Predictive design analysis utilizing ansys in an internet environment

Eriksson, Martin; Burman, Åke

2000

PREDICTIVE DESIGN ANALYSIS UTILIZING ANSYS IN AN INTERNET ENVIRONMENT

Martin Eriksson and Åke Burman

Division of Machine Design, Department of Design Sciences,
Lund Institute of Technology at Lund University, Sweden

ABSTRACT

During recent years the increasing power of computers and communications has led to the development and use of more advanced analysis techniques. Design of Experiment (DOE) has been used together with Finite Element (FE) analysis software such as ANSYS, see e.g. Johnson and Heald (1998) and Eriksson et al. (1998). Predictive Design Analysis (PDA) is an mechanical engineering design, or design for short, approach in which the final behavior of a product is predicted, at least to some extent, by combining information from different analysis techniques, see Eriksson and Burman (1999). This work presents an implementation of PDA consisting of DOE together with ANSYS.

The implementation is developed as a World Wide Web (WWW) Client/Server application. The user interface is constructed as a standard HTML page that can be viewed in a standard web. Java scripts and Java Applets are used to verify the data given by the designers or analysts before it is sent to a Common Gateway Interface (CGI) application on the server for evaluation. The result delivered from the CGI application is VRML files of selected responses along with GIF images and data based on the statistical evaluation of the studied responses.

The issue of decreasing the computation time is addressed in this work by running the FE analyses in parallel. The fact that all analyses based on DOE are performed independently of each other makes them well suited for distributed processing, for example on a cluster of computers connected through a network. Studies of the implementation regarding the timesaving of distributed computations are performed and documented. This application serves as a powerful tool for the designer or analyst at all levels of abstraction in the design process, when examining the influence of different design variables on the product studied.

INTRODUCTION

With the increasing power of computers and the development of efficient analysis techniques, the demand for more detailed analyses has grown. In order to satisfy the required level of detail, the analysis models have become more complex and thus more time-consuming. Techniques such as the statistical methods DOE and Monte Carlo simulations have successfully been introduced into the community of FE computing. These techniques organize a set of independent analyses with the emphasis on decreasing the required number of FE analyses to be performed and thus reduce time. The fact that they are independent makes them well suited for performing them with some form of parallel or distributed computing on either a cluster of computers or a Multiprocessor computer. By combining the mentioned statistical methods and parallel or distributed computing, the designer or analyst (hereafter referred to as designer) can take advantage of both the increasing computer power and also the increased level of knowledge that results from utilizing statistical methods.

OBJECTIVE

The objective of this paper is to present an implementation in which a computer network is utilized for performing a set of FE analyses established by DOE. The Client/Server WWW interface and DOE are briefly discussed, together with a thorough evaluation of the current implementation.

THE CONCEPTUAL APPROACH

The conceptual layout of the current implementation is shown in Figure 1. An investigation can be subdivided into several steps as indicated by the digits in Figure 1. In the first step (1) a designer opens up a standard WWW browser and connects to the WWW server and retrieves the client-side user interface. The problem to be evaluated is specified and the analysis data are submitted to the WWW server. The server starts the CGI application that handles the FE analyses. If the analyses are to be performed on a single machine, e. g. on the WWW server itself, they are sequentially started based on the DOE layout. The statistical evaluation is carried out by the CGI application the analyses are completed. When the statistical evaluation is performed the result is submitted back (5) to the client side where the designer can view the result. If, on the other hand, the analyses are chosen to be performed on a cluster of computers, the FE analyses information is passed on from the CGI application to the computing distributor (2) for further processing. The computing distributor establishes the computer resources available and divides up all the FE analyses (3) on the available computers in the network. When all FE analyses have been executed, the computing distributor passes the FE results back (4) to the WWW server where the CGI application performs the statistical evaluation. The result of the evaluation is submitted back (5) to the client side where the engineering designer can view the result. In both cases of evaluation, the results are now obtainable for other designers and design teams from all over the Internet or intranet by connecting (6) to the WWW server and retrieving data belonging to a specific evaluation.

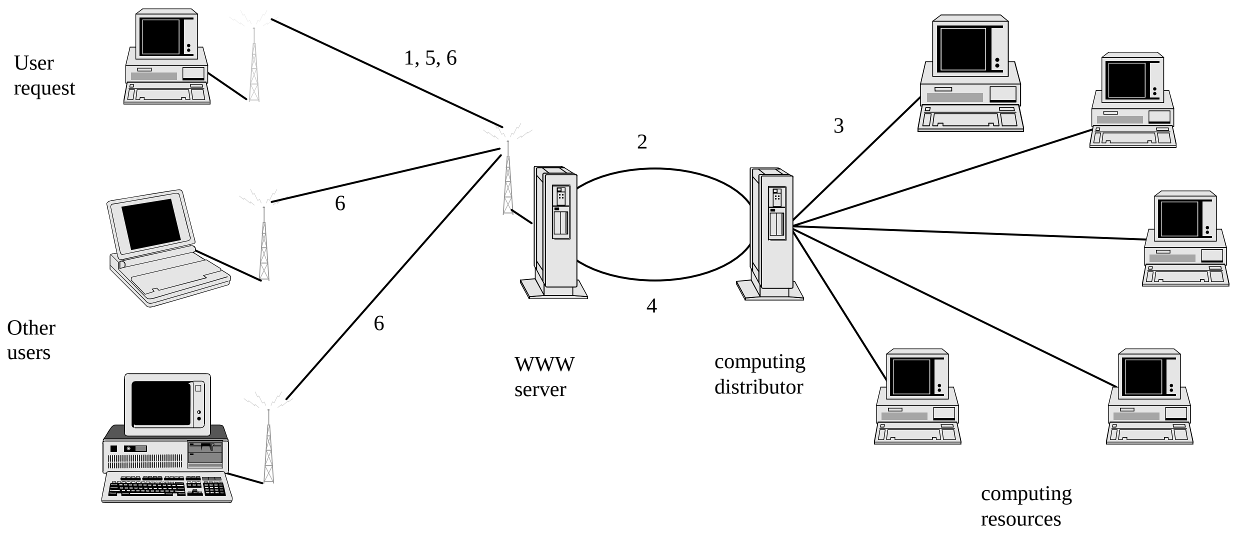


FIGURE 1. CONCEPTUAL LAYOUT OF THE PRESENTED IMPLEMENTATION.

