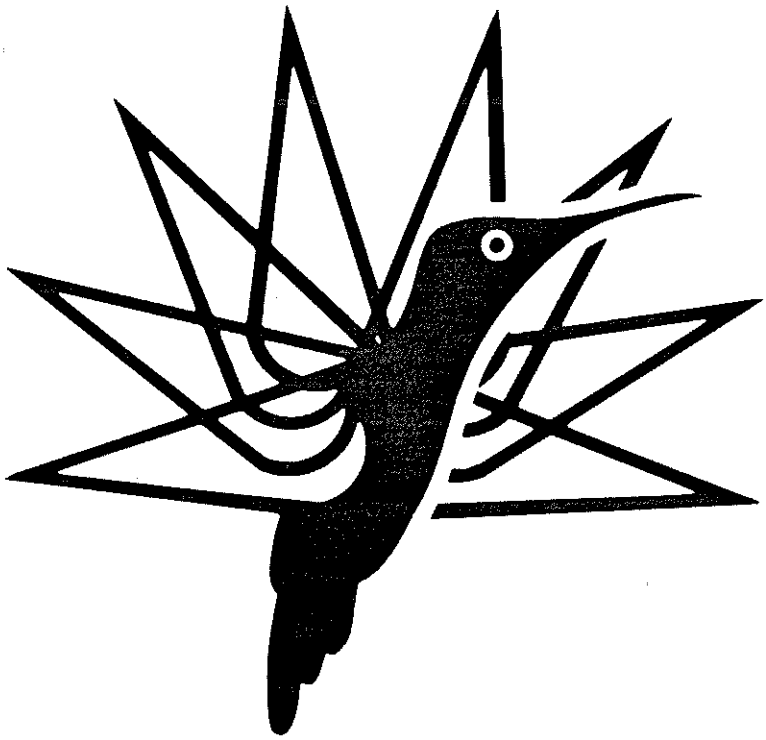
PAPER Nr. : 62



A NEW APPROACH IN HELICOPTER REAL-TIME SIMULATION

G. Lehmann
C.-H. Oertel
B. Gelhaar

Deutsche Forschungsanstalt für
Luft- und Raumfahrt
Institut für Flugmechanik
Braunschweig, FRG

FIFTEENTH EUROPEAN ROTORCRAFT FORUM

SEPTEMBER 12 - 15, 1989 AMSTERDAM

A NEW APPROACH IN HELICOPTER REAL-TIME SIMULATION

G. Lehmann
C.-H. Oertel
B. Gelhaar

Deutsche Forschungsanstalt für
Luft- und Raumfahrt
Institut für Flugmechanik
Braunschweig, FRG

ABSTRACT

Nowadays system simulation is an important task in the development procedure of new and improved helicopter systems. In addition to the typical off-line and non-real-time simulation of the system, special requirements on real-time computation speed exists for flight simulators. A further application of real-time simulation is the so called 'hardware in the loop simulation' where real components like new closed loop controllers are tested under realistic conditions.

In the past a lot of companies have designed and built special purpose simulation computers which are very powerful but expensive. The progress in computer science shows a trend to distributed systems where multiple processors are running in parallel to improve the performance dramatically.

At the DLR Institute for Flight Mechanics a computer system, based on the TANSPUTER which was developed by the British INMOS company, was built up to get the real-time simulation capabilities for the ROTEST model rotor.

1 DEMANDS UPON SIMULATION

At date simulation is a necessary step in the evaluation and design process of new concepts and systems. The complexity of the models used for simulation can vary from a very simple design to large and complex structures, depending on the goal and the system under simulation, [1]. Fig.1 depicts a block diagram of a typical helicopter simulation for basic research purpose where a visual system is not required, [2]. The main objective here is to simulate the flight mechanics under various flight conditions. The frequency range of interest is limited to up to the first flapping mode.

But the increasing demand of optimized vehicles also addresses the high frequency components in the helicopters frequency characteristic. Fig. 2 shows the bandwidth which must be considered if all effects are intendet to be modelled in a simulation program. If we assume, that generally simulation also includes the tasks without true real-time requirements, one can identify the following two ordering schemes for simulation in the field of helicopter research and development:

complexity of simulation model:

- isolated flexible blade for stress and frequency estimation
- main rotor with arbitrary blades combined with simple down wash models
- as above but with improved down wash models (incl. free wake)
- complete helicopter system with tail rotor, engines and drag and mass model of the fuselage
- as above but with additional simulation of avionics etc. (full scale cockpit)
- coupled rotor-flexible fuselage models for dynamic properties

computing time demands:

- more or less complex models of parts of a helicopter for design optimization (no real-time)
- more or less complex models of complete helicopters for pilot training (Basic, up to Class III, real-time)
- hardware-in-the-loop simulation for verification of real parts as controllers and flight control computers (real-time).

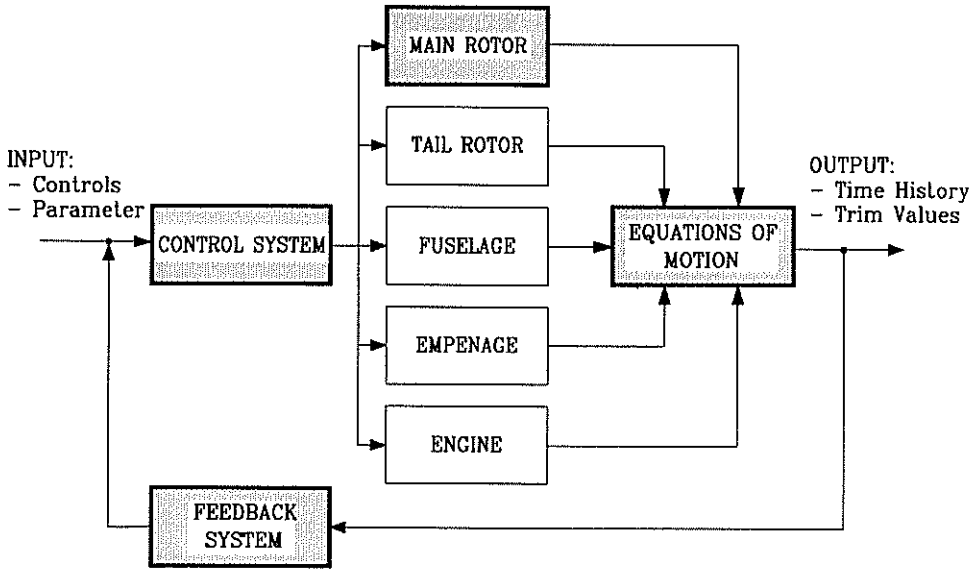


Fig. 1. Block diagram of a helicopter simulation.

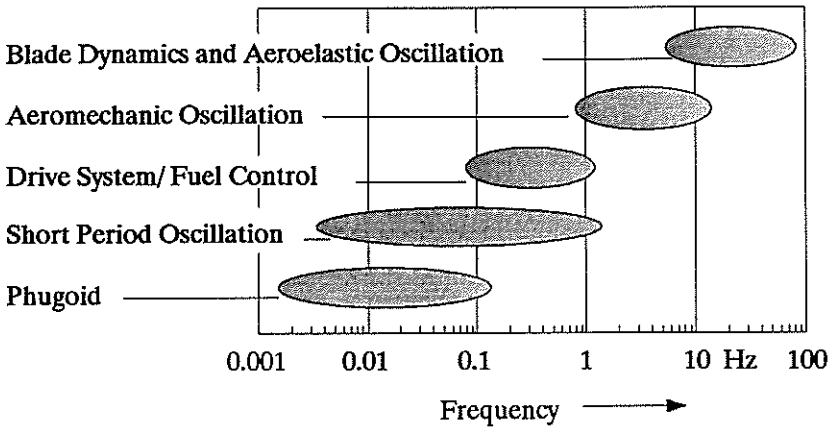


Fig. 2. Frequency characteristic of helicopter dynamics.

