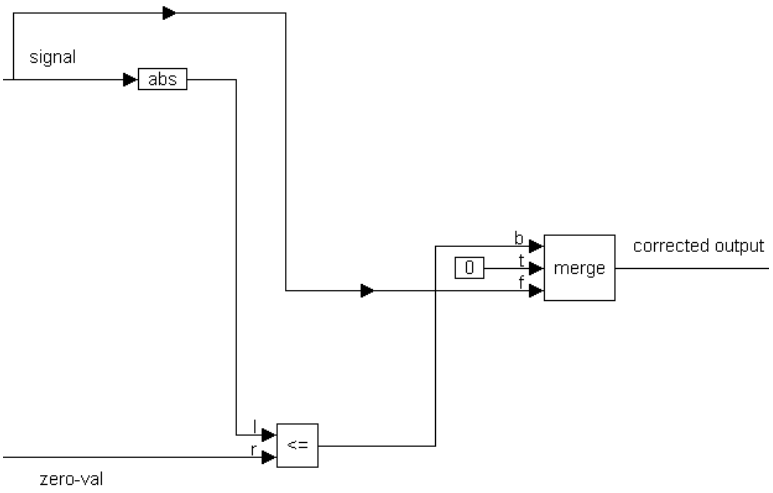

Figure 11. Velocity threshold to enforce zero movement (logic)

From the logical operations depicted in Figure 11, it is seen that when the absolute value of the block velocity is less than or equal to the zero threshold value, the velocity is corrected to be equal to zero. Otherwise, the velocity is left unchanged.

System Dynamics

The actual dynamics of the system are examined next. As with most numerical simulations, the solution to a dynamic system is obtained by rewriting the corresponding differential equations as equivalent integral equations.

To compute the acceleration v_1^{dot} of mass m_1, the net force acting on m_1 is divided by m_1. This operation is shown in Figure 12.

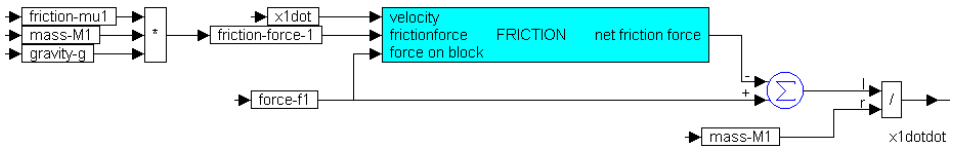

Figure 12. Computation of v_1^{dot}

As examined previously, the output of the `FRICTION` compound block is the net frictional force acting on m_1. As the frictional force always opposes the motion of m_1, the net force acting on m_1 is the difference between the external force f_1 and the net frictional force. The kinematic acceleration v_1^{dot} of m_1 is obtained by dividing the net force with m_1 Similar algebra is performed to determine the acceleration v_2^{dot} of mass m_2.

To obtain the instantaneous velocity and position of m_1, v_1^{dot} must be integrated twice. The first integration stage is shown in Figure 13. VisSim provides a `resetIntegrator` that can be reset to a particular value, whenever a Boolean input becomes "true." In this case, the Boolean input is `rst-x1dot` which was determined in Figure 6, and is "true" on the occurrence of either a slot collision or if m_1 collides with the left wall.

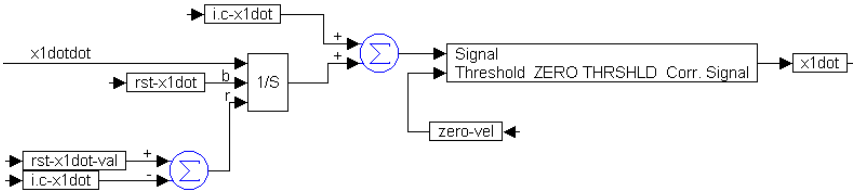

Figure 13. Integration of v_1^{dot} to determine v_1

Setting Initial Conditions Externally for Integrators

To specify an initial condition for an integrator externally, the `summingJunction` block can be used. The output of the integrator, represented in VisSim by `1/s`, is connected to one of the inputs of the `summingJunction` and the desired initial condition is connected to the other input. For this procedure to work, the internal initial condition for this integrator must be set to zero. (All VisSim integrators have a default internal initial condition of zero.)