DESIGN OF EXPERIMENTS

In full factorial experiments the response is calculated at every combination of design variable levels. Fractional factorial experiments are based on arrays that define the order in which the design variables should be altered. They are commonly used within DOE to achieve a more time- efficient evaluation. When the number of design variables is increased, the calculation time can be rapidly decreased by the use of fractional factorial experiments. In practice the task of finding a useful suitable array is easily reduced to selecting previously defined arrays that can be found in many reference books, see e.g. Box et al. (1978).

The choice of response function depends on the problem. In structural analysis, for instance, the weight, stresses and displacements can be chosen as response functions, and in contact analysis the contact pressure and penetration may also be interesting. The possibilities are many, and the purpose of the analysis will be the guideline when selecting the most suitable response functions.

Whichever analysis and response functions are chosen, the basic procedure for analyzing the data is the same. First all essential data must be given by the designer. The design variables are then used in the usual FE environment as parameters. The modeling and equation solving phases must be included in a sequence where the design variables are altered according to the DOE layout. The chosen response is also collected within the sequence for further statistical evaluation after all design variable configurations are performed.

BASICS WITHIN PARALLEL COMPUTING

The general idea of parallel processing is to utilize many small tasks in order to solve one large problem. The acceptance of parallel computing has mainly been facilitated by two major developments: massively parallel processors (MPP) and distributed computing. MPPs are currently the most powerful computers in the world. These machines combine up to thousands of CPUs in a single large cabinet connected to thousands of gigabytes of memory. MPPs have massive computational power and are used to solve challenging problems such as analyzing complete cars in crashes — see e.g. Marczyk (1999).

Distributed computing collectively uses a set of computers in a network to solve a single large problem. By using many general-purpose workstations interconnected by high-speed local area networks, the combined computational resources may exceed the power of a single high-performance computer. To achieve unequaled computational power, several MPPs can be combined using distributed computing.

In an MPP, every processor is exactly like every other in capability, resources, software, and communication speed. This is not necessarily the case on a network. The computers on a network may have different types of heterogeneity. The network can include several different types of architectures such as PC, Unix workstations, and shared-memory multiprocessors and vector supercomputers. Computational speed, machine load and network loads are three additional factors that must be considered when dealing with network computing.

One thing shared in common by distributed computing and MPP is the notion of message passing. Data must be exchanged between cooperating tasks performed with parallel processing. The message-passing model is the paradigm of choice, from the perspective of the MPP that supports it, as well as in terms of applications and software systems that use it. The most commonly used methods of parallel computing are P4 , see Butler and Lusk (1993), Express, see Flower et al. (1991), Linda, see Carriero (1989), the Message Passing Interface (MPI), see Message Passing Interface Forum (1994) and the Parallel Virtual Machine (PVM), see Geist et al. (1994). As mentioned by Geist et al. (1996) the PVM model is built around the virtual machine concept that provides a set of dynamic resource managing and process control functions. This, along with the fact that PVM has good interoperability between different hosts and different architectures, makes PVM a good message-passing approach when applications are to be run over heterogeneous networks, and thus PVM is chosen to handle the distributing in this work.

Parallel Virtual Machine

PVM can be dated back to 1989 when the development started at Oak Ridge National Laboratory. PVM is a software system that permits a heterogeneous collection of computers networked together to be comprehended as a single parallel computer by an application program. PVM is designed to link computing resources and provide users with a parallel platform for running applications, irrespective of the number of different computers used and the location of the computers. PVM transparently handles all message routing, data conversion, and task scheduling across a network of incompatible computer architectures. The principles upon which PVM is based include the following:

User-configured host pool: The application's computational tasks work on a set of machines that are selected by the user for a given run of the PVM program. Both single CPU machines and hardware multiprocessors may be part of the host pool. Adding and deleting machines during operation may alter the host pool.

Translucent access to hardware: Application programs may either view the hardware environment as an attributeless collection of virtual processing elements, or choose to exploit the capabilities of specific machines in the host pool by positioning certain computational tasks on the most appropriate computers.

Process-based computation: The unit of parallelism in PVM is a task, an independent thread of control that alternates between communication and computation. No process-to-process mapping is implied or enforced by PVM; in particular, multiple tasks may execute on a single processor.

Explicit message-passing model: Collections of computational tasks, each performing a part of an application's workload using data-, functional-, or hybrid decomposition, cooperate by explicitly sending and receiving messages with each other.

Heterogeneity support: The PVM system supports heterogeneity in terms of machines, networks, and applications. With regard to message passing, PVM permits messages containing more than one data type to be exchanged between machines having different data representations.

Multiprocessor support: PVM uses the native message-passing facilities on multiprocessors to take advantage of the underlying hardware.

The PVM system is composed of two parts. The first part is a daemon, which resides on all the computers constituting the virtual machine. The second part of the system is a library of PVM interface routines. It contains a functionally complete repertoire of primitives that are needed for cooperation between tasks of an application. This library contains user-callable routines for message passing, spawning processes, coordinating tasks, and modifying the virtual machine. The original PVM was developed for Unix platforms only, but today there exist other implementations such as Windows PVM (WPVM) running under Windows 95/98/NT on Intel processors. WPVM is built upon the standard features of PVM.

Parallel programming techniques

Parallel computing with respect to process structure can be approached from three fundamental viewpoints, based on the organization of the computing tasks. Within each, different workload allocation strategies are possible.

The first model is often termed "crowd" computing: a collection of closely related processes, typically executing the same code, performs computations on different portions of the workload. This model can be further subdivided into two categories depending on programming layout. A master-slave (or host-node) model is used in which a separate program termed the master is responsible for process spawning, initialization, collection and display of results and the other has the master's responsibilities built into the node (slave) program that initiated the computation.

The second model of parallel computing is termed "tree" computation, where the processes are spawned (usually dynamically as the computation progresses) in a tree-like manner. This model is a natural fit to applications where the total workload is not known prior to the computation.

