Journal of Computing and Information Technology - CIT 20, 2012, 2, 61–68
doi:10.2498/cit.1001771

Distributed Mathematical Model Simulation on a Parallel Architecture

Peter Kvasnica[1] and Igor Kvasnica[2]

[1] Centre of Information Technologies, Alexander Dubček University, Trenčín, Slovakia
[2] Faculty of Public Policy and Public Administration, College in Sladkovičovo, Slovakia

The aim of this article is to discuss the design of distributed mathematical models and suitable parallel architecture of computers. The paper summarises the author's experience with mathematical modelling of decomposed information systems of a simulator. Conclusions are based on the theory of the design of the computer control systems. The author describes computers that create a distributed computer system of a flight simulator. Modelling of a time precision of mathematical model of the speed of a simulator system is done by describing equations. The qualities of models depend on the architecture of computer systems. Some functions of other sections of POSIX are also analysed including semaphores and scheduling functions. An important part of this article is the implementation of computation speed of aircraft in multicore processor architecture.

Keywords: centralized and decentralized control system, mathematical modelling, flight simulator, operating system, thread

1. Introduction

Information on behaviour of a system is centralised in a system control theory. Design and creation of information systems with distributed databases are important assumptions for decentralised decision-making and control systems (Blakelock 1991).

Design decentralised control computer systems (DCCS) belong to the area of a control theory and information technology with the need of real-time control of a system. Problems arising in a centralised system control theory are very often transmitted to the area of decentralised control systems.

The efficiency of the various models in the range is characterized by the maximum number of processors that can be installed. After its introduction in the mainframe world, the approach was adopted by minicomputer manufacturers (in the 1970s) and then in the 1980s for UNIX systems based on standard microprocessors. The Windows NT systems were conceived from the very start to support symmetric multiprocessing (SMP), and today even desktop PCs are adopting the approach. Our research work is oriented to the design mathematical models, their decomposition and simulation on the computer systems. Nowadays computer architectures provide the various types of parallel computer system for simulation. It's from one multicore processor, through multiprocessor systems to the massively parallel systems. The parallel systems programming is rapidly moving towards introducing parallelism wherever performance matters. These systems provide entire models with its own set of enterprise computing environment, including application scalability, improved resource efficiencies, faster deployment time, etc. Accordingly, the area of interest in implementation of simulation is high.

2. Decentralized Mathematical Models of Simulator System

We get linear, manageable, dynamic system connected to the model (Clark 1996):

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t), \\ \mathbf{y}(t) &= \mathbf{D}\mathbf{x}(t), \\ \mathbf{x}(t) &= \mathbf{x}_0; \end{aligned} \tag{1}$$

where: $\boldsymbol{A}$ has the dimensions $n \times n$, $\boldsymbol{B}$ has the dimensions $n \times m$, $\boldsymbol{D}$ has the dimensions $r \times n$, $\boldsymbol{x}$ has the dimensions $n \times l$, $\boldsymbol{u}$ has the dimensions

$m \times l$, $\boldsymbol{y}$ has the dimensions $r \times l$ (columns). To make the task easier in the way that we will focus on the object of the control, the equation (1) can have a general shape:

$$\dot{\mathbf{x}}_i = \mathbf{A}\mathbf{x}_\mathbf{i}. \tag{2}$$

The equation (2) comprises 11 state variable sensors of information, 18 state variables that express situation coordinates of performing elements in the system. They are divided into two halves and the rest is divided into 38 state variables that represent unmeasured noise and sensor failures. These four parts of a piloting control system are expressed by the stage of the state vector $n = 4$ that represents a stage of a matrix.

The given system can be simulated by more computers or processors. In case more processors of a simulator system are involved $(P_1, P_2, \ldots, P_n)$, they communicate with each other by means of a shared memory, Figure 1. Processors unified in this way perform all necessary simulation services including directing, diagnosing, data flow managing etc.

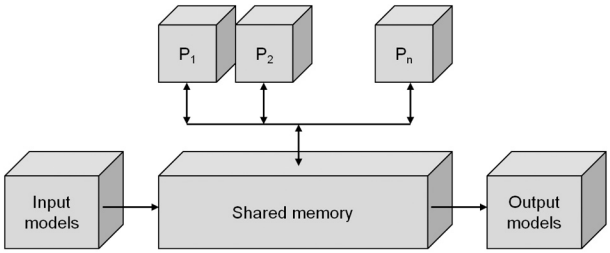

Figure 1. Block diagram processors in simulation system.

