There are several points of interest for discussion. The results show clearly the importance of the flow at a small region near the apex which forms the core of the vortex. The flow in this region is highly three-dimensional in nature and strongly influenced by the axial flow in most situations and thus is not accurately simulated by the impulsive flow analogy or by the conical flow assumption. By reducing the interaction or entrainment between the flows on the two sides, the flow asymmetry caused by the perturbation imposed at the apex is not being amplified farther downstream. In other words, the amplification of the asymmetry seems to take place mostly near the apex. While Moskovitz et al.[3] suggested that the effect of blunting the nose was mainly a reduction of the model asymmetry, because a blunted nose is easier to machine, it would seem that the effect may also be the result of the modification of the flow near the apex region. That is, blunting the nose has a similar effect of reducing the flow interaction as the strake.

References

[1]Ericsson, L. E., and Reding, J. P., "Aerodynamic Effects of Asymmetric Vortex Shedding from Slender Bodies," AIAA Paper 85-1797, Aug. 1985.

[2]Degani, D., and Schiff, L., "Numerical Simulation of the Effect of Spatial Disturbances on Vortex Asymmetry," AIAA Paper 89-0340, Jan. 1989.

[3]Moskovitz, C. A., Hall, R. M., and DeJarnette, F. R., "Effects of Nose Bluntness, Roughness and Surface Perturbations on the Asymmetric Flow Past Slender Bodies at Large Angles of Attack," AIAA Paper 89-2236, Aug. 1989.

Synchronization and Time Tagging in Distributed Real Time Simulation

Amnon Katz*, Daniel M. Allen†, and
Joseph S. Dickson‡
McDonnell Douglas Helicopter Company,
Mesa, Arizona

I. Introduction

BY definition, real time simulation runs as fast as the real world process it simulates. With proper calibration, this holds true on the average. However, "in the small" the digital processes proceed by leaps and bounds and deviate considerably from the uniform flow of time. Time evolution is computed in steps. Typically, time is divided into frames. All of the processes that occur in parallel during the frame are computed during the frame, in sequence. Some of the frame time is used for computer housekeeping and communications. Some usually is surplus and remains idle.

The frame becomes the natural unit of time for simulation. It may be neither possible nor useful to measure an increment of time finer than a frame.

Received Oct. 4, 1989; revision received Feb. 25, 1990.

*Manager, Advanced Development Office, Combat Simulation and System Integration. Member AIAA.

†Advanced Development Office, Combat Simulation and System Integration.

‡System Integration Engineer, Combat Simulation and System Integration.

The situation is more complex when multiple processors, computing in parallel, are involved. Different machines may run different cycles. Some may be asynchronous, and some may adjust their step time to the workload. Even when all frames are of equal duration, they are not directly comparable unless steps are taken to synchronize them. And even then, the correct time tag of any information may not be clear. Variables computed by processor X and used by processor Y may have been transmitted several times through a bus or local area network. The values used or recorded could be several frames old.

See Refs. 1 and 2 for a review and further references on the effects of temporal fidelity on simulator effectiveness.

As more and more demand is made on simulation in the fields of training and testing of airmen and design and evaluation of airframes and avionics, the need to quantify the meaning of time tagging becomes more crucial. This is required in the assessment of simulation fidelity and for the correct interpretation of information gathered from simulation studies.

This paper outlines a system of timing for distributed real time simulation. Both hardware and software are involved. Three different times are invoked: time of day, mission time, and dynamic time. All three are relative times referred to an arbitrary starting point. All three are reckoned in ticks. Time of day is a continuously running count. Mission time skips periods when the simulation is frozen. Dynamic time addresses the distinction between the time a computation is made and the time at which the result is valid.

Between them, the three time definitions provide the analyst with the tools to decide what takes place when, in simulation and in the real world process being simulated.

II. Time of Day

Time of day (TOD) is defined by a common timing signal distributed to all processor boards. This signal may be the basic simulation frame count or it may be a faster count whose frequency is an integral multiple of the frame count. In this case, a new frame is started every nth count. Individual counts are available for timing finer than a frame.

Mission time is maintained as a tick count in a register or memory location that is available to all processes, real time as well as control functions, to read. In a distributed system, each processor or chassis maintains its own TOD count. This permits independent operation. It also ensures that time can be determined locally without undue delay and without an undue burden on intercomponent communications.