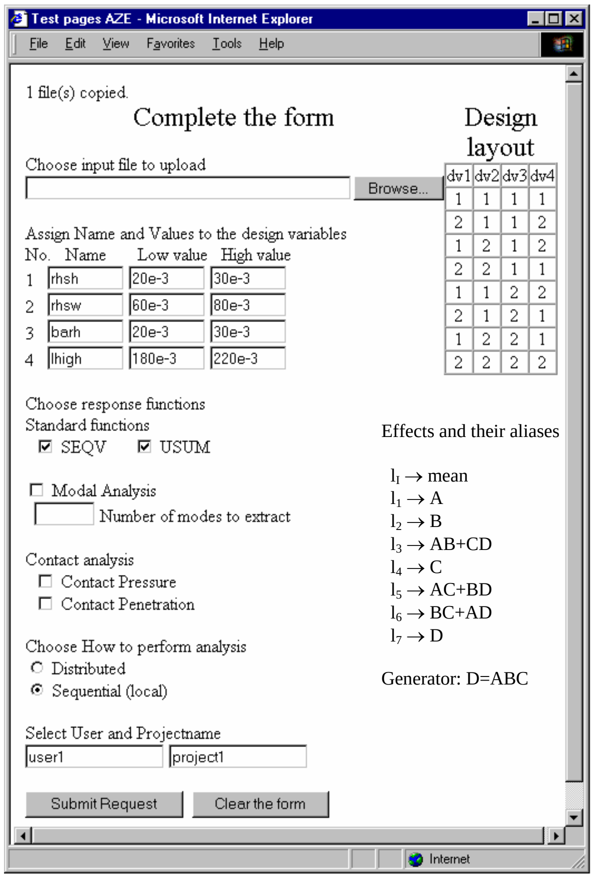
The third and last model can be seen as a combination of the two first models, in which the spawning structure is arbitrary. Which model to use is of course application-dependent and should be selected to best match the natural structure of the paralleled program.

The issue of workload allocation is subsequent to establishing process structure. Two general methodologies are commonly used. The first, termed data decomposition, or partitioning, assumes that the overall problem involves applying computational operations or transformational operations or transformations on one or more data structures. The second, called function decomposition, divides the work according to different operations or functions. Generally speaking, function decomposition can be said to use fundamentally different tasks to perform different operations, and data decomposition uses identical tasks to operate on different portions of the data.

IMPLEMENTATION

The implementation presented in this work is a Client/Server WWW application that combines FE analysis with DOE, in which computing resources within a network are efficiently utilized. The Client/Server application makes use of the advantages of the Internet/intranet in terms of accessibility and flexibility. The WWW server used in the implementation is the Sambar Server 4.2 from Sambar Technologies. This server provides many capabilities to the developer (virtual domains, document aliases, server-side includes, CGI/WinCGI programming etc.). The CGI/WinCGI specification of Sambar Server 4.2 allows server-side applications to be written in almost any language. The CGI application in this implementation is written in C.

Figure 2 below shows an example of the pre-processing user interface, in which the evaluation-input data are to be specified. The Ansys input file is chosen for upload to the server, and the appropriate name and values of the design variables are assigned. The appropriate responses that should be calculated in the analyses must also be defined. Clicking on the submit button sends the data to the server and the CGI application is executed. If the analyses are to be distributed over the network, the master program of the WPVM is started. The WPVM software is built upon the standard features of PVM compiled for the MS Windows operating system. WPVM is compatible with the original PVM, which means that virtual machines can simultaneously be composed of UNIX and MS Windows machines. The master program allocates the necessary or available computer resources and sends data to each slave program that executes the FE analyses.


Test pages AZE - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Browse...
Submit Request
Clear the form
Internet

1 file(s) copied.

Complete the form

Choose input file to upload

Assign Name and Values to the design variables

No.	Name	Low value	High value
1	rhsh	20e-3	30e-3
2	rhsw	60e-3	80e-3
3	barh	20e-3	30e-3
4	lhigh	180e-3	220e-3

Choose response functions
Standard functions
☑ SEQV ☑ USUM

☐ Modal Analysis
Number of modes to extract

Contact analysis
☐ Contact Pressure
☐ Contact Penetration

Choose How to perform analysis
○ Distributed
● Sequential (local)

