Simultaneous events, singularity in the space of models and chicken

Abstract

The discrete event simulation with simultaneous events is discussed. Discrete models are treated as some points in the general space of models. The main topic of the paper is the convergence of sequences of continuous models to the corresponding discrete event model. It is pointed out that the discrete event models may represent some singularities in the space of models. This fact is related to the problem of validity of discrete event simulation. The models presented here are not the topic of the paper. These are known models, used as examples. The model of the "chicken game" and multiple elastic collisions are considered. It is shown that if a discrete event model is some special limit case of a sequence of continuous models with finite event duration, then that limit model is a singularity. It may provide results very different from the expected limit value of the results from the converging sequence of models.

Keywords: simulation, discrete events, modeling

Introduction

According to the Wolfram Math World, a *singularity* is a point at which an equation, surface, or other mathematical object, blows up or becomes degenerate. Singularities are often also called singular points. At the singular point, a given mathematical object may not be defined, or it fails to be well-behaved in some particular way, such as differentiability.

Now, consider a *set of all models* or a set of all models of certain type. To define a *space of models*, we must define a structure in the set. In Raczynski (1998) a metric structure is defined in the set of models. This way we obtain a metric space of models with the corresponding induced topology. Recall that this is just a metric space. No operation like addition or multiplication of models is defined. In that definition, the model is treated like a black box that produces certain outcome. The outcome may be represented by a single number, a set of numbers or a set of points given by the graphs[1] of plots produced by the model. These may be, for example, deterministic time-plots or plots of the probability distributions produced by the model. The distance between models is defined as the Hausdorff distance (see Marošević 2018) between sets of the model output graphs. This permits to calculate the distance between two models, to define convergent sequences of models and to handle the mappings from model parameter space to the model space. The continuity of such mappings can be investigated. This may be useful while selecting a simplified model specification, deciding if a model component can be removed or if the model structure can be simplified. Note that the Hausdorff distance defines the *strong* metric. However, we define it for the outcome sets. This means that the fact that the distance between two outcome sets is equal to zero, does not mean that the corresponding models are equal to each other. In other words, our distance is a *pseudometric* in the model space, because it may not satisfy the *identity of indiscernibles* condition necessary for a strong metric (see Lawvere 2002).

In discrete event simulation methodology, the most important contribution is the Discrete Event System Specification (DEVS), see Chow (1996) and Zeigler (1987). DEVS is a modular and hierarchical formalism for modeling and analyzing general systems. This formalism helps to create big discrete event models that run fast and support the use of reusable modules. Many works on DEVS are dedicated to possible problems that may occur when in the model there are simultaneous events. Indeed, if we consider events that occur in certain time instant with event duration equal to zero, we must admit simultaneous events. If two or more events have to be executed in the same (model) time instant, then the implementing hardware executes them in some, implementation-dependent, order.

[1] The graph of a function $f(x)$ is the set of all pairs of points $(x,f(x))$ over a given interval for x.

Thus, the simulation result may be implementation-dependent. There are several algorithms that cam be applied to avoid such dependence. In the model definition of DEVS, there is a *select* element that is responsible for simultaneous event handling, consult Saadawi (2010).

While coupling small "atomic" models, the *select* element must be added. *Select* resolves possible conflicts and ambiguity in simultaneous event execution, see Saadawi and Wainer (2010). Observe, however, that using select, we add to the model a new, artificial component that has no corresponding component in the modeled real system. This issue has been discussed in Raczynski (2017).