Actual simulators are very individual and they cover the full range of the, naturally incomplete list above. It is difficult to exactly define a simulator and therefore to draw the line between varieties. In addition, the demands (can) vary with the progress of system development and thus the improvement of the simulation model is a steady process. On the other hand the modern computer technology offers a wide range of different systems which are more or less suited to match the performance of the individual user. A significant disadvantage of most typically used systems is the almost fixed performance. For a growing simulation model it is necessary to order

an overdimensioned computer, and it often happens that after two or three years this system is unable to match the actual performance requirements. A new system is necessary followed by all related and well known modifications in hardware, software, and system management.

As the DLR Institute for Flight Mechanics, Braunschweig, decides to build up a simulation of the model rotor test rig (ROTEST), the highest frequencies as depicted in Fig. 2, should be addressed. In the past the model rotor has covered a wide range of different test objectives like performance, acoustics, and active control technologies (HHC: Higher Harmonic Control). The potential of the capabilities grows with every test period. But in modern helicopter systems more and more computer based augmentation of dynamic characteristics of the rotor system like narrow band disturbance rejection, lead-lag damping and tip path plane stabilizing, is an essential part of improvement. It is evident that such parts must be tested under realistic conditions before they can be adapted for wind tunnel tests. These demands led to the design and evaluation of a completely new real-time simulator concept for hardware-in-the-loop simulation, which is the subject of this paper.

2 MODEL DESCRIPTION

The ROTEST model is a four-bladed rotor with elastic blades (without flap and lag hinges) with 4 meter diameter. It can be used with different sets of blades, one being a 2.5 scale of the BO-105 main rotor blades. The rotor blades are equipped with strain gauges to measure the elastic flap, lag, and torsion motion. Additional sensors are used to measure the blade root pitch angle and the pitch-link forces and shaft bending moments. In the fixed frame a six component balance is instrumented to measure the net rotor forces and moments. The normal operating speed of the model is 1050 rpm to match the same blade tip Mach number as the full scale version. The simulator model must cover these conditions.

For the first evaluation a 24 degree-of-freedom model was chosen. Each blade dynamic is described in a modal formulation with three flap modes, two lead-lag modes, and one torsion mode, which leads to the well known equation of motion for the i-th eigenmode:

$$\mathrm{d}^2 q_i/\mathrm{d}t^2 + \nu_i^2 q_i = m_i^* F_i\,(q, \mathrm{d}q/\mathrm{d}t, \mu, \theta, t) \qquad (1)$$

where

ν_i is the eigenfrequency,
m_i^* is the generalized mass, and
F_i is the generalized external force.

The q_i 's are the generalized co-ordinates of the eigenmodes which are normalized to unity blade tip deflection. The eigenmodes are evaluated with a finite element method [3], and adjusted by modal measurement. For the calculation the blade was modelled by 40 coupled beam elements. Special emphasis was taken on the blade torsion, because this degree-of-freedom is forced by the aerodynamics and essentially by moments of inertia from the blade root pitching. Especially in the case of HHC this contribution is not negligible. The control inputs are directly given by a harmonic series applied at the blade root because ideal transfer functions for the actuators and the swashplate are assumed. The net forces and moments of the complete rotor are calculated with the inertia forces and moments which are caused by the mode deflections and the aerodynamic forces.

The amount of computation mainly depends on the complexity of the right hand side of equation (1). We implemented the blade element formulation with nine different elements, that means, the span of the elements decreases from the root to the blade tip in such a manner that each element describes the same disc area during a rotor revolution, Fig. 3. The computation of the aerodynamic forces at each element is based on a fully nonlinear model with special c_z, c_x and c_M tables depending on the local Mach-Number in x and z-direction. Fig. 4 depicts the relationship between the conventional formulation and the coefficients used in the real-time

simulation. Note, that V_s in Fig. 4 is the speed of sound, which only depends on the air conditions (pressure, temperature). More informations about this new formulation of the aerodynamic coefficients can be found in [4]. The tables of coefficients are defined in the range $-0.15<Ma_x<0.9$ and $-0.1<Ma_z<0.16$, Fig. 5. A table-look-up procedure with a linear interpolation algorithm is implemented to give the actual coefficients for arbitrary Mach-Numbers within this range.

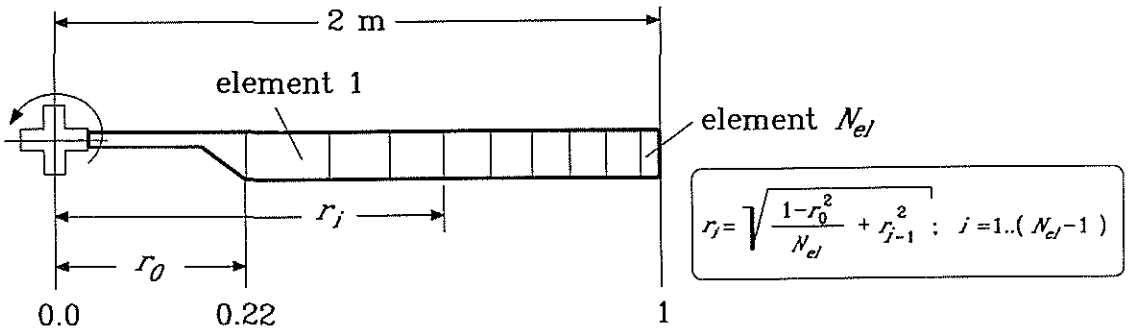


Fig. 3. Blade discretesation of the model rotor.

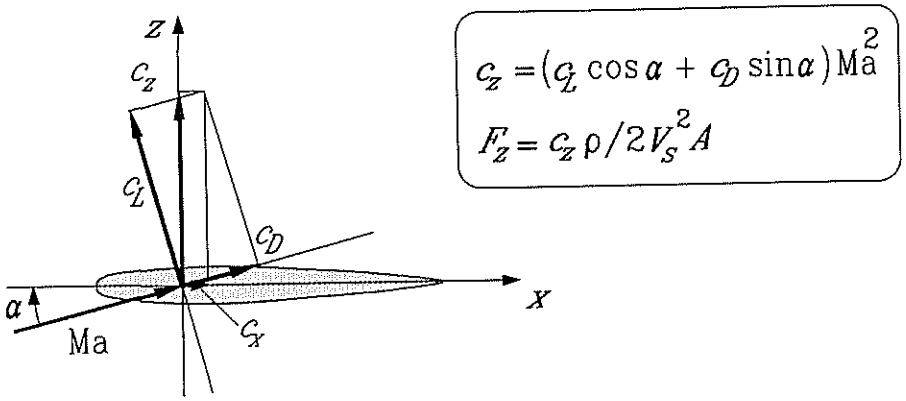


Fig. 4. Definition of aerodynamic coefficients (Example for Ma=1).