Select User and Projectname
user1 project1

Design layout

dv1	dv2	dv3	dv4
1	1	1	1
2	1	1	2
1	2	1	2
2	2	1	1
1	1	2	2
2	1	2	1
1	2	2	1
2	2	2	2

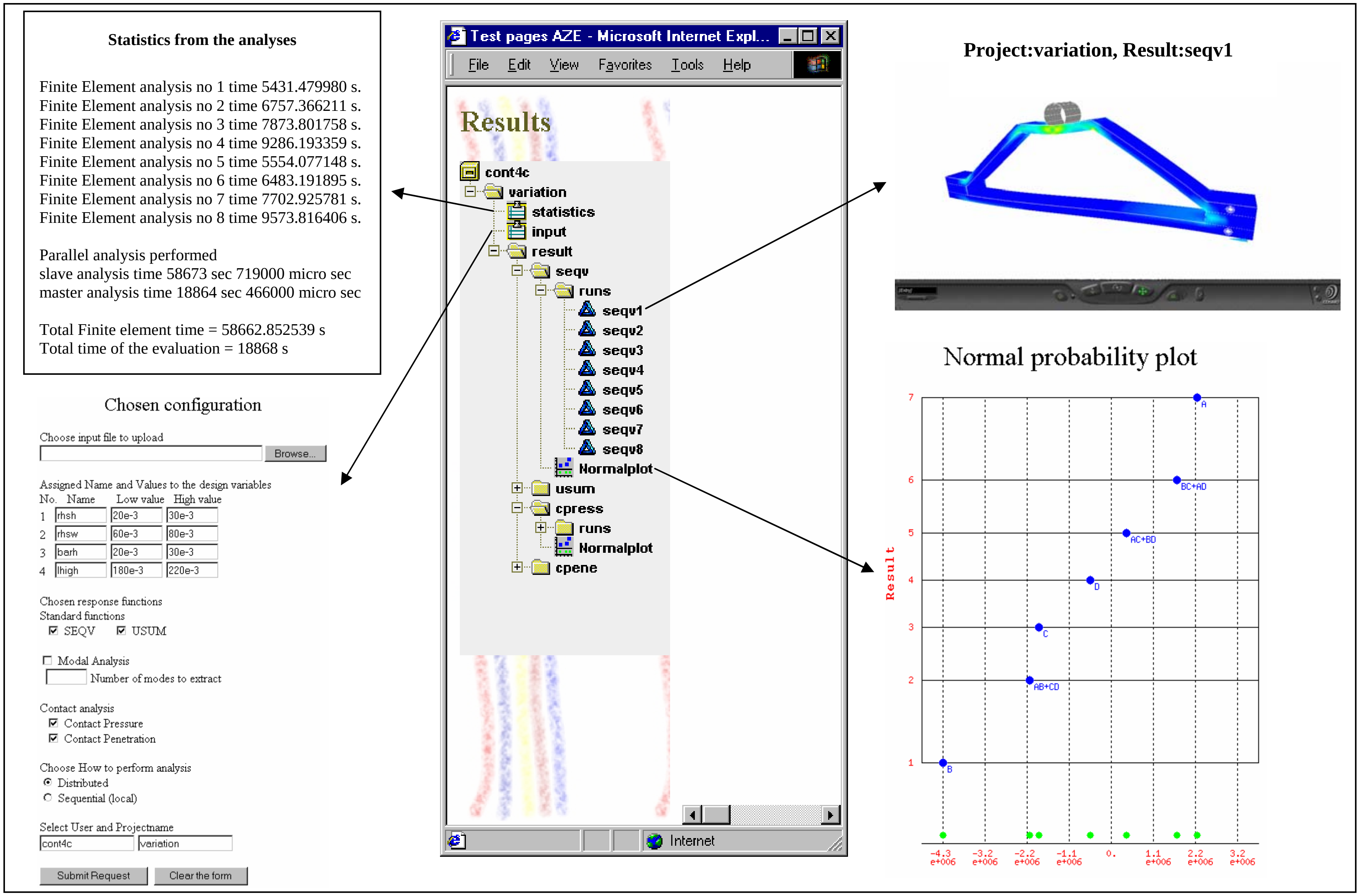
Effects and their aliases

$l_I \rightarrow$ mean
$l_1 \rightarrow$ A
$l_2 \rightarrow$ B
$l_3 \rightarrow$ AB+CD
$l_4 \rightarrow$ C
$l_5 \rightarrow$ AC+BD
$l_6 \rightarrow$ BC+AD
$l_7 \rightarrow$ D

Generator: D=ABC

FIGURE 2. CLIENT USER INTERFACE.

After all the FE analyses are performed, the resulting data are collected by the master and sent back to the CGI application for the statistical evaluation. After the CGI program is finished the server sends the post processing HTML page back to the user that requested the evaluation. A collection of different post-processing data is shown in Figure 3. The central part of the post-processing is a Java applet, see the center picture in Figure 3. The applet serves as a file manager for all the users' studied projects. Every project contains a listing of the chosen analysis data and a presentation of analysis statistics, see the left pictures in Figure 3. There are possibilities to choose other values for the current design variables or to test new design variables. Submitting the new data to the CGI application will extend the Java applet tree with a new project. The applet also contains a subfolder named *result* containing another subfolder for each of the chosen response functions. A response result folder includes a subfolder, named *runs*, with the FE analyses results visualized through Virtual Reality Modeling Language (VRML) files that can be seen in the top right picture in Figure 3. The user has the possibility to rotate, zoom, translate and seek certain model locations of the VRML model with the built-in mechanism in the browser's plugin for handling VRML files. The folder further contains graphical presentations of the statistical evaluations through normal probability plots as can be seen in the bottom right picture in Figure 3.



Statistics from the analyses

Finite Element analysis no 1 time 5431.479980 s.
Finite Element analysis no 2 time 6757.366211 s.
Finite Element analysis no 3 time 7873.801758 s.
Finite Element analysis no 4 time 9286.193359 s.
Finite Element analysis no 5 time 5554.077148 s.
Finite Element analysis no 6 time 6483.191895 s.
Finite Element analysis no 7 time 7702.925781 s.
Finite Element analysis no 8 time 9573.816406 s.

Parallel analysis performed
slave analysis time 58673 sec 719000 micro sec
master analysis time 18864 sec 466000 micro sec

Total Finite element time = 58662.852539 s
Total time of the evaluation = 18868 s

Chosen configuration

Choose input file to upload
[ ] Browse...

Assigned Name and Values to the design variables

No.	Name	Low value	High value
1	rhsh	20e-3	30e-3
2	rhsw	60e-3	80e-3
3	barh	20e-3	30e-3
4	lhigh	180e-3	220e-3

Chosen response functions
Standard functions
☑ SEQV ☑ USUM

☐ Modal Analysis
[ ] Number of modes to extract

Contact analysis
☑ Contact Pressure
☑ Contact Penetration

Choose How to perform analysis
[✓ Distributed]
Sequential (local)

Select User and Projectname
cont4c variation

Submit Request | Clear the form

FIGURE 3. EXAMPLES FROM THE POST PROCESSING USER INTERFACE.

EXAMPLE

The example chosen to exemplify the application is a support arm for a transportation vehicle. The support arm is built up from two RHS profiles made of steel with the dimensions shown in Figure 5. Bolts with a diameter of 12e-3 m are placed in both holes. The support arm will be studied with three different load cases in order to evaluate and sort out the most important overall variables. The first load case is a standard linear analysis where the support arm is loaded in the bolts with a total horizontal load of 11.2 kN and the top surface with a total load of 1 kN. The equivalent von Mise's stress and the total displacement are evaluated in the first load case and in the second case the first six eigen frequencies are studied. The third load case introduces nonlinear material behavior and contact between the arm and the surrounding environment as shown in the top right picture in Figure 3.