3. Decentralized Mathematical Model Aircraft in Simulator

According to the given facts the equation (2) can be expressed (Lazar 2007):

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}. \tag{3}$$

Let us decompose the given system into four subsystems. The first one shall be the subsystem of state variable sensors, the second and the third ones shall be a subsystem of performing items (two systems), noise and failures of the apparatus shall be measured by the fourth one. Thus the subsystems have the order $n_1 = 1$, $n_2 = 1$, $n_3 = 1$, $n_4 = 1$, of course, it does not have to be like this. The state space is divided into 4 parts:

$$\begin{aligned} \mathbf{x} &= (x_1, x_2, x_3, x_4)^T \\ &= (x_1x_1, x_1x_2, x_1x_3, \ldots, x_4x_4), \end{aligned} \tag{4}$$

where: x_ix_j represent the items of a state vector. If i represents order relevancy n, i. e. the number of the subsystem, then j – stands for sequential number of the items in the given subsystem. The architecture of the system matrix $\boldsymbol{A}$ in the state space after multiplication, the following shape can be formed (Lazar 2007):

$$\begin{aligned} \dot{x}_{11} &= a_{11}x_1x_1 + a_{11}x_1x_2 + a_{11}x_1x_3 + a_{11}x_1x_4, \\ \dot{x}_{12} &= a_{12}x_1x_1 + a_{12}x_1x_2 + a_{12}x_1x_3 + a_{12}x_1x_4, \\ &\vdots \\ \dot{x}_{44} &= a_{44}x_1x_1 + a_{44}x_1x_2 + a_{44}x_1x_3 + a_{44}x_1x_4. \end{aligned} \tag{5}$$

Then we decompose 16 subsystems into four parts called isolated subsystems:

$$\begin{aligned} \dot{x}_{11} &= a_{11}x_1x_1 + a_{11}x_1x_2 + a_{11}x_1x_3 + a_{11}x_1x_4, \\ &\ldots \\ \dot{x}_{14} &= a_{14}x_1x_1 + a_{14}x_1x_2 + a_{14}x_1x_3 + a_{14}x_1x_4, \\ &\vdots \\ \dot{x}_{41} &= a_{41}x_1x_1 + a_{41}x_1x_2 + a_{41}x_1x_3 + a_{41}x_1x_4, \\ &\ldots \\ \dot{x}_{44} &= a_{44}x_1x_1 + a_{44}x_1x_2 + a_{44}x_1x_3 + a_{44}x_1x_4. \end{aligned} \tag{6}$$

The above mentioned process is a process of decomposition carried out by matrix and vector operations. The division results in sixteen blocks that are formed in the matrix $\boldsymbol{A}$. They are marked according to the order "n": A_{11}, A_{12}, $\ldots$, A_{43}, A_{44}. Where:

$$\begin{aligned} \dot{x}_1 &= A_{11}x_{11} + A_{12}x_{12} + A_{13}x_{13} + A_{14}x_{14}, \\ \dot{x}_2 &= A_{11}x_{11} + A_{12}x_{12} + A_{13}x_{13} + A_{14}x_{14}, \\ &\vdots \\ \dot{x}_{16} &= A_{44}x_{11} + A_{44}x_{12} + A_{44}x_{13} + A_{44}x_{14}, \end{aligned} \tag{7}$$

Mathematical description of isolated subsystems has the following form:

$$\begin{aligned} \dot{x}_1 &= A'_{11}x_1,\ \dot{x}_2 = A'_{22}x_2, \\ \dot{x}_3 &= A'_{33}x_3, \dot{x}_4 = A'_{44}x_4. \end{aligned} \tag{8}$$

Where:

$$A_{11}^{'} = \begin{pmatrix} a_{11} \\ a_{12} \\ a_{13} \\ a_{14} \end{pmatrix}, \quad A_{22}^{'} = \begin{pmatrix} a_{21} \\ a_{22} \\ a_{23} \\ a_{24} \end{pmatrix}, \quad A_{33}^{'} = \begin{pmatrix} a_{31} \\ a_{32} \\ a_{33} \\ a_{34} \end{pmatrix}, \quad A_{44}^{'} = \begin{pmatrix} a_{41} \\ a_{42} \\ a_{43} \\ a_{44} \end{pmatrix}. \tag{9}$$

The mutual relations between the first and second isolated subsystems are:

$$l_{12}(x) = A_{11}^{'} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = (a_{11}, a_{12}, a_{13}, a_{14})^T \begin{pmatrix} x_{11} \\ x_{12} \end{pmatrix}. \tag{10}$$

The analysed case is simple and well described in the bibliography under number (Blakelock 1991). It is necessary to know whether every system can be decomposed. The capability of being decomposed can be calculated by means of incidental matrices that can be utilized in cases when the mathematical model of the system is known. Units of incidental matrix are placed when its elements are permuted or transformed so that they can appear diagonally. The number of items on the main diagonal is given by the number of subsystems.