A lot of emphasis was put on the calculation of the downwash distribution over the rotor disc. Because the well known linear inflow variation formulated by different authors does not achieve a sufficient estimation of higher frequency components in the downwash distributon, the Mangler-Squire model, [5], is used in the simulation program. The original derivation of the Mangler formulae is based on a typical pressure distribution as depicted in Fig. 6. Baskin et al. [6] have shown that the Mangler equations are also achieved with an undistorted wake model when an average is taken over all blades and one rotor revolution. This means, that local effects like blade vortex interaction or a dependency on the number of blades can not considered by this model. Nevertheless this model leads to a good first estimation of the downwash velocities in the most important range of trimmed flight conditions.

In a real-time simulation a continuously time discrete computation of the dynamic balance has to be performed. That does not allow any iteration processes where the amount of computation steps depends on the convergency of a formula. In the case of helicopter simulation one must avoid a thrust-downwash iteration which is common practice in most non-real-time programs. Here we use a dynamic inflow model for the mean value of the down-wash which is formulated as a first order filter. Filter input is the actual mean velocity which balances the aerodynamic forces

from the blade element calculation. Then the filter output is valid for the next simulation step. A validated model for estimation of the time constant is still subject of research, [7]-[9]. We chose 0.1 s, which gives a smooth convergency after a step input in the blade pitch.

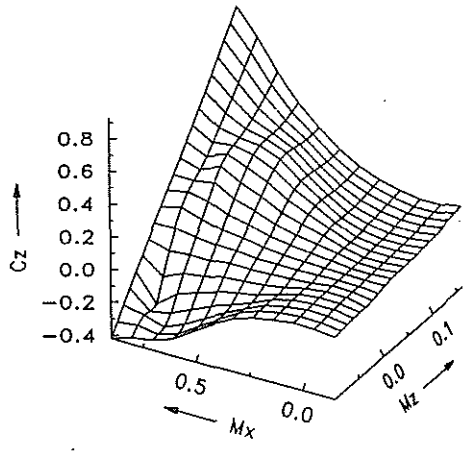


Fig. 5. Lift coefficient vs. Ma_x and Ma_z.

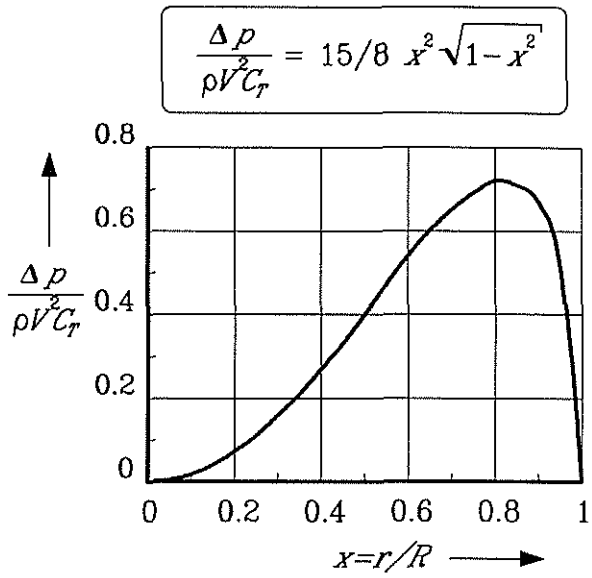


Fig. 6. Mean value of Mangler pressure distribution.

As mentioned above the new simulator has to cover the real-time conditions given by the ROTEST. This model operates with 17.5 Hertz rotor revolution. The frequencies of the model range up to the second lead-lag mode which lies above the 5/rev. This is more than 90 Hertz. It is well known that, in an accurate time integration procedure, the step width depends on the highest frequency in the model and on the stability of the integration algorithm. Basic research was conducted on integrating the equations of motion on a parallel processing system in real-time, [10]. Finally a one step predictor-corrector procedure was chosen which operates with 6 degree step size or 60 steps per revolution, so that the highest frequency component is evaluated with 12 steps per period. But in terms of computation steps per second one has 60*17.5 Hertz equal 1050 Hertz. So it is necessary to calculate the complete mathematical model described above in less than 952 μs.

Before we show how this can be solved, it is necessary to give some information about the newest computer technology, because a successful realization of the discussed simulation goes hand in hand with the knowledge about this technology.

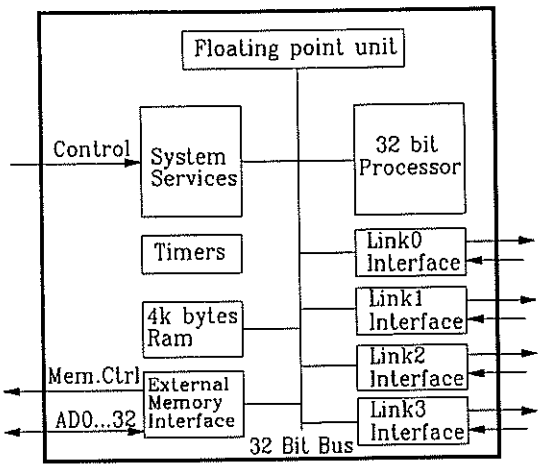
3 OCCAM AND TRANSPUTER

3.1 Philosophy

A system in the 'real world' can be described as a set of processes which work in parallel and exchange information between them. These processes are local and exchange their information only with neighbouring processes. A good approach for numerical simulation would be, to have the same set of information exchanging processes on a computer, too. This kind of mapping of communicating parallel processes in the real world to communicating parallel processes in the computer would provide a consistent relationship between the real world model and the realization in the numerical world inside the computer. Unfortunately traditional computers do not match the basic requirements of this approach i.e.

- parallelism / quasi-parallelism
- communication
- locality

in a sufficient manner. In addition, most of the traditional languages are not developed to do this job. Although most computers allow parallel communicating processes on one machine or even two, their operating systems, schedulers, semaphore-techniques etc. are the bottleneck for real-time simulation. They need a lot of code and a lot of time, because these computers have been optimized to do one job at a time. Parallelism has been introduced by software overhead to these machines, but the basic concept of the machines has not really been influenced by realtime simulation requirements.



Performance :	T800/20 MHz	T800/30 MHz
Bandwidth internal RAM	80 MByte/s	120 MByte/s
Bandwidth external RAM	26.6 MByte/s	40 MByte/s
Link speed per direction	800 kByte/s	800 kByte/s
On Chip RAM	4 kByte	
External RAM	4 GByte	

Fig. 7. Architecture of a TRANSPUTER T800.

For example, although a VAX 8600 is a very fast computer, it would not produce sufficient throughput if running, say 1000, parallel processes, communicating to each other. The machine has been designed for doing one job fast, and all the supported real-time systems and parallel programming tools can only do their job by software overhead. But the machine itself is a single task machine. Even special simulation machines only fulfil some numerical requirements by optimizing special hardware parts like table look-up, or special word lengths for integer computation, but not the structural requirements.

What we need for fast real-time simulation of complex systems is:

a) significant improvement of performance by having a lot of physical computing units,

b) moderate price of the units because we need a lot of them,

c) a good balance between performance, memory capacity and communication capability of each computing unit.

d) configuration capability of the units in order to have a consistent model,

e) a language allowing a simple and clear description of the configuration and communication of such multi-processor systems.