An alternative approach to model building is the use of *semi-discrete* events. In such kind of models, we permit events that have certain duration (in the model time). This is a methodological, rather than practical consideration, because the conventional discrete-event simulation will always be the fastest way to simulate great models. However, we also must be aware of the model validity. In this paper we discuss another example of semi-discrete and discrete-event model to show that the conventional discrete-event model may represent a singularity in the model space. This may cause problems with model validity questions. The semi-discrete events are supposed to have short duration D, but it is not a necessary property. Let us consider a model with semi-discrete events that permit the event duration to be changed. Now, consider a sequence of models with D approaching zero. In the limit, we obtain a conventional discrete event model. Using the distance between models mentioned before, we can construct a sequence of models and examine the convergence property. Unfortunately, there are cases when such sequence whit $D \rightarrow 0$ does not con converge to the limit model with D=0. In other words, D=0 is the singularity in the model space. In Raczynski (2017) some examples of this phenomenon are given. The very simple example is the model with one server without buffer. If a client appears it tries to occupy the server. If the server is free, the client occupies it, otherwise he goes away. If these two events are simultaneous, the simulation results may be wrong. However, such examples may appear to be too artificial. Many simulation packages use an implicit buffers before servers, to avoid such problems. Here, we present another example to show the singularity of discrete event models.

The example shown below, as well as examples discussed before may appear to be very simple. Note, however, that what are looking for are counter-examples. The simplest counter-example is always the best counter-example.

The chicken game

In the game called *Chicken*, two people, say Jim and Fred, drive heir cars towards each other from opposite ends of a road. If one of them swerves before the other, he is called a chicken and the other wins. Of course, if neither swerves, they will crash. So, the outcomes may be: 1. Fred wins, 2. Jim wins, 3. The cars crash.

In the model used here, we do not pretend to exactly simulate what appear in the real game. This is just a simple example used to investigate the convergence of a sequence of models in the model sapce.

The model rules are the following.

Car attributes:

Position *(x,y),* where x is the distance run, y is the deviation from the initial straight line.

Velocityies v_x and v_y, in the direction x and y, respectively.

Fear factor f_g. This is the fear growth rate. At certain fear threshold f_t, the player starts his decision event.

D - the duration of the "decision event", that is, the event of decision making

The initial positions for Fred and Jim are (-1,0) and (1,0), respectively. Initial velocity of Fred is positive, the velocity of Jim is negative (they are going towards each other).

After the game starts, the fear level for the two players begins to grow, and their distance decreases. For each player, if his fear reaches the threshold level, he triggers his decision-making event. During this event, the driver observes the movement of the competitor. If the other player quits (his y coordinate becomes different from zero), the driver follows with the straight movement. If the other player did not quit, the driver may take the decision to turn out. A "human factor" is taken into account. This means that the driver needs some (may be very small) time interval to decide if he quits or not..

The decision is taken at some time instant inside the decision making event. In addition, due to the human factor, the decision may be wrong or not be taken at all. This means that the driver may quit even if the other did, or he can follow the straight movement when the other does the same. Figure 1 shows the increasing fear and the decision making event according to the car movement. The end point is where the possible crash may occur.

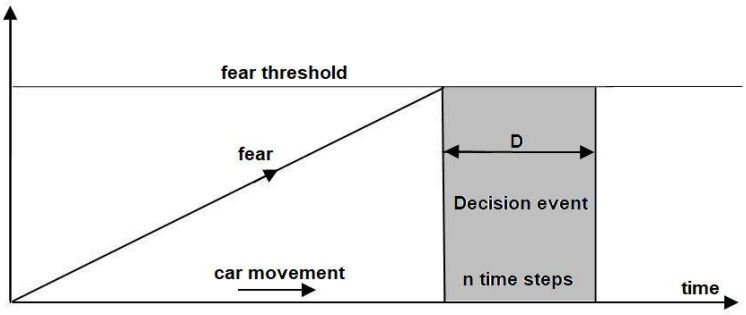

Figure 1. Car movement, increasing fear and the decisions event.

The decision event is an approximation of a continuous process. However, in our (recent) computers nothing is continuous. Thus, the event is executed in some discrete time steps, according to the following algorithm.

In each time-step, the player does the following:

1. If the other player quits, the player goes ahead. If the player velocity v_y is positive, it can be updated to be zero and the car follows the straight line. This occurs with probability equal to 0.1 (in each step).

2. If the other player did not quit, the actual player quits with probability equal to $1/n$ at each time-step (n is the number of time steps).