In most mathematical models of large pilot control complexes, connections comprise only few mutual variables and that is the reason why the process of decomposition can be carried out. A form of decomposition can be effective when it enables forming separate systems, forming subgroups of connections without mutual variables. In that case, it is possible to work with each system on an independent base.

4. Multiprocessor and Multicore System

The processing capability of a symmetrical multiprocessor is adjusted to meet the needs simply by installing additional processors until the requirements are met. This allows a customer to increase the capabilities of "*n*" installed system by simple addition of processor modules (see Figure 2). The workload is distributed fairly among all the installed processors. SMPs are also referred to as tightly-coupled multiprocessors (Chevance 2005).

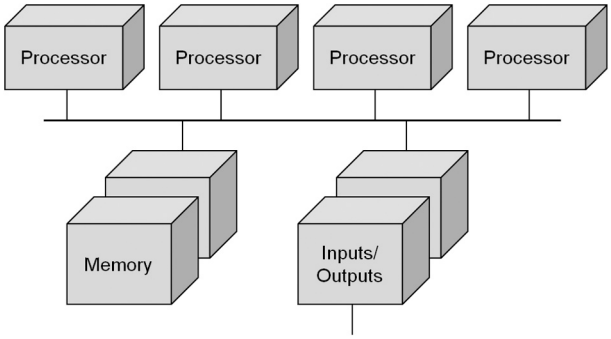

Figure 2. The multiprocessor architecture.

The core can be configured easily and can be adjusted according to the users' requirements or technical specifications of computers. The core configuration has an impact on the complex system behaviour, its performance, output, stability and response which results in increase of the general output of the operational system.

From the point of view of architecture, there can be either monolithic module aided core that means drivers can be implemented in the system in the form of modules while the system is operating. Multi core processors are launched into the market as they represent the cheapest alternation that helps meet the demand for an increased output (Martincova 2008).

Multi core processors can be successful and effective if huge bulks of information are processed and are optimized for the throughput. Recently, more complicated video in HD resolution has been introduced in the market; the quality of digital audio records has an increasing tendency.

To assess the operation of servers, there are more important factors, such as their performance, reliability – no failure in data processing. In the process of evaluation buses are employed and they are equipped with mechanisms of data consistence control and error correction. Another quality indicating the reliability is the ratio between the input power and performance.

Employing multi core processors means that there is still an option to employ more processors in one system. Graphic stations and servers can multiply the amount of the calculating performance in the way that the basic board can serve more processors at the same time. In case four cores are not sufficient, 4 x 4 can be used and the problem is resolved.

A lot of optional parameters have impact on a reliable performance of the processor and the complete computer system. The parameters can be adjusted at the core level of the operation system. The most important parameters are (Martincova 2008):

- Tickles' System (NO_HZ) – interrupting timers is processed differentially if the system is overloaded or not loaded sufficiently.
- High Resolution Timer Support (HIGH_RES _TIMERS) – switches the timer support with high resolution on.
- Symmetric multi-processing support (SMP)
- Sub architecture Type.
- HPET Timer support (HPET_TIMER) – to manage time HPET (high precision event timer is employed.
- Maximum number of CPUs (NR_CPUS) – to limit maximum number of processors.
- SMT (Hyper threading) scheduler support (SCHED_SMT) – to schedule processors Intel Pentium 4 with Hyper Threading support
- Multi-core scheduler support (SCHED_MC) – improves performance of the scheduler CPU in multi core processors.
- Memory split – controls the memory split for users' programmes and the core.
- Math emulation (MATH EMULATION) – math emulation of the co-processor.
- Timer frequency – is employed in switching tasks over (100, 250, 300, 1000 Hz).
- Kexec system call (KEXEC) – enables to switch over performing core for other one.

