Submitted to Mechatronics-2000, 1st IFAC Conference on Mechatronics Systems, Darmstadt, Sept. 2000

DISTRIBUTED NETWORK ENVIRONMENT FOR VIRTUAL VEHICLE SYSTEMS SIMULATION

G. E. Smid, Ka C. Cheok, J. L. Overholt
Electrical and Systems Engineering Dept.
Oakland University, Rochester, Michigan. USA.

Abstract

Emerging technologies in modeling and simulation of dynamic systems provide transportation industry with the challenging opportunity to migrate parts of the design and development process of new vehicles to the virtual environments.

This paper presents the application of a full vehicle simulation in a distributed real-time simulation configuration. The purpose of the system is to provide for an easy-to use full-vehicle simulation platform that allows human-in-the-loop interactive driving with force and motion feedback, and for hardware-in-the-loop prototype evaluation through simulation.

The contribution that is highlighted in this paper is the networking configuration for distributed simulation. The user front-end in Matlab/Simulink provides a familiar interface where the designer/engineer is able to pull up any dynamic signal from the vehicle simulation, and is able to replace any subsystem from a remote unit as well.

Keywords: Vehicle Simulation, real-time, Human in the loop, Hardware in the loop, Virtual Reality

Figure 1: The simulation outputs are presented in real-time animation

1 Introduction

Whereas the aerospace industry has successfully adopted simulation technology for engineering and training, the automotive society is traditionally more used to conventional tools and engineering methods. It is only in recent years that the virtual simulation of full vehicle systems has become a serious effort for the automotive industry. Most of the large companies (the Big Three and their European and Japanese competitors) have developed their own facilities to simulate vehicles in virtual realistic environments. Each of them in a slightly different setting, each of them with different research objectives. Objectives could be to study the subjective performance of passenger vehicle with mathematical prototype sub-systems, or it could be for training of engineers or users, or to develop safety and driving stability systems, or it could be in the area of unmanned vehicle operations.

Commercial packages like for example ADAMS, DADS or VDANL [1] are available, as well as commercialized academic packages like CarSim and Madymo. However, generally they contain black-box simulators and are expensive. This means that a file of configuration parameters defines the simulation environment, but there is no easy access to the models. Also, these simulators will typically utilize a predefined scenario for the driver. This can be either a driving scenario (steering, throttle and barke trajectories) or a driving behavior characterization (delay, bandwidth, gain). The simulators will usually generate graphical output in the form of curves and plots on axes. Additional software needs to be aquired to vizualize the data in a 3D scenery. Hence, most of the commercial solutions will not give the user the ability to run arbitrary interactive test scenarios, and do therefore not give the subjective feel of a systems performance.

In this paper, for the simulation of vehicle dynamics, a closed-loop system will be defined that includes the driver, a virtual environment and the mathematical synthesis of an automobile. It is the principle objective of this research to develop a system that can emulate the behavior of a vehicle, using a multi-processing environment, and that can present the solution in a user-interactive immersive virtual realistic environment, which can be used as a tool in the design process.

The multi-processing environment suggests a network for *real-time* multi-CPU simulation and the development of a driver interface with a graphics front-end that can provide the simulation output to the driver.

Immersion of the human driver opens a wide area of research in the project. A human driver will feel to be immersed in the simulation when the environment *looks, feels* and *performs* realistically. In order to achieve this level of perception, the simulation required full 3D audio-visual feedback, force and motion feedback and an absolute alignment with absolute time.