Note that the last decision has the probability 1/n in each step, which means that the decision can be taken at some moment inside the decision event. The total probability of taking the decision is approximately equal to one. As for the point 1, there is some uncertainty about the decision. It may result to be mistaken, due to the human error.

If the two players have different fear factors or fear thresholds, the player that first enters the decision event will quit with high probability. This probability decreases if the decision events overlap. Here, we focus on the case when the two players have identical parameters. This means that the decision events start and terminate in the same time instants. With such, strictly overlapping events, the final outcome is uncertain.

Simulation and model convergence

Assume the following model parameters (equal for the two players):
Absolute initial velocity v_x= 0.1, v_y=0
Fear grow rate f_g=0.1
Fear threshold f_t=0.6
Decision event duration D=0.5
Number of steps in decision event n=50

The simulation of the car movement is rather trivial and will not be discussed. What is relevant here is the decision event execution and final outcome. The simulation with the above parameters provides the following results:

Crash probability: 0.150
Fred wins: 0.424
Jim wins: 0.407
Crash due to false turn decision: 0.02

These are results obtained for the average the outcome from 5000 repetiitions of the simulation run.

Observe that looking at the rules of the game (previous section), we can see that the outcome does not depend on the decision event duration *D*. Indeed, running the same experiment with different values of *D*, we obtain the same results (up to small random fluctuations of the statistics). Now, suppose that *D*=0. Taking this value, the model operations do not change. Simply all *n* steps of the decision event are executed for the same model time. The time advance step inside the event becomes equal to zero, but there are still n steps, so the algorithm cannot stop in a "zero-advance" loop.

Consider a sequence of models with *D* approaching zero. As mentioned before, the "limit model" with D=0 is the discrete event model with simultaneous events. It might appear that the final outcome for the limit case D=0 should be the same. However, this is not the case. With *D*=0 we obtain the following.

Crash probability: 0.341
Fred wins: 0.231
Jim wins: 0.220
Crash due to false turn decision: 0.197

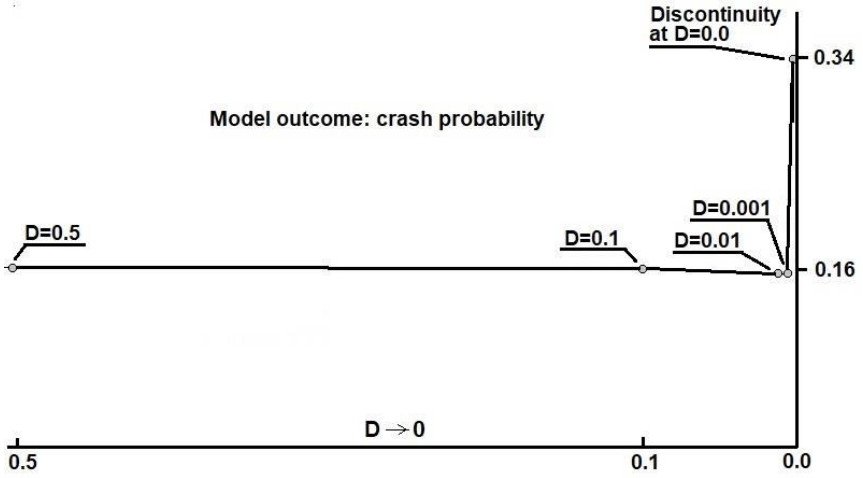

Figure 2. Model outcome with D approaching zero.

The other results also reveal discontinuity at D=0, as shown in the following table.

Interval D	Crash %	Fred wins %	Jim wins %	Crash%(bad turn)
1.0	15.0	42.4	40.7	2.0
0.5	16.2	41.0	40.9	1.9
0.1	16.1	42.3	40.0	2.0
0.01	15.5	40.9	41.5	1.8
0.001	15.8	40.4	42.5	2.0
0	34.1	23.1	22.0	19.7

Table 1. Simulation results with different values of *D*. 5000 repetitions of the simulation run..

Figure 2, as well as table 1, show the simulation results with *D* tending to zero, including the limit point *D*=0. It can be clearly seen that the last point (discrete event model) represents a singularity in the model space.