5. Softwares to Design Process and Threads

The main task is to create a program based on a mathematical notation model in "symbol presentation". The program shall simulate the given model. For this purpose, more integrated development environments (IDE) with code writing support in advanced program languages like C/C++, C#, Pascal, Java and others can be used. Writing the resource code is an inevitable condition to create a process based on this code in a real computer system with a built-in operation system. The simulation speed and accuracy are influenced by many parameters that have already been pointed out in the paper.

It has been proved that one core systems are able to form just one thread at the same time. The computing time of this core is shared by all processes and threads via the function OS called time split. More often the multi processors or multi core processors systems are utilized, which results in improving system possibilities for forming processes and threads at the same time, but this is also possible with the systems without multi processors or multi cores.

Processes are often considered to be the same as a program or application. But the application that is considered to be simple by the user can be a real set of co-processes. To make the processes communications easier, most OS offer way of inter-process communication like pipes and sockets.

Software designed for one type processor deployment shall work perfectly well on a compatible SMP, but the performance available shall be unchanged, one-type processor software cannot harness the power of multi-core processors. The software must be designed appropriately fundamentally; it must be designed as multiple units - threads that can execute instructions concurrently.

For multi core processors it is necessary to decompose the mathematical model in the way that one core can simulate the distributed mathematical model. This conception has many advantages for large systems and is demanding from the synchronization point of view. These qualities enable communication with models in other systems.

The SMP programming model is based on the concept of shared memory, which provides a simple and effective way for multiple processes to cooperate in simulation. Variables may be shared between processes; when this is done, access to those variables requires some discipline. Operating systems provide mechanisms to declare regions of memory (Chevance 2005). These applications may be organized as a collection of multiple elements structured as cooperating processes.

Maintaining much of the concept, mainframe systems long ago introduced the idea of a subprocess, which was borrowed and reused in the UNIX and Windows worlds and renamed as threads. The approach simply multiplexes

multiple lightweight processes within a single process.

6. Programming and Application

Threads and also processes offer the environment for running and setting the program up. Forming a new thread is easier than creating a new process. Threads act inside the processes, there is at least one thread employed in each process. Threads share the process means, including the memory and open sets.

The mathematical model that has been formed needs programming in the programme language, and then translating, linking with the libraries involved and then the process can start. For implementation of the resource code of the mathematical model, it is advised to use threads at the stage of the operation system. Parallel or narrowed threads can be used.

The thread is sometimes called lightweight process – LWP and it forms an important, basic unit in using the processor. The unit comprises the programme counter, set of registers and a stack. The code, data part of the process comprising the programme counter and the resource division OS are shared by all process threads. Traditional or heavyweight process represents a task with only one thread (Martincova 2008).

Easy sharing CPU represents switching over among threads and difficult sharing, then forming threads and comparing them to switching over for the context in heavyweight processes.

Some systems implement the users' level of threads in users' libraries instead of threads implemented via systems calls. Then, switching over the threads does not require OS aid and does not initiate interruptions. If there is one thread in the OS core that calls, each core by each process stops its complete task until the core responds.

Thread functions are clearer than the ones in multi core and multi process systems. Every process in the multi process system has its own programme counter, stack and addresses space. This method of organization is suitable if the processes performing the programmes in the system are independent on themselves. If only one thread of the task is blocked, it is possible to match the processor to another one (as it is in the same address space). Threads that form a part of the same task and collaborate, improve the throughput and the performance of the system, what can be used in simulation.

6.1. Libraries Pthread

The user space is reused for each process. That means, every time a user process starts execution, the operating system sets up the virtual to real address translation tables in the Memory Management Unit (MMU) to map the 0 to 2 GB virtual address space onto the portions of real memory that hold that process.

Cooperation among processes is based on information exchange using shared files or interprocess communications mechanisms, these being themselves either message-based (for example, using pipes or shared-memory mechanisms).

Using shared memory implies that the user processes must declare to the operating system their intention to share memory, whereupon the system arranges a shared region in the virtual address space of the cooperating processes. Interprocess communication by means of shared memory is by far the most efficient mechanism available (Chevance 2005).

Each lightweight process has its own procedure call stack (as does the owning process) and its own context saves area (register contents, instruction pointer, and so on).

The programming interface for threads has been standardized for UNIX (POSIX 1003.4a); lightweight processes may be supported either directly by the operating system (as in Windows Server 2003), or as a separate subsystem (as in some UNIX variants). The availability of a new release of Pthreads-win32, an Open Source Software implementation of the Threads component of the POSIX 1003.1 2001 Standard for Microsoft's Win32 environment was implemented.