The system that emerges is called the *Virtual Vehicle*

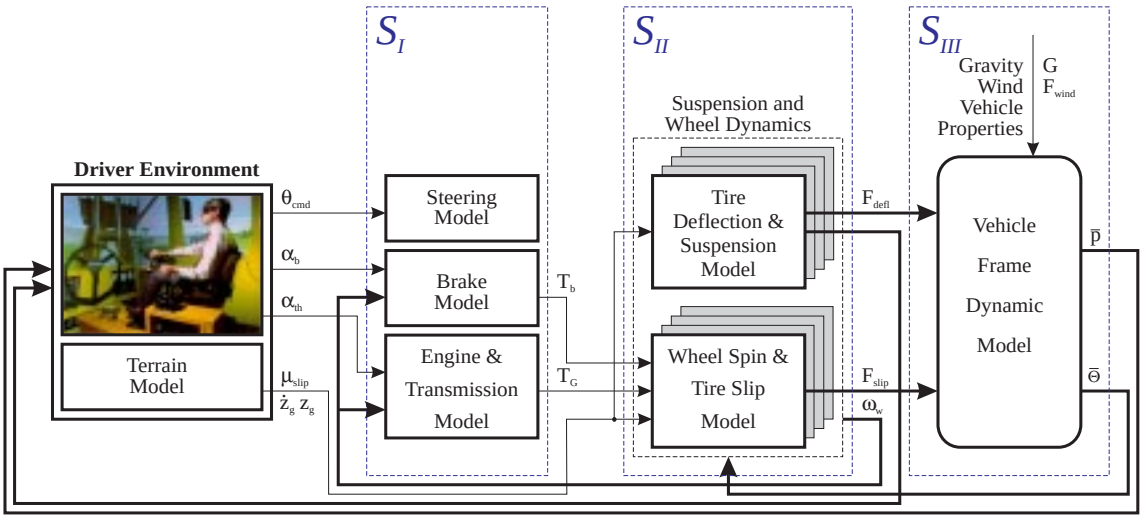


Figure 2: System

System Simulator, further referred to as VVSS. The requirements for the VVSS to be an effective and useful facility are: 1) to have a comprehensive full vehicle model with sufficent level of detail, 2) the system should be easy to use[1], a user should not need to understand the system in order to use it, 3) the real-time performance of interfative simulation at desktop level for which we need a distributed configuration, 4) a modular configuration of dynamic models provides flexible exchangability of prototype subsystems and reconfiguration, 5) and finally the system should be affordable.

2 Real time simulation

Real-time is typically referred to as the utmost strict requirement to provide a numerical output at a specified fixed time instance. Special-purpose computers with a real-time operating system are often used for the control of hardware-in-the-loop systems. For the purpose of interactive vehicle simulation, the human driver could be considered hardware-in-the-loop. The term real-time will have the similar definition. However, under-performance will not necessarily be disastrous. The term *real-time* in the context of simulating vehicle dynamics will be used in the following definition:

> The simulation of the vehicle dynamics is performed in *real-time* when the dynamics are numerically integrated, in parallel with the graphics rendering process, over a time interval that equals the graphics update frame interval, *and* the application is generating the virtual environment in the selected frame-rate.

The objective for the multi-processing environment is to allow engineers to communicate with the real-time vehicle dynamics simulation. The simulation platform with integrated network environment (hereafter referred to as SPINE) combines the convenient accessible simulation interfaces like Simulink with the virtual realistic driving simulation.

[1] The Matlab/Simulink interfaces will provide the desktop front-end

3 Vehicle Modeling

The vehicle system is considered to be constructed from a rigid frame and four wheels, hence a five-body system. The vehicle model was derived at Oakland and has been presented at CAINE [5], and has a total of 16 degrees of freedom.

The model can be partitioned in three submodels. Partitioning is beneficial for simulation, especially with the use of real-time integration methods [3].

The most suitable modeling methodology, with respect to modularity and to digital simulation, is the discrete-time state-space representation, with extension to non-linear dynamics. In the Newtonian modeling method, the forces on a body are assumed as inputs, and the dynamic motion of the body is assumed as the states or the output. When the state-space matrices are derived, they can be substituted in the full system model. This strategy is also used for the modeling of the Jumbo Container Crane [2].

The VVSS is a closed loop system that incorporates the human driver in the loop. The user inputs will change the torque and angle on the wheels. The wheels on their turn will apply inputs to the vehicle frame model. The vehicle system will then be rendered in the drivers' environment where the driver will react and generate new inputs to the simulation. This human-in-the-loop model is graphically illustrated in Figure 2.