In Figure 13, it is seen that the output of the `resetIntegrator` block, and the initial condition `i.c-x1dot` are connected to the `summingJunction` block. This assigns the value of the variable `i.c-x1dot` as the initial value for the output of the integrator, which in this case is the block velocity v_1, which in turn is obtained by integrating v_1^{dot}. The output of the `summingJunction` block then passes through the zero thresholding process shown in Figures 10 and 11, and the output of the thresholding process is set to be the actual value of v_1.

Now comes another tricky part – the actual reset value. When the Boolean `rst-x1dot` is "true," v_1 must be reset to `rst-x1dot-val` as computed in Figure 9. However, since the initial condition is being set externally as described above, an equivalent amount must be deducted from the reset value, so that the correct value of v_1 is observed at the output of the `summingJunction` block. Consequently, the difference (`rst-x1dot-val - i.c-x1dot`) is provided as the reset value to the `resetIntegrator`. A similar sequence of operations is performed to obtain v_2 by integrating v_2^{dot}.

Integrating Velocities to Obtain Positions

The ideas presented in the preceding section are carried a little further, by integrating the velocities obtained above to compute positions. Since each block is subjected to external, physical travel stops that constrain its motion, `limitedIntegrator` blocks are used in VisSim to perform this state of numerical integration, as shown in Figure 14.

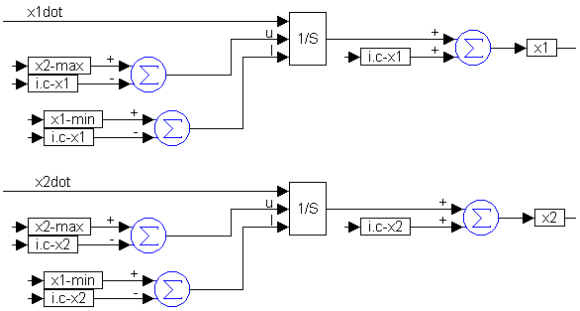

Figure 14. Integration of v_1 and v_2 to obtain x_1 and x_2

The `limitedIntegrator` accepts user-specified upper and lower limits, denoted by "u" and "l" on the connector tabs. The `limitedIntegrator` integrates the input value and limits the internal state to user-specified upper and lower limits. The initial positions of the two masses, `i.c-x1` and `i.c-x2`, are specified as shown in Figure 14, by adding them to the outputs of the appropriate integrators.

For both masses, the upper and lower travel stops are specified by the problem statement to be `x2-max` and `x1-min`, respectively. However, following the earlier discussion on specification of external initial conditions, an amount equal to the initial condition must be subtracted from the upper and lower limits to account for the externally specified initial condition. Consequently, in the integration for x_1, `i.c-x1` is subtracted from the upper and lower limits, and in the integration for x_2, `i.c-x2` is subtracted from the upper and lower limits.

Calculation of Kinetic Energy

The total kinetic energy of the system, assuming ideal conditions, is given by

$$KE = \tfrac{1}{2} m_1 v_1^2 + \tfrac{1}{2} m_2 v_2^2$$

This expression may be calculated using the diagram segment shown in Figure 15. Monitoring the value of kinetic energy is a good way to ensure that the simulation does not violate any basic principles of physics.

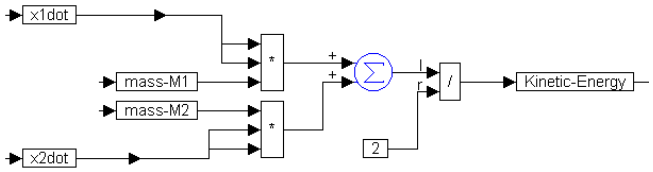

Figure 15. Calculation of total kinetic energy

Specifying Simulation Parameters

The parameter values required for simulating the dynamic system can be set directly in VisSim or accessed from an external data file. Since it is likely that the simulation will be run several different scenarios, an `import` block is used in VisSim to access data from ASCII data files. The diagram segment for initializing parameters from an external file is shown in Figure 16.