In all cases the FE model of the support arm is built up with shell 93 elements with the thickness of 2.5e-3 m, and the bolts are modeled with pipe 16 elements. The left-hand side of the support arm is constrained in all degrees of freedom. In the first two load cases the bolts are only allowed to move in the direction of the arm and rotate around their own axes. In the third load case the bolts are given a 1e-3 m displacement in the vertical direction. Figure 5 below shows the boundary conditions of load case 1 and in case 2 the forces are deleted.

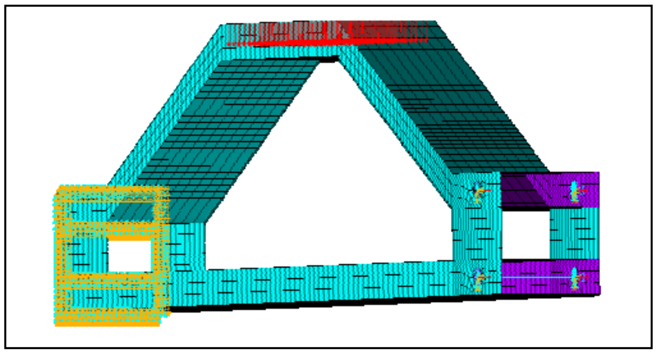


FIGURE 4 BOUNDARY CONDITIONS OF CASE 1.

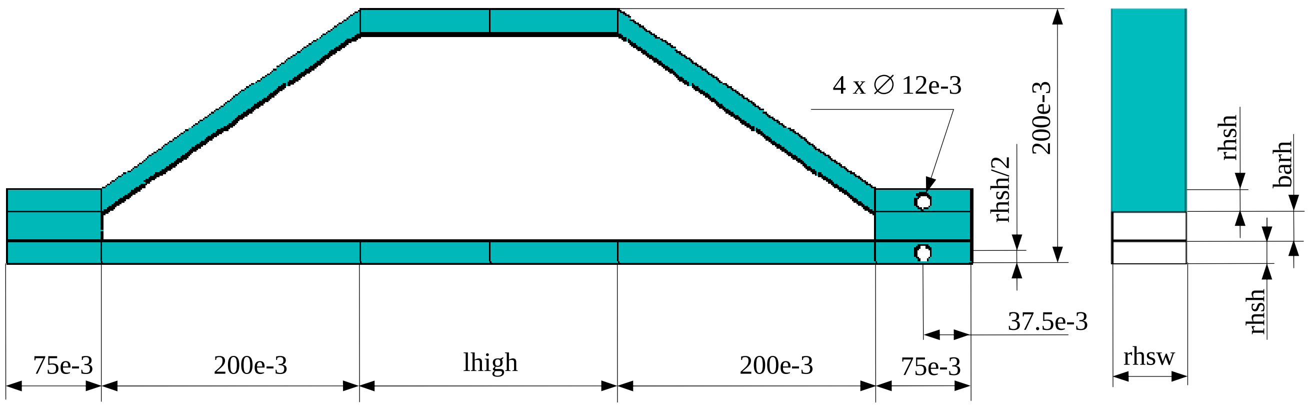


FIGURE 5. DIMENSIONS OF THE STUDIED SUPPORT ARM (IN M).

An analysis, not presented here, was performed with the objective of sorting out the few variables from a set of 10 studied variables that appeared interesting to evaluate further. The variables that were found interesting were the ones presented in Table 1. To evaluate the four variables, a design layout for four variables at two levels in eight runs is chosen. The chosen layout, 2_{IV}^{4-1}, is of resolution IV, which means that first order values cannot be separated from third order combinations of design variables in the statistical evaluation. In Figure 2 the analysis layout and the estimation of effects along with the aliases can be seen. The values assigned to the studied variables are shown in Table 1, and in all load cases the same set of four design variables are used.

The FE analyses are distributed over a cluster of PC computers running Windows NT 4.0 connected with an Ethernet network with 100 Mbps. The computers used have 450 MHz Pentium III processors, contain 256 MB PC100 SDRAM Internal memory. All computers have the same hardware and software configuration to guarantee as equal FE execution time as possible.

TABLE 1. VARIABLES FOR THE FE ANALYSES.

Design variable	Description	Low value (m)	High value (m)
A	Height of the RHS profiles (rhsh)	20e-3	30e-3
B	Width of the RHS profiles (rhsw)	60e-3	60e-3
C	Distance between the two RHS profiles (barh)	20e-3	30e-3
D	Length of the parallel part of the middle section (lhigh)	180e-3	220e-3

RESULTS

Verification of the implementation

To verify the implementation, the first load case was performed with all design variables at their lower values. Table 2 below shows the resulting execution times for different computer setups. The first row is results from the case when all analyses are performed sequentially on a single computer. The following rows show results from distributed analysis with the number of active computers in the first column. The following eleven columns contain the actual FE analysis time for that setup along with a summation and the calculation of mean value and standard deviation. Columns twelve and thirteen presents both the master and slave execution times for the distributed calculation. The last three columns contain data related to the total execution time of the CGI application, i.e. the total evaluation time. The first of these three columns shows the total evaluation time of the CGI application. The column named "FE_time" shows the value calculated as

$$FE\_time = total\, evaluation * no.\, computers / 8$$

and the last column named "relative" shows the following value

$$relative = total\ evaluation / total\ FE\ analysis\, time$$

TABLE 2. RESULT SUMMARY FOR THE FIRST LOAD CASE WITH SAME VARIABLE VALUES (IN S).