When the drivers' actions and the terrain data are taken as inputs, and the vehicle configuration as a set of parameters, then a representation of the simulation system can be expressed in the following non-linear state-space form

$$\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u} + \bar{\eta}(\mathbf{x}, \mathbf{u}) \tag{1a}$$

$$\underline{\mathbf{y}} = C\mathbf{x} + D\mathbf{u} + \bar{\zeta}(\mathbf{x}, \mathbf{u}) \tag{1b}$$

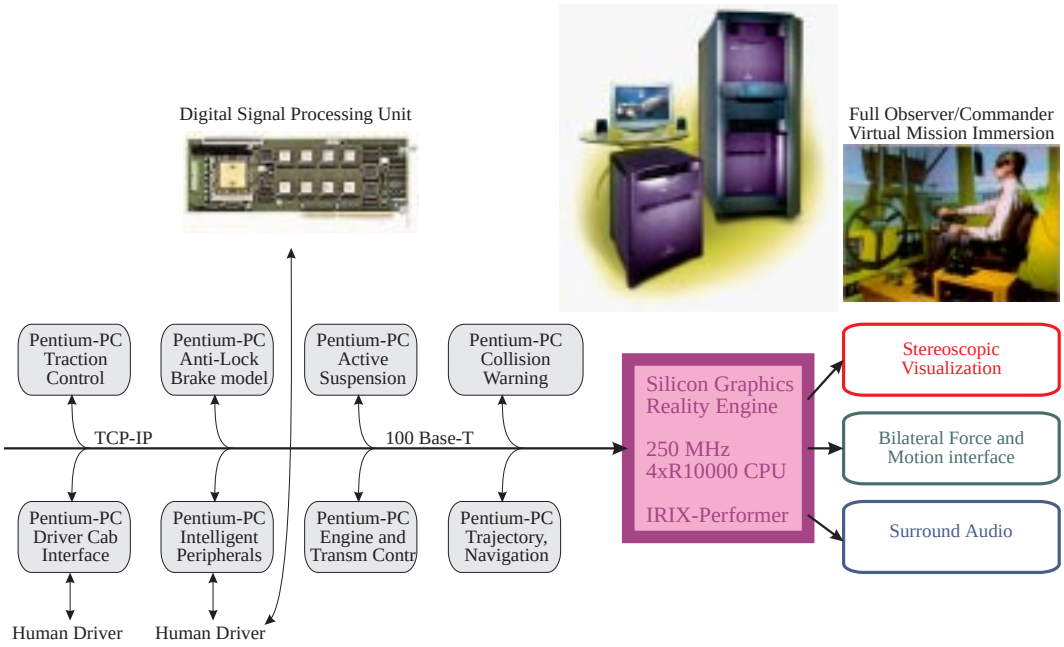


Figure 3: Hardware

with $\bar{\eta}(\mathbf{x}, \mathbf{u})$ and $\bar{\zeta}(\mathbf{x}, \mathbf{u})$ being non-linear functions of the states $\mathbf{x}$ and the inputs $\mathbf{u}$. Note that these non-linear functions are independant of time t. Time varying parameters in these functions are considered constant or near-constant. This condition implies some limitations in the generality of the formulation in that Equations 1 model the system around a certain equilibrium condition.

The system S is thus characterized by the four matrices A, B, C and D and the non-linear driving functions $\bar{\eta}(\mathbf{x}, \mathbf{u})$ and $\bar{\zeta}(\mathbf{x}, \mathbf{u})$. A compact descriptor model for the system can be formulated as follows

$$\begin{bmatrix} \dot{\mathbf{x}} \\ \mathbf{y} \end{bmatrix} = \mathcal{D} \begin{bmatrix} \mathbf{x} \\ \mathbf{u} \\ 1 \end{bmatrix} \tag{2}$$

where, for a system with n states and m inputs and k outputs, the descriptor matrix $\mathcal{D}$ will be given as

$$\mathcal{D} = \left[\begin{array}{ccc|ccc|c} A_{1,1} & \cdots & A_{1,n} & B_{1,1} & \cdots & B_{1,m} & \eta_1(\mathbf{x}, \mathbf{u}) \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ A_{n,1} & \cdots & A_{n,n} & B_{n,1} & \cdots & B_{n,m} & \eta_n(\mathbf{x}, \mathbf{u}) \\ \hline C_{1,1} & \cdots & C_{1,n} & D_{1,1} & \cdots & D_{1,m} & \zeta_1(\mathbf{x}, \mathbf{u}) \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ C_{k,1} & \cdots & C_{k,n} & D_{k,1} & \cdots & D_{k,m} & \zeta_k(\mathbf{x}, \mathbf{u}) \end{array}\right] . \tag{3}$$

Using this representation in Equation 2, the vehicle dynamics model that is given in Figure 2 has been formulated. From Figure 2, it can be seen that the model can be partitioned in a series of three submodels $\mathcal{D}_I$, $\mathcal{D}_{II,i}$ and $\mathcal{D}_{III}$, for the steering, brake and powertrain, for the suspension and wheel dynamics and for the chassis respectively. Further detail on the vehicle modeling aspects are not outlined here, and [4] is referred to.