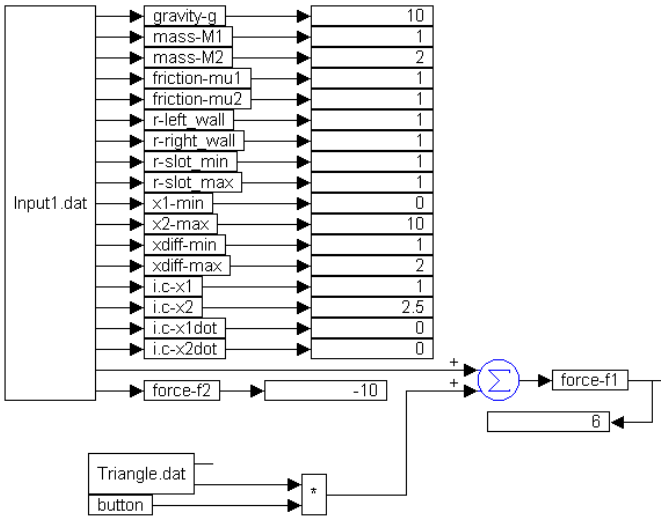

Figure 16. Initialization of simulation parameters

Recall that f_1 and f_2 are the external forces acting on m_1 and m_2. Additionally, f_1 can be specified directly from another data file, `Triangle.dat`, as shown in Figure 16. When `button` is selected, the force profile from `Triangle.dat` is added to the value from `Input1.dat`, and the resulting value is assigned to f_1. This setup becomes useful for running one of the test cases discussed later in this tutorial.

Additionally, two other simulation variables need to be specified. The variables `clock` and `zero-vel` are used in several diagram segments that were described earlier. The `clock` variable is a sequence of impulses with a time period equal to the simulation time step, and `zero-vel` is a zero threshold value for ensuring zero motion. These two variables are constructed as shown in Figure 17.

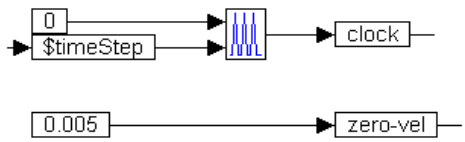

Figure 17. Initialization of simulation variables

Two external inputs can be assigned to the `pusleTrain` block. The desired amount of delay must be connected to the top input, and the bottom input specifies the time between pulses. Since no delay should be imposed, a constant value of zero is connected to the top input, and since the time between pulses must be same as the simulation step size, the system variable `$timeStep is connected` to the bottom input. Recall that `$firstPass`, `$lastPass`, `$runCount`, `$timeEnd`, `$timeStart`, and `$timeStep` are VisSim system variables that can be accessed through the VisSim variable block. The zero threshold value is set to be 0.005.

Monitoring Simulated Results

In addition to observing the dynamics visually by means of `plot` and `display blocks`, and other means, the simulation output can also be recorded in an external data file. This is set up by means of an `export` block in VisSim, which in Figure 18, is configured to save the output to `Output1.dat`.

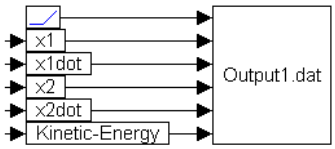

Figure 18. Saving simulation data to a data file

The `ramp` block connected to the top input tab of the `export` block outputs a value that is always equal to current time *t*. This is the easiest way to access the simulation time in VisSim. In addition we will monitor the positions x_1, x_2, velocities v_1 and v_2, and the kinetic energy of the system.