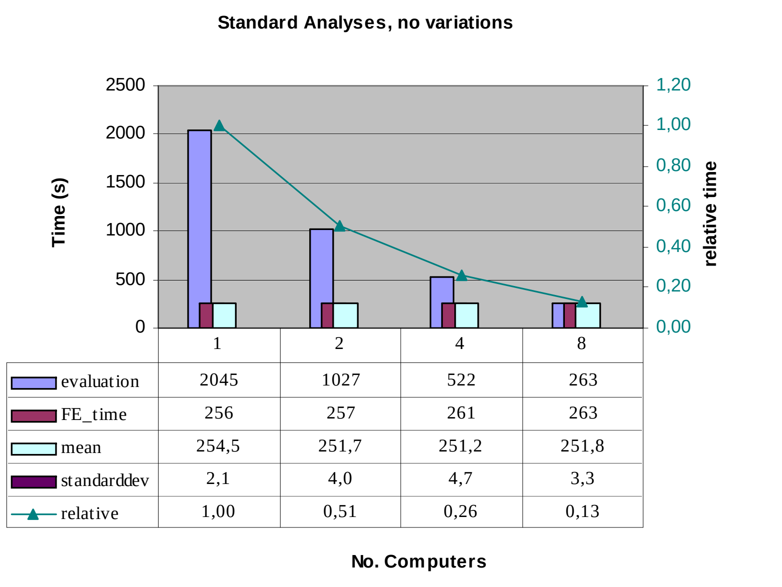
No of Computers	FE analysis times											Distribution		Evaluation		
	1	2	3	4	5	6	7	8	total	Mean	standarddev	Slave	master	total	FE_time	Relative
1	253	252	254	254	249	251	251	249	2013	252	2.0	sequential		2017	252	1.00
1	256	255	255	256	255	256	251	251	2036	254	2.1	2041	2042	2045	256	1.00
2	258	249	250	252	248	253	247	257	2013	252	4.0	2022	1023	1027	257	0.51
4	259	247	249	249	249	250	248	258	2009	251	4.7	2014	519	522	261	0.26
8	259	250	251	248	250	252	253	250	2015	252	3.3	2017	260	263	263	0.13

The small standard deviation in Table 2 indicates that all FE analyses have similar execution times. Table 2 further indicates that the distributed implementation is correct since the "FE_time" differs very little. The fact that the value FE_time increases as the number of computers increases is related to the nature of distributed computing. This can also be seen from the value of "relative" in which the value is close to the optimal sequence of (1, 1, 0.5, 0.25, 0.125) as it should be if the properties of distributed computing had no effect on the total execution time. Table 3 shows results from an evaluation of the same load case with the design variables chosen according to the values in Table 1. In Table 3 the standard deviation has higher values, which indicates that the execution times of the FE analyses differ. The total evaluation time is still decreased considerably by using 8 computers, but the relative values are further away from the optimal values.

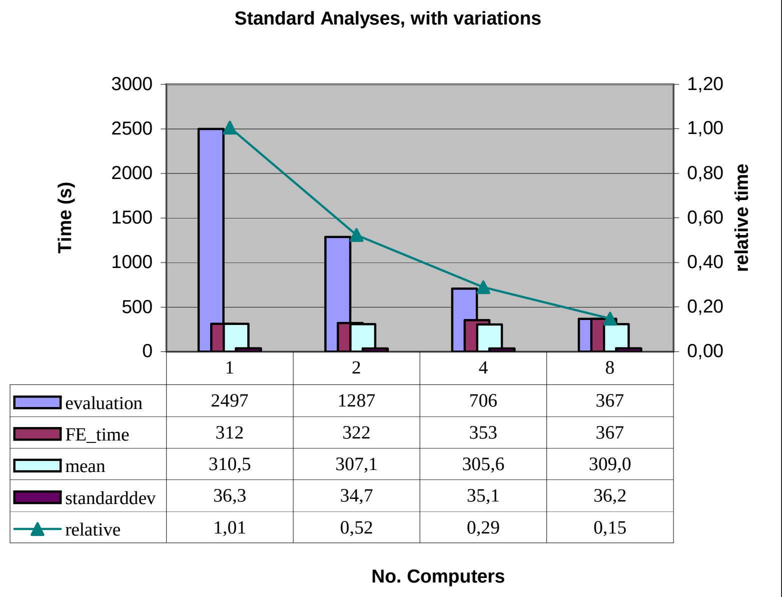
TABLE 3. RESULT SUMMARY FOR THE FIRST LOAD CASE WITH VARITIONAL VARIABLE VALUES (IN S).

No of Computers	FE analysis times											Distribution		Evaluation		
	1	2	3	4	5	6	7	8	total	Mean	Standarddev	Slave	master	Total	FE_time	Relative
1	264	297	322	348	274	290	318	364	2477	310	34.9	sequential		2482	310	1.00
1	255	300	328	346	274	295	325	362	2484	310	36.3	2492	2493	2497	312	1.01
2	254	295	328	340	275	288	320	356	2457	307	34.7	2464	1284	1287	322	0.52
4	252	297	316	341	275	289	314	361	2445	306	35.1	2451	703	706	353	0.29
8	260	294	323	348	272	294	318	365	2472	309	36.2	2477	366	367	367	0.15

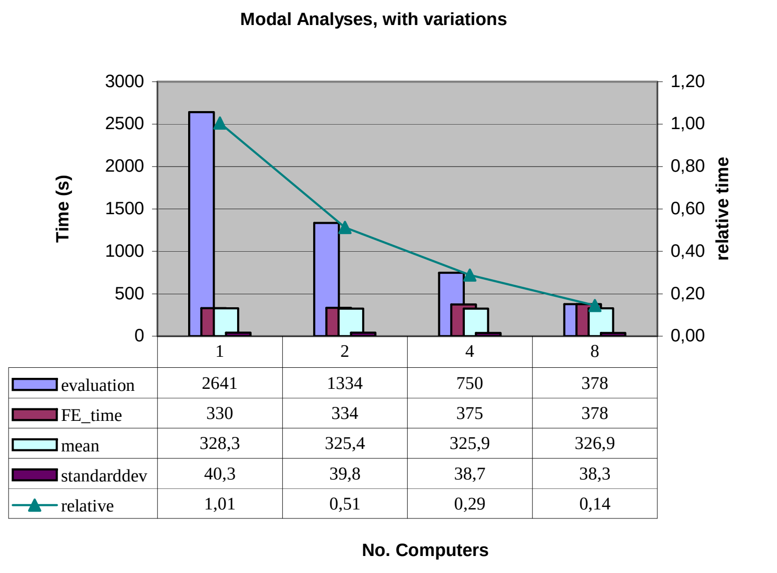
In Figure 6 four graphs show the data from the distributed analyses discussed above along with results from the distributed modal analyses and contact analyses. The graphs show the total evaluation time along with the mean values and standard deviations of all FE analyses. These values and the calculated "FE_time" value are presented as bars, and the "relative" value is shown as lines. All load cases have individually similar mean values independent of computer configuration, which indicates that all computers used behave equally. The influence of the standard deviation is once more highlighted in the contact load cases where the values are notably higher than in the other load cases. The calculation of the relative value is higher as a result of this fact. Throughout all load cases the configuration of four computers is further away from the optimal value then the other configurations. This is related to the order in which the design variables are organized and the different FE analyses are performed. In this example the most time-consuming analyses are 4 and 8 for all load cases, which can be seen in Table 3. When the configuration of four computers is used, one single computer will execute both these time-consuming analyses.