In the McDonnell Douglas Helicopter Company (MDHC) facility, TOD is a 59.94 Hz signal derived from the image generation system and defining the basic frame of the out-the-window visual. The signal is distributed by coaxial cable to the many processors. Typically, these are 68020 processor boards in VME chassis. The timing signal is wired into an unused line in the backplane, where every processor board picks it up. Each processor then maintains its own TOD count in an onboard memory location. The processor can access its own time count over a local bus. At the same time, the location is also available to external processes over the VME bus. In particular, the system control station (SCS) has access over Ethernet to each VME bus and can read and write each processor's memory.

III. Synchronization

The distribution by wire assures that the ticks counted by the different processors are synchronized to within nanoseconds (representing the differences in wire length). However, at this point, the TOD count maintained by different processors is different because the processors are independent and start operation asynchronously. Synchronizing the TOD counts of the different processors is a task to be accomplished as part of the global system initialization by the SCS. The process consists of SCS reading its own TOD and writing the same

Table 1 Synchronization by SCS

SCS TOD	SCS Action
117	Set TOD for processor 1 to 117
118	Verify that TOD for processor 1 is 118
119	Set TOD for processor 2 to 119
120	Verify that TOD for processor 2 is 120
121	Set TOD for processor 3 to 121
122	Verify that TOD for processor 3 is 122

Table 2 Synchronized transition from FREEZE to RUN

		RUN flags processors			
SCS TOD	SCS Action	1	2	3	...
117	Instruct Processor 1 to RUN at 122	F	F	F	...
118	Instruct Processor 2 to RUN at 122	F	F	F	...
119	Instruct Processor 3 to RUN at 122	F	F	F	...
.		.	.	.	.
.		.	.	.	.
121		F	F	F	...
122		T	T	T	...

value into the TOD register of another processor. Subsequently, the SCS reads the TOD of the other processor to verify that a correct setting has been achieved.

The SCS must achieve the reading of its own TOD and the writing to the remote processor within a single tick period. Similarly, the verification must occur within a single tick period. Note, however, that the verification need not take place in the same period as the writing. Note also that TOD for different processors may be set in different periods. The process may proceed as in Table 1.

The need of achieving each of the actions in Table 1 within a single tick places an upper bound on the frequency of the timing signal. But this bound is not overly restrictive, 60 Hz timing presents no difficulty.

In the MDHC installation, writing and verifying the tick count in each chassis were carried out over Ethernet by means of the Stealth protocol. This is an original proprietary protocol residing above TCP/IP which permits peeking and poking into memory of remote chassis transparently to the application residing and executing there. The Stealth protocol is described elsewhere.[3,4]

IV. Mission Time

One of the advantages of simulation over training or testing in flight is the ability to freeze the action. A difficult or otherwise significant situation may be "frozen" for discussion, consideration, or study. Subsequently the simulation may be "unfrozen," and the flight continued from the point at which it was stopped.

Mission time (MT) is a count of the TOD ticks that increments while the simulation is running but remains constant when the simulation is frozen. More formally the definition is mission time is the count of TOD ticks that occur while the simulator is in the RUN mode. Ticks that occur while the simulator is in any other mode—FREEZE, INITIALIZE, SHUTDOWN,...—are ignored.

Several obvious applications of mission time are 1) timed maneuvers such as holding patterns and procedure turns need MT to complete correctly, if frozen; 2) MT represents the cumulative time that has been flown and the simulator time that should be logged; 3) Where environmental conditions, such as sun angle, or ambient temperature, are programmed to change in time, it is mission time that they should be tied to; and 4) Time histories of anything from variables of state to a complete tactical mission are most meaningful when stated in terms of MT.

V. Synchronized FREEZE and RUN

It is fairly straightforward to count ticks when the RUN flag is true and ignore them when it is false. But a method is necessary to cause multiple processors to raise and lower their RUN flag on the same tick.

At this point we assume that the TOD synchronization described in section III has been carried out. Let us further assume that all processors are frozen, i.e., their RUN flag is false. Our purpose is to cause all processors to raise their RUN flags in unison, i.e., on the same tick. It may not be possible to communicate with all processors within the same tick. On the MDHC control network, which does not support broadcast, it is definitely impossible. The way around this limitation is to lead the RUN command by several ticks. SCS does not issue an immediate RUN command. Rather, it commands all processors to unfreeze on a named future tick. All processors have the command ahead of time. Once TOD reaches the value named, all processors raise the RUN flag together. This might proceed as in Table 2. A synchronized FREEZE is achieved the same way.