Test Cases

In the following test cases, the value of gravity is assumed to be g = 10 m/sec^2. The masses are assumed to be m_1 = 1 Kg and m_2 = 2 Kg. The left wall limit $x_{left\text{-}wall}$ is assumed to be 0.0 m and the right wall limit $x_{right\text{-}wall}$ is assumed to be 10.0 m. Six test cases are performed and the corresponding input data is summarized below. However, results from only two cases are presented.

Table 1. Simulation parameters for the six test cases

Parameter	Test No. 1	Test No. 2	Test No. 3	Test No. 4	Test No. 5	Test No. 6
m_1	1.0	1.0	1.0	1.0	1.0	1.0
m_2	2.0	2.0	2.0	2.0	2.0	2.0
μ_1, μ_2	1.0	1.0	0.0	1.0	0.0	1.0
$r_{left\text{-}wall}$	1.0	1.0	1.0	1.0	0.5	1.0
$r_{right\text{-}wall}$	1.0	1.0	1.0	1.0	0.5	1.0
$r_{slot\text{-}min}$	1.0	1.0	1.0	0.0	1.0	1.0
$r_{slot\text{-}max}$	1.0	1.0	1.0	1.0	1.0	1.0
$x_{left\text{-}wall}$	0.0	0.0	0.0	0.0	0.0	0.0
$x_{right\text{-}wall}$	10.0	10.0	10.0	10.0	10.0	10.0
$x_{diff\text{-}min}$	1.0	1.0	1.0	0.0	0.0	0.0
$x_{diff\text{-}max}$	2.0	2.0	2.0	2.0	50.0	2.0
x1(0)	1.0	1.0	1.0	1.0	1.0	1.0
x2(0)	2.5	2.5	2.5	2.8	9.0	2.0
v1(0)	0.0	1.0	-9.0	10.0	0.0	0.0

Table 1(cont.). Simulation parameters for the six test cases

Parameter	Test No. 1	Test No. 2	Test No. 3	Test No. 4	Test No. 5	Test No. 6
v2(0)	0.0	1.0	1.0	0.0	0.0	0.0
f1	6.0	0.0	0.0	0.0	-2.0	triangular profile
f2	-10.0	0.0	0.0	0.0	4.0	0.0

Test Case 3: Conservation of Energy

The input data for this case is available in `Input3.dat`. The simulation duration in this case is 6.0 sec with a simulation step size of 0.001 sec. It is assumed that there are no frictional forces acting on the two masses. Consequently the total kinetic energy of the system must remain constant. Several collisions are observed, of all possible types: `left-wall`, `right-wall`, `slot-min`, and `slot-max`. The coefficients of restitution are all assumed to be unity, such that no energy is lost due to collisions.

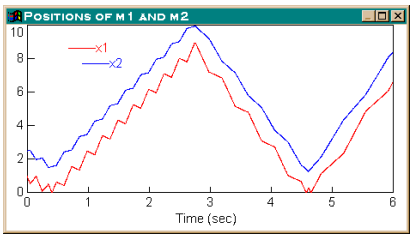

Figure 19. Case 3: Mass positions

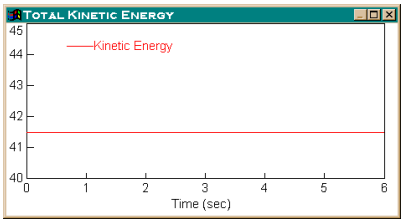

Figure 20. Case 3: Kinetic energy

The results obtained during this test are shown in Figures 19 and 20. The mass positions show a series of `slot-min` and `slot-max` collisions, along with a right wall collision around 2.73 sec and a left wall collision around 4.6 sec. During this entire series of collisions, the total kinetic energy remains constant at 41.5 *J*.

Test Case 5: Wall Collisions with Rattling Stop

The input data for this case is available in `Input5.dat`. The duration of the simulation is 3.2 sec and the simulation step size is 0.001 sec. An external force of -2 *N* is applied to m_1 such that it moves towards the left wall, and a force of 4 *N* is applied to m_2 such that it moves towards the right wall. Under the constant external forces, both m_1 and m_2 experience a series of collisions with the left and right walls respectively. Each collision is associated with energy absorption. The results obtained are shown in Figures 25 to 27.