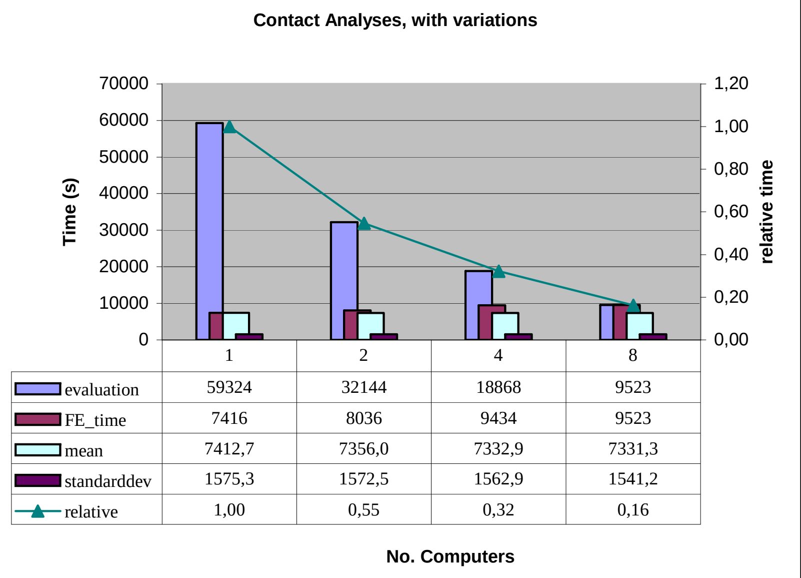
	1	2	4	8
evaluation	2045	1027	522	263
FE_time	256	257	261	263
mean	254,5	251,7	251,2	251,8
standarddev	2,1	4,0	4,7	3,3
relative	1,00	0,51	0,26	0,13



	1	2	4	8
evaluation	2497	1287	706	367
FE_time	312	322	353	367
mean	310,5	307,1	305,6	309,0
standarddev	36,3	34,7	35,1	36,2
relative	1,01	0,52	0,29	0,15



	1	2	4	8
evaluation	2641	1334	750	378
FE_time	330	334	375	378
mean	328,3	325,4	325,9	326,9
standarddev	40,3	39,8	38,7	38,3
relative	1,01	0,51	0,29	0,14



	1	2	4	8
evaluation	59324	32144	18868	9523
FE_time	7416	8036	9434	9523
mean	7412,7	7356,0	7332,9	7331,3
standarddev	1575,3	1572,5	1562,9	1541,2
relative	1,00	0,55	0,32	0,16

FIGURE 6. GRAPHS ON THE EXECUTION TIME OF THE STUDIED LOAD CASES.

Statistical evaluation

The statistical evaluation of the support arm is summarized in Table 4. Mean values of all studied responses are given in the second column. Along with the mean values, the implementation presents all individual values of the studied variables. Thus, if a particular variable combination is of interest that specific response value can easily be accessed. The last column shows the significant variables. The Letters correspond to the Letters assigned to each design variable in Table 1.

TABLE 4. RESULT OF THE STATISTICAL EVALUATION.

Analysis types and studied response functions	Mean value	Significant variables
Linear analysis		
Von Mises Stress	210 MPa	A, C
Displacement	0.29e-3 m	A
Modal analysis		
1 eigen frequence	307 Hz	A, D
2 eigen frequence	404 Hz	A
3 eigen frequence	506 Hz	A
4 eigen frequence	660 Hz	
5 eigen frequence	807 Hz	A
6 eigen frequence	950 Hz	
Contact analysis		
Von Mises Stress	182 MPa	A
Displacement	1.0e-3 m	A, C
Contact pressure	21 MPa	A, C, D
Contact penetration	0.23e-3 m	A

Figure 7 shows the normal probability plot that is the result reported by the implementation. All variables and combinations of them that fall off a line through x-value 0 and y-value 4 should be interpreted as significant. Figure 7 shows that variables A and D are significant for the first eigen frequency. Table 4 contains the result from all evaluations performed and shows that design variable A (Height of the RHS profile) is a significant variable for almost all other load cases also. Variables C and D are significant in some of the studied responses. The data representation in the normal probability plot is abstract but nevertheless an essential part of the evaluation, as it is the basis for further statistical evaluation of the results. Based on the significant variables in the normal probability plot, the more concrete representation of cube plots can be produced. The cube plot presented in Figure 8 shows the result from the first eigen frequency. It can easily be seen in Figure 8 that the influence from variable B is much smaller then from the other two variables. The normal probability plots and the cube plots are important tools to the designer in the evaluation of the results. The results have also increased the designer's insights on which the subsequent decision-making procedures are based.