The requirements a) - d) are fulfilled by the TRANSPUTER, a new processor which has been developed by INMOS, funded by the European ESPRIT project. The high level language OCCAM, which has been designed by the same company, focuses on the requirement e).

3.2 TRANSPUTER

The TRANSPUTER (Fig. 7) is a Von Neumann machine with a built in floating point unit and 4kB on-chip RAM. It has been optimized to run OCCAM programs. Two special features make it an ideal machine for parallel processing:

- *Hardware scheduling:* All scheduling between processes is done in hardware. No operating system to schedule processes is needed. All this is done inside the transputer and is complete transparent to the programmer. Scheduling takes only some microseconds if, e.g. a process becomes ready to run.

- *Communication links:* Each transputer has four bidirectional serial 20 Mbit/s links, each working with a separate link engine using DMA. To send, e.g. 1000, words via a link to another TRANSPUTER, the user only has to start the transfer and nothing else. If the receiving TRANSPUTER is not ready to receive the message, the sending process is descheduled automatically until the message has been received. Even if the processor is sending and receiving on all four links simultaneously with full speed, the internal bandwidth guarantees, that the CPU's work goes on. As we will see later, the hardware links correspond to logical channels between OCCAM processes.

- *Performance:* Fig. 8 shows some benchmarks of the TRANSPUTER and its competitors. The Whetstone benchmark is a typical sequential program. So the TRANSPUTER doesn't work in the parallel world it has been designed and optimized for.

	T800/20		T800/30	
	single length	double length	single length	double length
add	350 ns	350 ns	233 ns	233 ns
sub	350 ns	350 ns	233 ns	233 ns
multiply	650 ns	1050 ns	433 ns	700 ns
divide	950 ns	1700 ns	633 ns	1133 ns
	10 MIPS and 1.5 MFLOPS		15 MIPS and 2.25 MFLOPS	

Whetstone - results

processor	clock	Whetstones/sec single length
INTEL 80286/80287	8MHz	300k
IMS T414-20	20MHz	663k
NS 32332/32081	15MHz	728k
MC 68020/68881	16/12MHz	755k
Fairchild Clipper	33MHz	2220k
IMS T800-20	20MHz	4000k
IMS T800-30	30MHz	6000k
ROLM HAWK32		1500k
VAX11/780 with FPA		1083k
MVII with FPA		925k

Fig. 8. Whetstone performance of different processors [from 11].

3.3 OCCAM

The basic concepts of OCCAM have been proposed by Hoare, [12]:

- at runtime, a process is loaded and fixed on its processor,

- at runtime, processes can not be created, i. e. all processes of the system exist when the program starts, whether they do anything or not,

- processes only work with local memory,

- a process communicates to another via any kind of communication network.

OCCAM provides the following language elements to achieve the above rules :

- processes
- constructs
- procedures
- communication .

Processes may consist of other processes. The atomic process is SKIP which does nothing. The assignment (f.e. a:=b) is a process, too. The constructs WHILE, FOR, IF etc. are well known from other languages. But there are some additional new constructs like SEQ, PAR and ALT.

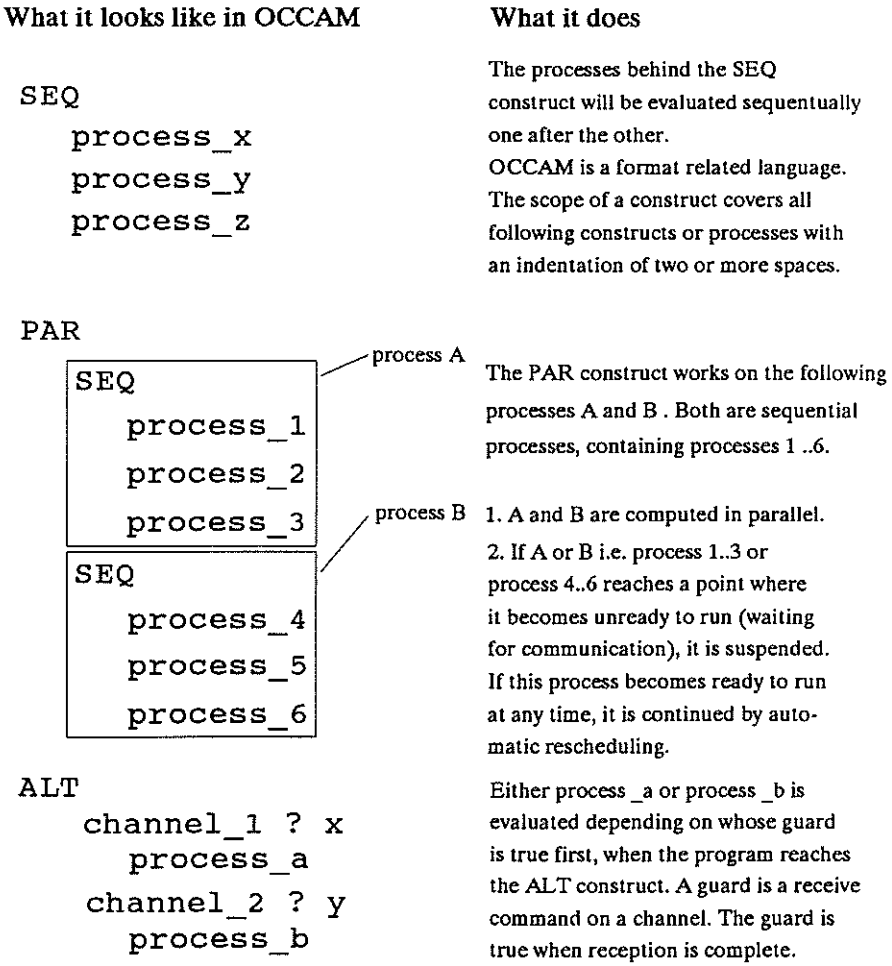


Fig. 9. OCCAM language constructs.

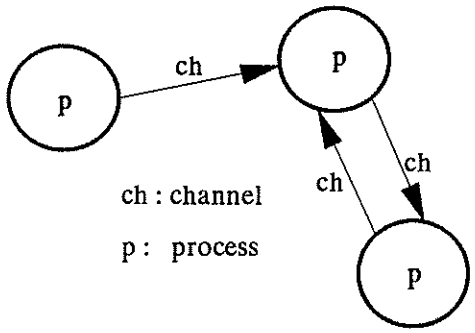
Fig. 9 shows what constructs in OCCAM look like and what they do. These are the important basic language elements to formulate sequential, parallel, or alternative processes. OCCAM is a format related language, i.e. a construct works on all following processes which are indented more than the construct itself.

3.4 Communication

The most important feature of OCCAM is the communication between processes. Two processes communicate if one process sends a message on a channel and another receives a message on the same channel.

```
First process:     channelname ! message.to.send
Second process:    channelname ? message.to.receive
```

If both processes above work in parallel, they communicate if both reach their send respective their receive point. If one reaches this point first it waits until the other one reaches its respective point. Of course both processes are de-scheduled automatically by the processor if another process is ready to run. This is done in microseconds. The channel 'channelname' is a logical channel, connecting two processes. Fig. 10 gives the rules of the process-channel concept of OCCAM.