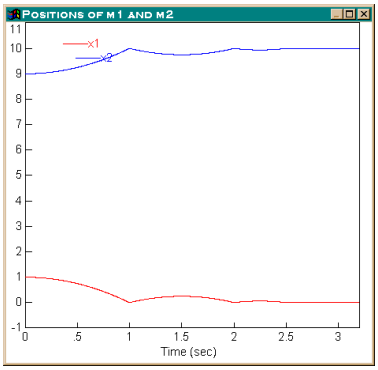

Figure 21. Case 5: Mass positions

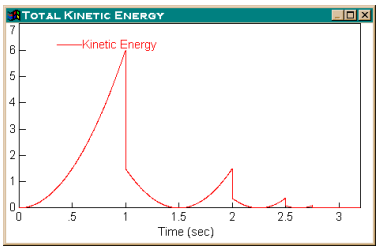

Figure 22. Case 5: Kinetic energy

Mass m_2 starts from an initial position of x_2 = 9.0 m and is forced into the right wall by the external force f_2 = 4.0 *N*. m_2 collides with the right wall at t = 1.0 sec and bounces back with some energy absorption. The external force causes m_2 to repeat the collision with decreasing time between collisions. The mass finally rattles to a stop at t = 3.0 sec.

Mass m_1 starts from an initial position of x_1 = 1.0 m, and is forced into the left wall by the external force f_1 = -2 *N*. Mass m_1 collides with the left wall at t = 1.0 sec and bounces back, absorbing some energy. Mass m_1 repeats collisions, with decreasing time between collisions, and finally rattles to a stop at t = 3.0 sec. Slot collisions are not observed in this case.

The kinetic energy of the system, shown in Figure 22, is initially zero as both masses are at rest. As the velocities of the two masses increase, the kinetic energy also increases, to about 6 *J* at t = 1.0 sec. At this instant, both masses collide with the walls as indicated by a sudden drop in the kinetic energy. At this point, the velocities change direction (and sign). Due to the external forces, the velocities start to build up again.

Depending on the case, one or both velocities may pass through zero and become positive. For the first collision, this phase of rebuilding system energy occurs during the time period from t = 1.0 sec to t = 2.0 sec. The energy at this point is about 1.45 *J*, and another set of collisions occur at

this instant. The whole process repeats itself from this point on, with decreasing times between successive collisions. More importantly, the magnitude to which the kinetic energy builds up between collisions, decreases from collision to collision. Finally, at t = 3.0 sec., kinetic energy reaches zero, and remains zero for the rest of the simulation.

Figure 23 indicates the times at which m_1 and m_2 collide with the left and right walls respectively. The first collision occurs at t = 1.0 sec, the second at t = 2.0 sec, the third at t = 2.5 sec., and so on. As time progresses, the time interval between successive collisions decreases, causing a rattling effect. As can be seen from the figure, starting at t = 3.0 sec, the collision indicator stays high for the remainder of the simulation, as both blocks rattle and stop against the two walls. The constant high value of the collision indicator represents constant contact between the concerned objects.

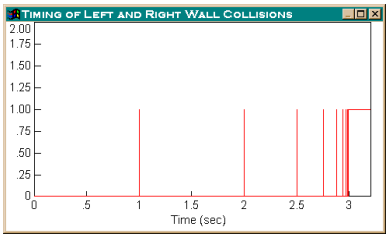

Figure 23. Case 5: Occurrence of left and right wall collisions

Concluding Remarks

In this tutorial, we have shown the steps involved in formulating a mathematical model of a dynamic system; developing a simulation in VisSim—the block diagram based nonlinear modeling and simulation software; running several test cases from the same VisSim diagram; analyzing and visually interpreting the results.