Figure 4: Seat

4 Distributed Simulation

The setup for the multi-processing vehicle simulation is presented in Figure 3. The concept is to replace one or more states in the general vehicle dynamics state space system matrices, with outputs of a model that is simulated elsewhere on the integrated network. For the interaction this means that in each integration step, the VVSS will request the modules for replacement of system states, and then return a set of updated states. The distributed configuration that is hereby defined is called *Simulation Platform Integrated Network Environment* (SPINE).

The objective for parallel simulation through SPINE, is so that states or submodels of the vehicle model can be selected and configured dynamically, without re-organizing

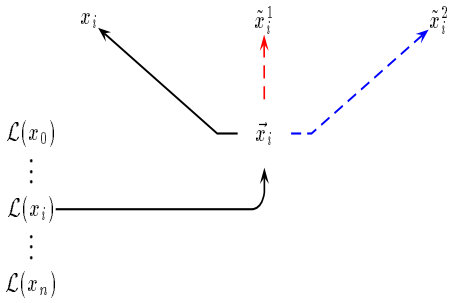


Figure 5: Indexed states represent state values through indirect addressing. The library index $\mathcal{L}(x_i)$ allocates the pointer $\vec{x}_j$ which is the index reference pointer for the actual state value x_i. The contents of the pointer can be changed to reference another state value, for example $\tilde{x}_i^1$ or $\tilde{x}_i^2$.

the structure of the vehicle dynamics model itself.
In other words, the system must be able to allow for any state at any given time, to be separated from the model and be computed in a parallel process through SPINE. The methodology that has been adopted to achieve this objective is called *indexing*, which makes use of a *pointer table*. The pointer table in the application of SPINE will be referred to as *index library* $\mathcal{L}(\cdot)$. The method has been widely used in computer science, in the application of interrupt handling.

The idea is that instead of the actual state values, a pointer is used in the state equations. The pointer directs the simulation to the location of the state value. The term *indirect addressing* is generally adopted by software developers for this method, where an interrupt pointer table is used to identify the locations of interrupt handler routines.

In the application of the virtual vehicle system simulation, it means the following. As soon as we would like to replace a state with the output of a parallel simulation, we will only need to change the pointer for the particular state, to direct the algorithm of the simulation to the location of the new value. The notation $\vec{x}$ is adopted to denote the pointer to the state value x. The concept of the state value pointer is shown in the diagram in Figure 5.

The following example shows the use of state pointers for a linear system.

EXAMPLE 1 Parallel simulation using state pointers.

The implicit state equations, using the bilinear approximation method will be expressed in terms of the state pointers.

$$
\begin{aligned}
x_{1,k+1} &= \vec{x}_{1,k} + ah\vec{u}_k \\
x_{2,k+1} &= \vec{x}_{2,k} + b\frac{h}{2}\left(\vec{x}_{1,k+1} + \vec{x}_{1,k}\right) \\
x_{3,k+1} &= \vec{x}_{3,k} + c\frac{h}{2}\left(\vec{x}_{2,k+1} + \vec{x}_{2,k}\right)
\end{aligned}
$$

The input u_k is considered to be a zero-order hold input for simplicity. By default the pointers represent the location of the state value;

$$
\begin{aligned}
\vec{x}_1(t) &\rightarrow x_1(t) \\
\vec{x}_2(t) &\rightarrow x_2(t) \\
\vec{x}_3(t) &\rightarrow x_3(t)
\end{aligned}
$$

When we would like to replace the model for state x_2, the second state equation will be simulated in a parallel process, as

$$\tilde{x}_{2,k+1} = \vec{x}_{2,k} + dh\vec{x}_1,$$

and the pointer to the second state will be changed to

$$\vec{x}_2(t) \rightarrow \tilde{x}_2(t).$$

where $\tilde{x}_2$ denotes the remote state value. □

Initialization of a remote process through SPINE takes place when a *state request form* is submitted to the VVSS. In the request form, the remote processor will list a set of state indexes which it will need to receive, and a set of state indexes for which it will submit values at each simulation interval.

