Comp3252 Modelling and Simulation. Worksheet Verification & Validation

CBPrice January 2021

Color Coding for this Worksheet	
	General Information
	Information you will need for the activities
	From Math Model to Code
	Guided Activities
	Investigations

Purpose

(a) To understand how to *verify* and to *validate* a coded mode. **This is important work since it hits one of the ILOs required for your first assignment.**

Console Commands for the Oscillator Model

The following console commands are available for this simulation. Here "X" refers to a number which you will be asked to input. Remember to include spaces as indicated below.

Console Command	Effect
setparam stiffness X	sets the stiffness of the suspension springs.
setparam mass X	sets the mass of the monster truck.
setparam damping X	sets the damping coefficient of the shock absorbers
setparam Euler	changes the solver algorithm to **"Euler"**
setparam EulerCromer	changes the solver algorithm to **"Euler Cromer"**
setparam deltaT X	sets the time interval for the solver algorithm

Activities

1 VERIFICATION of the Simulation Code by Simple Observation

Here we are asking the question "Are we coding the model right?" As explained in class, the simulation should produce a "sine curve" where the amplitude remains constant over time, when there is no damping. In this activity you will be asked to use two different computational approaches (the "solver"). One produces correct data (passes the *verification* test) and the other does not. You must decide!

Find the monster truck – the one in the garage.

(a) Run your level and check that the **solver** is **SOLVER_EULER_CROMER** before you start the simulation

run. If it's not, then use a console command (blue section) to set it. Also check that **deltaT** is **0.01** and that **damping** is **0.0,** and if it's not use a console command to set it.

(b) Start the simulation and observe the oscillations for 10 cycles, this should take about 20 seconds. Does the amplitude appear to remain constant when you observe the oscillator (eyeball)? If not, then what happens to it? Press **F1** to stop and **F2** to reset.

(c) Bring up the console and set the solver to **EULER** (**setparam EULER**) and repeat (b). Again, does the amplitude appear to remain constant. If not then what happens to it?

(d) So which solver is a likely candidate for **verification** and which solver is not?

2 VERIFICATION of the Simulation Code by Measurement (EULER Solver)

(a) In the Editor double left-click on the monster truck and open up the "mas14" tab. Set the log Interval to 0.01. Now run the level

(b) Set the solver to **EULER** and make sure that **deltaT** is **0.01** and that **damping** is **0.0.** Run the simulation for about 20 cycles of oscillation and then exit the level.

(c) Set up Octave as usual and run the script **PlotMonsterTruck.** The top plot shows theory (blue) and simulation (red circles). You should see that the *envelope* of the graph is increasing. How does it increase? Is it a straight line (steadily increasing) or is it exponential (exploding)? If it is exploding then this is really bad. Any form of increase is bad, since we are not pumping any energy into the system.

(d) Let's compare the simulation values with the theory values. Choose a peak near the end of the plot and note down the simulation and theory values. Let's call these *dispZ* and *dispZTheory*.

Now use these values like this to calculate the "percentage error" in the computation over this time interval.

$$100 * \frac{(dispZ - dispZTheory)}{dispZTheory}$$

(e) This error should be less than 10% if the code is to be verified.

3 VERIFICATION of the Simulation Code by Exact Measurement (EULER_CROMER Solver)

Repeat 2(a) – (e) for the **EULER_CROMER** solver. Then state which solver is verified and support this with your calculations from activities 2 and 3.

4 VALIDATION of the Simulation code by comparison with real experimental data

Here we are asking the question "Are we coding the right model?" Here we are going to look at a second Monster Truck where the suspension has been modelled mathematically for a Land Rover Defender. We shall compare the behaviour of the suspension when it is loaded with experimental data.

For your interest, the code for the suspension **forceZ** expressed in terms of the spring displacement **dispZ** looks like this. No coding is required.

forceZ = k*A/(A*(x0 + dispZ))1.3 - 5500 + load - damping*velyZ;**

Here, **k, A, x0** are all constant coefficients. But the dependency of force on displacement (the first term on the right hand side is nasty and nonlinear.

Here's the experimental data (Personal communication JLR):

load (Newtons)	extension (dispZ) (metres)
2500	0.2937
1250	0.1048
-2500	-0.1320
-5000	-0.2034
-7500	-0.2489
-10,000	-0.2828
-12,500	-0.3075

Find the model MAS14_MonsterTruck_AirSuspension. It should be about here, just outside the garage

(a) Fire up the simulation. Use the console command **setParam load X** where X are values from the above blue table. For each load make a note of the final **dispZ**

(b) Copy and paste the experimental data in the blue table above. Add two columns to the right. In the first extra column add your simulated values for **dispZ.** In the second extra column calculate the percentage error like this

$$100 * \frac{(dispZSimulation - dispZExperiment)}{dispZExperiment}$$

(c) Are all the percentage errors less than 10%? Only then is the model *validated* which means that it is the correct model.