Pthreads-win32 currently implements a large subset of the POSIX standard threads related API. The Win32 pthreads are normally implemented as a dynamic link library (DLL). This has some notable advantages from the Win32 point of view, but it also more closely models existing pthread libraries on UNIX which are usually shared objects.

The library is being used in many projects, either migrating from UNIX platforms or developing cross-platform applications.

The one thread computes simulation 1st item from equation (15), other thread computes the simulation 2nd item from equation (15), etc. Every thread is created *CreateThread* function's call with parameters define the normal priority class (Siram 2009). A *synchronization object* is an object whose handle can be specified in one of the *wait functions* to coordinate the execution of multiple threads. Synchronization of simulation between running threads performs functional call *WaitForSingleObject*, *functions* allow a thread to block its own execution.

The function WaitForSingleObject requires a handle to one synchronization object. These functions return when one of the following occurs (Synchronizing 2010):

- The specified object is in the signaled state.
- The time-out interval is set to 5ms, in our case, to specify that the wait will not time out.

The wait functions do not return until the specified criteria have been met. How did we use the *semaphore* object? When a wait function is called, it checks whether the wait criteria have been met. This call enables cooperation threads, the inter-process communications mechanisms, based on information exchange using shared memory.

6.2. Application Model of Speed Depending on Fuel Supply

From mathematical model, in the longitudinal direction arise (Krasovskij 1980):

$$W_V^{\delta_T}(s) = -a_x^{\delta T}\frac{\Delta_{11}(s)}{\Delta(s)}, \qquad (11)$$

where $a_x^{\delta T}$ is speed coefficient with respect to fuel supply, $\Delta(s)$ – determinant of the transfer function, $\Delta_{11}(s)$ – algebraic adjunct. In the numeric formulation valid $\Delta(s) = s^4 + 1.1338s^3 + 62.7975s^2 + 28.6585s + 4.09291$ and $\Delta_{11}(s)$ is $= s^3 + 1.12s^2 + 62.782s + 25.32$. The transfer function (11) has a form:

$$\begin{aligned} \Delta V_V^{\delta_T}(s) &= W_V^{\delta_T}(s) * \Delta\delta_T(s) \\ =5\frac{s^3+1{,}12s^2+62{,}782s+25{,}32}{s^4+1{,}1338s^3+62{,}7975s^2+28{,}6585s+4{,}09291} &* \Delta\delta_T(s). \end{aligned} \qquad (12)$$

From the equation (12) ensure: $b_3 = 1$; $b_2 = 1.12$; $b_1 = 62.782$; $b_0 = 25.32$; $a_4 = 1$; $a_3 = 1.1338$; $a_2 = 62.7975$; $a_1 = 28.6585$; $a_0 = 4.09291$. The input quantity $\Delta\delta_T(s)$ is one step, i. e., $\Delta\delta_T(s) = 1/s$. If we define new variables $x = W_V^{\delta T}(s)/-5$; $u = \Delta\delta_T(s)$, the transfer function is obtained:

$$\begin{aligned} &\left(a_4s^4 + a_3s^3 + a_2s^2 + a_1s + a_0\right)x \\ &= \left(b_3s^3 + b_2s^2 + b_1s + b_0\right)u, \end{aligned} \qquad (13)$$

The examine going from a differential equation to state space. We'll do this with a system (12), that will demonstrate the usefulness of a standard technique. After rewriting transfer function (13) to the framework differential equations of the first order in state space we have (Driels 1996). Converting from state space form to a transfer function is straightforward, the transfer function form is unique. Converting from transfer function to state space is more involved, largely because there are many state space forms to describe a system (Glasa 2009):

$$\begin{aligned} x_1 &= x, \\ x_2 &= \dot{x}_1 = sx, \\ x_3 &= \dot{x}_2 = s^2x, \\ x_4 &= \dot{x}_3 = s^3x, \end{aligned}$$

$$\dot{x}_4 = \frac{1}{a_4}u - \frac{a_3}{a_4}x_4 - \frac{a_2}{a_4}x_3 - \frac{a_1}{a_4}x_2 - \frac{a_0}{a_4}x_1. \qquad (14)$$