It is the timing of a synchronized RUN that is the main function of TOD. All simulation data are more meaningfully tagged by MT.

VI. Dynamic Time

An essential ingredient of any real time simulation is the integration in time of equations of motion. This proceeds in steps and uses the variables of state at the end of the last step (and possibly of previous steps) together with other variables (such as control inputs) to predict the values of the variables of state at the end of the step.

Values produced in this way are valid (i.e., the approximation selected) for the precise time of the end of the integration step.

The computation normally takes place within the step and is finished before the end of the step. At that time the new values are usually put in memory to replace the earlier values. Yet these values are not, strictly speaking, valid all the time they reside in memory, nor even at the time they are first placed there. They are valid at the end of the step for which they are computed.

Even under the simplest circumstances, where a step, a frame, and a tick are all equal, an ambiguity exists. Values read from memory while MT has a certain value could be the values of the end of the last frame (if this frame's computation is not complete) or of the end of this frame (if it is).

Many situations can arise that are considerably more complicated. A frame could last several ticks. A step of a slow process may last several frames. An integration process may run asynchronously with the frames and ticks to adapt its step to the workload. Any of these circumstances increase the ambiguity. Values that are available in memory over a number of ticks are actually valid for a specific tick.

The situation is even further complicated in a distributed system with multiple processors. Variables may be transmitted from processor to processor. Each transmission may occur in a different frame. In this way, values may reside in one processor's memory long after the step in which another processor originally computed them and long past their instant of validity.

It is obviously desirable to permanently tag dynamic variables by the point in time for which they are valid. This is accomplished by dynamic time (DT), which is defined as follows: dynamic time is a dynamic variable whose time derivative is unity.

We append a variable T to any set of variables that are integrated together along with an equation of motion

$$\frac{dT}{dt} = 1$$

and an initial condition

$$T_{\text{init}} = \text{MT}$$

T becomes the dynamic time for that set of variables. The DT is kept in memory with the other variables and transmitted with them to other processors. In this way, the precise moment of validity is always available. This device automatically takes care of the complexity of long integration steps and variable integration steps.

The equations above assume that ticks are the unit of time used. If this is not so, a conversion of units is required.

The DT so defined keeps pace with MT. This is so because, by definition, integration of equations of motion is suspended in all but the RUN mode. DT thus provides a permanent tagging of dynamic variables by the appropriate value of MT.

VII. Discussion

Mission time and dynamic time are both tools for the analyst. By reading dynamic variables together with their DT and also reading MT at the same time, the analyst secures a complete picture. DT is the time at which the values obtained were valid. MT is the time at which they were available and presumably being used. Both methods of tagging convey meaningful information. The spread between them measures the delays inherent in the system.

TOD is merely a tool necessary to insure synchronized changes in the RUN flag over the distributed system. This, in turn, is required for correctly accumulating both mission time and dynamic time.

The current trend in simulation is toward parallel processing on a large scale. This compounds the difficulties of correct and consistent time tagging. Yet, as the demands on simulation increase, so does the need for precise and rigorous timekeeping. Management of time along the lines suggested here satisfies that need.

References

[1]Levison, W. H., "Model-Based Guidelines for Simulator Temporal Fidelity Requirements," *Proceedings of the AIAA Flight Simulation Technologies Conference*, AIAA, Washington, DC, Aug. 1989, pp. 68–77.

[2]Kennedy, R. S., Allgood, G. O., and Lilienthal, M. G., "Simulator Sickness on the Increase," *Proceedings of the AIAA Flight Simulation Technologies Conference*, AIAA, Washington, DC, Aug. 1989, pp. 62–67.

[3]Katz, A., Allen, D. M., and Rao, N., "Networking in a Distributed Simulation System," *Proceedings of the 11th Interservice/Industry Training Conference*, Defense Preparedness Association, Washington, DC, Nov. 1989, pp. 70–74.

[4]Allen, D. M., and Rao, N., "Real Time Data Collection and Control in a Distributed Simulator System Using Ethernet TCP/IP," Society of Automotive Engineers, New York, Paper 892356, Oct. 1989.