- Processes contain the activities in OCCAM
- Channels are bi-directional data links between two processes
- Processes exchange data via channels.
- A connection is established between two processes if a process sends and another process receives on the same channel
- There is no explizit synchronisation between processes. Synchronisation is done implizit by the channel protocol.

Fig. 10. OCCAM's process-channel concept.

3.5 TRANSPUTER-Networks

If several processes run in parallel on one machine, of course they can only run quasi-parallel. If it is nessesary to run these processes faster, they can be placed on several TRANSPUTERs. Without modifying the source code of the program the same processes can run, each on a different TRANSPUTER, by connecting the hardware links of the TRANSPUTER and placing the logical channels on physical links. All placing is done outside the source code. Therefore a system of hundreds of processes can be tested numerically on one machine and then be placed on a lot of TRANSPUTERSs in order to improve the performance. Fig. 11 gives an example for placing channels and processes.

It should be noted that, if all communication between the processes has been defined and it has been shown that the program runs without deadlock, the processes can be modified and developed independently. The programmer modifying the process only has to be aware, not to change the interface (e.g. channels and channel protocols) w.r.t. to the rest of the program, The first step in developing an OCCAM program is therefore to evaluate a communication model which represents the problem's structure. After this, the different processes are evaluated.

TRANSPUTERs can be connected in different network structures. Because each problem has an optimal network structure, one has to decide, which structure provides the best 'fit' to the dedicated requirements. Basic structures are, for example :

data flow processing	--> pipeline
2-D-Euler	--> grid
minimum distance between 2 processors	--> hypercube
problems with feedback	--> tree

etc. Fig. 12 shows some of these structures.

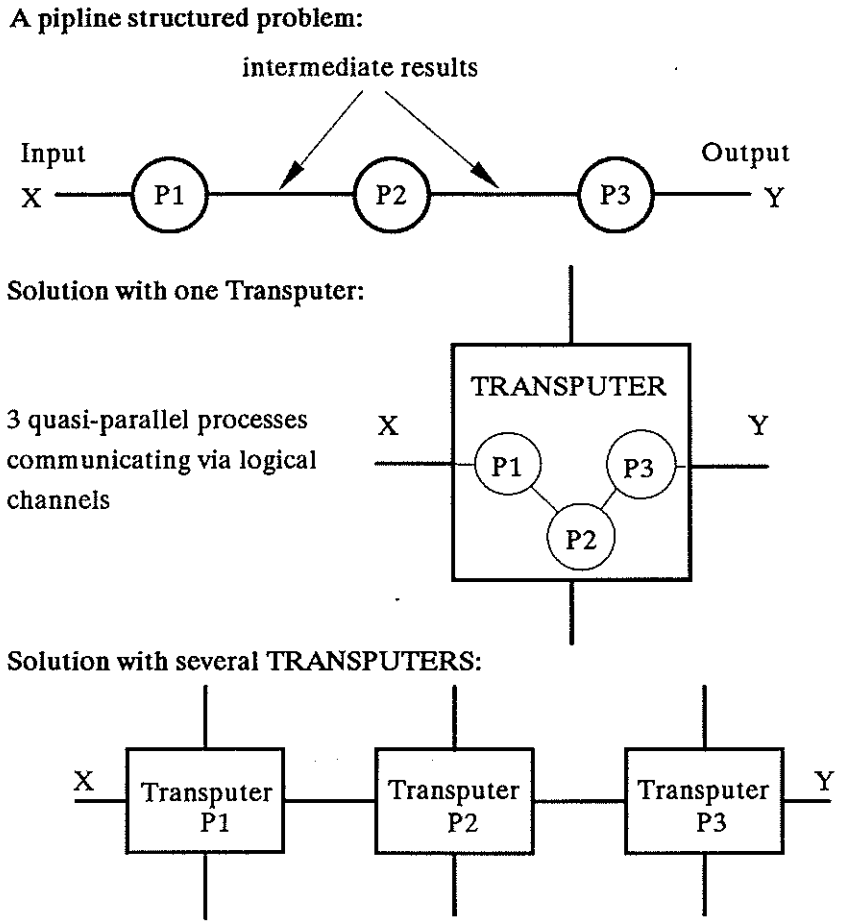


Fig. 11. Example for different hardware solution for the same process model.

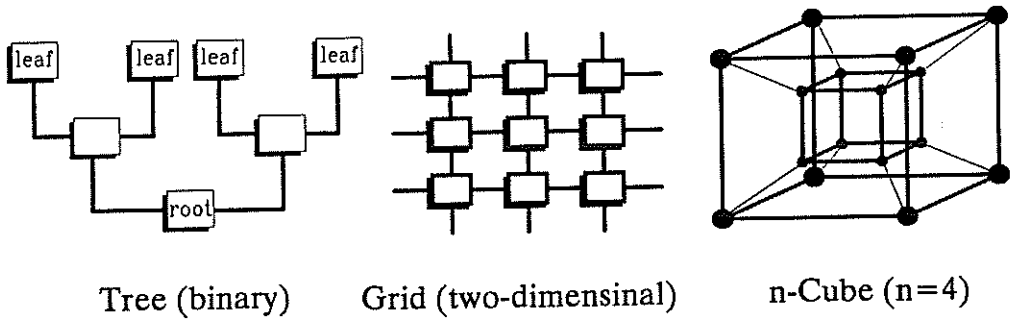


Fig. 12. Three possible network structures for TRANSPUTERs.

4 DESCRIPTION OF THE REALIZED SYSTEM

4.1 Selection of a numerical integration method

The mathematical model of a technical system is defined, in general, by a set of partial differential equations. By describing the movement of the rotor system with elastic blade bending modes, a set of coupled ordinary differential equations is derived. This initial-value problem can be formulated with difference equations and is solved by integrating the equations of motion in the time domain. Therefore, a suitable numerical integration algorithm is required. Some points have to be considered with the choice:

- *realtime application*
  This excludes all algorithms with variable stepsize, because all output signals have to be generated with equally spaced time steps. The required calculation time for one integration step has to correspond with the fixed stepsize.

- *change of input signals during the running of the program*
  All predictor-corrector methods of higher order use results which belong to the past, i.e. to a period before the jump in an input signal occured. Thus, all changes of the input signals are filtered. This is not desired. It can also be shown, that these methods are less stable, compared with other algorithms, when they are applied to the highly nonlinear simulation of the rotor system.

- *numerical stability, accuracy and parallel processing*

The numerical integration of the initial-value problem for an ordinary differential equation by finite differences is a sequential calculation. This means, that the approximation to the solution of an ordinary differential equation, obtained by methods like Runge-Kutta, evolves one point at a time. The solution at each new mesh point is a prescribed function of the values of the solution at certain previous mesh points. The Runge-Kutta algorithm has one great disadvantage. It needs four evaluations of the aerodynamical function for one integration step. A predictor-corrector algorithm is able to handle changing input and output signals in a quarter of that time. Its operation is shown in Figure 13. This first order predictor-corrector algorithm therefore seems to be best suited for the parallel processing in real-time applications. Its numerical stability is very good, but its integration error exceeds that of Runge-Kutta. Nevertheless one can obtain sufficient agreement of both methods, [10].