After initilization, the computation of the vehicle dynamics model will use the values of the remote process $\tilde{x}(t)$, for the states that are defined in the submitted request form as outputs of the remote simulation.

When the remote simulation terminates, it will close the SPINE connection with the VVSS. The original state values will automatically be replaced in the vehicle dynamics model, and the pointer $\vec{x}(t)$ will be restored to direct to the original state value.

Parallel simulation has also an impact on the remaining vehicle dynamics system as it is simulated in the VVSS. Whereas bilinear integration is preferred for the vehicle dynamics, the integration of state values from two separated processes can only be implemented by forward Euler transformation, when no recursion is allowed.

It is therefore important to consider the dynamic behavior of the state values that will be replaced in the vehicle model. The following example will demonstrate this case.

EXAMPLE 2 Implication for integration when simulating x_2 in parallel.

The original system, using the implicit bilinear approximation for internal state integration was described by

$$
\begin{aligned}
x_{1,k+1} &= x_{1,k} + ahu_k \\
x_{2,k+1} &= x_{2,k} + b\frac{h}{2}\left(x_{1,k+1} + x_{1,k}\right) \\
x_{3,k+1} &= x_{3,k} + c\frac{h}{2}\left(x_{2,k+1} + x_{2,k}\right)
\end{aligned}
$$

The input u_k is again considered to be a zero-order hold input. By simulating the equation for x_2 in a parallel process, we cannot longer use $x_{1,k+1}$ for computing $x_{2,k+1}$, nor $x_{2,k+1}$ for $x_{3,k+1}$. Table 1a shows how the equations of the system approximation are changed. The table is chronologically, in that is shows the exchange of state values, then the evaluation of the models in the main system (left) and the subsystem (right) for the integration step at time $t = kh$.

Alike in the preceding examples, the original system with bilinear approximation could be modeled, as the set of explicit state equations in Table 1b. This explicit form would not require a change of the integration method. The original system for the VVSS however, is not explicitly available in the bilinear approximation expressions. □

	$\longleftarrow \tilde{x}_{2,k}$	
$t = kh$	$x_{1,k} \longrightarrow$	
$t = kh + h$	$x_{1,k+1} = x_{1,k} + ahu_k$ $x_{3,k+1} = x_{3,k} + ch\tilde{x}_{2,k}$	$\tilde{x}_{2,k+1} = \tilde{x}_{2,k} + dhx_{1,k}$

(a) Explicit integration of a distributed model in SPINE using forward Euler

	$\longleftarrow \tilde{x}_{2,k}$	
$t = kh$	$x_{1,k}\ u_k \longrightarrow$	
$t = kh + h$	$x_{1,k+1} = x_{1,k} + ahu_k$ $x_{3,k+1} = x_{3,k} + ch\tilde{x}_{2,k}$ $+dc\frac{h^2}{2}x_{1,k} + adc\frac{h^3}{4}u_k$	$\tilde{x}_{2,k+1} = \tilde{x}_{2,k} + dhx_{1,k} + ad\frac{h^2}{2}u_k$

(b) Explicit integration of a distributed model using Bilinear Transformation

Table 1: Tables

Time	**VVSS**		**Submodel**
$t = kh$		$\longleftarrow$	Request form of indexes 102 202 105 205 2 -160 -260 *
	Values of Requested states 12.34 12.34 12.34 -1.1 0.4 12.34 1.1 0.4 0.0 *	$\longrightarrow$	
$t = kh + h$		$\longleftarrow$	Simulated state values 0.0 0.0 *
	Values of Requested states 12.34 12.34 12.34 -1.1 0.4 12.34 1.1 0.4 0.0 *	$\longrightarrow$	
$t = kh + 2h$		$\longleftarrow$	Simulated state values 0.0 0.0 *
	Values of Requested states 12.34 12.34 12.34 -1.1 0.4 12.34 1.1 0.4 0.0 *	$\longrightarrow$	
$t = kh + ih$		$\vdots$	

Table 2: Table with an example of communication

In order to administer replacements in the state variable pointer, a library index has been defined with the locations of the pointers to the actual values.