The Frobien matrix notation of the framework equations (14) is:

$$\begin{vmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{vmatrix} = \begin{vmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -\frac{a_0}{a_4} & -\frac{a_1}{a_4} & -\frac{a_2}{a_4} & -\frac{a_3}{a_4} \end{vmatrix} \cdot \begin{vmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{vmatrix} + \begin{vmatrix} 0 \\ 0 \\ 0 \\ -\frac{1}{a_4} \end{vmatrix} u. \qquad (15)$$

When calculated numeric values are implemented, we shall obtain the next shape of differential equation with fourth item (McCormic 1995):

$$\begin{aligned} \dot{x}_4 = 1 * u - 1,1338 * x_4 - 62,7975 * x_3 \\ - 28,6585 * x_2 - 4,09291 * x_1. \end{aligned} \qquad (16)$$

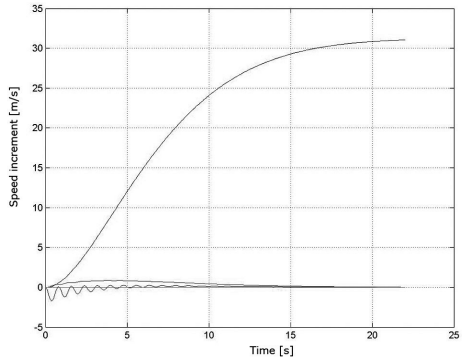

Figure 3. The time dependency increase of speed from fuel supply.

Output vector has a form:

$$y = [21,22709; 34,1235; -61.7975; -0,1338] * [x_1, x_2, x_3, x_4]^T . \quad (17)$$

Mathematical model aircraft displacement of speed (ΔV) is given by equation (17). The one step unit of the fuel supply $\Delta\delta_T(s)$ has sign "+", however any coefficient has sign "−", equation (16). After multiplying dot product from this equations the result has sign "+". There is a very similar situation in the second equation (17), for state variable y. Information about mathematical solving of these equations is given in (Clark 1996). These results match the physical substance of solving problem. The graphic results from equations (16) and (17) modelling by thread running on a processors core are in the previous picture.

If the mathematical model speed of aircraft from fuel supply is modelled at the decomposed system by threads on computer, the steady state value is 25.016 m/s.

6.3. Comparing Results with Matlab

Results gained in mathematical model simulation by means of threads in multi-processor system aided by inter-process communication can be compared to the results of the famous software tools Matlab. The method of implementing our solution variant is a new one, while utilizing the given computing architecture in model simulation and design.

The Matlab is a world-wide known system and a programme enabling different mathematical calculations like vector and matrix simulations in controlling systems, calculating differential equations, etc.

We compare the Matlab and the tool Simulink as a model environment and we employ the equation (13) as a mathematical model for Simulink to execute the simulation. A result is obtained and has graphical form in Figure 4.

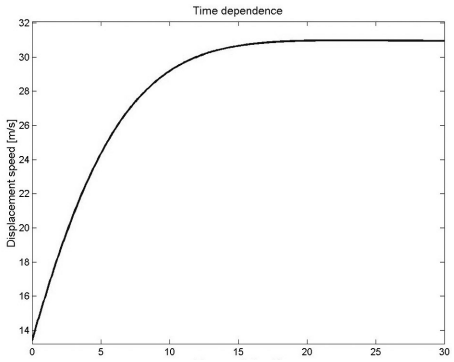

Figure 4. The time dependency increase of speed from fuel supply in Matlab.

Results gained in the process of comparing mathematical model simulation in our way show following:

- The value of speed increase at steady level is identical if compared to the Matlab or Simulink.
- Dynamics of simulation of a mathematical model in our solution is delayed by 2-3 sec if compared to the dynamics in Matlab.
- This simulation is more efficient than the one in Matlab.

The delay is caused by the distributed mathematical model designed in this way. It is caused by decomposition into parallel simulation and results transfer between parallel processes simulating mathematical model defined in the equation (14).

7. Conclusion

Modelling the speed of an aircraft at the computer in the form of mathematical notation equations (17) or (12) is made by thread of simulating process on the processor.

Multi core-processor architecture of a decomposed system is only considered in the analysis of the task. Small distortions, if compared to original system, are caused by this configuration.

The above mentioned method can also be used in other modelled configurations of an aircraft simulator according to (Rolfe 1986) and (Blakelock 1991). A mathematical notation of calculation modelling the speed of an aircraft for the decomposed subsystems of the application on computers in an aircraft simulator must be clarified. These notations include commercial, research and other applications, many of which are accessed in the Google search.