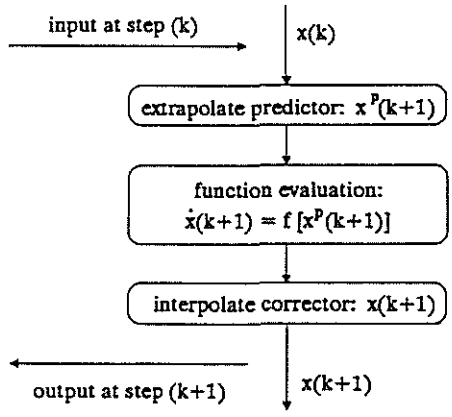


Fig. 13. Principle of the integration procedure.

4.2 Description of the parallel running tasks

The distribution of the mathematical description into parallel running processes is a result of the chosen integration algorithm and the possibility, of splitting the simulation into local physical effects. The distribution is done in two ways:

- tree-like structure

Each rotor blade can be modelled seperately. Therefore, four processes can run in parallel, each containing the integration algorithm and the function evaluation. Describing the complete aerodynamical effects of the blade, the function evaluation time is much larger than the linear-combination time associated with any given integration algorithm. Since local physical evaluations are carried out, each part of the blade can be described in parallel to the other parts. The TRANSPUTER is provided with four links. In a tree-like structure one link is needed for the connection to the upper level and three links can be used for the leaves.

This knowledge, combined with some runtime examinations, leads to splitting the blade into nine elements, that means nine processes, of which always three are evaluated on one TRANSPUTER. E.g. from the 'AERO'- TRANSPUTER in Fig. 14, one is always describing the tip part, one the middle part, and one the root part of the blade. The numerical integration algorithm is run on the 'ROOT'- TRANSPUTER. Since all local physical effects are summarised on this TRANSPUTER, it represents the connection of the blade to the rotor head. Thus, the tree-like arrangement of the TRANSPUTERs in Fig. 14 represents the physical structure of the rotor. This leads to another advantage: On all twelve 'AERO'-TRANSPUTERs the same program can be run. The same applies to the four 'ROOT'-TRANSPUTERs.

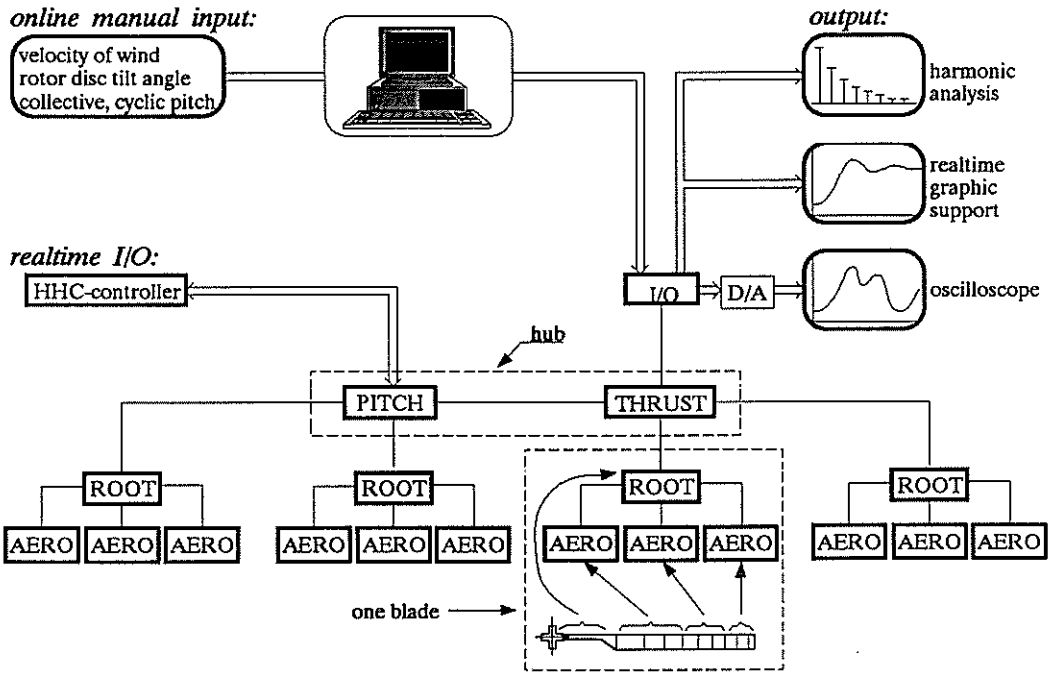


Fig. 14. TRANSPUTER network for the real-time simulator.

- pipeline structure

The above mentioned summarizing of the forces and moments on the 'ROOT'-TRANSPUTER can be done while the 'AERO'-TRANSPUTERs evaluate the function. This takes approximately the same amount of time. Thus the function evaluation at the mesh point k runs in parallel to the summarizing of the forces at the time step (k-1). The consequent

continuation of this idea leads to Fig. 15. At any given moment k on the horizontal axis (time-axis), quite a lot of preparatory work and work which belongs to older mesh points is done. (Figure 15 just shows the main tasks of the TRANSPUTER network.) On the other hand, if one follows the information flow which belongs to any given mesh point k, dark line in Figure 15, the pipeline structure of the information processing can be seen. The following advantages are gained:

- the sequence 'predictor-function evaluation-corrector' can be repeated at a high frequency, this determines the resulting stepsize,
- for one integration step just one data transmission in each direction between the TRANSPUTERs is necessary. This point is very important in speeding up calculations through parallel processing.

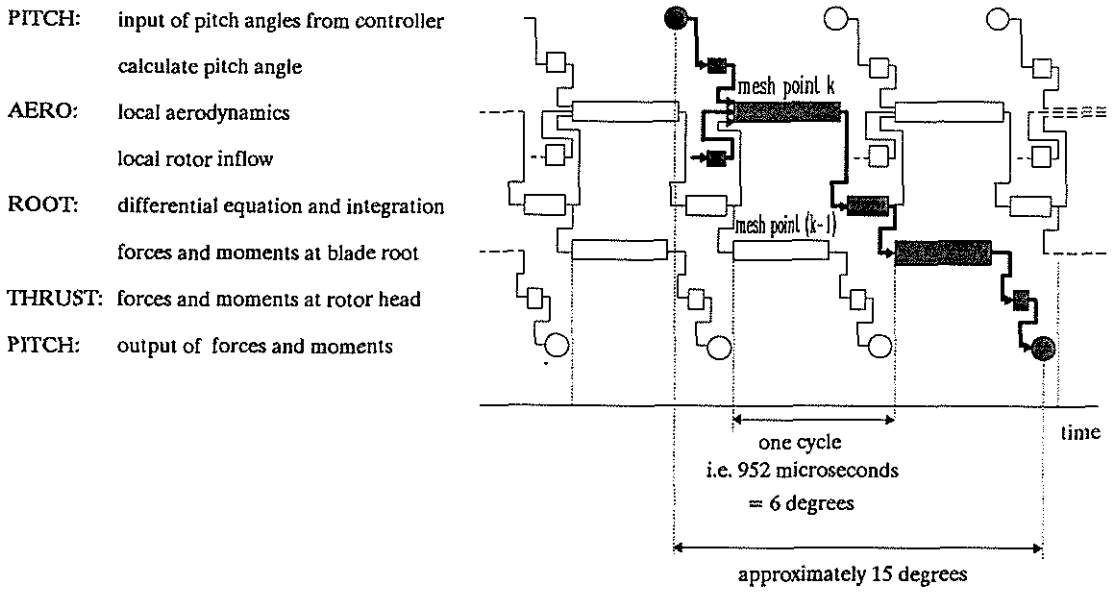


Fig. 15. Time diagram of the simulation main tasks.