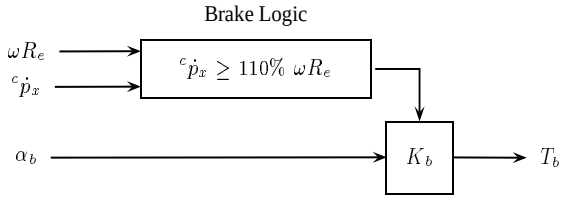
The index library $\mathcal{L}$ is a table which holds the pointers to the pointers $\vec{x}$, which on their turn locate the current state value for x. In order to be consistent throughout the modeling, reference to state variables and parameters must always be made through the pointers $\vec{x}$. The following example will be used to demonstrate the use of the pointer table.

EXAMPLE 3 Replacing the model for braking

A simplified model for ABS brake logic is presented as an example on how to use SPINE to replace a partition of the vehicle dynamics. The brake system computes a brake torque T_b, based on the brake pedal angle α_b. When the longitudinal vehicle speed is 10% larger than the wheel travel ωR_e (spin times effective radius) than the brake torque will be minimized. The actual model for the brake torque is not of interest in this example. We will assume

$$T_b = K(\alpha_b,\ {}^c\dot{p}_x, \omega R_e).$$

The diagram for the brake model is as follows.



In the index library we find the indices for the following pointers as

$$\mathcal{L}(\vec{\omega}_i) = 100i + 2$$

$$\mathcal{L}({}^c\vec{p}_{x,i}) = 100i + 5$$

$$\mathcal{L}(\vec{\alpha}_b) = 2$$

$$\mathcal{L}(\vec{T}_{b,i}) = 100i + 60$$

For implementation of this example it means that the first string will contain the indexes with a terminating character,

```
102 202 105 205 2 -160 -260 *
```

The VVSS will respond by sending the values for the requested states. Realize that ${}^c\dot{p}$ is is a pointer that consists of three values. For example,

```
12.34 12.34 12.34 -1.1 0.4 12.34 1.1 0.4 0.0 *
```

From then on, the VVSS only expects the value of the states that will be sent from the remote process (the two brake torques $T_{b,i}$).

```
0.0 0.0 *,
```

and the VVSS will respond by sending the values for the requested states as before. The process will continue until the connection closes. The VVSS will then restore the pointers for the torque to the refer to the original torques.

The process can be depicted in the chronological order of tasks as depicted in Table 2. □

Besides the change in integration scheme there is another important consideration for using SPINE to simulate subsystems. In the aim for real-time simulation there are two parameters that can be tuned for optimized performance. They are the frame-rate f and the subsampling factor n.

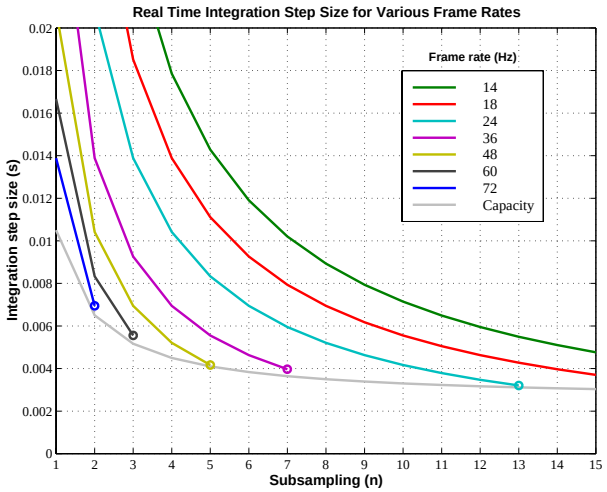


Figure 6: Frames

The consideration here is that communication for data exchange on SPINE takes place only once per frame. Hence, the more integration cycles in a frame (hence, the higher the number n), the larger the integration interval for the state that is to be replaced.

The vehicle dynamics model is simulated in a Silicon Graphics ONYX-II computer with 4 processors and *reality engine* graphics boards. The timing event for the visual output is generated by the hardware of the computer. This timing facility will be used as a reference to real-time. The hardware can be configured to generate a particular frame rate (i.e. 72, 36 or 24 Hz).