Three body collision

Two models are considered: The bodies with compliance and the ideal rigid bodies.

Using *compliance collision* we assume that the body deforms during the contact with other body as if there were a small spring between them. The "spring" accumulates the potential energy and then devolves it through a force that make the bodies reboot. The total energy and the momentum of the whole system are preserved. Simulating this kind of collision we need a great number of steps in a continuous simulation to integrate the movement that occurs during the contact. This makes the simulation considerable slow and may result in stiff model equations.

In the *ideal rigid collision* of two bodies, known formulae for the resulting velocities can be used. This kind of simulation is fast and exact. A more difficult case is a simultaneous collision of three or more bodies. In this case a *serial collision* handling method can be used, also known by the terms *propagating impulses* or *sequential collision* handling. Using this method, we imagine that each of the multiple collisions happens one at a time in sequence; we resolve each collision as a single collision, and then based on the resulting velocities there are subsequent collisions to resolve. All this happens instantaneously. Here, we will discuss the simultaneous collision of three identical balls, two-dimensional case.

Compliance collision

In this model, when the bodies collide, a small deformation u is assumed. Due to elasticity of the material, a force is produced repelling the bodies from each other. The force is supposed to be equal to $F = u/c$, where c is the compliance (cm/N).

A good example of the application of such kind of simulation (Complance coliision, CC) is the problem of breaking the rack in billiards. This probem, treated as the ideal elastic collision of multiple bodies is considered as numerically intractable (see Traub and Wozniakowski,1994). However, if we treat the balls as real elastic bodies and use the CC model, there is no problem with the modeling and simulation.

Figure 3 shows the collision scenario. Three balls (circles) of radius r move toward the collision positions, marked with small circles (A). They collide simultaneously and their velocities change. The part B of the figure shows the situation after collision. The parameters are as follows.

Mass = 1, r = 0.04, initial velocity =0.5, for all three bodies.

If the distance d between the centers of two balls become less then $2r$, then the value $u = 2r-d$ is used to calculate the repelling force $F=u/c$, where c is the compliance. The bodies satisfy the Newton's movement equations. With very small c some integration problems occur, but cam be solved by decreasing the integration step. In our case, the simulations have been run with varying from 0.1 to 0.0001 without problems. All these simulations provide the same result, as shown on fig.3 B. It should be noted that for these, as well as for the simulations of ideal elastic collision discussed in the following section, the preservation of total energy and momentum i checked at each simulation step.

Now, suppose that c reaches the limit value of zero. Whit c=0 we cannot use this model. With no compliance, we have ideal elastic collision, and instead of integrating the model equations we can use the known formulae for elastic collision, as in the following section.

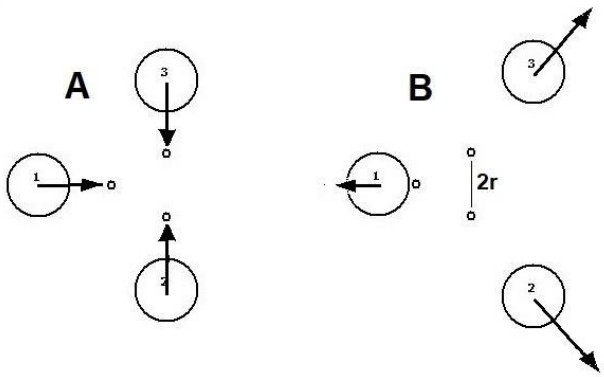

Figure 3. Ball movement before (A) and after collision (B).

Elastic collision: zero compliance

In this case, each collision is a discrete event and the model becomes of discrete event type. There are, executed simultaneously, namely the collision between bodies 1 - 2, 2 - 3, and 3 - 1.