This method involves some disadvantages. The high frequency integrating satisfies the requirements of the system dynamics. But the time delay associated with the pipeline gives a phase lag between input (blade pitch) and the rotor reactions. The delay is small, about 1 ms or 9 degrees rotor azimuth. In the case of 5/rev this value increases to 45 degrees. There are two methods of considering this effect in a correct way. First, a model of a virtually digital data acquisition system can be implemented where the time delay is considered as a part of the related transfer function. Or, second, the computation must be accelerated. This will be possible very soon because the new version of the TRANSPUTER works at 30 MHz. Nevertheless two points have to be considered:

1.) The observer wants to see input and output signals which belong together. Therefore, input signals which act as output signals as well, e.g. the pitch angle, have to be kept until the responding output signals are calculated. Then they may be sent to the output system.
2.) A special view has to be given to the output signals which are gained on the upper levels of the tree structure and subsequently are fed back to the bottom leaves of Figure 14. This effects just one signal, describing the rotor inflow. Since it is filtered by a lowpass, only a time constant has to be modified.

With reference to Figure 14 and Figure 15, the simulation of the rotor system can be split up into several tasks which are related to the executing TRANSPUTERs:

'AERO': - local pitch angle and local rotor inflow
- local evaluation of the aerodynamics for three parts of a blade

'ROOT': - integration algorithm for one blade
- velocities of a blade
- forces and moments at the blade root

'PITCH': - pitch angle and differentiation of pitch angle for all blades
- communication with controller, if connected
- harmonic analysis for controller

'THRUST': - summarizing forces and moments of all blades
- Mangler downwash model - global part of calculation
- working up input signals
- communication with I/O system
- generation of real-time events with internal timer

In Fig. 16 and 17 some results of the simulation model are plotted. Fig. 16 shows the case of a step response at hovering coditions where the collective pitch was suddenly increased by one degree. The plot gives the time history of the blade tip deflection, the thrust and the mean induced velocity. A typical steady state flight codition at μ=0.3 is shown in Fig. 17. Here the blade pitch angle, tip deflection and the corresponding hub reactions F_z, M_x and M_y are plotted over three rotor revolutions. The numerical accuracy of the real-time model was compared with a conventional simulation on a mainframe computer, coded in FORTRAN. The results are in agreement by more than five digits, if the identical model was implemented.

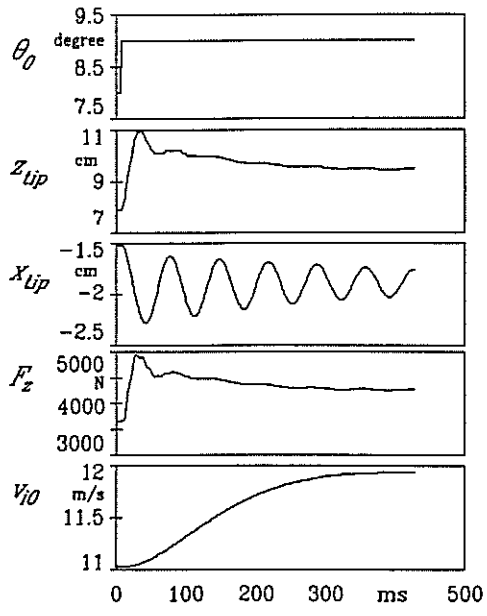


Fig. 16. Rotor response to collective step input.

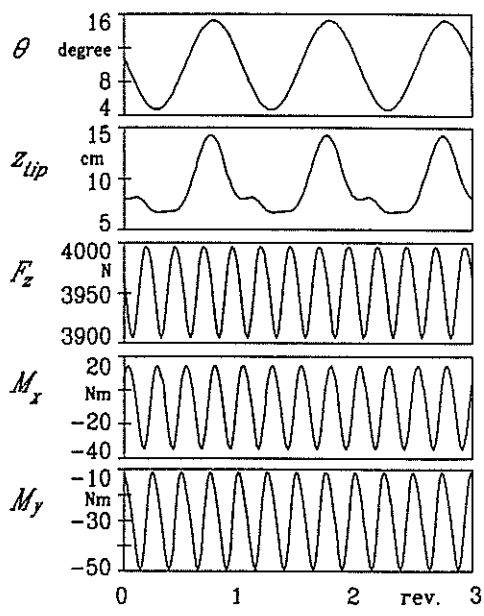


Fig. 17. Rotor components at μ=0.3.

4.3 I/O systems

The I/O system works as the interface between the simulated model and the real world. Normally the ROTEST is operated by specially trained personel. The control inputs are set manually while the instruments and displays are observed for correct (or better: expected) reaction of the model. The same procedure is implemented in the simulator. From a PC the five control parameters tunnel speed, shaft tilt, and collective and cyclic pitch can be manually adjusted. The simulator reaction can be seen on oscilloscopes (32 channels) or, with enhanced features, on a realtime graphic support. Here, up to six selectable signals can be displayed, either as discrete value 'distribution' in the time domain, or as distribution at a given azimuth step in the space domain. Thus makeing it possible to look at e.g. blade deformation, blade loadings, velocity distribution etc. Fig. 14 can only sketch these possibilities. Another way of data reduction is the harmonic analysis of the signals. This is performed optionally by the 'PITCH'-TRANSPUTER associated with small additional penalty of delay (< 0.2 ms). The algorithm implemented here is based on a recurrent calculation of the cross correlation as derived in [13].

5 HARDWARE

In the previous four chapters we have described the model and the performance of the simulator. At this point the hardware expense should be mentioned. Fig 18 shows a basic module with one T800 and one megabyte memory. This module contains all parts of a computer, and has its performance listed in Fig. 7 and Fig. 8. For realisation of the simulator we have connected 18 modules like this in a way shown in Fig. 14. At least one additional module is used in the PC for software development, one module for D/A controlling and four are used in the realtime graphics support. Four of the basic modules can be placed on one card, Fig. 19. So it is possible to house all the computing power, inclusive the I/O, in a 19" cabinet as shown in Fig. 19. The power consumption is less than 80 Watts. For software development a standard PC/AT is very suitable.