During each frame, a minimum number of tasks needs to be completed by the processor that computes the vehicle dynamics. Chronologically they are:

1. Compute the next state of the vehicle dynamics.
2. Update states from the vehicle dynamics which are animated in the animation to the graphic scenery rendering processes in CPUs 1, 2 and 3.
3. If there is data ready from SPINE in the network connection buffer, then
   (a) receive the data from the buffer. Interpret the input and insert the new values for the states into the vehicle dynamics model,
   (b) compile a set of current or future state values for the remote processes and pass them to the transmit buffer for SPINE.
4. Process keyboard and mouse inputs, and optionally receive peripherals for vehicle dynamics inputs.
5. Wait for the next frame rate synchronization

The selection of sampling time is critical with respect to convergence and real-time. In order to get more freedom for the selection of integration step size, we will apply *subsampling* for the evaluation of the vehicle dynamics.

Subsampling means that for task 1 in the above task list, the output for the vehicle dynamics model will be integrated n times, using a sub-sampling integration step size h_s. The result is a frame rate integration interval of

$$h = n\ h_s. \tag{4}$$

The actual duration of one frame, hence one total simulation interval, depends on the selected video frame rate,

$$T_f = \frac{1}{f} \qquad f\text{=14, 24, 36, ... 96 Hz} \tag{5}$$

It is now important to guarantee that execution of the task list (Tasks 1...4) will not exceed T_f seconds. Denoting the time that it takes for the computer to execute Tasks 2...4 by t_0, and the time that it takes to compute one iteration step of the vehicle dynamics model by t_s, this requirement can be expressed by[2]

$$t_0 + n\ t_s < T_f \tag{6}$$

At the same time, we must guarantee that the simulation of the vehicle dynamcis will evaluate a total amount of T_f seconds per frame, in order to be real-time. In other words,

$$h_s = \frac{T_f}{n} \tag{7}$$

From the above discussion, the maximum constraint for real-time simulation can be expressed for h_s in terms of the time that it takes to execute the task list,

$$h_s > \frac{t_0 + n\ t_s}{n} \tag{8}$$

This constraint is graphically represented in Figure 6, for several values of f, where for $t_0 = 0.008$ sec, $t_s = 0.0025$ sec, which are realistic estimates. For example, the VVSS is currently configured for $h_s = 0.004$ sec, and $n = 7$ at 36 Hz. If it was desirable to decrease the integration step size, we would need to lower the frame rate to 24 Hz, and use for example $h_s = 0.0038$ sec at $n = 12$.

The same idea applies to the modules. The sub-systems could be evaluated in a smaller integration step-size. The module will be integrated for the respective number of times, before actually communicationg with the VVSS. The advantage is that more accuracy or stability at greater stiffnesses can be achieved. That is, if the processor can perform the higher integration frequency in real-time.

5 Conclusions

A distributed simulation environment has been presented using vector based technology. In contrary with many known distributed simulation methods, the model does not need to be formally separated.

The application of the VVSS requires the real-time explicit formulation of the mathematical state space model. The configuration for distributed simulation that was presented in this paper could also be used for recursive integration schemes that do not require strict timing performance

[2] The requirement in Equation 6 applies to the sequencial processing of the simulation of the vehicle dynamics.

References

[1] *Vehicle Dynamics and Analysis Software*. a comprehensive 6 degree of freedom vehicle simulation program for the PC.

[2] J. B. Klaassens, G. Honderd, A. El Azzouzi, Ka C. Cheok, and G. E. Smid. 3d modeling visualization for studying controls of the jumbo container crane. In *Proceedings of ACC*, 1999.

[3] P. Y. Papalambros and N. F. Michelena. Optimal model-based decomposition of powertrain system design. Technical Report Technical Report UM-MEAM 95-03, Department of Mechanical Engineering and Applied Mechanics. University of Michigan., 1995.

[4] G. E. Smid. *Virtual Vehicle Systems Simulation*. PhD thesis, Oakland University, School of Electrical and systems Engineering, 1999.

[5] G. E. Smid, Ka C. Cheok, and K. Kobayashi. Simulation of vehicle dynamics using matrix-vector oriented calculation in matlab. In *Proceedings of CAINE (ISCA)*, pages 115–120, Orlando, FL, December 1996.