The velocities of collision between body *k* and *j* change according to the following formula:

$$w_k = v_k - \frac{2m_j}{m_k + m_j} \frac{\langle v_k - v_j, x_k - x_j \rangle}{\|x_k - x_j\|^2} (x_k - x_j)$$
$$w_j = v_j - \frac{2m_k}{m_k + m_j} \frac{\langle v_j - v_k, x_j - x_k \rangle}{\|x_k - x_j\|^2} (x_j - x_k), \quad (1)$$

where $v_{k,j}$ are the velocities before collision and $w_{,k,j}$ are the velocities after the collision. Angle brackets indicate the inner (or dot) product of two vectors. In our case the mass $m_1=m_2=m_3=1$.

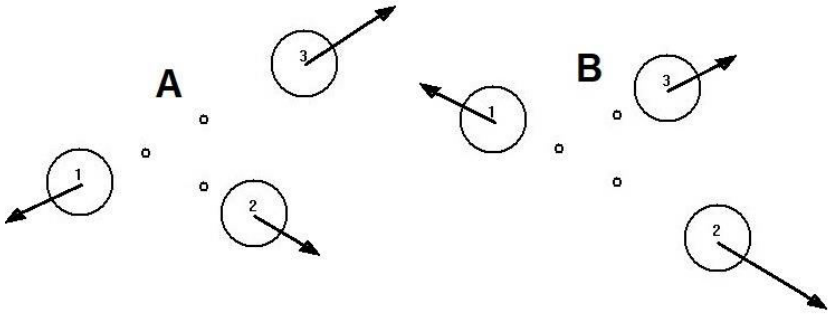

Figure 4. The velocities resulting from the triple ideal elastic body collision.

Figure 4 shows the results of two runs of the simulation, with identical parameters. Only the order of the execution of the simultaneous evens change. The result is completely different for each simulation, and it never coincides with the results of the continuous simulation with compliance.

Again, considering any sequence of continuous CC models with compliance approaching zero, we see that the sequence converges to the (correct) result shown in figure 3 B. However, the case c=0 provides completely different result. This means that the corresponding discrete event model represents a singularity (discontinuity) in the model space. Moreover, the result of the discrete simulation with simultaneous events is implementation-dependent because the result depends on the

order of event execution determined by the computer or by the particular simulation software. The DEVS *select* model component cannot resolve this.

Conclusion

The main topic of this paper is the convergence of models in the model space, and not the construction of particular model. Once a distance between models is defined, we can construct sequences of models and see if they converge. The point is that for some sequences of continuous models that approach the limit discrete event model, the limit point (the discrete model) represents a singularity in the model space. This occurs when the results of the sequence models approach certain point, but the results of the limit model are very different. Such singularity of discrete event models may cause some doubts about the validity of discrete event models, in general. Recall that if we admit discrete events, we must admit simultaneous events, as well. This may be the cause of singularities, as shown on the presented simple counterexamples.

References

Chow A.C. (1996) A parallel, hierarchical, modular modeling formalism and its distributed simulator. Transaction of the Society for Computer Simulation, The Society for Modeling and Simulation, 13(2):55-67.

Lawvere F.W. (1973) Metric spaces, generalised logic, and closed categories. Reprints in Theory and Applications of Categories, 1:1-37.

Marosevic T. (2018) The Hausdorff distance between some sets of points. MATHEMATICAL COMMUNICATIONS, 23:247-257.

Raczynski S. (2017) Semi-discrete events with fuzzy logic. Journal of Simulation, Palgrave MacMillan - Springer, 12(3):248-257, DOI: https://doi.org/10.1057/s41273-017-0050-4 , ISBN/ISSN 1747-777

Raczynski S. (1998) On the metric structure in the space of dynamic system models. Transactions of the Society for Computer Simulation International, Society for Computer Simulation, 15(2):70-75.

Saadawi H., Wainer G.A. (2010) Rational Time-Advance DEVS (RTA-DEVS). Conference paper: Spring Simulation Conference (SpringSim10), DEVS Symposium, Society for Computer Simulation (SCS).

Traub J.F, Wozniakowski H. (1994) Breaking Intractability. Scientific American, 270(1):102-107, URL: http://www.jstor.org/stable/24942555..

Zeigler B.P. (1987) Hierarchical, modular discrete-event modelling in an object-oriented environment. Simulation, SCS, 49(5):219-230.