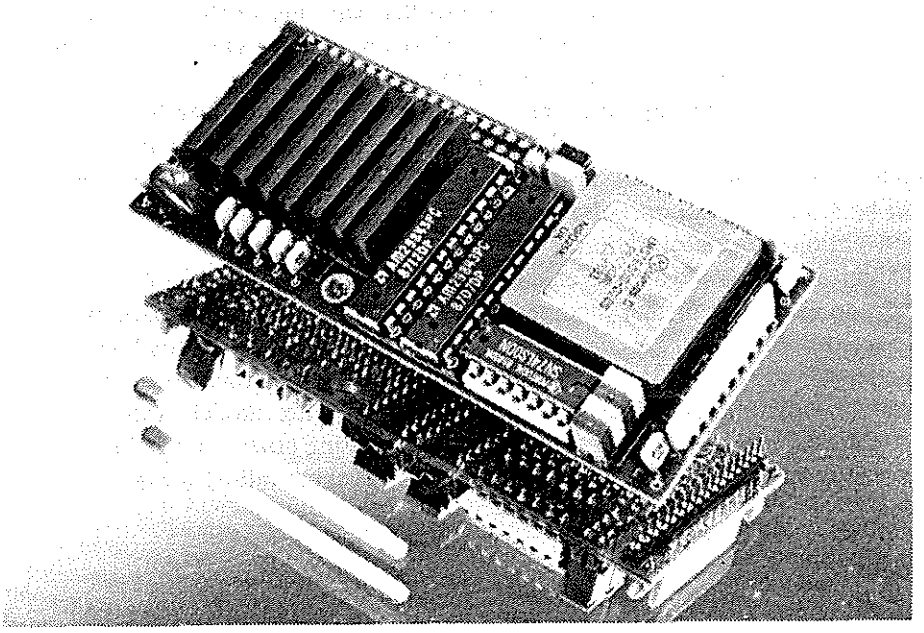


Fig. 18. Basic module (100mm x 42mm) with T800 and 1 Mbyte memory.

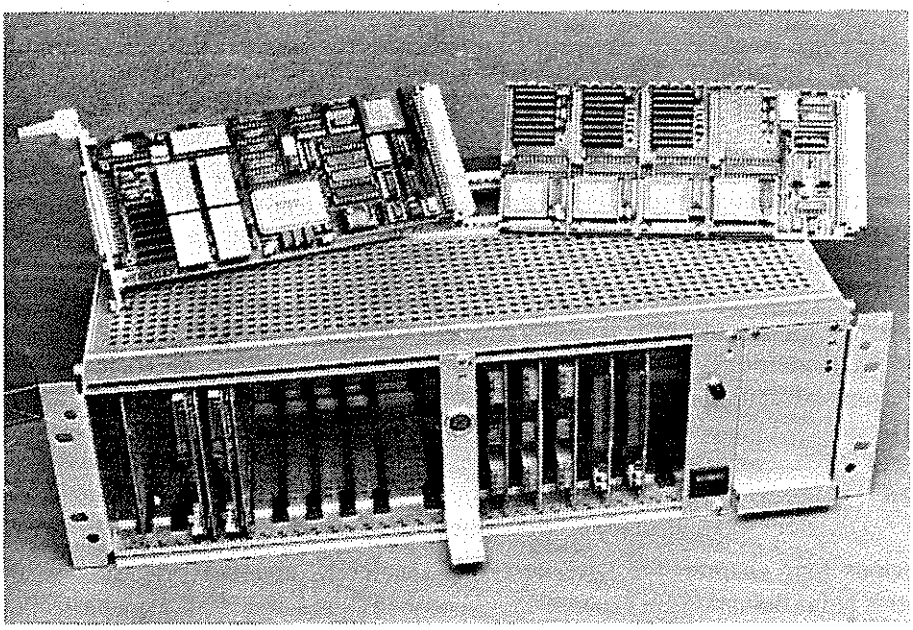


Fig. 19. Complete simulation computer including digital and analog I/O and power supply.

6 APPLICATIONS

The first application of this simulator was the testing of a new HHC-controller, which works in the frequency domain. It is a enhanced version of the system described in [13]. The controller is based on a TRANSPUTER, too. Therefore a connection to the simulator can be easily performed via a TRANSPUTER-Link. At date a different controller, working in the time domain, is in development. With the simulator a powerful test bed is provided to check this system under realistic conditions without the cost and risk of a wind tunnel test.

But not only the true real-time applications with hardware-in-the-loop can be addressed. The implemented rotor model reflects the conditions of the ROTEST model rotor. It is now possible to change physical parameters of the model in real-time while the system is running. The parameters can be selected by an optimisation algorithm. Thus an optimisation process can be handled as a conventional regulator option.

For future application it is possible to enhance the system capabilities in order to simulate a complete helicopter. Therefore further modules can be added to simulate the parts fuselage, tail rotor etc. as depicted in Fig. 1. On the other hand, a real helicopter operates at a lower frequency range as the model rotor (scale 1:2.5). So it is possible to extend the mathematical model without loss of real-time conditions. Nevertheless, much more complex models can be taken in consideration than it was ever possible in the past. The computing power we need to run in real-time grows with the complexity of the model and thus no physical limitations are obvious today.

7 CONCLUSIONS

It was shown that a new technology leads to a new design in real-time simulation. The consistent mapping of real world processes into a corresponding hardware-software system based on TRANSPUTER and OCCAM provides a clear and, this is the most important feature, easily extendable data processing structure with real-time capabilities. The hardware expense is as low

as possible. Programming of the processes is done in a high level language (like Pascal or C) which incorporates the constructs for parallel programming. Because the software development is independent of the amount of processor modules in the target system, only the logic and physics of the real world processes must be met by the program.

The computing speed is an additional and independent degree of freedom, which can be modelled by the network structure and the amounts of computing elements. The discussed real-time simulator clearly shows a lot of advantages in performance, modular design, and expense. The only disadvantage at date is that there are only four links available. Thus possible network structures are restricted. But this limitation will be dropped with the next generation of TRANSPUTERs.

1) H. Huber, P. Krauspe, The Role of Simulation. Helicopter Aeromechanics. AGARD Lecture Series No. 139, Braunschweig, 2-3 May 1985

2) P. Saager, W. v. Grünhagen, Real Time Helicopter Simulation. INFAUTOM International Symposium on Simulation, Toulouse (France) 2-3 March 1989

3) W. v. Grünhagen, Bestimmung der gekoppelten Schlagbiege-, Schwenkbiege- und Torsionsschwingungen für beliebige Rotorblätter mit Hilfe der Finite-Element-Methode. DFVLR-IB 154-80/21, Braunschweig, 1980

4) B. v. d. Wall, Verfahren zur Berechnung der instationären Luftkräfte in der Hubschraubersimulation. DFVLR IB 111-87/38, 1987

5) K. W. Mangler, H. B. Squire, The Induced Velocity Field of a Rotor. R.&M. No. 2642, 1950

6) V. E. Baskin, et al., Theory of the lifting airscrew. Moscow, 1973, NASA TT F-823, 1976

7) D. M. Pitt, D. A. Peters, Theoretical Prediction of Dynamic-Inflow Derivatives. Sixth European Rotorcraft Forum, Paper No. 47, Bristol 1980

8) T. N. Chen, W.S. Hindson, Influence of Dynamic Inflow on the Helicopter Vertical Response. NASA TM 88327, 1986

9) S. Houston, Identification of a 3 DOF Body/Coning/Inflow Model in Hover. HTP-6 System Identification Workshop, March 21-25, 1988, Bedford.

10) C.-H. Oertel, Untersuchungen zur Echtzeitsimulation eines Hubschrauberrotors auf einem Parallelrechnersystem. Diplomarbeit am Institut für Regelungstechnik der TU Braunschweig, Braunschweig 1988

11) IMS T800 TRANSPUTER. INMOS 72 TRN 117 02, April 1987

12) C. A. R. Hoare, Communicating Sequential Processes, Comm. ACM 21 (8), 666-677 (1978)

13) G. Lehmann, R. Kube, Automatic Vibration Reduction at a Four-Bladed Hingeless Model Rotor - A Wind Tunnel Demonstration. Proceedings of the 14th ERF, Mailand, Paper No. 20, Sept. 1988