Received: January, 2010
Revised: July, 2012
Accepted: July, 2012

Contact addresses:
Peter Kvasnica
Centre of Information Technologies
Alexander Dubček University
Trenčín, Slovakia
e-mail: peter.kvasnica@tnuni.sk

Igor Kvasnica
Faculty of Public Policy
and Public Administration
College in Sladkovičovo
Slovakia

References

[1] J. H. BLAKELOCK, *Automatic Control of Aircraft and Missiles*, Second Edition, John Wiley & Sons. Inc.: New York, 1991.

[2] R. N. CLARK, *Control System Dynamics*, First Published, Cambridge University Press, New York, USA, 1996.

[3] M. DRIELS, *Linear Control Systems Engineering*, McGraw-Hill Inc., San Francisco, USA, 1996.

[4] J. GLASA, Computer simulation and predicting dangerous forest fire behaviour. In *International Journal of Mathematics and Computers in Simulation*, Vol. 3, Issue 2, pp. 65–72, 2009.

[5] R. J. CHEVANCE, *Server Architectures: Multiprocessors, Clusters, Parallel Systems, Web Servers, And Storage Solutions*. Elsevier, Digital Press, 2005.

[6] A. A. KRASOVSKIJ, *Sistemy avtomaticeskogo upravlenja poletom i ich analiticeskoje konstruirovanie* (In Russian), Nauka, Moskva 1980.

[7] P. MARTINCOVÁ, K. GRONDŽÁK, M. ZÁBOVSKÝ, *Programovanie v jadre operačného systému Linux* (In Slovak), Žilinská univerzita, 2008.

[8] B. W. MCCORMIC, *Aerodynamics, Aeronautics and Flight Mechanics*, John Wiley & Sons, Inc.: New York, Second Edition, USA, 1995.

[9] J. M. ROLFE, K. J. STAPLES, *Flight Simulation*, Cambridge University Press: Cambridge, 1986.

[10] T. LAZAR, F. ADAMČÍK, J. LABÚN, *Modelovanie vlastností a riadenia lietadiel* (In Slovak), first ed., Technická univerzita v Košiciach, Košice, 2007.

[11] S. SIRAM, S. S. BHATTACHARYYA, *Embedded Multiprocessors: Scheduling and Synchronization*, Second Edition (Signal Processing and Communications), Taylor & Francis Group, Inc.: New York, Boca Raton USA, 2009.

[12] SYNCHRONIZING Execution of Multiple Threads, [online], [cit 13.1.2010], On web: http://msdn.microsoft.com/en-us/library/windows/desktop/ms687069(v=vs.85).aspx, 2010.

PETER KVASNICA is the deputy director of the Centre of Information Technologies at Alexander Dubček University of Trenčín. He has been involved in research on mathematical models of flying objects and programming virtual reality applications focusing on verification and collaboration in a computer network. He has published papers on applications of mathematical methods for flying objects, in scientific programs with the emphasis on modeling these systems. He graduated from the University of Technology in Brno (VUT Brno), in the field of study: digital computers. He was awarded the degree of Doctor of Philosophy (PhD.) at M. R. Štefánik Military Academy of Aviation in Košice, in the specialization in computer science application (aircraft model cybernetic characteristics). He was awarded the degree Magister (Mgr.) at Faculty of Public Policy and Public Administration, College in Sladkovičovo, in 2012. He has been involved in the development of special adapted mathematical models of objects from the point of view of programming and use of distributed computer system in flight simulators for real-time applications. He is interested in software tools for parallel programming MPI, Open MP and information about Open CL for creation GPU software application.

IGOR KVASNICA graduated from the Slovak University of Technology in Bratislava, Faculty of Chemistry and Technology. He intensively studied fuel microbiological contamination and kept studying at M. R. Štefánik Military Academy of Aviation in Košice. He was awarded the degree Doctor of Philosophy (PhD.) in 2004. He was awarded the degree Magister (Mgr.) at the Faculty of Public Policy and Public Administration, College in Sladkovičovo, in 2012. He is interested in simulation and its application in implemented systems, development and tools for verification similar systems. He is interested in using formal methods applied in personal computers with an option to represent these results in virtual reality. At present, he is involved in the research on applications. He has worked out many papers on environmental issues and has prepared materials to be implemented into Slovak legislation on water quality and water and air pollution. Working on above mentioned issues, he has been employing different software tools and computer simulation.