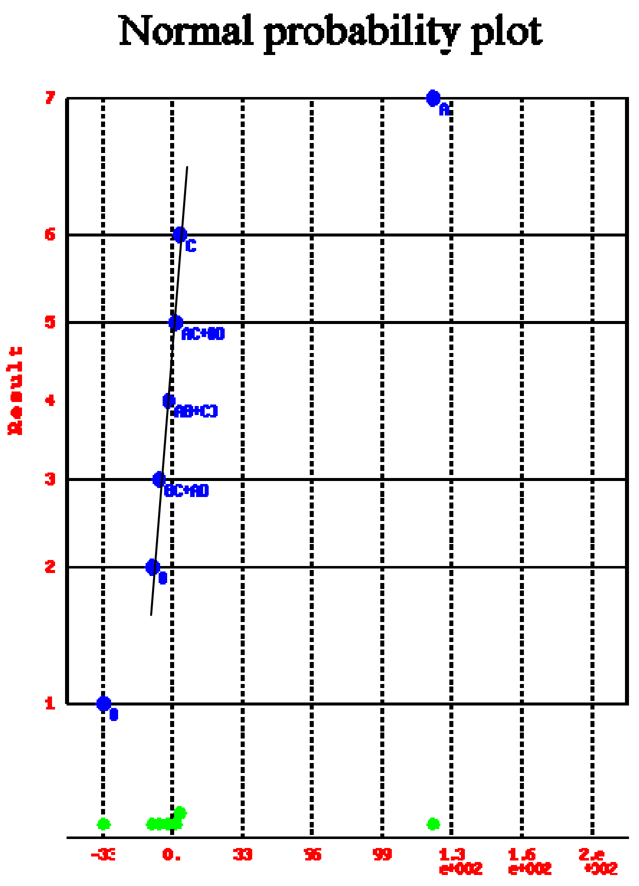


FIGURE 7. NORMAL PROBABILITY PLOT CORRE-SPONDING TO THE FIRST EIGEN FREQUENCE.

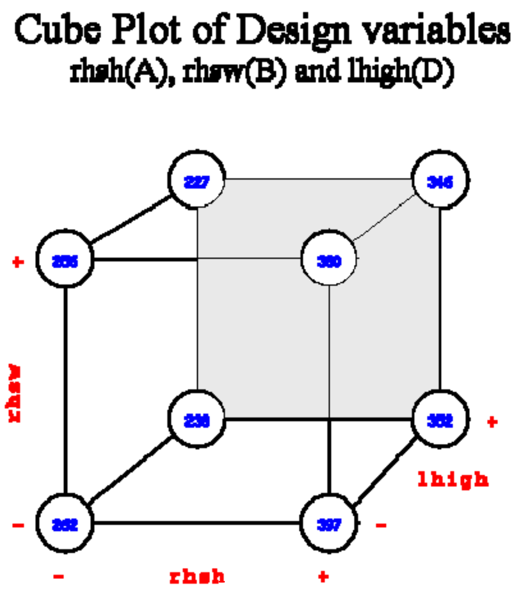


FIGURE 8. CUBE PLOT OF THE FIRST EIGEN FREQUENCY SHOWING VARIABLES A, B, AND D.

CONCLUSIONS

The example presented shows that additional time introduced by distributing the FE analyses over a network is small in comparison to the total evaluation time. On the other hand, the total evaluation time decreases rapidly when the number of computers utilized increases. The implementation highlights an example of efficient use of existing computing resources. In a company the computing resources can effectively be allocated whenever the computers are not allocated for other duties. By using this analysis structure the computer resources can grow in stages and take advantage of the latest computational and network technologies. The Client/Server application also makes it possible for designers in different design teams connected to the Internet or intranet to view and evaluate the statistical result. The implementation facilitates the possibilities for design teams located all over the world to evaluate the result simultaneously. An improvement of the implementation that would further enhance the capabilities of collaborative engineering between different design teams would be an interactive channel for "chat-like" discussion concerning the results. A load balancing routine would in some cases decrease the total evaluation time even further.

REFERENCES

Box, G. E. P., Hunter, W. G. and Hunter , J. S., Statistics for Experimenters, John Wiley & Sons, New York, 1978

R. Butler and E. Lusk, Monitors, messages, and clusters: The p4 parallel programming system., Technical Report Preprint MCS-P362-0493, Argonne National Laboratory, Argonne, IL,1993.

Nicholas Carriero and David Gelernter, LINDA in context, communications of the ACM, 32(4): 444-458, April 1989.

Eriksson, M., Andersson, P. O., Burman, Å.,"Using Design-of-experiments techniques for an efficient finite element study of the influence of changed parameters in design", 1998 Ansys Conference Proceedings Vol.2, Ansys Inc., 1998, pp.63-72

Eriksson M., Burman Å., "The Role of Design of Experiments in Predictive Design Analysis", 1999 ICED Conference Proceedings Vol.1, WDK 26, Munich, 1999, pp. 385-388.

J. Flower, A. Kolawa, and S. Bharadwaj, The Express way to distributed processing, Supercomputing Review, pages 54-55, May 1991.

Geist, Al et. al, PVM: Parallel Virtual Machine, A Users' Guide and Tutorial for Networked Parallel Computing, The MIT Press, Cambridge, Massachusetts, USA, 1994.

Geist, G.A, J.A. Kohl, P.M. Papadopoulos, `` PVM and MPI: A Comparison of Features '', Calculateurs Paralleles , 8(2), pp. 137--150, June, 1996.

Johnson, D.H., Heald, E. E., "Design of experiments methods for a nonlinear buckling FEA study", 1998 Ansys Conference Proceedings Vol.1, Ansys Inc., 1998, pp. 513-520

Marczyk, J., Recent Trends in FEM, Proceedings of the NAFEMS World Congress'99 on Effective Engineering Analysis, vol. 2, Newport RI, 1999, pp. 1327-1342.

Message Passing Interface Forum, Mpi: A message-passing interface standard. Computer Science Dept. Technical Report CS-94-230, University of Tennessee, Knoxville, TN, April 1994

GLOSSARY

CGI	— Common Gateway Interface
CPU	— Central Processing Unit
DOE	— Design of Experiments
FE	— Finite Element
GIF	— Graphics Interchange Format
HTML	— Hypertext Markup Language
JAVA	— Object-orientated program language
Mbps	— Megabits per second (one million bits per second)
MHz	— Megahertz
MPI	— Message Passing Interface
MPP	— Massively Parallel Processors
PDA	— Predictive Design Analysis
PVM	— Parallel Virtual Machine
RHS	— Rectangular Hole Section
SDRAM	— Synchronous Dynamic Random Access Memory
VRML	— Virtual Reality Modeling Language
WINCGI	— Windows Common Gateway Interface
WWW	— World Wide Web
WPVM	— Windows Parallel Virtual Machine

VARIABLES USED IN THE EXAMPLE

barh	— Distance between the two RHS profiles
lhigh	— Length of the parallel part of the middle section
rhsh	— Height of the RHS profiles
rhsw	— Width of the RHS profiles